Muhammad Kashif Nawaz

Exploring End-to-End Data Engineering A GCP Case Study

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Master of engineering, information technology, cloud-based software engineering
May 2024

ABSTRACT

Centria University of Applied Sciences	**Date** May 2024	**Author** Muhammad Kashif Nawaz
Degree programme Master of engineering, information technology, cloud-based software engineering		
Name of thesis Exploring End-to-End Data Engineering: A GCP Case Study		
Centria supervisor Ali Asghar Khavasi		**Pages** 55 + 20

In an era dominated by data, the role of Data Engineering (DE) has become quite important in crafting robust solutions for data management. Data Engineering has become an integral part due to the rise of machine learning and artificial intelligence. In machine learning we need big amounts of data so handling that data is an integral part before starting the ML adoption in an organization. Data engineering ensures that we have efficient data processing which includes storage and retrieval along with maintaining data quality and supporting batch processing or real time analytics which are essential for machine learning and analytics workflows.

This thesis provides a hands-on exploration of new cloud technologies, offering insights into real-world challenges and solutions. By looking into a complete flow of data from collection to reporting this thesis aims to link the theoretical knowledge and practical application, preparing individuals for the demands of the evolving DE field. This thesis proposes a comprehensive case study, aiming to construct an end-to-end solution using Google Cloud Platform (GCP) and other tools. Study covers the scenarios of how data engineering simple workflow looks like in real world. For this purpose, Raw data from multiple sources is collected into cloud storage via the process called Extract, Transform, Load (ETL) and then apply the transformations into the data warehouse and in last build a business dashboard containing different key performance indicator (KPI). For orchestration Mage AI and for transformations and dbt library is used. Business users are more interested in the data in the context of their business domain, so dashboards are quite useful for them when they want to analyse the historic trends or analyse the performance.

This thesis introduces libraries databases and different modern tools which are being used in data engineering along with use of cloud providers like Google cloud provider in this case how to integrate open-source tools like mage AI in that workflow. This work produces different Python scripts, dbt scripts, terraform scripts and database views into the production environment.

Key words
Data Engineering, Data warehouse, Extract Transform Load (ETL), Google Cloud Platform (GCP)

ABSTRACT
CONTENTS

1 INTRODUCTION

People often use the term that data is the ‘new oil’ (Parliament, 2020) because data is so valuable. In this digital age, data is a big deal But, unlike oil, data keeps growing and can be used again and again. Data is a big asset for businesses, helping them improve what they do now, create new things, and grow. It helps the economy grow and changes how we do things. Data helps us in improving our decision making and picking up smart choices so the government, businesses, and common people all can do better when we have access to the data. It helps organizations make smart decisions based on solid evidence, preventing small problems from becoming big ones.

With data, you can see if your strategies are working and find effective solutions to problems. Plus, it helps you back up your arguments, explain decisions, and be more efficient with your resources. Artificial intelligence innovations are shaping our future by driving technologies such as big data, robotics, and IoT, with generative AI adoption by common people, is making AI even more popular across various industries. Data lifecycle in shown below in Figure1 is the complete cycle of data.

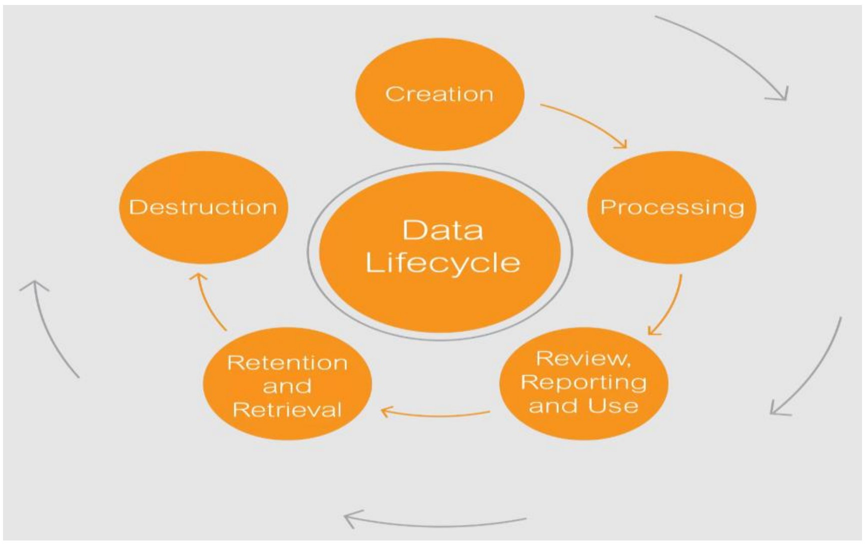


FIGURE 1: Data Lifecycle

Data engineering makes sure that the data used for training AI models is high quality and readily available, which is crucial because how well this data is handled can make or break AI projects, allowing data scientists to focus on working with models instead of spending time in data prep and management.

In 2023, a survey by IBM found that 42 percent of big businesses are using AI in their work, with 40 percent more thinking about using it, and 38 percent have started using generative AI, with another 42 percent thinking of giving it a try (IBM, 2024). In today's world we can see the explosion of AI and machine learning and the quality of an AI system depends on the models and the data it trains from which makes data engineering crucial for organizing and managing data.

Data engineering is about using tools to handle lots of data by processing, transforming, and storing it and cloud computing plays an important role here, as it offers scalable, and cost-friendly ways to manage these tasks, letting data engineers use on-demand resources and SaaS services which greatly reduce the time and effort. In this thesis the focus will be on some of those services offered by Google Cloud Services and some open-source tools like Docker, dbt, mage AI and terraform.

2 BACKGROUND

In today’s data-centric era, the management and utilization of data are one of the most important businesses’ processes in every industry. Digital technologies produce more and more data, which grows exponentially every year. Therefore, the problem for businesses is how to be able to use the potential of data to make strategic decisions and create a competitive advantage. Thus, the discipline called Data Engineering emerges as an art to design, develop, and maintain data pipelines and infrastructure to support the efficient and reliable processing of data.

On the one hand, data engineering combines data collection, storage, analysis, and transformation, which on the other hand, have the aim of extracting actionable insights from raw data. However, as today more and more businesses use cloud computing services for data storage and handling, platforms like Google Cloud Platform have become essential tools for every data engineering project within an organization. Hence, to start with, discuss the high-level overview of Google Cloud Platform.

There are four ways via which GCP can be accessed which are Web Console, Command Line Interface (CLI), API, and Mobile App. Every GCP resource is organized into the GCP resource hierarchy as follows. All resources are grouped into projects. Projects can also be organized into folders. These folders and projects must be structured. All structured folders and projects are organized together under an organization node. Project folders along with organization nodes are where policies are defined. Policies are inherited downstream, dictating who can access which resources. Everyone must belong to a project, and a billing account must be associated with it. In this context, this thesis is set to explore end-to-end data engineering process in the GCP ecosystem.

The thesis looks forward to demonstrating how organizations can effectively design, implement, and manage data pipelines to extract valuable insights and support informed decision-making using GCP’s strong features and capabilities.

2.1 Motivation

In recent years we have seen a remarkable growth in the adoption of artificial intelligence which has created a demand for data engineering skills. Companies are collecting massive amounts of data so they need professionals who can effectively manage and process large amounts of data. The motivation behind writing this thesis is to look into the world of data engineering and to gain a deeper understanding of tools and techniques that are being used and how they can be applied to real world scenarios.

Professional's forecasts that the worldwide big data analytics market will experience a healthy growth rate of 30.7%, reaching an estimated value of $346.24 billion by 2030 (Market Research Future, 2022). This trend is expected to continue as more and more companies adopt artificial intelligence and machine learning, which means we will be having significant volume-s of data so we need professionals who can design and implement scalable and efficient data processing solutions.

This case study implementation serves as a guide for providing hands-on experience with various data engineering tools and technologies both with GCP and open source. It also gives the user a high-level view about data engineering workflow and how different components are pulled together and what is the role of each technology. In Figure 2 below here data flow is presented how in a typical real world scenario data flow normally is and how different components interact with each other.

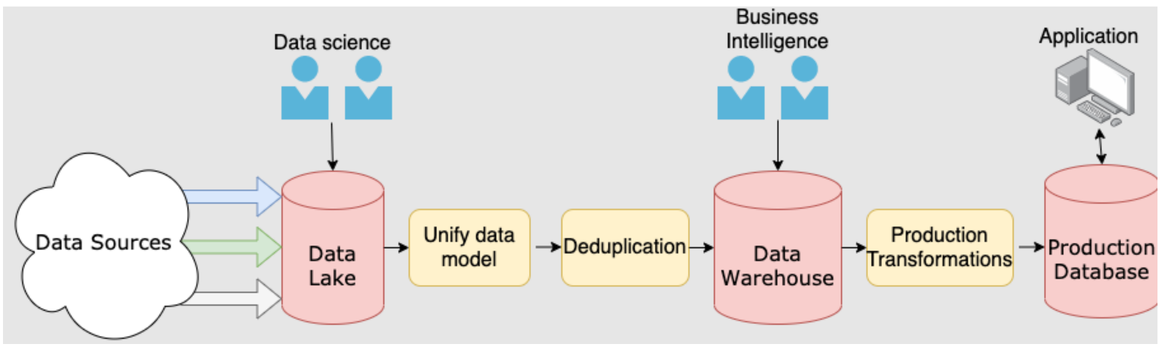


FIGURE 2 Data Flow

The aim is to provide how high-level understanding of how engineering works in real life and the basic processes that the majority of data engineering projects follow. Different tools used in data engineering such as Google cloud services, docker, dbt, terraform, MageAI will be examined and how they fit in and the overall data pipeline. By focusing on these we can have a better understanding of how the raw data is converted into the insights that can be utilized in making business decisions. The aim is to gain hands-on experience with these technologies and tools to be market ready and have basic skills required to be productive.

2.2 Focus and constraints.

The focus of this thesis is only to give you a general idea and this only covers some of the tools like Google Cloud services, docker, terraform, dbt, python and SQL. In the industry there could be a variety of tools or variety of combinations being used and companies that really depend on their needs. We will be using open-source data as our raw data here and we will be applying transformations and then building that dashboard to show how the flow works in real life. It only covers batch processing and data from static sources.

The thesis seeks to identify and provide coverage of all the steps and processes involved in executing data engineering within the GCP platform. This necessitates coverage of the GCP data collection, storage and processing, transformation, and analysis of data as well as the visualization aspects of data treated in this research. By covering these aspects adequately, the thesis aims to educate the reader on how the entire data engineering systems work from the beginning to the end. The second point revolves around the use and interactions with the various services and tools offered in the Google cloud platform. For instance, the data warehousing and analytics tool such as BigQuery, storage tool such as cloud storage and workflow orchestration tools such as Apache Airflow are vital services and tools that will be extensively used in this research. By highlighting their utilization this thesis aims to present the actual applicability and use cases for the GCP-based data engineering solutions.

The third introductory point seeks to address the integration of prevalent programming languages and other technologies. This involves python and SQL programming languages and containers such as docker and infrastructure technology like Terraform. This aspect is meant to provide a clear depiction of how to integrate these languages and technologies in actual practice to develop efficient data engineering tools. Lastly, the proposed methodology seeks to provide insight and justification for real-life scenarios and practicality. The main goal of this methodology is to elaborate on existing frameworks and coverage to address data engineering problems as they manifest in practical life.

Constraints:

The thesis has several constraints regarding resources, including time, budget, and access to GCP services. Since the data engineering process is extensive and complicated, it must be limited by the following resources to ensure its feasibility and implementation.

Although the thesis attempts to explore as many dimensions of the data engineering process as possible, its depth and breadth are limited by the time frame. The chosen aspects of data engineering are of primary importance for describing and analyzing the process, so some of the activities may be described in more detail than others according to their relevance and complexity.

In addition to that, the thesis must adhere to technical limitations resulting from the specifics of GCP service configurations and optimization, program languages and technologies integration, and consideration of the existing infrastructure and systems throughout the execution.

2.3 Research Questions

Research questions help to structure this study since they guide the researcher to focus on specific areas and aspects of the topic of interest. Since the purpose of the research is to examine end-to-end data engineering within the Google Cloud Platform environment and integration of other open-source tools with it, these questions are designed to cover the most significant elements of the thesis.

Some of the questions discussed in this thesis focus on the implementation and best approaches of integration of tools. This study discusses how can GCP services and tools be used to develop and implement end-to-end data engineering solutions. The question revolves around understanding the ability of the defined GCP services and tools, such as BigQuery, Cloud Storage, and Apache Airflow, to build high throughput, smoothly working data engineering pipelines.

Thesis will be focused on some of the best approaches when collecting, storing, processing, and analysing the data within GCP. The second question targets the exploration of approaches and strategies that might help perform basic data engineering works, such as data collecting, storing, and analysing/processing in one place. It also implies the investigation of some performance and integrity measures.

One question to consider is how programming languages and relevant technologies, such as Python and SQL, can be incorporated into the pre-set concept of data engineering within the confines of GCP.

This question encompasses the usage of programming languages and technology, Python and SQL, respectively, in the course of work on defining how to incorporate the concept into GCP. In addition, non-target programming language technology, Docker, is defined. In GCP, data engineering includes programming design directly executing typical GCP instruments, using Python as the manager language, and employing SQL, a language specifically developed to ask relevant questions.

It is important to investigate what the main issues experienced and factors to consider are when efficiently developing and implementing a GCP data engineering solution. The fourth question aims at identifying the main issues and concerns that might occur in the course developing the GCP data engineering solution and what are the different challenges which are there when it comes to integrating the open-source solutions with the Google cloud ecosystem and how do both of these complement each other.

It is important to consider what the applicable real-world uses and conceivable consequences of a solution centred around GCP might be. The final question addresses real-life usage and outcomes of GCP-centred solutions. By highlighting the answers to the research questions in the study, it is possible to guide people and entities interested in the GCP data engineering and seek to address the issue successfully. These issues can help those interested in GCP data engineering to make the right choice by addressing GCP data engineering on the identified dimensions.

When efficiently developing and implementing GCP data engineer solutions this study aims at identifying the main issues and concerns that might occur in the course developing the GCP data engineering solution. And also, implementing the best approaches when collecting, storing, processing, and analysing the data within GCP. Data warehouses are a cornerstone of data-based decision-making that gives users access to needed data and the ability to analyze it. This, in turn, leads to the generation of actionable insights that spur business growth. One of the major data warehousing issues is the breakdown of data silos and fragmentation. Data lying in organizational systems, databases, and apps is hard to find, combine, and analyze. Data warehouses bring all the data scattered throughout an organization into one centralized data store, negating the need for data silos.

2.4 Thesis Organization

The organization of thesis is presented in this section. It has been reported so that the reader can get the idea what to expect from each chapter and get a better understanding of this whole thesis.

Essential tools and methodologies used in data engineering, covering programming languages, containerization, orchestration, infrastructure as code, version control, agile principles, data warehousing, and preprocessing techniques will be covered in chapter 3. In chapter 4 discussion about existing work on data engineering challenges, the role of Google Cloud Platform in data engineering solutions, and relevant tools and technologies is covered.

Chapter 5 mainly discusses the details of research methodology, a case study approach with action research principles along with the research design and data collection methods. Next, chapter 6 provides a step-by-step guide to implementing the data engineering framework involves Google Cloud Platform setup, data pipeline construction, infrastructure configuration, data transformation, and dashboard creation.

In Chapter 7 demonstrates the applications of the case study implementation on Google cloud platform along with opensource tools and it also contains summarized key findings and conclusions. Lastly in chapter 8 the results will be shared, and future work related to this thesis.

3 TOOLS AND METHODS

In this chapter most of the tools will be covered that we used in this case study. A variety of tools have been used such as Python, SQL, MAGE AI, Google cloud services, github, terraform, Docker and Looker. Deep dive on how these tools are integrated with the Google cloud services and what their role in the data engineering workflow is. Also, looking at how these tools are addressed in the analytics ecosystem and how we can manage these in the cloud as well as on the on Prem systems. Also, discussion about some of the concepts related to data warehousing, Master data management and project management like agile methodology and its types is covered in this chapter.

3.1 Technical Tools

3.1.1 Python

Python is a high-level object-oriented programming language. Dynamic typing along with its built-in data structures and dynamic typic makes it a preferred choice when it comes to RAD and for integrating different components together (McKinney, n.d.). Python gained popularity because of its simple syntax and readability. It has got a lot of libraries related to web development, data analysis, machine learning and desktop development.

Python is a great choice when it comes to data engineering because it tends to integrate seamlessly with the big data processing frameworks like Apache spark via pyspark library which enables the data engineers to scale their data pipelines easily and process large amounts of data. Python is also a multi paradigm language which supports both object-oriented and functional programming styles. It can be used for many tasks related to data engineering like data cleaning, data transformation, aggregation, and visualizations. A large community of people use and contribute to python libraries, provide help in learning it via video tutorials too which makes it easy for new users to adopt this.

Python also offers the libraries to process real time streaming data which enables real time data analytics. GCP also provides its sdk in python via which we can integrate our application with cloud services and control those services from there. The best thing about this is that a lot of libraries and tools in python are opensource which provide us the flexibility to customize the library according to our needs.

3.1.2 SQL

SQL stands for Structured Query Language, is a powerful tool used for managing and manipulating the databases. It is commonly used to create and manage relational databases and query the results from them (Rockoff, 2022). This makes it easy to structure and organize large volumes of data in a logical manner.

In data engineering, SQL plays quite a crucial role as it provides a standardized way to interact with databases which makes it easier to extract insights from data. In data engineering it is immensely helpful as it allows data engineers to write queries to extract specific information from data efficiently. These queries can filter, sort, aggregate, and join datasets. It also enables engineers to prepare the data for analysis or further processing. With the help of SQL, we can create and modify database schemas where we can easily manage our structured data. It helps us in designing tables defining relationships between them and ensuring the data integrity.

Additionally, SQL is used to create views which simplify data access and ensure consistency. A lot of tools that are used in data engineering domain also use the SQL format which makes it easy for data engineers to adapt and quickly get hands on with that.

3.1.3 Docker

Docker is an open-source containerization platform. It helps developers package all the dependencies like files, libraries, and settings into one package called a container. This container is like a portable, lightweight version of your program that can run on any computer that has Docker installed. So, instead of worrying about whether your program will work on different machines, you can just ship it to a Docker container and know that it will run the same everywhere. Docker makes developers' lives easier by ensuring consistent and reliable software deployment (Karl Matthias, 2015).

