COMM034 Coursework Report

Student Name	Aaditya Singh
Course Name	COMM034
Date Submitted	04/05/2025

Table of Contents

A. Project Design & Planning

Since we are dealing with a large static data source, we need to integrate a cloud storage model like ‘Object storage’ with the help of GCS. We are making use of GCP as it is required for this coursework. To access this data from a Data Warehouse, GCP offers BigQuery where we can query large datasets. To clean and transform data, we can use BigQuery for SQL based cleaning or Vertex AI workbench which are Python-based notebooks to build machine learning models. Google Looker Studio is one of the biggest visualization tools available which can be connected with BigQuery for seamless data integration.

A list of functional Requirements:

- Store and access a large static data source
- Load this data into a warehouse
- Run SQL queries on this data for analysis
- Build dashboard for data visualization

List of Cloud services:

- BigQuery Editions
- Vertex AI pipelines
- Dataflow classic
- Cloud Storage

Approximate Cost:

$6.11 per month

Limitation and opportunities for improvement:

Cloud based systems can become costly at scale for small to medium scale business with huge amounts of data.

B. Explore the Environment

Successfully deployed the solution and understood what is needed to be done in the coursework. Used BigQuery as a data warehouse and Looker Studio for Data Visualization. Development was done using Google Cloud console and a step-by-step walkthrough was provided in the form of tutorial. When the 'Data warehouse with BigQuery' Solution was deployed, there were 83 different resources spanned out across multiple Products which took about 10 minutes to initialize. The Dataset used in this project is called 'thelook-ecommerce'. It consists of 8 tables with user and item related data with table sizes ranging from 9 MB to 200 MB. The status for all these Products became 'Created' once they were successfully deployed. Our dataset – 'the-look' has 7 tables which are related to different aspects of a fictitious e-commerce clothing brand. All the tables had a primary key – 'id'. Lastly the looker studio dashboard provides a real-time overview of the dataset. The first page was the 'Top-line Business Pulse' and it includes cards with details about revenue, inventory costs, and profit trends. A few visualizations were about Revenue over the years, Revenue by category and the fastest growing category. The second page was named 'Distribution Center Efficiency' which featured key metrics such as total products processed, the average hours it took to process and the average hours it took from order to delivery. It also included a map of average order processing time in hours, a table which compares various metrics and a historical demand trend by location.

C. Familiarize with Solution

All tutorials within Data Warehouse for BigQuery had a minimal learning curve and therefore were easy to navigate. By default, all queries are named 'Untitled Query' until they are saved, and this makes having several open queries confusing. It is inconvenient to have to manually place a lengthy <walkthrough-project-id/> in every code snippet. One potential improvement would be to have users input their project ID once and have this copied to all snippets. Following is the individual tutorial wise improvements or errors I encountered-

Sr. No.	Tutorial name	Errors and Improvement
1	BigQuery Machine Learning	Suggestion: Request changes or modification of the code from user instead of providing complete solutions. This will reduce repetitive occurring queries.
2	BigQuery for SQL Translations	Error: The PostgreSQL code shouldn't start with 'CREATE TABLE…' since the tables are already created.
3	BigQuery Fine Grained Security	Suggestion: $ *session_user* is misleading in context of using a cloud shell terminal, since we obtain the username from the SQL query. Error: We cannot manage Data Policies to create masking rules for policy tags as we need to belong to an organization. Doubt: Why is it so easy to define access to row-level access controls in contrast with column-level access controls?
4	BigQuery for Analytics	Error: FORMAT_DATE function- An error occurred when trying to ORDER BY a column from APPROX_QUARTILES; so, assigned [OFFSET(2)] to an alias 'median' which allowed proper sorting.
5	Gen AI for BigQuery ML	Error: thelook doesn't have any model called 'text_generate_model' and creating one requires additional steps which are not currently enabled.

6	Connecting BigQuery data to apps	The notebook was not preloaded in the project environment (cc-coursework-456415). I had to manually initialize the notebook from GitHub to continue the tutorial.

D. Query the Dataset

Query 1:

