GCP Project Setup

- Created a new GCP project, enabled billing, set Budget & Alerts to trigger alerts at a certain threshold (£10).
- Enabled APIs: BiqQuery, Cloud Resource Manager, IAM, Cloud Storage.
- Installed Terraform and gcloud CLI locally, and authenticated on my local machine.

I created a new GitHub repository and cloned locally.

CSV File Processing

I used a Google Colab notebook to explore and clean the file with Pandas and Python.

- Checked for missing values and duplicates.
- Replaced "INVALID DATE" with a default value and converted it to a datatype.
- Replaced stings like "One hundred pounds" with integers. If the file was larger, I would consider removing those records or setting a default value instead, as replacing them with corresponding integers is not efficient and prone to errors.
- Replaced missing ProductID values with a default value.
- Converted Quantity into integer, OrderAmount as float, kept OrderDate as date, and converted the remaining columns into string.
- Added a new TotalOrderValue column and calculated the values.
- Saved the cleaned data into a new csv file.

Infrastructure As Code

I used Terraform to provision the following infrastructure:

- A GCS bucket for data files;
- A GCS bucket for Terraform state files;
- A service account with access to Cloud Storage and Big Query;
- A BigQuery dataset;
- A BigQuery table with a specified schema and partitioning by date.

After I had my SA created, I obtained credentials for it and downloaded them as a JSON file. I added the file to GitHub secrets for my repository.

GitHub Actions

I used GitHub Actions to create a Terraform Workflow to authenticate to Google Cloud, set up resources, check terraform files format, and add Plan and Apply stages. The Apply stage can only be executed when the code is merged to the main branch.

Uploading CSV to Cloud Storage and Loading Data into BigQuery

I created two Python scripts: one to upload the cleaned csv to the Cloud Storage bucket, and the other to load the data from the file to an existing empty BQ table with specified partitioning.

Querying the Data

Since BigQuery does not support RANGE with INTERVAL for date or timestamp columns, I used a subquery instead to calculate a rolling average of customer spending over the last 30 days.

I used GROUP BY with the aggregate function to obtain a sorted list of regions based on the highest average spending.