In dta engineering we must build complex data pipelines with multiple steps, each requiring specific software configurations and dependencies. It allows data engineers to create isolated containers for each step of the pipeline. This means we can have one container for data cleaning, another for transformation, and so on. Since each container is isolated and self-contained and has everything, it needs to run it makes very easy for us to share the projects and work that we did to deploy to other environments like from development to staging or from staging to production.

Docker makes it easy to scale our pipelines by spinning up multiple containers in parallel, handling large volumes of data efficiently whenever needed. Almost all the cloud providers allow deploying the docker containers to their platforms, in fact it has become kind of a de facto industry standard when we must do the deployments and their easy integration with other tools, which makes it a must have in the world of data engineering. (Naik, 2017)

3.1.4 Mage

Mage is an open-source data orchestration tool used to manage data pipelines. Mage gives us an edge in one thing that we can write code and use the modular code blocks available there previously (Mage, 2024). We can do the extract transform and load along with the orchestration of data pipelines where we run and monitor the statuses of data pipelines. Orchestration tools are very important in manging data pipelines and Mage helps us in managing that. As its quite interactive and has user friendly features it is the best choice when it comes to using this for a small team or a small project.

We can also deploy it on any cloud of our choice with just two commands, and it gives us the flexibility to write code in multiple languages like python, R or SQL which is an added benefit of flexibility. Mage was developed by keeping developer experience in mind as developers will be using this product most of the time so it is good as it has user friendly features which helps them in collaboration, observability, and deployment (Mage, 2024).

3.1.5 Terraform

Terraform is an Infrastructure as Code (IaC) tool which lets users define resources in human readable configuration files which can be reused, versioned and shared (Hashicorp, 2024). It works with both on-prem and cloud resources. It really helps us in scenarios where we have to manage all the infrastructure in its lifecycle and have to manage from high level components to low level components like storage and compute. Terraform has a multi-cloud support which allows users to manage infrastructure across different cloud platforms like aws, azure and gcp at the same time.

In data engineering we know how difficult it is to manage infrastructure across different cloud platforms and on premises environments and when we must manage a number of data pipelines that is where Terraform really shines because we can manage everything via code so we can maintain the consistency and standardization in that.

Also Terraform helps us in implementing the security and compliance requirements across the whole infrastructure along with disaster recovery and backup. It is state management feature helps us in recovery from those kinds of scenarios. It really reduces our effort required in provisioning and managing the infrastructure components and helps us in improving the reliability and scalability of our systems (Brikman, 2022).

3.1.6 dbt

dbt or Data Built Tool is an open-source framework which helps us in transforming and modelling data in data warehouse using SQL (Handy, 2017). It is built on top of SQL so users can easily write SQL queries as models which are used to transform the data and also, we can write the test using the simple yaml based syntax. dbt makes it very easy to manage complex data workflows and collaborate in large teams.

It is integration with popular data warehouse technologies like snowflake, Google big query, AWS Redshift and others makes it a good choice. In data pipelines we must simplify the pipelines by breaking them into small and reusable components which can be combined to create more complex pipelines and via using dbt we can achieve that with the help of components called models, macros and packages. Dbt helps in streamlining the repetitive tasks such as data transformations and testing.

It is the ability to apply the transformations incrementally and loading all the models that are related to the affected model helps us in managing the consistency. Using Git developers can manage and track all the changes made to the dbt pipelines over time and if something needs reversal that can be managed by git too. As all the code is on git developers can easily collaborate and work on different modules in the same codebase.

3.1.7 Git

Git is an opensource version control system that helps developers in tracking the changes in codebase along with collaboration in teams and managing different versions of their projects (Jon Loeliger, n.d.). It was created by Linus Torvalds in 2005 and has now become a de facto standard when it comes to version control systems in code management.

With git, developers can track changes that have been made in their repositories over time. This enables them to fix bugs concurrently without affecting others and it provides different options like branching, merging and resolving conflicts between different branches, so it makes it easier for teams to collaborate on a single codebase.

Every developer has a local git repository on their machine which allows them to work offline and then they can merge those changes with the central repository where they can add new code or in case if there are some conflicts resolve those using the pull requests which senior developers review before merging it into the central repository.

In data engineering it is very important to manage the data pipelines so data engineers use git to manage different versions of data pipelines. They also have different branches for different tasks like development branch, testing branch or a new feature branch and then there is this main branch which represents the production so the teams working on different projects can have their own branches and work independently. It is integration with different cloud platforms like Azure data factory, Google data cloud flow etc where they use the gitOps to enable the continuous integration and continuous deployment (CI/CD). It helps data teams in collaboration due to which they have less errors, more reliability and maintainability of their data infrastructure.

3.2 Agile Methodology

Agile methodology is the core when we have to implement any project related to IT or in data engineering projects, we tend to follow this to plan and execute the projects. To ensure that a project is completed on time and meets its targets we divide the tasks into small chunks and then we make the sprints if we go with the scrum. In Scrum, the Product Owner (PO) decides what to build, the development team builds it, and Agile emphasizes delivering value to the customer in small increments, with crucially gathered feedback incorporated into the process (Williams, 2023).

Study focuses on using scrum to manage all the project but there are other ways to manage the projects also. Scrum teams operate in a series of sprints. Before each sprint, a sprint planning meeting is conducted by the Scrum Master, where the PO and the development team attend, deciding on the tasks to be added to the sprint backlog. Progress can be tracked via a Scrum board. Daily standup meetings are held, allowing team members to share their completed tasks, ongoing work, and any blockers. Each sprint concludes with a sprint review, showcasing completed features, and a sprint retrospective to assess areas for improvement. Kanban, in contrast, lacks sprints; it operates as a continuous process and doesn't have a sprint backlog. Tasks are chosen based on team capacity, with items moving from development to testing, creating space for new items to be pulled from the product backlog. Daily standups, demos, and retro meetings are part of the Kanban process, resembling Scrum practices.

In Scrum, without estimates, we cannot set boundaries or soft targets that we must achieve. Without estimates, we do not know the target or how long it is going to take, which business users or investors may not understand, and they usually want something in terms of timelines. We know that most of the time estimates are not perfect, but it is an estimate; it doesn't have to be perfect. That's why each project or activity allocates some time as a cushion period, hoping that if something goes wrong, we can use this time to meet the deadlines. Millions of software have been developed with the help of estimation. Only a novel idea with no requirements may not work with estimates, but much of the work is repetitive, or we are building on what was previously developed, so we know how much time it will take. Counting stories and making projections is also a form of adding estimates. Counting stories via the stories map is something business users do not agree upon because they are used to having estimates, and they plan things accordingly. Counting stories is also a task that they must do every time, so it is kind of dynamic, which businesses do not like.

3.3 ETL vs ELT

ETL and ELT are two types of approaches via which we can stage, transform, and load the data. In ETL approach we do the transformations first then we go for the data storage like in a cleaned format and in a standard format in our data warehouse but in contrast the ELT approach focuses primarily on the loading of data into the data lakes and then after it has been staged, we perform the transformations. As seen in Figure 3 below the difference is quite clear and both approaches have their own advantages and disadvantages and their own use cases where these approaches are best suited.

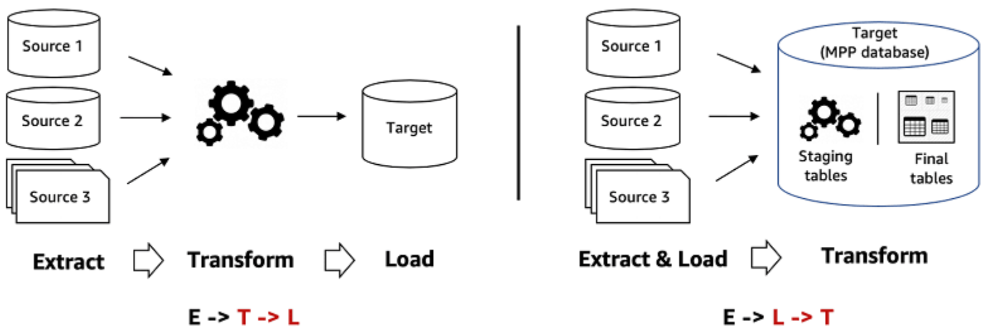


FIGURE 3 ETL vs ELT

If the data is less, then we normally go for the ETL approach as we have to do the transformation first and then we can carry on the loading process so with less data we can transform the data quite quickly. When we have plenty of data or we have a big data problem or we have a scenario where we have to retain the original project data then we first save the data into the data lakes or into the databases and after that we apply the transformations on top of it so now we have two copies like one original which contains no changes or modifications. This approach is also useful when we have to refer back to the original raw data in case of any mishap or in case we have to verify some of the numbers or some of the data then we have the original copy retained or in some cases we have to build transformations are a bit differently so when we have the availability of raw data then it is an added advantage that we do not have to pull the data into the data lakes again because we have the data available over there and we can easily do the transformations on top of that.

3.4 Data Warehouse

Data warehouse is used to store data for analysis and reporting of structured and semi structured data from multiple data sources (Nambiar & Mundra, 2022). Data warehouses are central to the field of data engineering since they are used for storing and handling vast amounts of the data from various sources. A data warehouse is a database that contains data from various sources and allows the data to be queried and accessed together for business purposes. The ETL process is frequently utilized to transport data to a data warehouse.

Data engineers are responsible for the design, construction, and maintenance of data warehouses that are scalable, performant, and reliable. Data warehouses are a cornerstone of data-based decision-making that gives users access to needed data and the ability to analyze it. This, in turn, leads to the generation of actionable insights that spur business growth. One of the major data warehousing issues is the breakdown of data silos and fragmentation. Data lying in organizational systems, databases, and apps is hard to find, combine, and analyze. Data warehouses bring all the data scattered throughout an organization into one centralized data store, negating the need for data silos (Carruthers, 2022).

Data warehouses are also critical for data analysis and report generation, enabling users to draw useful conclusions from examining several data sources. Data storage and retrieval efficiency is the one other issue that a data warehouse helps address. Databases may be unable to conduct analytical queries and manage considerable data volumes. Data warehouses have been architected with optimized data processing structures, indexing, and optimized querying abilities in mind. As a result, they enable rapid data retrieval and data analysis. Data engineers build a reliable storage pool that enables organizations to make informed decisions about their data (Garani, 2019).

3.5 Data Preprocessing

Within the field of data engineering, data preprocessing is of primary significance due to its role as the foundation upon which all subsequent data analysis, modelling, and visualization efforts rest. More specifically, data engineers are tasked with developing and implementing data preprocessing workflows that are tailored to meet the unique needs and constraints of their respective organization’s data landscape. Using these techniques and tools we can make use of raw data to transform and clean this so that it can be easily utilized by the data scientists, data analyst or anyone interested in working with the data which will make it easy for them to analyze or use it for machine learning inputs (Gonzalez Zelaya, 2019).

Data cleaning involves the identification and removal of errors and inaccuracies, such as apostrophes, blank entries, upper case spelling errors and others. Data transformation converts data from its original format, such as an Excel table, into a format conducive to analysis and modelling by standardizing or normalizing it, or by aggregating or disaggregating.

Data integration involves the merging or combining of data tables or sources that involve similar information. Data reduction is the scaling down or simplifying of data sets to their most germane and meaningful subset. Data preprocessing ensures that data sets are clean, consistent, and structurally prepared for more accurate and precision tools by analysts and other professionals (Nada Elgendy, 2016).

Data storage and retrieval efficiency is the one other issue that a data warehouse helps address. Databases may be unable to conduct analytical queries and manage considerable data volumes. Data warehouses have been architected with optimized data processing structures, indexing, and optimized querying abilities in mind. As a result, they enable rapid data retrieval and data analysis.

4 LITERATURE REVIEW

This chapter contains the discussion related to literature review in review of work in terms of the data engineering and workflows. Data engineering makes sure that the data used for training AI models is high quality and readily available, which is crucial because how well this data is handled can make or break AI projects, allowing data scientists to focus on working with models instead of spending time in data prep and management. The chapter has been organized in a way that we can have a discussion related to what is available and the challenges and tools we have now.

4.1 Research Background

In this section, the discussion is related to previous work that has been done on this topic. Data is a big asset for businesses, helping them improve what they do now, create new things, and grow. It helps the economy grow and changes how we do things. Data helps us in improving our decision making and picking up smart choices so the government, businesses, and common people all can do better when we have access to the data. It helps organizations make smart decisions based on solid evidence, preventing small problems from becoming big ones.

With data, you can see if your strategies are working and find effective solutions to problems. Plus, it helps you back up your arguments, explain decisions, and be more efficient with your resources. Now storage is cheap as predicted in the paper by J. Gary also and we can make use of that by separation of compute and storage (Gray, 2000). Artificial intelligence innovations are shaping our future by driving technologies such as big data, robotics, and IoT, with generative AI adoption by common people, is making AI even more popular across various industries. These solutions must be scalable horizontally and vertically and be reliable.

Resources can easily be added or removed on demand rather than having our own on premises data centers cloud providers easily solve this challenge giving us a variety of options in terms of additional locations and new technologies and tools' availability on the fly. It can be noted that such an approach is aimed at a comprehensive investigation of the data engineering process on GCP from the beginning to the end stages. For example, it includes data collection, processing, analysis, and visualization. By using representative research, the thesis aims to explain how GCP solutions can be used to solve practical problems in data engineering.

4.2 Challenges in Data Engineering

Currently there are some challenges when it comes to the data engineering field like data volume, variety, velocity, quality, privacy and security and most importantly, a shortage of skilled professionals who understand the domain and can create the scalable and good solutions. A tremendous amount of data is being generated by different sources and, with the introduction of IOT, we have seen a major growth in data generation from different sensors. Data comes in a variety of formats and shapes and from different sources and it is a challenge to integrate that and make it easy for analysis.

Generally, data is divided into two major categories; structured data and unstructured data. Structured data is kind of organized data and is often stored in tables, databases and in form of rows and columns. While unstructured data has no predefined format and comes in multiple format and shapes like photos, videos and documents. Data velocity is related to the fact that data is being generated at an impressive pace like never before. IoT generates enormous amount of data for which we need cloud computing which offers robust resource for processing and analysing data.

Research in this area focuses on enhancing the accuracy of big data analytics for IoT, enabling the extraction of insights from generated data (Parth Goel, 2017). The data quality is a major challenge as we are now focusing on training AI models, so we need good quality data to train those models to expect good results and accuracy.

A study by researchers in data domain found that almost 20-71% of the labels in real world datasets were wrong and 17-99% of data points were duplicated which will impact the downstream model's trainings that depend on this dataset (Croft, et al., 2023). With enormous amount of data comes huge responsibility to handle that carefully. As almost everything is online these days from our banking records to our medical health records our most sensitive data is in soft form. New methods are needed to make sure that data is safe and prevent it from any unauthorized and unauthenticated access. As the volume of data grows, we need scalable solutions so that they can handle the increased loads and adapt to the changing requirements.

These solutions must be scalable horizontally and vertically and be reliable. Here cloud computing plays an important role in solving this challenge as we can easily add or remove the resources on demand rather than having our own servers and computing resources on on-premises datacentres. cloud providers easily solve this challenge giving us a variety of options in terms of additional locations and new technologies and tools availability on the fly.

The most important challenge that we are facing now is the lack of skills in this domain. We are lacking professionals who can deal with data efficiently and can create scalable solutions to handle the data. LinkedIn listed data engineers as one of the professions which is expected to grow more and has seen almost 35% growth in 2021 (Wilkins, 2021). In recent years we have seen that even mid-tier and small companies have even realized the importance of having their own data team and are doing the analytics inhouse.

4.3 Data Engineering and Google Cloud

Data engineering on Google cloud has been evolving and Google is also launching tools and trainings to help professionals train on GCP. Data engineering on GCP is one of the popular specializations that Google is offering. They have a whole separate track for data (Wijaya, 2022) where they cover different specializations like Data Analyst, Data Engineer, and Database Engineer.

Google Cloud is also pushing hard for the data engineering solutions as due to the boom of AI and machine learning we have seen that cloud providers are growing in the recent years, and this trend continues in coming years too as evident in the image below Figure 4.

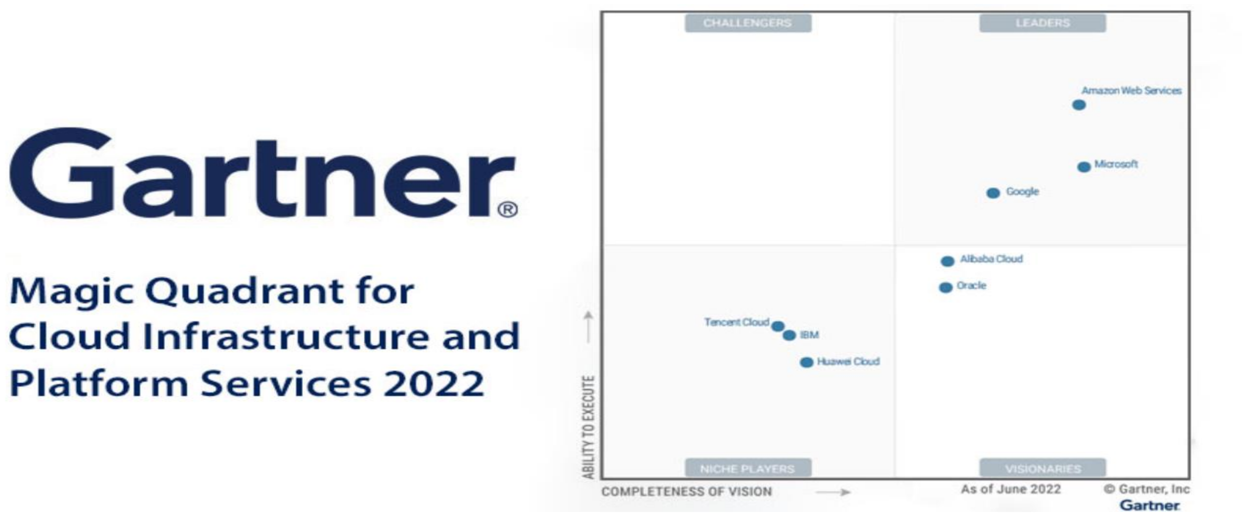


FIGURE 4 Gartner GCP Magic Quadrant

GCP is a suite of cloud computing services offered by Google. It provides a wide range of tools and services for data engineering, including data storage, data processing, and data analysis. Google Cloud offers different kinds of services for data engineering, which includes BigQuery, Cloud Storage, Cloud SQL and Looker. We'll be focusing on a variety of the services related to data that will be used in this case study.