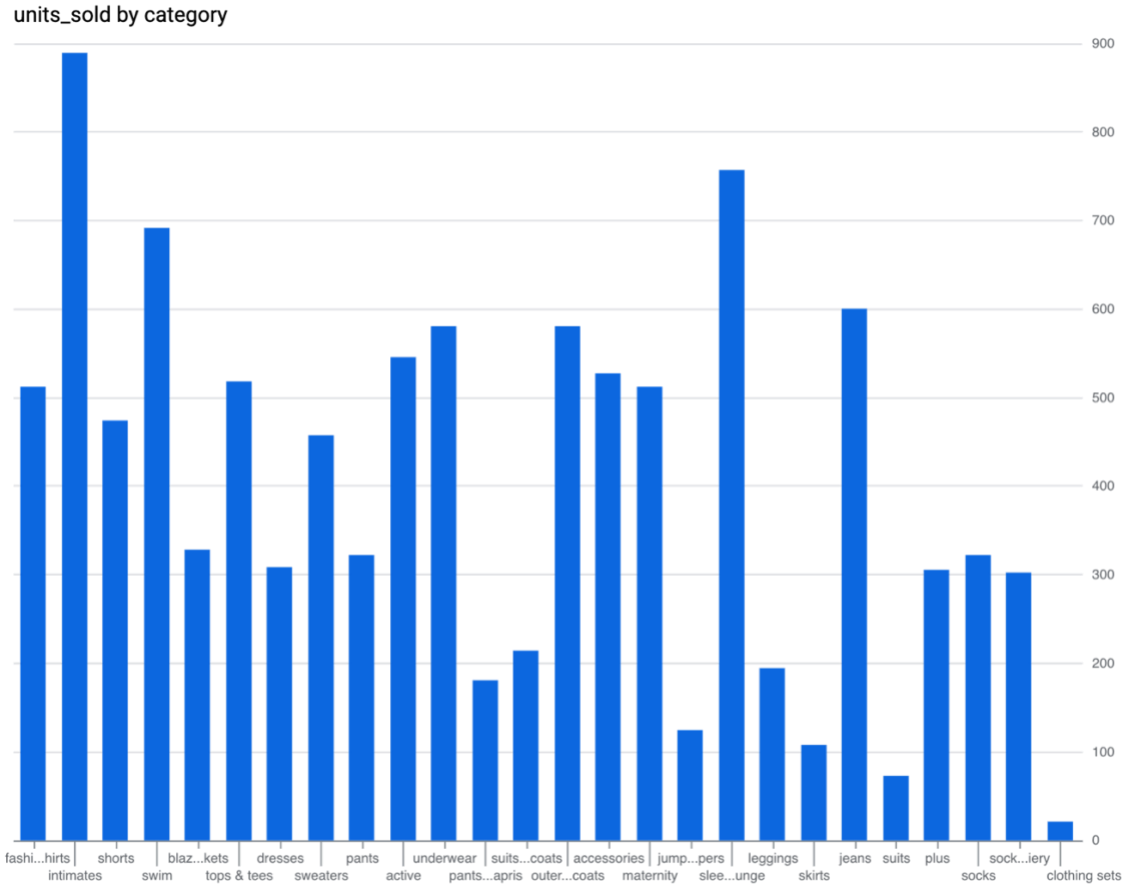
```
SELECT
dc.name AS center,
LOWER(TRIM(p.category)) AS category,
--tackles case insensitivity and spaces
COUNT(order_items.product_id) AS units_sold,
--total number of units
SUM(order_items.sale_price) AS total_sales
FROM `cc-coursework-456415.thelook.order_items` AS order_items
--primary transactional table
JOIN `cc-coursework-456415.thelook.inventory_items` AS inventory
ON order_items.inventory_item_id = inventory.id
--connects each order-item to an inventory record
JOIN `cc-coursework-456415.thelook.products` AS p
ON order_items.product_id = p.id
--straightforward relational link between order_items and products
JOIN `cc-coursework-456415.thelook.distribution_centers` AS dc
ON inventory.product_distribution_center_id = dc.id
--maps inventory to center
WHERE order_items.status = 'Shipped'
--consider only actually fulfilled order only
GROUP BY dc.name, category
ORDER BY dc.name, units_sold DESC;
```

1. Explanation- This query helps us understand sales performance of product category which were shipped from various centers. This query makes use of tables such as order_items, inventory_items, products and distribution_centers. We take information from certain columns from these tables and combine them to understand which categories are sold the most and from which center, how many units have been sold and the revenue that was generated from those sales.
2. Working- The order_items table includes information of every item in a customer order, we make use of JOIN syntax to join order_items table with inventory_items table which tells us where that product was specifically shipped from. JOINing products on their id allows us to group any metrics by product category or name, since product_id is common in both order_items and products. We JOIN distribution centers to associate each item with its distribution center. Here, inventory.product_distribution_center_id is the key which maps the inventory to the centers. We are only considering fulfilled items by using the WHERE clause, this ensures that cancelled, returned, or still processing types of transactions are not being considered. The GROUP BY clause sorts the data by grouping it based on the category of products for each center. The ORDER BY statement allows us to organize the result in a