Google Cloud Services is also pushing hard for the data engineering solutions as due to the boom of machine learning we have seen that cloud providers are growing in the recent years, and this trend in coming years too as evident in the image below too. Data and GCP go long way because Google has been adopting the techniques of artificial intelligence and machine learning quite a lot and this is the future also.

4.4 Tools and Technologies

In this sub-chapter the tools that we have used for different purposes will be discussed in the implementation of this case study from Google cloud storage for the data ingestion and storage and Google big theory for data warehousing solutions along with MageAI for data pipeline orchestration and dbt for the data transformation and data modeling as shown in Table 1 below, Along with their alternative options available which could be from other cloud providers but from the open source technologies.

TABLE 1. Tools Used along with alternatives.

Tool	**Purpose**	**Alternative**
Google Cloud Storage	Data Ingestion and Storage	Amazon S3, Microsoft Azure Blob Stor-age, IBM Cloud Object Storage
Google BigQuery	Data Warehousing and Data Modelling	Amazon Redshift, Microsoft Azure Syn-apse Analytics, Snowflake
MageAI	Data Pipelines Orchestration	Apache Airflow, Zapier, Microsoft Power Automate (formerly Microsoft Flow)
dbt	Data Transformation and Data Modelling	Informatica PowerCenter, Talend, Mi-crosoft SQL Server Integration Services (SSIS)
Looker	Data Visualization and Dash-board Creation	Tableau, Power BI, QlikView
Terraform	Building Infrastructure as Code	AWS CloudFormation, Azure Resource Manager (ARM), Ansible
Git	Version Control System	SVN (Subversion), Mercurial, Perforce
Docker	Containerization	Kubernetes, rkt, Open Container Initiative (OCI)
Python	Programming Language	Java, C++, R
Google Compute	Virtual Machine and Compu-ting	Amazon EC2, Microsoft Azure VM’s, IBM Virtual Server

These tools basically gave us the insight about how to implement the IT data engineering process a workflow in the cloud environment along with the purpose that it serves and with the alternative options if you do not have the skills needed to operate the tool or if you are interested in other options available on the market that also depends on the skill level and the technology stack that the company is using but is planning to use in future. Waka along with docker and Python previously implementing the containerization that will enable us to deploy our code and do the continuous integration and continuous deployment and also scale out application easily.

5 METHODOLOGY

The methodology used in this thesis is a systematic method that is to be used to research the use of Google Cloud Platform tools and services for data engineering. Thus, the methodology is aimed at addressing the investigated research questions in a holistic manner in terms of collecting, storing, processing, and analysing data. It comprises both quantitative and qualitative research methods such as case study analysis, action research, and empirical study. The research aims to investigate and report on practical research findings related to how users employed GCP and its applicability as a data engineering framework.

5.1 Case Study

The primary research strategy implemented in this thesis is a case study that involves a detailed analysis of the use of GCP tools and services in a particular case for data engineering. It can be noted that such an approach is aimed at a comprehensive investigation of the data engineering process on GCP from the beginning to the end stages. For example, it includes data collection, preprocessing, analysis, and visualization. By using representative research, the thesis aims to explain how GCP solutions can be used to solve practical problems in data engineering. Simultaneously, the case study helps to integrate the features of action research when a researcher addresses stakeholders to identify, solve, and evaluate practical problems or problems in specific areas of activity, in this case, the use of GCP. This methodology promotes research that can be categorized as a case study because it tracks GCP usage and evaluates the researcher’s experience. Moreover, action research is a participatory approach to research in which a researcher attempts to find a solution for a significant problem. The action research method focuses on a specific case that must be properly investigated and studied after the GCP usage.

It is also worth mentioning the practical, hands-on attitude of action research as one of the reasons why it was chosen as a particular methodology as a way of exploring the use of Google Cloud Platform. Action research enables an in-depth investigation of the stakeholders. As a result, it is possible to gather information about the stakeholders’ needs, attitudes, and real-life challenges accompanied by the use of GCP. Additionally, by including the participants in the research process, it is possible to ensure that the findings can be used in practice. Moreover, this strategy also promotes a sense of concernment and investment among stakeholders, contributing to the effective use and application of the findings in GCP implementations. As a result, the framework of action research built in this approach is flexible and explorative. Since it is iterative and based on constant feedback and adjustment, it is a valuable approach to the exploration of practical and developing phenomena where practical findings are almost as instrumental as the theoretical ones.

5.2 Action Research Methodology

‘Research strategies that tackle real-world problems in participatory, collaborative, and cyclical ways in order to produce both knowledge and action.’ (O’Leary, 2007). Action research is a cyclical process of identifying problems or issues and next step is collecting data to related to that and developing solutions to fix those problems or issues. Lastly implementing those solutions in a production environment and then doing analysis on that and at last listings the learnings. It differs from traditional research methods in that it focuses on practical problem-solving rather than purely theoretical analysis. Action research allows researchers to work closely with stakeholders to identify their needs and develop solutions that are tailored specifically for their organization's requirements (Davison, 2021).

The aim of action research is not only to find a solution, but also to improve the understanding about the processes involved in finding one. This approach allows for iterative development cycles where you can learn from each stage of your work before moving on to the next stage. In our case study we are going through an action research cycle where firstly identifying problem statement then collect data and analyse it and then implement the solution as shown in Figure 5 below onto the production environment using open-source tools such as mage AI docker, dbt and using the Google cloud platform.

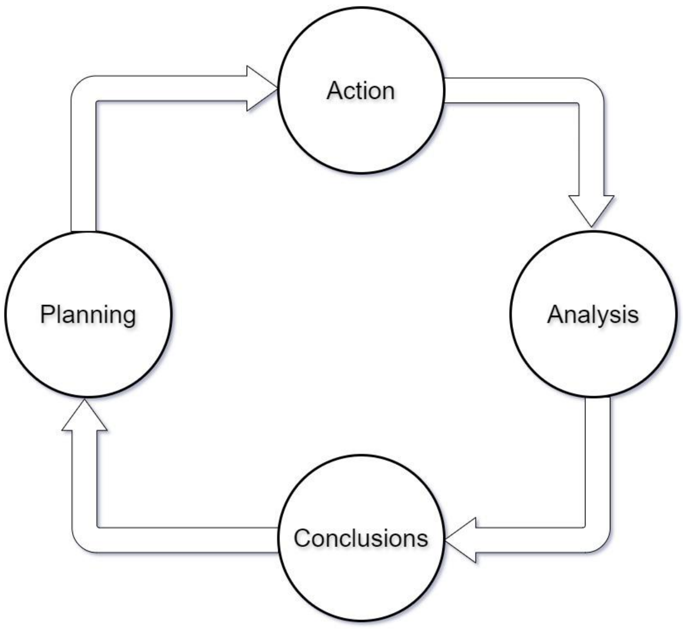


FIGURE 5 Action Research

This approach allows for iterative development cycles where you can learn from each stage of your work before moving on to the next stage. In our case study we are going through an action research cycle where firstly identifying problem statement then collect data and analyse it then implement the solution. It differs from traditional research methods in that it focuses on practical problem-solving rather than purely theoretical analysis. Action research allows researchers to work closely with stakeholders to identify their needs and develop solutions that are tailored specifically for their organization's requirements (Davison, 2021).

5.3 Research Design

In this section the research design will be discussed and how we'll carry out this whole research part. As this is an action study our focus is on implementation and so the research is practical. By looking into a complete flow of data and integration of tools this thesis aims to bridge the gap between theoretical knowledge and practical application, preparing individuals for the demands of the evolving DE field. This thesis proposes a comprehensive case study, aiming to construct an end-to-end solution using Google Cloud Platform (GCP) and other tools.

It is data engineering simple workflow looks like in real world scenarios. For this purpose, we'll collect the raw data from multiple sources into cloud storage via the process called Extract, Transform, Load

(ETL) and then apply the transformations on that raw data to convert it into meaningful information into the data warehouse and in last build a business dashboard containing different key performance indicator (KPI). For orchestration we will be using Mage AI and for transformations we will be using dbt library and for infrastructure management we will use terraform which is a popular tool for infra as code along with Docker. Business users are more interested in the data in the context of their business domain, so dashboards are quite useful for them when they want to analyse the historic trends or analyse the performance.

It helps to integrate the features of action research when a researcher addresses stakeholders to identify, solve, and evaluate practical problems or problems in specific areas of activity, in this case, the use of GCP. This methodology promotes research that can be categorized as a case study because it tracks GCP usage and evaluates the researcher's experience. Below in Figure 6 are the steps shown how it will be proceeded. Moreover, action research is a participatory approach to research in which a researcher attempts to find a solution for a significant problem. The action research method focuses on a specific case that must be properly investigated and studied after the GCP usage.

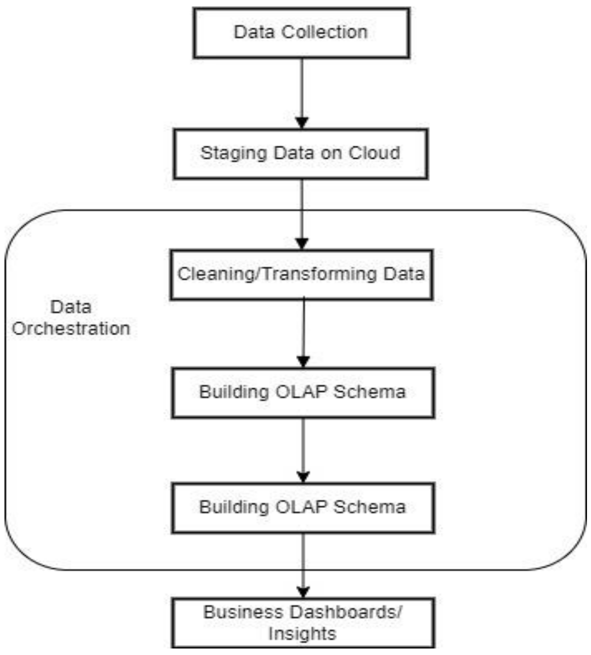


FIGURE 6 Research Design

For this purpose, the raw data from multiple sources will be collected into cloud storage via the process called Extract, Transform, Load and then apply the transformations on that raw data to convert it into meaningful information into the data lakes and in last build a business dashboard containing different key performance indicator. For orchestration we will be using mage AI and for transformations we will be using dbt library and for infrastructure management we will use terraform which is a popular tool for infra as code along with docker. Business users are more interested in the data in the context of their business domain, so dashboards are quite useful for them when they want to analyse the historic trends.

5.3.1 Research Objective

There are several steps in the research objective from data collection to findings. In data collection the researcher collects relevant information or other necessary sources related to the topic. In Data preprocessing data is cleaned, transformed, and prepared research data or easily accessible information. In analysis, pre-processed data is analysed, and researcher interprets to show meaning. Outcomes of the analysis are evaluated by the researcher and researchers' implications are revealed. Finally in findings stage information or result after the analysis is shared and evaluation of the research.

5.4 Data Collection

For data collection an open-source data set will be used, that is available for public use. The data contains information related to the electric vehicles collected by the state of Washington in the USA (government, 2024). It contains 29 columns of different data types, and a variety of information is present in it from electric vehicle type to its model year to make to its range to its price and also information related to the government fees which have been paid. Data with the help of an API is loaded provided by the state of Washington, and Mage AI tool, its data loading module will be used to pull data from the open data platform into the Google cloud storage. Partition the data on the basis of years so that we have a separate folder for each year it will help us in managing the data and also, improves the data storage and query performance. All Columns and their description are shown in Table 2 below which will give us the information about the data set.

TABLE 2. Dataset description

Column Name	**Data Type**	**Description**
electric_vehicle_type	string	Type of electric vehicle

vin_1_10	string	Vehicle Identification Number (first 10 characters)
dol_vehicle_id	string	Vehicle ID assigned by the Department of Licensing
model_year	int64	Year of the vehicle model
make	string	Vehicle make (e.g. Honda, Toyota etc.)
model	string	Model of Vehicle (e.g. Accord, Civic etc.)
vehicle_primary_use	string	Primary use of the vehicle (e.g. personal, commercial, etc.)
electric_range	int64	Electric range of the vehicle (in miles)
odometer_reading	int64	Odometer reading at the time of sale
odometer_code	string	Odometer code (e.g. actual, exempt, etc.)
new_or_used_vehicle	string	Vehicle is new or used flag
sale_price	int64	Price at which the vehicle was sold
date_of_vehicle_sale	string	Date of vehicle sale
base_msrp	int64	Base Manufacturer's Suggested Retail Price
transaction_type	string	Type of transaction (e.g. sale, lease, etc.)
transaction_year	int64	Year of the transaction
county	string	County where the vehicle was sold
city	string	City where the vehicle was sold
state_of_residence	string	Buyer state of residence
zip	string	ZIP code of the buyer's residence
meets_2019_hb_2042_sale_price_value_requirement	bool	vehicle meets the HB 2042 sale price value requirement or not flag
2019_hb_2042_sale_price_value_requirement	string	Description of the HB 2042 sale price value requirement
electric_vehicle_fee_paid	string	Whether the electric vehicle fee was paid
transportation_electrification_fee_paid	string	Whether the transportation electrification fee was paid

hybrid_vehicle_electrification_fee_paid	string	Whether the hybrid vehicle electrification fee was paid
census_tract_2020	string	Census tract of the buyer's residence (2020)
legislative_district	string	Legislative district of the buyer's residence
electric_utility	string	Electric utility company serving the buyer's residence
transaction_date	string	Date of the transaction

The data set consists of multiple columns containing different kinds of information ranging from demographic information about the vehicle about the government fees along with the history of purchase and about the price of vehicle.

6 IMPLEMENTATION

In this chapter discussion related to the implementation and techniques used in deployment of tools on Google cloud services is discussed. Implementation requires Google account which is necessary to perform all the cloud related tasks.

6.1 Architecture

The architecture of the project is designed to extract, transform, and load data from an open-source data source related to electric cars. MageAI is used to create two pipelines, the first pipeline extracts data from the data source and stages it on Google Cloud Storage. To optimize data management, data is partitioned based on the registration year, which enables efficient data retrieval and analysis. This partitioning approach also allows for scalability and flexibility in data processing. Data is staged in parquet format because it is efficient data compression format due to which we get good query performance and reduced space requirements also. The main architecture diagram in Figure 7 shows all the components how they interact and what is the flow of data on the whole application As the size of data is reduced, it gives us the edge in improving the speeds and it improves our data management.

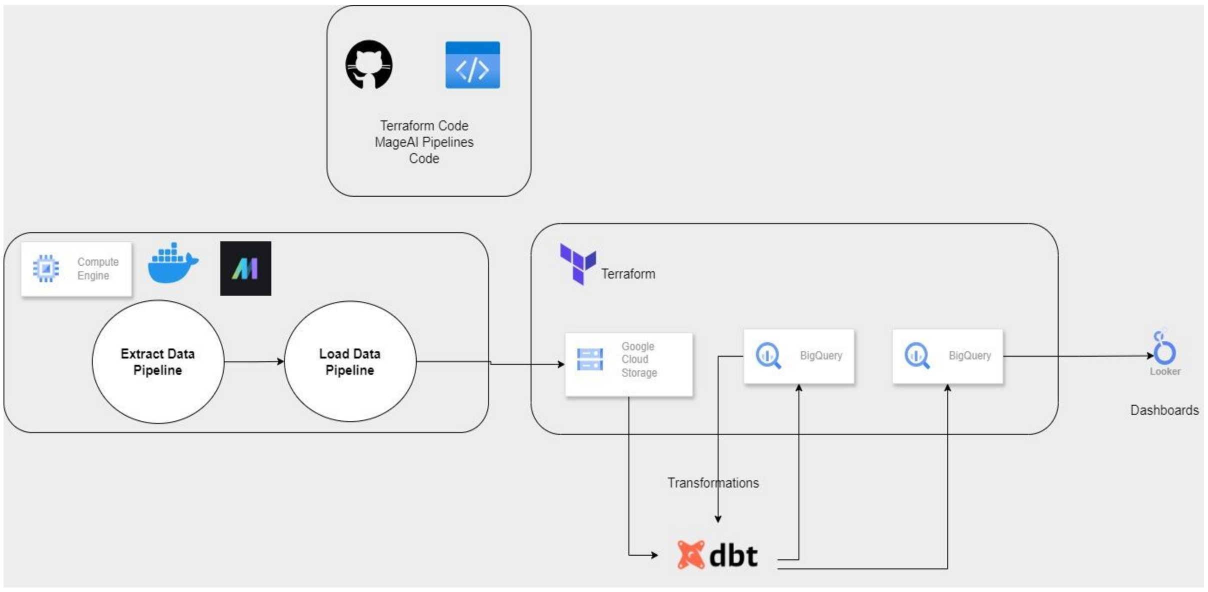


FIGURE 7 Architecture Diagram

The second data pipeline is using dbt for the data transformations. As we have data staged in Google cloud storage, the next step is to load this into our data warehouse that is in big query. First, we pick the data from GCS as that is in the OLTP Schema and we must convert it into OLAP Schema so that we can use it for analytical purposes. We will split this into facts and dimension tables with fact containing the records that tell us about the history and dimensions are used to filter the data or analyse the data from different perspectives. We pick data then write the dbt models which generate the respective tables in Google Big Query respective to those models and fills those with the data. Lastly, we are going to build the dashboards on that so that we can share the insights with the business users, and they can see the data in a meaningful way. With the help of filters, they can filter the visuals from different perspectives and visuals convey the information about the history and trends which can add value in decision making process.

For managing the infrastructure easily Terraform is used which is an Infrastructure as Code (IaC) tool to build Google Cloud Storage and Google BigQuery resources. Google Compute resources build a VM on which Docker is used to pull the MageAI image and saved all code and credential files. This approach provides flexibility, scalability, and ease of access to the project resources. For code management GitHub was used as code repository, where Terraform, Docker, and mage pipeline code is hosted. This enabled to track changes and maintain a version-controlled codebase. Overall, the architecture of this project is designed to efficiently extract, transform, and load data, and to provide a scalable and flexible infrastructure for data analysis and reporting.

6.2 Setting up Google Cloud Account

First create a Google Account which will be used for this project for that go to the Google Account sign-up page and follow the prompts to create a new Google account. Fill in the asked required information, like name, email and password then verify the account by verifying the contact information that was provided to get access to the new account.

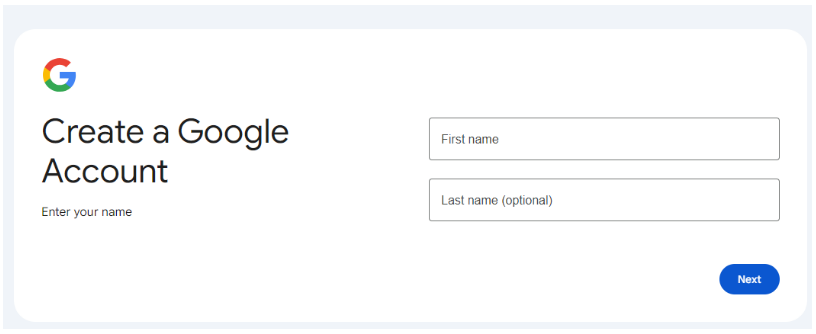


FIGURE 8 GCP Signup Page

6.2.1 Create a New GCP Project

After signing in open Google Cloud Console to access all the options. Click on the 'Select a project' dropdown menu at top and then select 'New-Project' from there. Enter a project name, company and location where these resources will be deployed. Click on New Project on top right corner as shown in Figure 9.

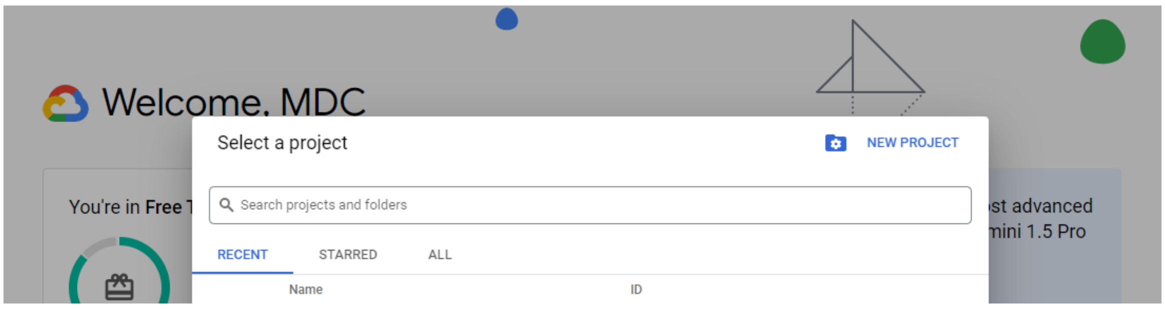


FIGURE 9 Google new Project

6.2.2 Enable the Free Trial

Open Google Cloud Console and on the 'Navigation menu' select 'Billing' or search for 'Billing' on the search bar to open the billings menu. Click on 'Free Trial' and follow the prompts to enable the free trial. You will need to provide a credit card or other payment method to verify your account, but you won't be charged until the trial period ends.

6.2.3 Set Up Your Billing Account

After signing into Google Cloud Console click on the open ‘billing’ from the navigation bar or by searching it into the search bar on top. Create a billing account by selecting ‘Create a billing account’ option. Fill in all required information, like name, billing address, and payment method for charge. Click on Create to create the account and start the computing.

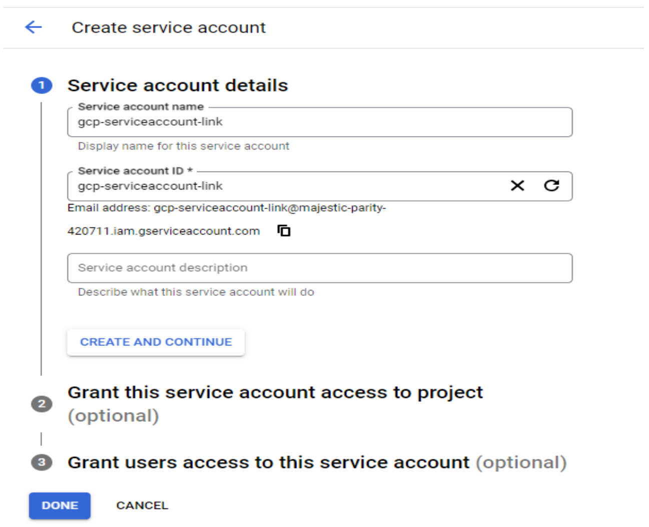
6.3 Service Account Creation

We use service accounts authenticate and authorize connections to GCP resources. Which provides a secure way to delegate access to specific resources and APIs, allowing for fine-grained control over permissions and access. This method helps us to reduce the risk of credential exposure and improves the overall security of cloud applications.

For this project create a GCP service account with the following roles ‘Big query Admin, Storage Admin, Object Storage Admin’. Once you have signed in or created an account, navigate to the Compute Engine section by clicking on ‘ENABLE APIS AND SERVICES’ from top navigation bar then ‘APIS and SERVICES’ from left navigation menu and searching for ‘Compute Engine API’. Enable this service for all regions that will be used by clicking on ‘ENABLE’.

6.3.1 Create Service Account

For service account creation so that GCP can be connected to other tools like MageAI login into Google Cloud Console and open ‘IAM and Admin’ by searching it on top search bar or from the left menu. Select Service accounts to access more options. Click on Create service account. Fill in all information in the form as shown in Figure 10 below including the service account name and email address. Click on ‘Create’.


Create service account

Service account details

Service account name: gcp-serviceaccount-link

Display name for this service account

Service account ID *: gcp-serviceaccount-link

Email address: gcp-serviceaccount-link@majestic-parity-420711.iam.gserviceaccount.com

Service account description

Describe what this service account will do

CREATE AND CONTINUE
Grant this service account access to project (optional)
Grant users access to this service account (optional)
DONE
CANCEL

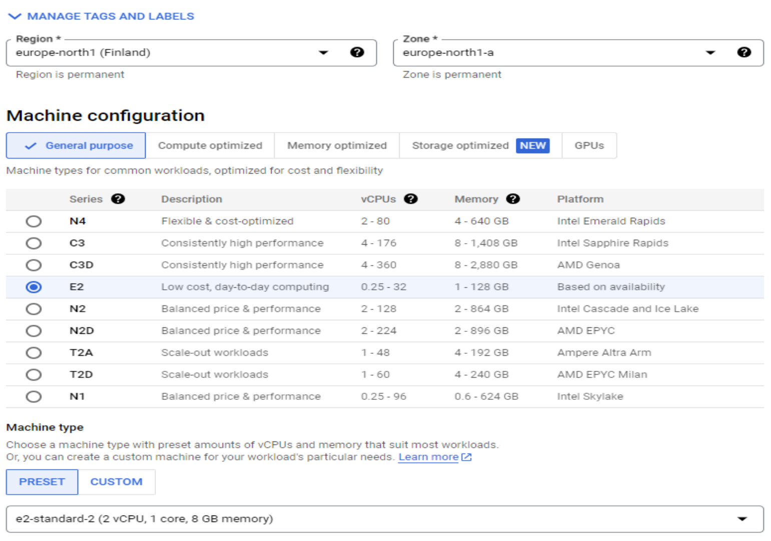
FIGURE 10 Service Account Creation

6.3.2 Generating Service Account Key

After creation of service account find the service account you created (Appendix 1) and click on the three vertical dots at the end of the row. Click on 'ADD SSH KEY' (Appendix 2). Select 'JSON' as the key type and click on 'Create'. Download the JSON key file in a secure location because it grants access to your GCP resources.

6.4 Configuring Compute on GCP

To configure the compute navigate to the Compute Engine Dashboard on GCP portal. Open Compute Engine dashboard searching for ‘Compute Engine’ from the search menu. On Compute Engine dashboard, click on ‘VM instances’ from the left menu. Select ‘Create VM instance’ button at the top of the page. Select the ‘Region’ close to your location and the ‘Zone’ in that region. Select the ‘e2-standard-2 (2 vCPU, 1 core, 8 GB memory)’ machine type, which has 2 vCPUs and 8 GB of RAM. As shown in figure 11 here.


MANAGE TAGS AND LABELS

Region *: europe-north1 (Finland)
Region is permanent

Zone *: europe-north1-a
Zone is permanent

Machine configuration

General purpose | Compute optimized | Memory optimized | Storage optimized NEW | GPUs

Machine types for common workloads, optimized for cost and flexibility

	Series	Description	vCPUs	Memory	Platform
○	N4	Flexible & cost-optimized	2 - 80	4 - 640 GB	Intel Emerald Rapids
○	C3	Consistently high performance	4 - 176	8 - 1,408 GB	Intel Sapphire Rapids
○	C3D	Consistently high performance	4 - 360	8 - 2,880 GB	AMD Genoa
●	E2	Low cost, day-to-day computing	0.25 - 32	1 - 128 GB	Based on availability
○	N2	Balanced price & performance	2 - 128	2 - 864 GB	Intel Cascade and Ice Lake
○	N2D	Balanced price & performance	2 - 224	2 - 896 GB	AMD EPYC
○	T2A	Scale-out workloads	1 - 48	4 - 192 GB	Ampere Altra Arm
○	T2D	Scale-out workloads	1 - 60	4 - 240 GB	AMD EPYC Milan
○	N1	Balanced price & performance	0.25 - 96	0.6 - 624 GB	Intel Skylake

Machine type

Choose a machine type with preset amounts of vCPUs and memory that suit most workloads.
Or, you can create a custom machine for your workload's particular needs. Learn more

PRESET | CUSTOM

e2-standard-2 (2 vCPU, 1 core, 8 GB memory)

FIGURE 11 VM Configuration

Configure the Boot Disk and OS. Click on ‘Change’ next to ‘Boot disk’ and select ‘Create a new boot disk’. Choose ‘Ubuntu’ as the operating system and select the ‘ubuntu-2004-lts’ image. Set the ‘Size’ to 300 GB. In the ‘Operating system’ option, select ‘Ubuntu’ as the operating system from the drop-down menu (Appendix 3). Create the VM. Click on the ‘Create’ button to create the VM. Wait for the VM to be created. Wait for the VM to be created (Appendix 4). This may take a few minutes. Once the VM is created, you can connect to it using SSH. Click on the ‘SSH’ button next to the VM machine on the Compute dashboard.

6.5 Installing Docker and MageAI

Connect to GCP Machine via SSH on VS Code. Open VS Code and install the ‘Remote - SSH’ extension if you haven't already. Click on the ‘Explorer’ icon in the left sidebar and then click on the ‘New Folder’ button. In the ‘Folder’ input field, enter the path to your GCP machine's SSH configuration file (e.g., ~/.ssh/config). Click on the ‘Open’ button to open the SSH configuration file. Update the SSH configuration file to include the IP address and credentials of your GCP machine.

```
Host gcp-machine
HostName <GCP_MACHINE_IP_ADDRESS>
User <GCP_MACHINE_USERNAME>
IdentityFile ~/.ssh/<PRIVATE_KEY_FILE>
```

Save the SSH configuration file (Appendix 5). On bottom of VS Code select the remote connection and then select ‘SSH’ as the connection type and enter the hostname gcp-machine (Thesis_VM in our case). Click on the ‘Connect’ button to establish an SSH connection to your GCP machine.

6.5.1 Installing Docker on the GCP Machine

Open a new terminal in VS Code by clicking the ‘Terminal’ icon in the top menu bar or pressing Ctrl + Shift (Windows/Linux). Run the following command to update the package list. Install Docker on the new virtual machine via command line by this command. ‘sudo apt install docker.io’ and then start the Docker service.

6.5.2 Install Docker Extension on VS Code

Open the Extensions panel in VS Code by clicking the ‘Extensions’ icon in the left sidebar. Search for ‘Docker’ in the VS Code Extensions marketplace. Click on the ‘Docker’ extension by Microsoft and click the ‘Install’ button. Wait for the extension to install and reload VS Code (Appendix 6).

6.5.3 Deploying MageAI Docker Container

Open a new terminal in VS Code by clicking on the ‘Terminal’ icon. Create the docker file as shown in the Appendix 19 below in the same directory as the project directory. Run the docker compose file as shown in Appendix 20 to start the MageAI Docker container and start that on port 6789. Make sure that docker file and docker-compose file are in same directory or change the path in docker-compose file in case of different paths.

Run this command ‘docker run -p 6789:6789 mageai/mageai’ to start a new MageAI Docker container. Forward the port 6789 from VM to local from VS Code. Open a web browser and navigate to http://localhost:6789/overview to access the MageAI web interface. You should also see the MageAI container if you open up the docker extension in VS Code

6.6 Building Data Pipelines

In this section Data pipeline-built techniques are discussed and how user can follow the steps and build the pipelines on their own systems, or they can observe the whole process of managing the data pipelines which are used to pull data and then stage it on GCP cloud storage and transform it into OLAP schema for data warehouse.

6.6.1 Pipeline 1: Stage Data into Google Cloud Storage

Define the source of your open-source electric vehicle data. This will be the API to fetch data from. For Data Cleaning Clean and pre-process the data as needed. Here we have to focus on standardizing formats, and validating data types as shown in Figure 12 below. For Year extraction extract the registration year from the data and create a new column. For parquet conversion, convert the data into Parquet format, which is efficient for storage and querying. We will stage the data in parquet format because it is an efficient data compression format due to which we get good query performance and reduced space requirements also. As the size is reduced so it gives us the edge in improving the speeds and it improves our data management.

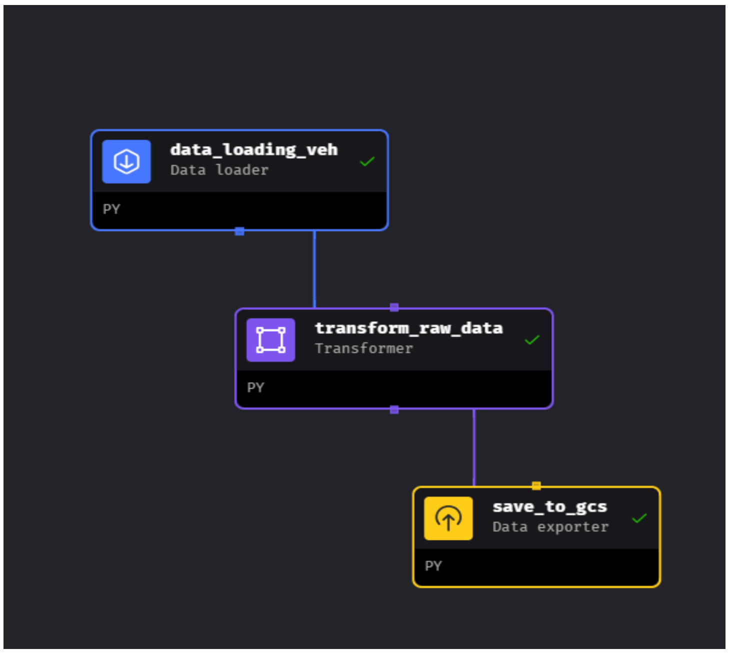


FIGURE 12 Pipeline 1

For destination we will be staging data into GCS Bucket, and we have to specify the Google Cloud Storage bucket where we want to store the data. Partition the data by registration year. This will improve query performance when accessing data for specific years. To optimize data management, Data is partitioned based on the registration year, which enables efficient data retrieval and analysis. This partitioning approach also allows for scalability and flexibility in data processing

6.6.2 Pipeline 2: Load Data into BigQuery

Define the source as the GCS bucket containing the partitioned Parquet files. Storing data in parquet format because it is an efficient data compression format due to which we get good query performance and reduced space requirements also. As the size is reduced so it gives us the edge in improving the speeds and it improves our data management. For schema validation ensure the data schema matches the BigQuery table schema as shown in Figure 13. For data Filtering we have to initiate the optionally filter the data based on specific criteria.

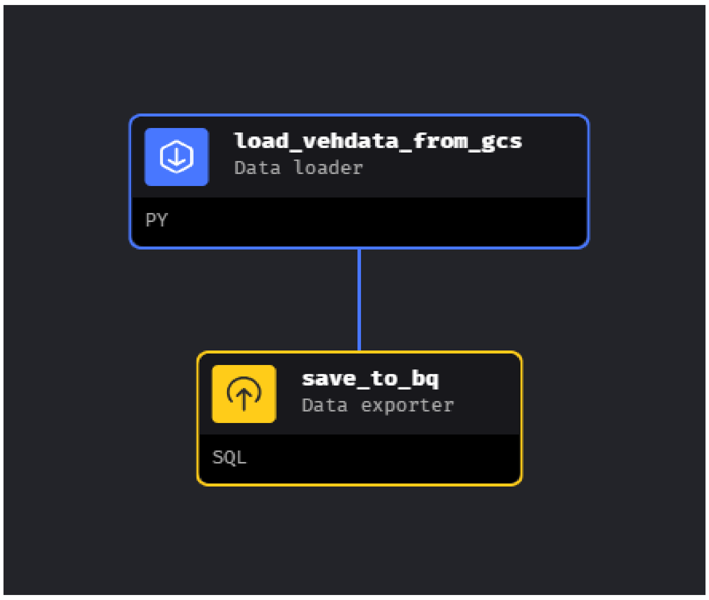
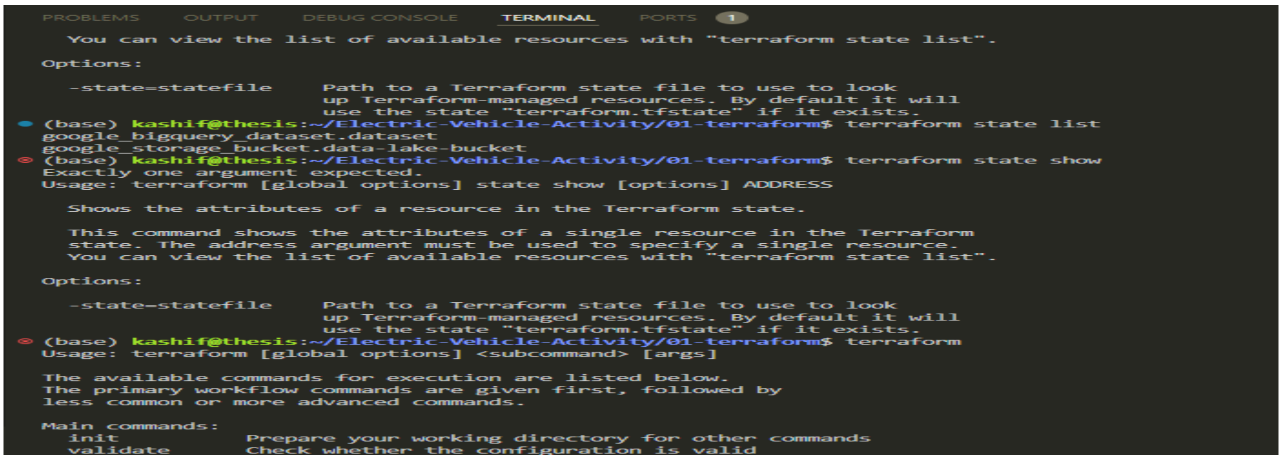


FIGURE 13 Pipeline 2

Data loading to BigQuery table where data is loaded. Structured data is kind of organized data and is often stored in tables, databases and in form of rows and columns. While unstructured data has no pre-defined format and comes in multiple format and shapes like photos, videos, documents. Data velocity is related to fact that data is being generated at impressive pace like never before.

6.7 Installing Terraform for IaC

For staging the infrastructure on the cloud install Terraform on your VM. Open a terminal on your VM and run this command 'sudo apt-get install terraform -y' to install Terraform using sudo. Create a separate directory for Terraform project and navigate into it. Copy your keys.json file from your local machine to your VM using a secure method such as scp or sftp. Configure Terraform to use your service account keys. Create a new file called varibales.tf (Appendix 16) in your project directory. Then initialize Terraform in the main directory and lastly configure your Terraform scripts. Upon execution of your Terraform script it will create the BigQuery dataset and Google Cloud Storage bucket as shown in Figure 14 specified in your main.tf file (Appendix 17).



```
  You can view the list of available resources with "terraform state list".

Options:

  -state=statefile    Path to a Terraform state file to use to look
                      up Terraform-managed resources. By default it will
                      use the state "terraform.tfstate" if it exists.
(base) kashif@thesis:~/Electric-Vehicle-Activity/01-terraform$ terraform state list
google_bigquery_dataset.dataset
google_storage_bucket.data-lake-bucket
(base) kashif@thesis:~/Electric-Vehicle-Activity/01-terraform$ terraform state show
Exactly one argument expected.
Usage: terraform [global options] state show [options] ADDRESS

  Shows the attributes of a resource in the Terraform state.

  This command shows the attributes of a single resource in the Terraform
  state. The address argument must be used to specify a single resource.
  You can view the list of available resources with "terraform state list".

Options:

  -state=statefile    Path to a Terraform state file to use to look
                      up Terraform-managed resources. By default it will
                      use the state "terraform.tfstate" if it exists.
(base) kashif@thesis:~/Electric-Vehicle-Activity/01-terraform$ terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
```

FIGURE 14 Terraform State File

6.8 dbt Cloud for transformations

Create a dbt Cloud account. Go to the dbt Cloud website and sign up for an account. Fill out the registration form with your email address, password, and other details. Verify your email address by opening the verification link sent by dbt Cloud. Create a new project in dbt Cloud. Log in to your dbt Cloud account and click 'New Project' from there. Enter a name of project and select 'Google BigQuery' as the target database. Click on the 'Create Project' button.

Link Google BigQuery with dbt Cloud. Click on the 'Connect to BigQuery' button in your dbt Cloud project. Enter your BigQuery project ID and credentials (using the keys.json file). Click on the 'Connect' button to establish a connection between dbt Cloud and BigQuery. Configure the schema.yml file (Appendix 18). In your dbt Cloud project, click on the 'Schema' tab. Click on the 'Edit' button next to the 'schema.yml' file. Update the schema.yml file to include the following configurations:

Develop models for fact and dimension tables. Create a new file called models in your dbt Cloud project. Create separate files for each model, e.g. ‘dim_location_info.sql, dim_vehicle_info.sql, fact_vehicle_data.sql, and stg_electric_vehicles.sql’ (APPENDIX 10-13). Write the SQL code for each model using dbt's syntax. Click on the ‘Run’ button in your dbt Cloud project to execute the models. dbt Cloud will create the fact and dimension tables in your BigQuery data warehouse.

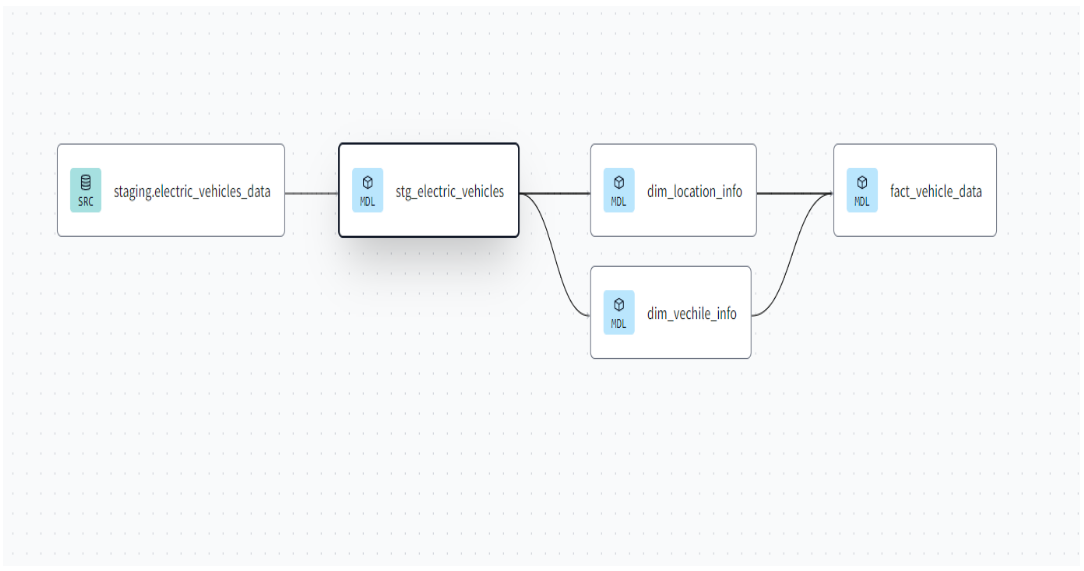


FIGURE 15 dbt Lineage Graph

Running the models will create the tables and big query database as shown in Figure 15 if the execution is successful and tables will be filled up with new data and the fact and dimensions that were created in the previous step. Then this data will be used and the tables to build a dashboard so that users are enabled to take businesses decisions from this data.

6.9 Business Dashboards

Business dashboards are quite critical in today's data-driven world. The enhanced decision making that they provide for the business users by which they can track the progress and measure this success of their KPIs that they are tracking allows them to quickly identify the trends patterns and potential issues (Mario Nadj, 2020). They also give you the competitive advantage of other businesses by utilizing their data and businesses are just saving the data into their databases but by analyzing that data we can quickly identify the new opportunities and make informed decisions quite quickly and also, we are not just saving the data we have making use of that another powerful way. Create a new dashboard on Looker and click on the 'New Dashboard' button. Give the dashboard a name, such as 'Electric Vehicle Registrations'.

6.9.1 Dashboard 1

Viewers can understand the overall growth of electric vehicle adoption, compare the performance of different companies, and identify trends in registration activity over time by using this dashboard show in Figure 16 below.

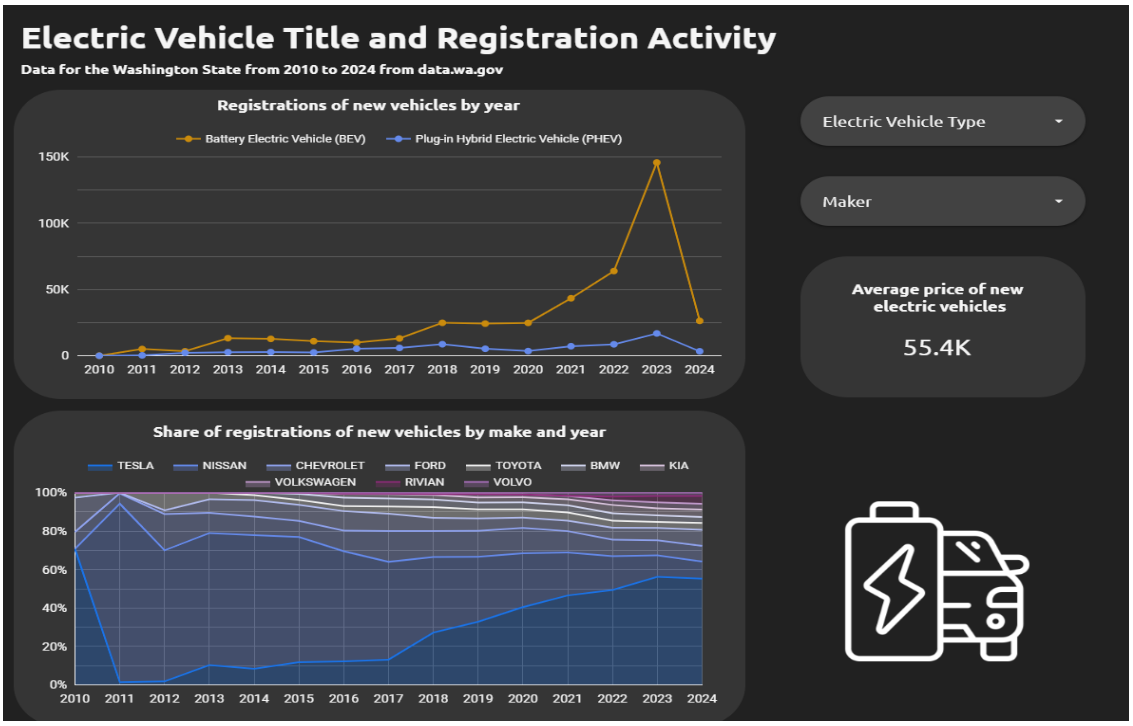


FIGURE 16 Dashboard 1

Line Chart shows registrations by year, which helps in identifying the trend of electric vehicle registrations over time, highlighting growth or decline in adoption. Use the 'year' dimension on the x-axis and the 'registrations' measure on the y-axis.

Area Chart shows company comparison over time which adds information in identifying the market share of different electric vehicle companies over the years, revealing which companies are leading or gaining traction. Use the 'year' dimension on the x-axis, the 'company' dimension for colour segmentation, and the 'registrations' measure for the area.

6.9.2 Dashboard 2

Viewers can explore the popularity of specific electric vehicle models, identify regional trends in adoption, and understand the geographic distribution of electric vehicles across the USA by using this dashboard as shown in Figure 17.

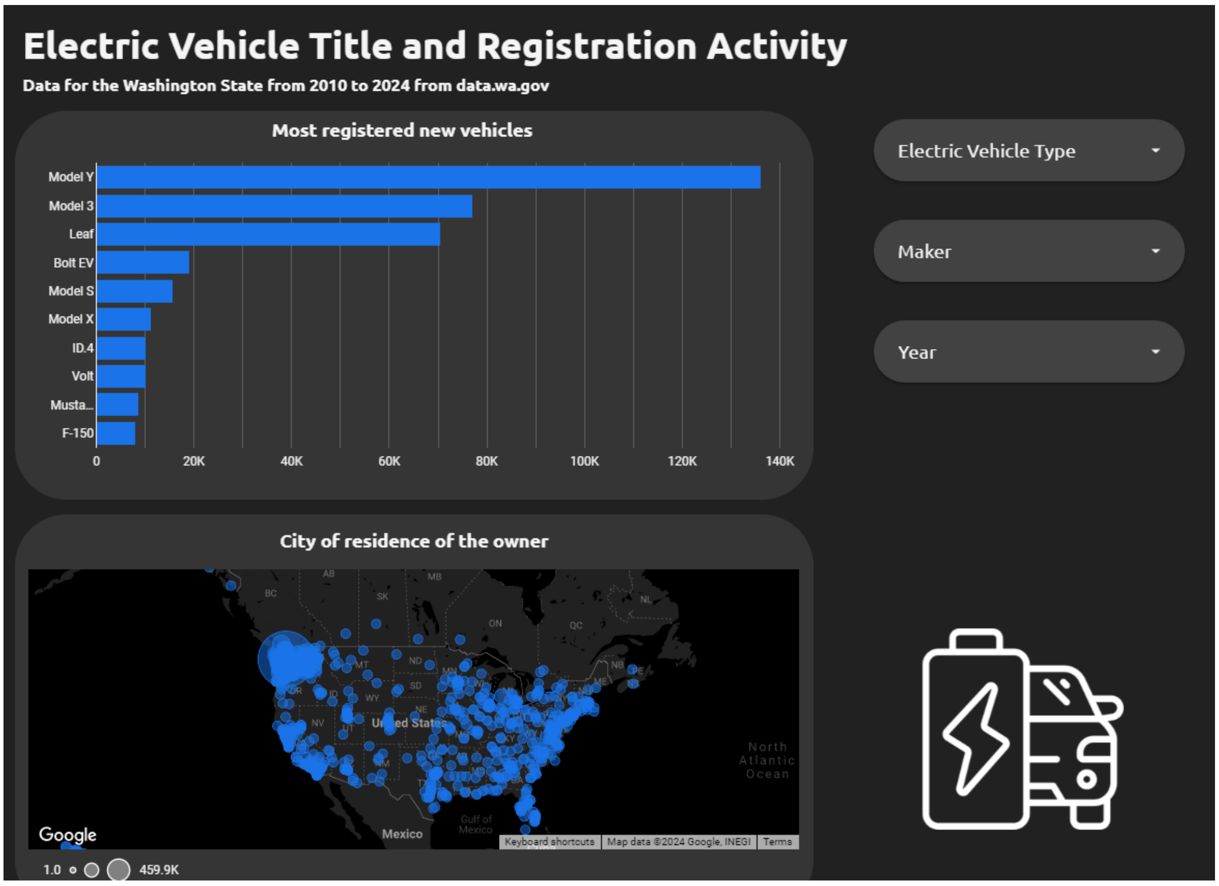


FIGURE 17 Dashboard 2

Column Chart shows model registrations by year where popularity of different electric vehicle models over time, identifying trends and shifts in consumer preferences can be seen. Use the 'year' dimension on the x-axis, the 'model' dimension for the columns, and the 'registrations' measure for the height of the bars.

Map visual for state-wise registrations in the USA helps in visualizing the geographic distribution of electric vehicle registrations across the United States, highlighting states with higher adoption rates. Use the ‘state’ dimension for location and the ‘registrations’ measure for size of markers.

Include filters for electric vehicle type, maker, and year to allow users to drill down into specific segments of the data. Drill-down capabilities allow users to click on chart elements to filter data based on their selections. Data labels and tooltips provides clear labels and informative tooltips to enhance understanding of charts. Dashboard layout and design organizes the visuals and filters in a clear way for easy navigation and interpretation. Interactivity is incorporating interactive elements like dropdown slicers and click to filter option for dynamic exploration.

7 RESULTS

In results chapter focus is on the results that were generated after the implementation of case study on Google cloud platform. This case study demonstrates the real-world use case which helps the readers to get an understanding of tools and technologies that are available in google cloud and also some of the open source technologies like docker, MageAI for data handling and data transformations.

7.1 Results

The project demonstrates a real-world implementation of a data engineering project showcasing the entire data lifecycle from data ingestion to insights generation. The project's objective was to collect, process, and analyse raw data related to electric vehicles to extract the actionable insights from that.

Processes that have been implemented in this case study are related to the data ingestion where raw data related to electric vehicles was collected and stored in Google Cloud Storage. Data Warehousing where raw data was then loaded into a data warehouse built on Google BigQuery, a fully managed enterprise data warehouse. Data Transformation is done using dbt (Data Build Tool), the data was transformed into facts and dimensions, creating a structured and organized data model. Finally, dashboards were built on Looker, a cloud-based business intelligence platform, to provide interactive and actionable insights to stakeholders.

This project showcases the entire data engineering workflow, from data ingestion to insights generation, highlighting the importance of each component in the data engineering workflow. The project demonstrates how data can be collected, processed, and analysed to provide valuable insights and how different tools connect and how they complement each other.

7.2 Thesis Evaluation

This study was the action research focused on the implementation of basic workflow that we follow in data engineering and how the objectives that were defined in the beginning are achieved. The data was gathered from an opensource data source which is staged and analyzed using the orchestration tool and from the validity standpoint the outcomes of this thesis align with the objectives of upskilling and walking through the whole process that involves the basic components and technologies used in data engineering related to Google Cloud and open-source technologies.

The flow of thesis is also aligned with research design that we had proposed first was the collection of data for that we have used the open-source data. Then stage it on the raw layer for further processing in Google Cloud and then build the data warehouse by building an OLAP schema but the analytical processing and reporting and that last enables the users to make use of that data. Business dashboards which give them the ability to try and see the information that is available in raw form coming from different business processes and present that in a meaningful way to help them in taking the business decisions.

This case study also demonstrated the importance of page components and tools and where we can use them and how that tool adds value to the whole process and what are the alternate options available in the market. We have also used the testing techniques in dbt, we have developed the test cases which tracks the data quality and the other conditions that we have configured over there before the processing of these data and if those cases are successful only then it is going to upload the data into the data warehouse.

This study also answers the key questions surrounding the development and implementation of end-to-end data engineering solutions using GCP. This showed how different GCP services and tools along with open-source technologies, can be effectively integrated to build a robust and scalable data processing solutions.

Thesis also serves as a guide for collecting, storing, processing, and analyzing data within the GCP ecosystem. It showcased the use of Google Cloud Storage for data ingestion and storage, Google BigQuery for data warehousing and modelling, and MageAI for orchestrating data pipelines. The integration of programming languages like Python and SQL was demonstrated using dbt for data transformation and modelling. Finally, the project highlighted the importance of data visualization and dashboard creation using tools like Looker for effective communication of insights. SQL is a de facto language when it comes to data querying and transformation within BigQuery. The seamless integration of these technologies and flexibility and power of GCP make it an ideal solution.

The project's real-world application was clearly demonstrated through the implementation of a complete data engineering solution. The tools and technologies combined with the best practices here in the study, offer a broiler plate for organizations looking to try GCP for their data processing needs.

The findings highlight the potential of GCP-centric solutions to improve data-driven decision-making, enhance efficiency, and unlock valuable insights from data.

7.3 Comparative Study

For the comparative study we analysed the same solution in multiple different environments and calculated the different KPIS used to track the data. The results of these analyses show significant differences in the performance when compared with other clouds and on prem solution. The on prem solution had the same specs as the other solutions deployed on cloud. Results of that experiment is shown in the table 3 below.

TABLE 3. Cloud Providers Comparison

Category	**Data Ingestion Time (sec)**	**Data Transformation Time (sec)**	**Query Response Time (sec)**	**Data Storage Cost ($/month)**
GCP	120	60	2	50
AWS	150	80	3	70
Azure	180	90	4	80
On-Premises	300	120	6	150

For data ingestion time GCP stood out to be the fastest as compared to the other cloud providers and on-prem took a lot of time due to the network as network on cloud machines due to the bandwidth networks and optimized services as compared to on-prem solution which really affected the performance here. For data transformation actually we are focusing on the aspect that how much time it takes to execute the data pipelines and produce the results. For data transformation we are transforming the data and checking for different conditions first and data is going through the number of tests like null value tests, checksum verification to ensure that data is same and has not been updated from source to target, checking for anomalies like out of range values we configure the range or expected values and the algo checks the values against these values first before executing the data pipeline.

Query response time measures the time it takes to retrieve data from the cloud platform or from on-prem system in response to a query. This is also a benchmark used to evaluate the performance of systems and lowest the query response the better is the performance. GCP also gets an edge here because we are using the BigQuery database here that is a very powerful database that can handle a lot of data. It is fast and very good at handling massive amounts of data.

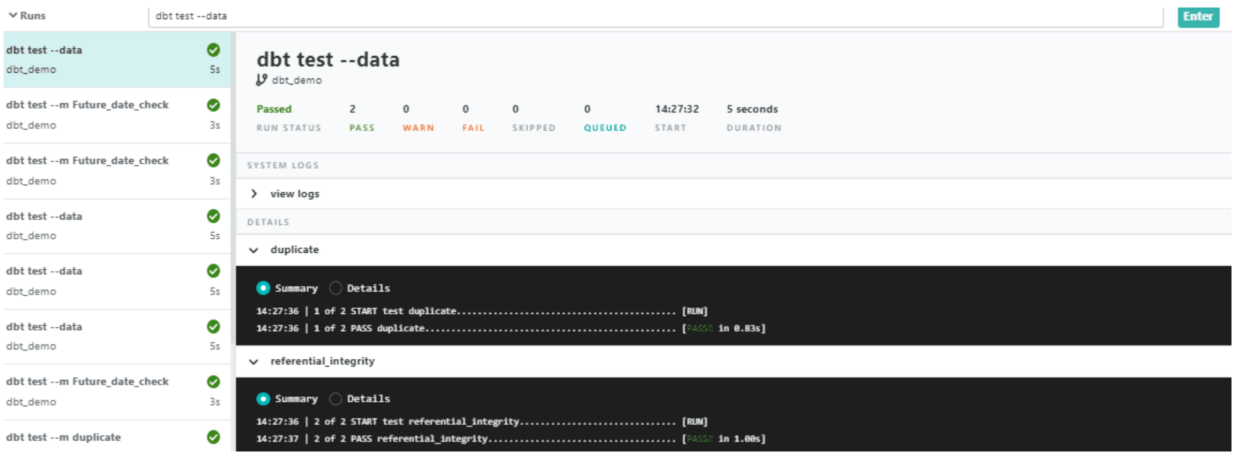
Google BigQuery achieves this optimized performance by using parallel processing. Data storage cost measures the cost of storing data in the cloud or on-prem systems and more the data higher the cost of storage is. Cost also depends on multiple conditions too from region to type of storage to time commitment made for the storage, but we found that Google is the cheapest option here too as it cost only 50 Euro monthly to store 30 GB of data. Overall, we can see the gcp stands out in our case when we had the data to deploy the ETL and store the data for processing the fast and economical option is the gcp. It also depends on region from where you are accessing the cloud and for on prem as you must take care of everything from infrastructure related that cost in most cases goes up.

7.4 Testing

Testing in the data engineering is really important to ensure that we are having a good quality and correct data as business dashboards are using the data and business decisions depend on that so to ensure correctness testing of data plays an important role here. Different kinds of testing techniques are discussed here that were used to ensure the data validation and quality.

7.4.1 Data validation testing

First and foremost, check in data validation is to check for any duplicates because we have to ensure that we are counting the data only once and it is not affecting the numbers and totals that we are displaying on the dashboards which are also used for taking the business decisions. We have used the dbt built in testing environments to test the data validation import the library and add the function and it will check the primary key or any surrogate key column and ensure that all data loaded is unique and is not being repeated.


Runs
dbt test --data
Enter
dbt test --data dbt_demo 5s
dbt test --m Future_date_check dbt_demo 3s
dbt test --m Future_date_check dbt_demo 3s
dbt test --data dbt_demo 5s
dbt test --data dbt_demo 5s
dbt test --data dbt_demo 5s
dbt test --m Future_date_check dbt_demo 3s
dbt test --m duplicate
dbt test --data
dbt_demo
SYSTEM LOGS
view logs
DETAILS
duplicate
referential_integrity
Summary
Details

Passed	2	0	0	0	0	14:27:32	5 seconds
RUN STATUS	PASS	WARN	FAIL	SKIPPED	QUEUED	START	DURATION

```
14:27:36 | 1 of 2 START test duplicate........................................ [RUN]
14:27:36 | 1 of 2 PASS duplicate............................................. [PASS in 0.83s]
```

```
14:27:36 | 2 of 2 START test referential_integrity............................ [RUN]
14:27:37 | 2 of 2 PASS referential_integrity.................................. [PASS in 1.00s]
```

FIGURE 18 dbt Tests

As shown above in the Figure 18 above tests related to the uniqueness of the records have been executed and it shows us that they have passed the tests. In case we experience the problem then the dbt model loading would have failed and triggered the workflow like sending an email to the watchers to inform them that this model loading needs to be fixed.

7.4.2 Data Quality

For data quality issues we must ensure the scale and units issue also so that they are consistent across the data lifecycle and are not being updated or converted without any knowledge at any stage if they are they should be kept in a separate column or in separate data frame. Schema consistency for a data set is quite crucial as we must plug in the schema for various data pipelines and if the schema names are different then it tends to break the pipeline so on the first step, we make sure that all column names are consistent and are following the same format as expected. Along with the column names, data types also should be consistent which gives us the edge in data validation along with the data export without any issues.

Identifying the relationships between variables helps us in identifying the data quality issues. Outliers and anomalies which are affecting the relationships and missing, and inconsistent data are the common issues we have to deal with when we are dealing with such data. As shown in image in Figure 19 below the correlation matrix is shown which helps us in identifying the key relationships between different variables and getting a deeper understanding of the data.

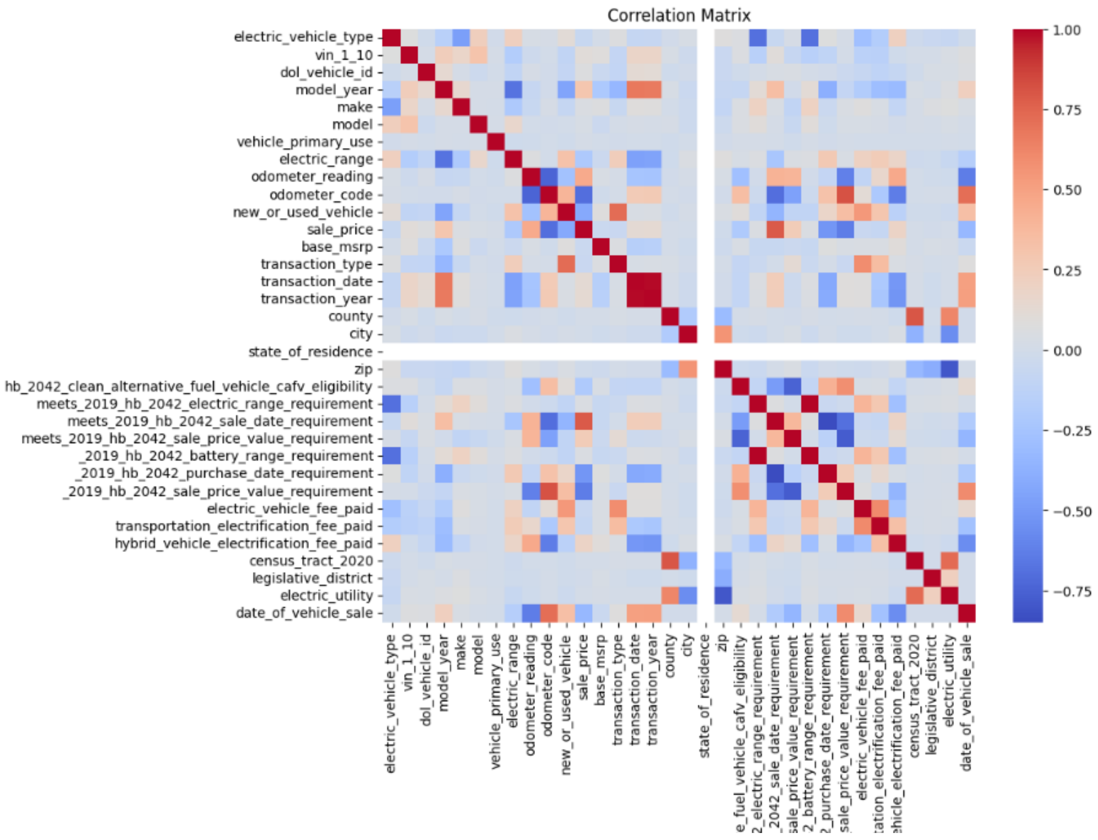


FIGURE 19 Correlation Matrix

For the data consistency check we can see in the dashboard where we have a column chart, and it displays the data on year-to-year basis we can see that there are no outliers or any anomalies as for difference between two years is not that big and no spikes were shown so we can say that data is consistent and according to the standards.

8 CONCLUSIONS

This case study implementation serves as a guide for providing hands-on experience with various data engineering tools and technologies both with GCP and open source. It also gives the user a high-level view about data engineering workflow and how different components are pulled together and what is the role of each technology.

Through this project, a complete end to end flow of data engineering tools and technologies from Google cloud and open-source system were used and discussed. Along with that the reasons why these tools exist and what problems they solve in real world scenarios and how they can be integrated with other systems is discussed.

For data ingestion and storage on Google Cloud Storage is used and for data warehousing and data modelling on Google BigQuery is used. Data Pipelines are crucial when it comes to loading data, transforming, and then staging it on data warehouse so for data pipelines orchestration MageAI was utilized. dbt is utilized for data transformation and data modelling. Business dashboards for the users were created using Looker. For managing the Infrastructure as Code Terraform is used.

As a thesis project, this implementation has achieved its objectives, demonstrating a real-world application of data engineering principles and tools. The project also serves as a guide for new users who wants to get into the world of data engineering.

8.1 Findings and Insights

Our case study implementation had several important findings. By combining cloud-based services with open-source tools, it is possible to build a robust and scalable data solution which is capable of handling large amount of data. The study demonstrated how data can be efficiently ingested from multiple sources then transformed into a usable format and loaded into a data warehouse for analysis.

Implementation of data engineering workflow requires careful consideration of data ingestion, data storage, data warehousing, data modelling, pipelines orchestration, transformation, and visualization.

The choice of data storage solution has a major impact on the performance and scalability, while data warehousing and data modelling organize data for robust querying and analysis.

Data pipelines orchestration automates data processing, and data transformation and data modelling prepares the data for analysis and dashboards. Lastly, data visualization and dashboard are an effective tool to communicate insights to stakeholders. Project demonstrates a real-world application of data engineering concepts and tools which makes it a good resource for those looking to enter the field and want to see everything in action and how these services interact with each other.

8.2 Further Considerations

Further consideration regarding this case study is to incorporate more features in this regarding data security and privacy, Because we deal with potentially sensitive user data it is necessary that we implement standard security measures and follow the data privacy regulations like masking the personal info and not saving the data onto outside locations. For scalability and automation, we can scale our data pipeline to handle the growing data volumes and automate that for minimum human intervention.
This study mainly focused on managing the batch processing, but more advanced real time data processing can be discussed in future studies. Lastly for advanced analytics we can explore how adding machine learning and AI techniques can predict future trends and personalizing user experiences.

9 REFERENCES

Brikman, Y., 2022. Terraform: Up and Running. s.l.:O'Reilly.

Carruthers, A., 2022. Building the Snowflake Data Cloud: Monetizing and Democratizing Your Data. s.l.:Springer.

Croft, R., Babar, M. A. & Kholoosi, M. M., 2023. Data Quality for Software Vulnerability Datasets. IEEE.

Davison, R. M. M. a. M. J., 2021. Research perspectives: Improving action research by integrating methods. Journal of the Association of Information Systems.

Garani, G. a. C. A. a. S. I. a. B. M., 2019. A Data Warehouse Approach for Business Intelligence. IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE).

Gonzalez Zelaya, C. V., 2019. Towards Explaining the Effects of Data Preprocessing on Machine Learning. IEEE 35th International Conference on Data Engineering (ICDE).

Gray, J. a. S. P., 2000. Proceedings of 16th International Conference on Data Engineering (Cat. No.00CB37073). IEEE.

Washington state government, 2024. Data Source. [Online]
Available at: https://data.wa.gov/Transportation/Electric-Vehicle-Title-and-Registration-Activity/rpr4-cgyd/data_preview
[Accessed 10 Mar 2024].

Handy, T., 2017. What, exactly, is dbt?. [Online]
Available at: https://www.getdbt.com/blog/what-exactly-is-dbt

Hashicorp, 2024. What is Terraform?. [Online]
Available at: https://developer.hashicorp.com/terraform/intro

IBM, 2024. IBM AI Adoption Index 2023. [Online]
Available at: https://www.multivu.com/players/English/9240059-ibm-2023-global-ai-adoption-index-report/
[Accessed 22 Mar 2024].

Jon Loeliger, M. M., n.d. In: O'Reilly, ed. Version Control with Git. s.l.:s.n.

Karl Matthias, S. P. K., 2015. Docker: Up & Running: Shipping Reliable Containers in Production. s.l.:s.n.

Mage, 2024. Welcome to Mage. [Online]
Available at: https://docs.mage.ai/introduction/overview
[Accessed 25 Mar 2024].

Mario Nadj, A. M. C. S., 2020. The effect of interactive analytical dashboard features on situation awareness and task performance. Decision Support Systems.

Market Research Future, 2022. Data Analytics Market Worth To Be USD 346.24 Billion at a CAGR of 30.7% by 2030 - Report by Market Research Future (MRFR). [Online]
Available at: https://www.globenewswire.com/en/news-release/2022/10/06/2529492/0/en/Data-Analytics-Market-Worth-To-Be-USD-346-24-Billion-at-a-CAGR-of-30-7-by-2030-Report-by-Market-Research-Future-MRFR.html
[Accessed 11 Apr 2024].

McKinney, W., n.d. Python for Data Analysis. In: Python for Data Analysis. s.l.:O'Reilly, p. 2022.

Nada Elgendy, A. E., 2016. Big Data Analytics in Support of the Decision Making Process. Procedia Computer Science.

Naik, N., 2017. Docker container-based big data processing system in multiple clouds for everyone. IEEE International Systems Engineering Symposium (ISSE).

Nambiar, A. & Mundra, D., 2022. An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management. Big Data and Cognitive Computing.

O’Leary, R. &. B. L. B., 2007. Conclusion: Conflict and Collaboration in Networks. International Public Management Journal.

Parliament, E., 2020. Is data the new oil? Competition issues in the digital economy, s.l.: s.n.

Parth Goel, D. G., 2017. The Internet of Things: A Main Source of Big Data Analytics. Computer Engineering and Intelligent Systems.

Rockoff, L., 2022. The Language of SQL. In: Pearson, ed. s.l.:s.n.

Wijaya, A., 2022. Data Engineering with Google Cloud Platform. In: s.l.:Packt.

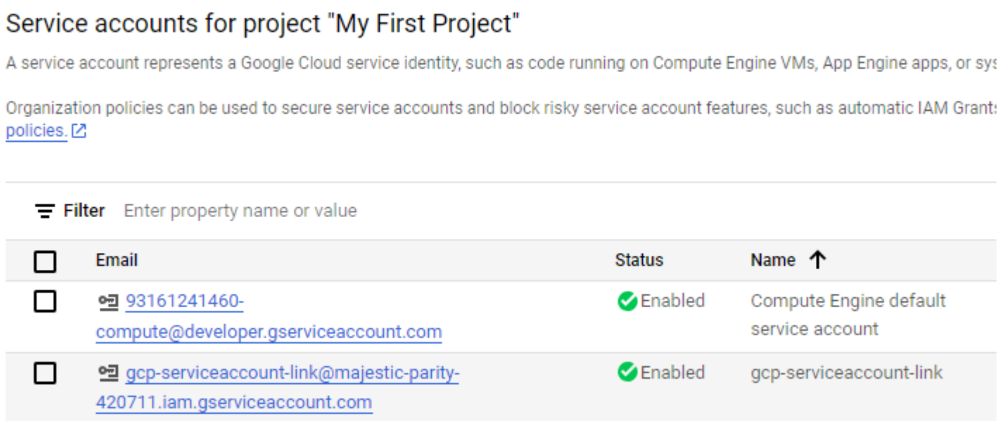
Wilkins, D., 2021. An In-Depth Analysis of the Data Analytics Job Market.

Williams, L. a. C. A., 2023. Agile software development: It's about feedback and change. IEEE.
APPENDIX 1

Service Account in Google cloud account

APPENDIX 1

Service Accounts in your GCP accounts


Service accounts for project "My First Project"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or sys

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grant: policies.

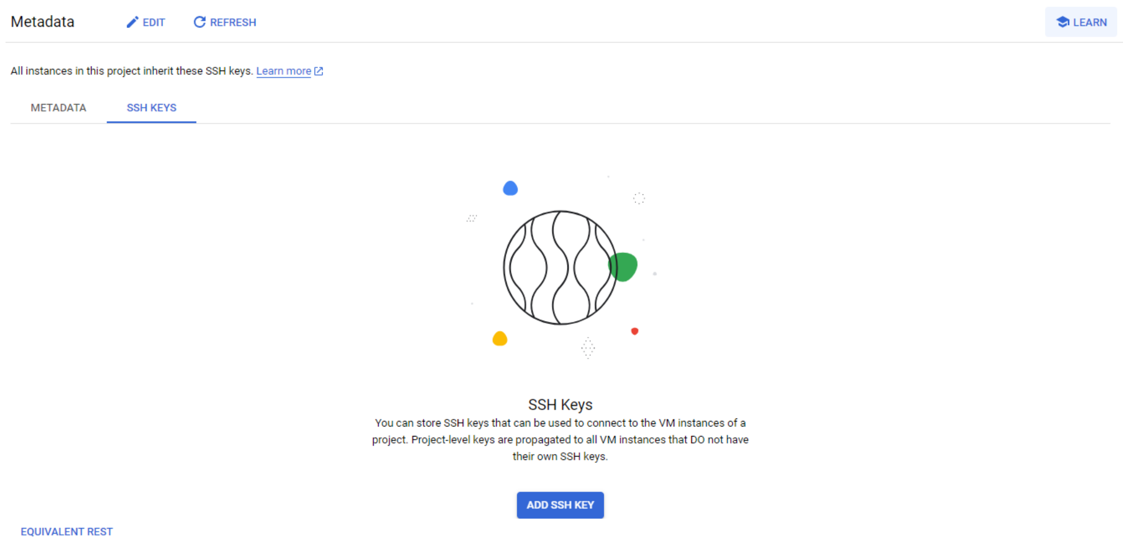
Filter Enter property name or value

☐	Email	Status	Name ↑
☐	93161241460-compute@developer.gserviceaccount.com	✓ Enabled	Compute Engine default service account
☐	gcp-serviceaccount-link@majestic-parity-420711.iam.gserviceaccount.com	✓ Enabled	gcp-serviceaccount-link

FIGURE 20 Service Account List

APPENDIX 2

Configuring the SSh keys for accessing the GCP resources.



All instances in this project inherit these SSH keys. Learn more

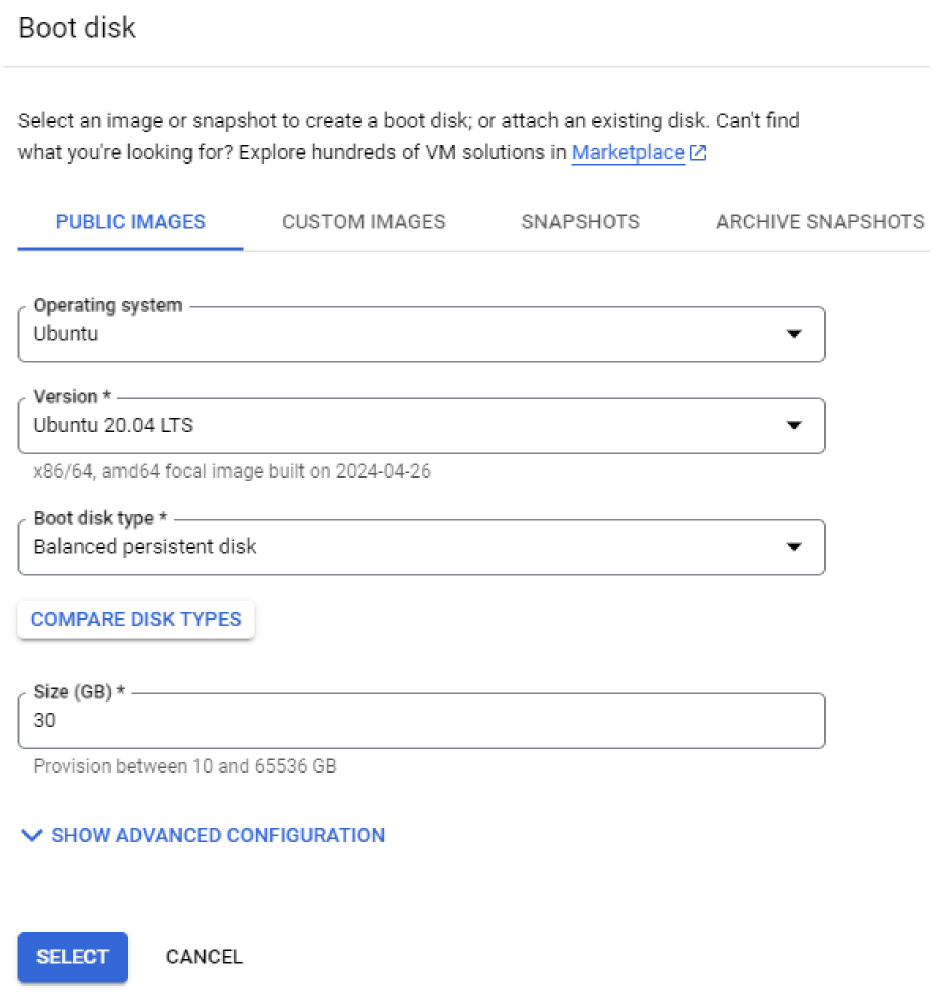
SSH Keys

You can store SSH keys that can be used to connect to the VM instances of a project. Project-level keys are propagated to all VM instances that DO not have their own SSH keys.

FIGURE 21 Adding SSH Key

APPENDIX 3

Building the Virtual Machine on GCP.


Boot disk

Select an image or snapshot to create a boot disk; or attach an existing disk. Can't find what you're looking for? Explore hundreds of VM solutions in Marketplace

Operating system: Ubuntu

Version *: Ubuntu 20.04 LTS

x86/64, amd64 focal image built on 2024-04-26

Boot disk type *: Balanced persistent disk

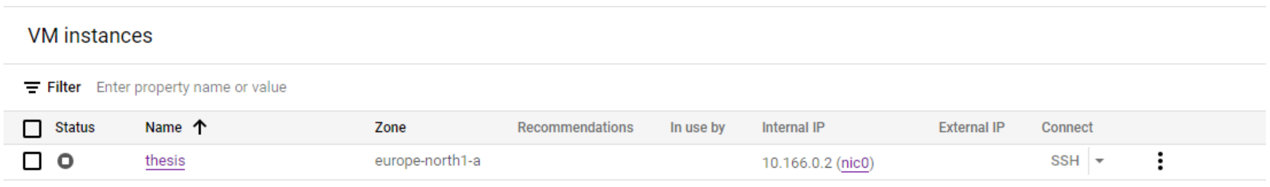
Size (GB) *: 30

Provision between 10 and 65536 GB

FIGURE 22 Configure VM Storage

APPENDIX 4

Virtual Machines available on GCP.



VM instances

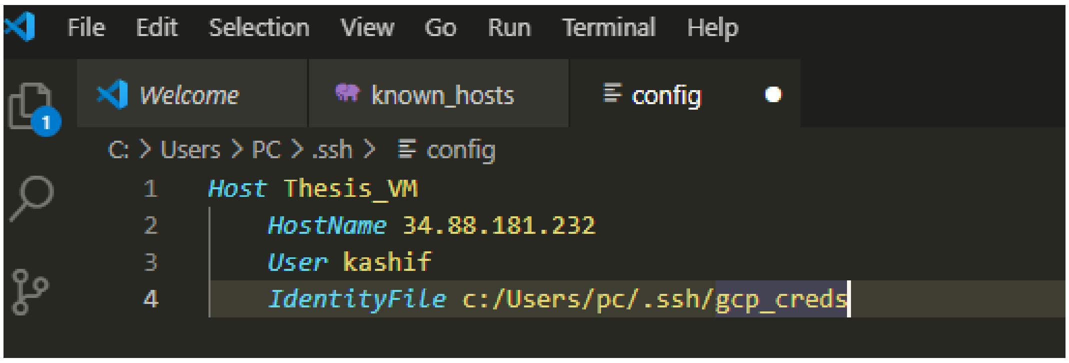
Filter Enter property name or value

Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
☐	thesis	europe-north1-a			10.166.0.2 (nic0)		SSH

FIGURE 23 VM Instance

APPENDIX 5

Configuring the SSH client in VS Code to access VM from local system.



```
Host Thesis_VM
    HostName 34.88.181.232
    User kashif
    IdentityFile c:/Users/pc/.ssh/gcp_creds
```

FIGURE 24 VM SSH Config file

APPENDIX 6

MageAI Interface.

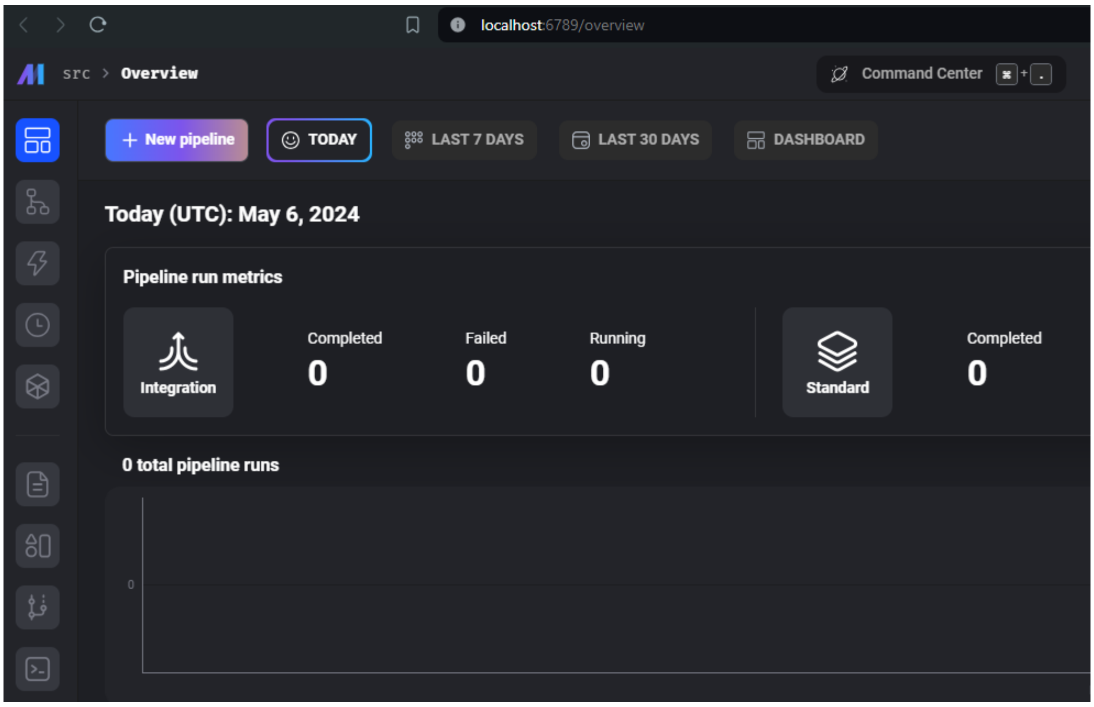


FIGURE 25 Mage Interface

APPENDIX 7

Docker Containers on VM.

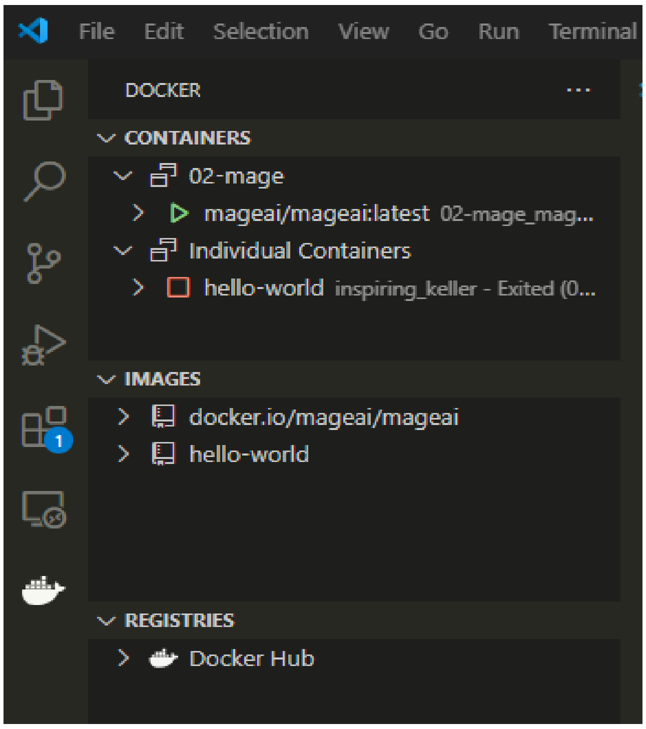
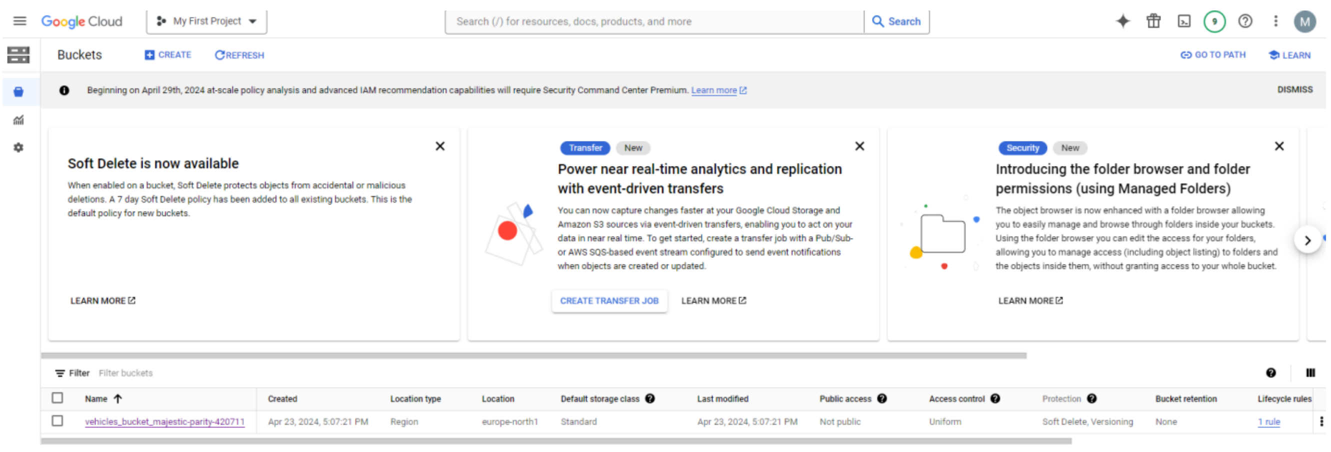


FIGURE 26 MageAI Container on Docker

APPENDIX 8

Google Cloud Storage Bucket created with the help of Terraform.



Beginning on April 29th, 2024 at-scale policy analysis and advanced IAM recommendation capabilities will require Security Command Center Premium. Learn more

Soft Delete is now available

When enabled on a bucket, Soft Delete protects objects from accidental or malicious deletions. A 7 day Soft Delete policy has been added to all existing buckets. This is the default policy for new buckets.

LEARN MORE

Transfer New

Power near real-time analytics and replication with event-driven transfers

You can now capture changes faster at your Google Cloud Storage and Amazon S3 sources via event-driven transfers, enabling you to act on your data in near real time. To get started, create a transfer job with a Pub/Sub- or AWS SQS-based event stream configured to send event notifications when objects are created or updated.

CREATE TRANSFER JOB LEARN MORE

Security New

Introducing the folder browser and folder permissions (using Managed Folders)

The object browser is now enhanced with a folder browser allowing you to easily manage and browse through folders inside your buckets. Using the folder browser you can edit the access for your folders, allowing you to manage access (including object listing) to folders and the objects inside them, without granting access to your whole bucket.

LEARN MORE

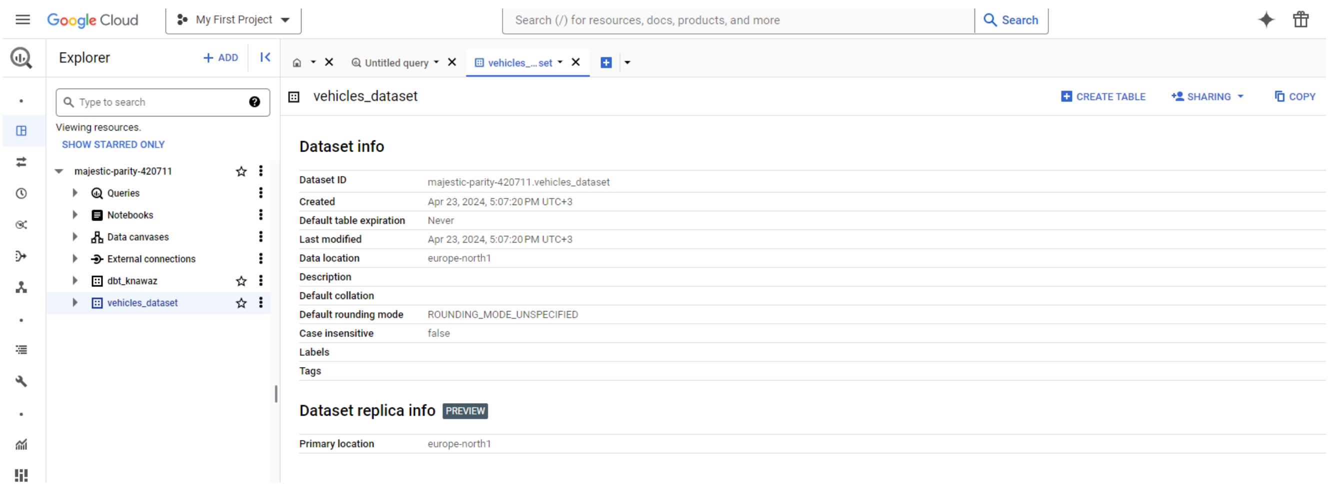
Filter Filter buckets

Name ↑	Created	Location type	Location	Default storage class	Last modified	Public access	Access control	Protection	Bucket retention	Lifecycle rules
vehicles_bucket_majestic-parity-420711	Apr 23, 2024, 5:07:21 PM	Region	europe-north1	Standard	Apr 23, 2024, 5:07:21 PM	Not public	Uniform	Soft Delete, Versioning	None	1 rule

FIGURE 27 Google cloud bucket

APPENDIX 9

Google BigQuery Dataset created via using Terraform scripts.



vehicles_dataset

Dataset info

Dataset ID	majestic-parity-420711.vehicles_dataset
Created	Apr 23, 2024, 5:07:20 PM UTC+3
Default table expiration	Never
Last modified	Apr 23, 2024, 5:07:20 PM UTC+3
Data location	europe-north1
Description	
Default collation	
Default rounding mode	ROUNDING_MODE_UNSPECIFIED
Case insensitive	false
Labels	
Tags	

Dataset replica info PREVIEW

Primary location	europe-north1

FIGURE 28 Google Big Query dataset

APPENDIX 10

Dim_location_info model in dbt used to build the table in bigquery in the form of dimension.

```
{{ config(materialized=‘table’) }}

with unique_locations as (
        select
            location_id,
            county,
            city,
            state_of_residence,
            zip,
            legislative_district
        from {{ ref(‘stg_electric_vehicles’) }}
        GROUP BY
            location_id,
            county,
            city,
            state_of_residence,
            zip,
            legislative_district
    )

select * from unique_locations
```

APPENDIX 11

dim_vehicle_info.sql model in dbt used to build the table in bigquery in the form of dimension.

```
{{ config(materialized=‘table’) }}

with
    unique_vehicles as (
        select
            vin,
            make,
            model,
            model_year,
            electric_vehicle_type,
            electric_range,
            vehicle_primary_use,
            base_msrp,
        from {{ ref(‘stg_electric_vehicles’) }}
        group by
            vin,
            make,
            model,
            model_year,
            electric_vehicle_type,
            electric_range,
            vehicle_primary_use,
            base_msrp
    )

select
*
from unique_vehicles
```

APPENDIX 12

fact_vehicle_data.sql used to build the table in bigquery in the form of dimension.

```
{{ config(materialized='table') }}

with transaction_data as (
    select
        transaction_id,
        sale_price,
        transaction_date,
        transaction_year,
        sale_date,
        transaction_type,
        new_or_used_vehicle,
        odometer_reading,
        odometer_code,
        vin,
        location_id,
    from
        {{ ref('stg_electric_vehicles') }}
)

select
        td.transaction_id,
        td.sale_price,
        td.transaction_date,
        td.transaction_year,
        td.sale_date,
        td.transaction_type,
        td.new_or_used_vehicle,
        td.odometer_reading,
        td.odometer_code,

        -- location info
        td.location_id,
        l.county,
        l.city,
        l.state_of_residence,
        l.zip,
        l.legislative_district,

        --vehicle info
        td.vin,
        v.electric_vehicle_type,
        v.make,
        v.model,
        v.model_year,
        v.vehicle_primary_use,
        v.electric_range,
        v.base_msrp

from transaction_data td
left join
    {{ ref('dim_location_info') }} l ON td.location_id = l.location_id
left join
    {{ ref('dim_vechile_info') }} v ON td.vin = v.vin
```

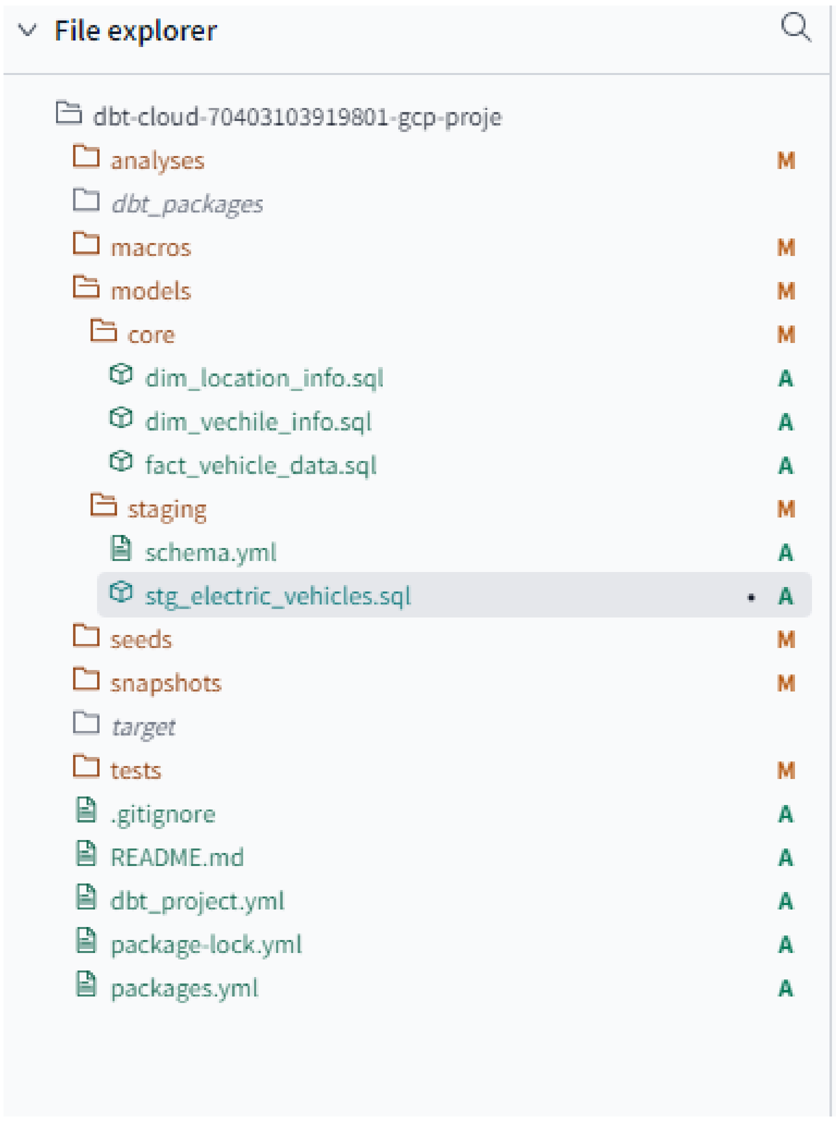
APPENDIX 13

fact_vehicle_data.sql used to build the table in bigquery in the form of Fact.

```
{{
    config(
        materialized='view'
    )
}}

with

source as (

    select * from {{ source('staging', 'electric_vehicles_data') }}

),

renamed as (

    select
        -- Transaction information
         {{ dbt_utils.generate_surrogate_key(['vin_1_10', 'transac-
tion_date']) }} as transaction_id,
        cast(transaction_date as datetime) as transaction_date,
        transaction_year,
        cast(date_of_vehicle_sale as datetime) as sale_date,
        transaction_type,
        sale_price,
        new_or_used_vehicle,
        odometer_reading,
        odometer_code,

        -- Vehicle information
        vin_1_10 as vin,
        dol_vehicle_id,
        electric_vehicle_type,
        make,
        model,
        model_year,
        vehicle_primary_use,
        electric_range,
        base_msrp,

        -- Location information
        {{ dbt_utils.generate_surrogate_key(['zip', 'city', 'state_of_res-
idence']) }} as location_id,
        county,
        city,
        state_of_residence,
        zip,
        legislative_district,

    from source
)

select * from renamed
```

APPENDIX 14

View of dbt cloud showing the staging and core schema and the dimensions and fact models arrangement.



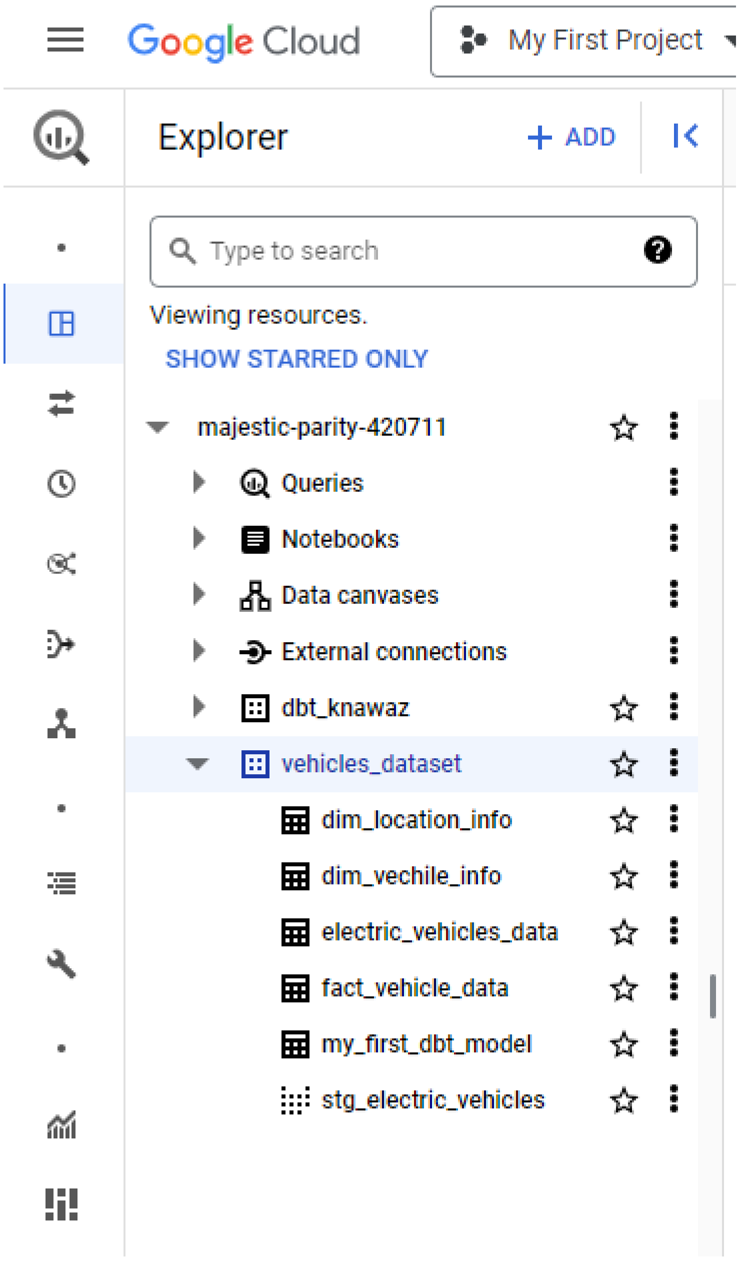
File explorer

```
dbt-cloud-70403103919801-gcp-proje
  analyses                      M
  dbt_packages
  macros                        M
  models                        M
    core                        M
      dim_location_info.sql     A
      dim_vechile_info.sql      A
      fact_vehicle_data.sql     A
    staging                     M
      schema.yml                A
      stg_electric_vehicles.sql • A
  seeds                         M
  snapshots                     M
  target
  tests                         M
  .gitignore                    A
  README.md                     A
  dbt_project.yml               A
  package-lock.yml              A
  packages.yml                  A
```

FIGURE 29 dbt Project

APPENDIX 15

View of tables generated in Google BigQuery with the help of dbt Cloud.



Viewing resources.

SHOW STARRED ONLY

- majestic-parity-420711
  - Queries
  - Notebooks
  - Data canvases
  - External connections
  - dbt_knawaz
  - vehicles_dataset
    - dim_location_info
    - dim_vechile_info
    - electric_vehicles_data
    - fact_vehicle_data
    - my_first_dbt_model
    - stg_electric_vehicles

FIGURE 30 Generated Tables

APPENDIX 16

Variables.tf file for terraform used to define the variables that will be referenced in main.tf for building the infrastructure on Google cloud.

```
locals {
  data_lake_bucket = ‘vehicles_bucket’
}

variable ‘project’ {
  description = ‘Your GCP Project ID’
  default     = ‘majestic-parity-420711’
}

variable ‘region’ {
  description = ‘Region for GCP resources.’
  default     = ‘europe-north1’
  type        = string
}

variable ‘storage_class’ {
  description = ‘Storage class type for your bucket.’
  default     = ‘STANDARD’
}

variable ‘BQ_DATASET’ {
  description = ‘Big Query Dataset that raw data’ll be written to’
  type        = string
  default     = ‘vehicles_dataset’
}
```

APPENDIX 17

Main.tf file for terraform used to define the resources used for building the infrastructure on Google cloud.

```
  required_version = ‘>= 1.0’
  backend ‘local’ {} # Can change from ‘local’ to ‘gcs’ (for Google) or
‘s3’ (for aws), if you would like to preserve your tf-state online
  required_providers {
    Google = {
      source = ‘hashicorp/Google’
    }
  }
}

provider ‘Google’ {
  project = var.project
  region  = var.region
  // credentials = file(var.credentials)  # Use this if you do not want to
set env-var GOOGLE_APPLICATION_CREDENTIALS
}


resource ‘Google_storage_bucket’ ‘data-lake-bucket’ {
  name     = ‘${local.data_lake_bucket}_${var.project}’ # Concatenating DL
bucket and Project name for unique naming
  location = var.region

  # Optional, but recommended settings:
  storage_class               = var.storage_class
  uniform_bucket_level_access = true

  versioning {
    enabled = true
  }

  lifecycle_rule {
    action {
      type = ‘Delete’
    }
    condition {
      age = 30 // days
    }
  }

  force_destroy = true
}
```

```
# DWH
# Ref: https://registry.terraform.io/providers/hashicorp/Google/lat-
est/docs/resources/bigquery_dataset
resource ‘Google_bigquery_dataset’ ‘dataset’ {
  dataset_id = var.BQ_DATASET
  project    = var.project
  location   = var.region
}
```

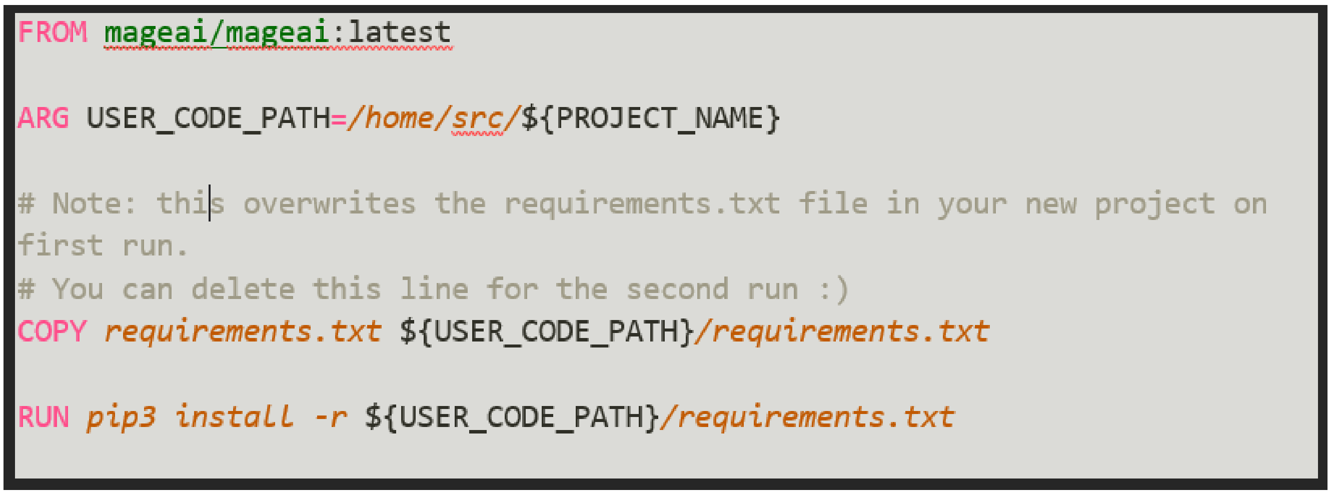
APPENDIX 18

Schema.yaml file to configure the dataset and tables in Google BigQuery.

```yaml
version: 2

sources:
  - name: staging
    database: majestic-parity-420711
    schema: vehicles_dataset

    tables:
      - name: electric_vehicles_data
```

APPENDIX 19

Docker file to create the Mageai image that will be used for data orchestration.

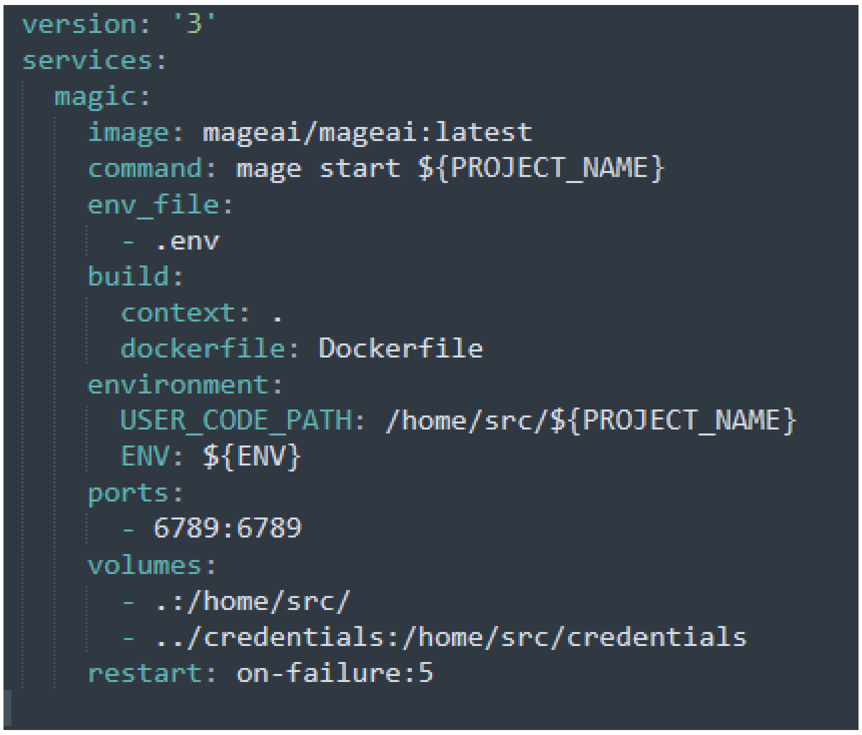


```
FROM mageai/mageai:latest

ARG USER_CODE_PATH=/home/src/${PROJECT_NAME}

# Note: this overwrites the requirements.txt file in your new project on
first run.
# You can delete this line for the second run :)
COPY requirements.txt ${USER_CODE_PATH}/requirements.txt

RUN pip3 install -r ${USER_CODE_PATH}/requirements.txt
```

FIGURE 31 Docker File

APPENDIX 20

Docker-Compose file to create the Mageai container and running that on port 6789 that will be used for data orchestration.



```yaml
version: '3'
services:
  magic:
    image: mageai/mageai:latest
    command: mage start ${PROJECT_NAME}
    env_file:
      - .env
    build:
      context: .
      dockerfile: Dockerfile
    environment:
      USER_CODE_PATH: /home/src/${PROJECT_NAME}
      ENV: ${ENV}
    ports:
      - 6789:6789
    volumes:
      - .:/home/src/
      - ../credentials:/home/src/credentials
    restart: on-failure:5
```

FIGURE 32 Docker-Compose file