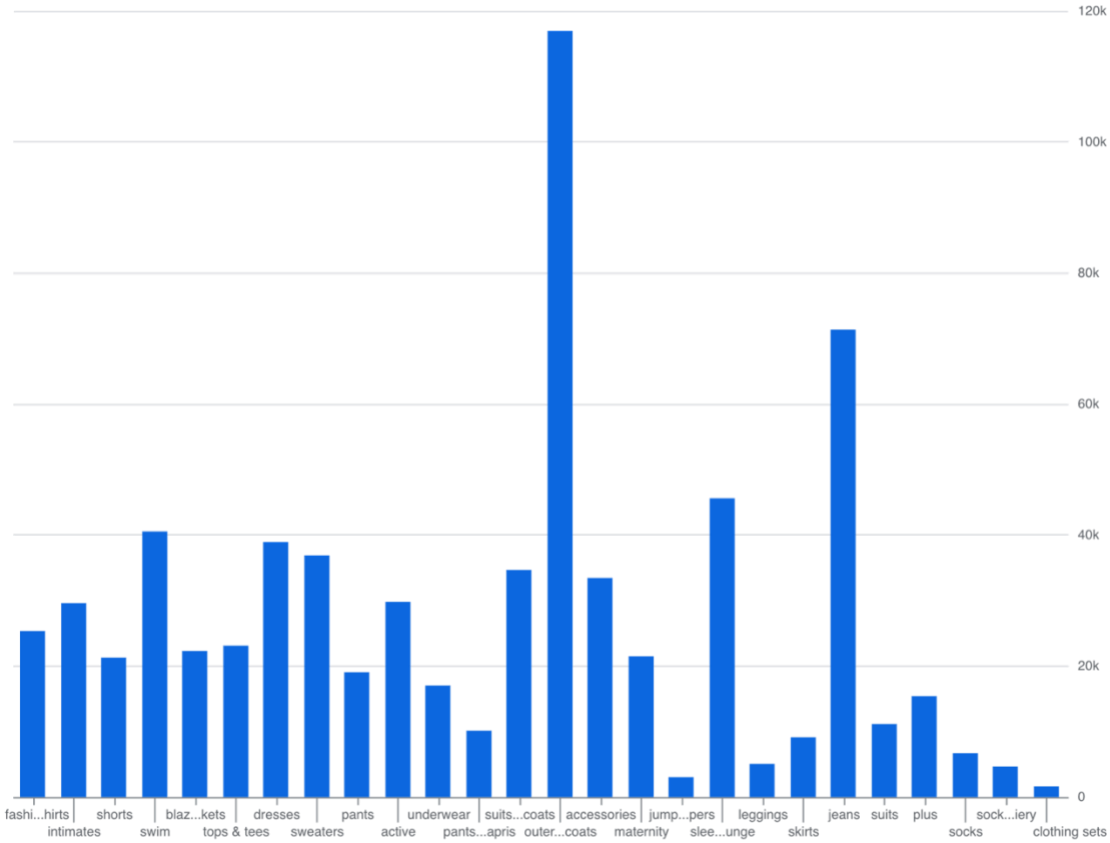
descending order such that the highest selling categories appear first. The result shows us

Result 1: Query 1 Result

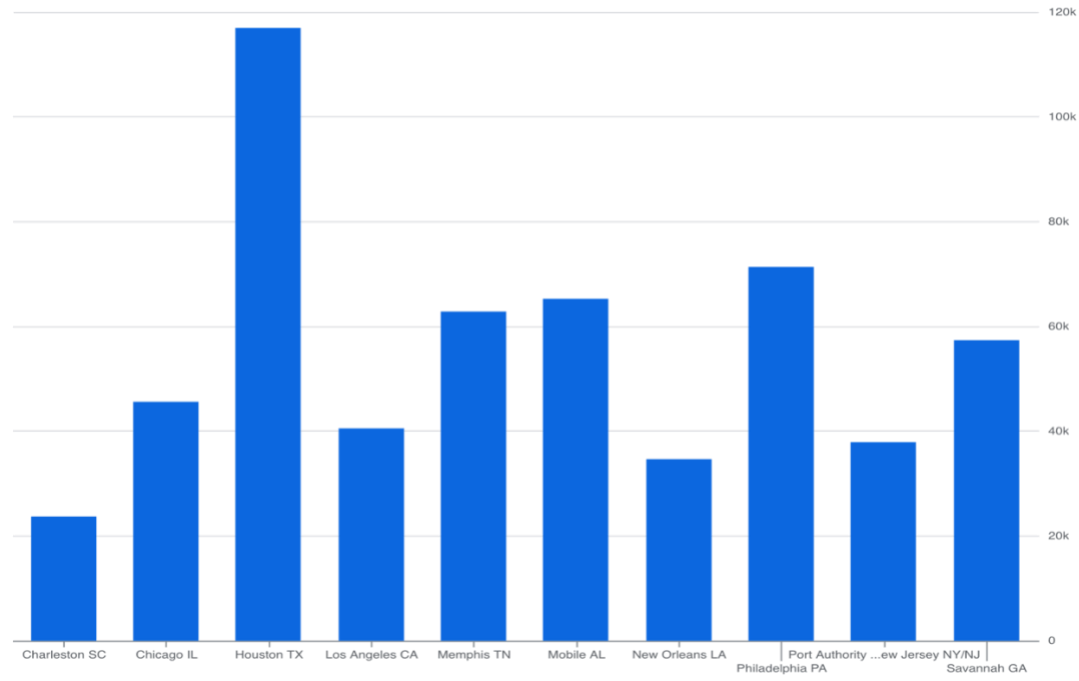
The category with highest number of units sold was intimates which was followed by sleep and lounge and swim. The lowest selling units were clothing sets, suits and skirts. The most revenue was brought by outer wear and coats, followed by jeans clothing sets along with jump suits and rompers brought the least revenue. Houston, TX recorded the highest total sales and Chicago, IL sold the most units.



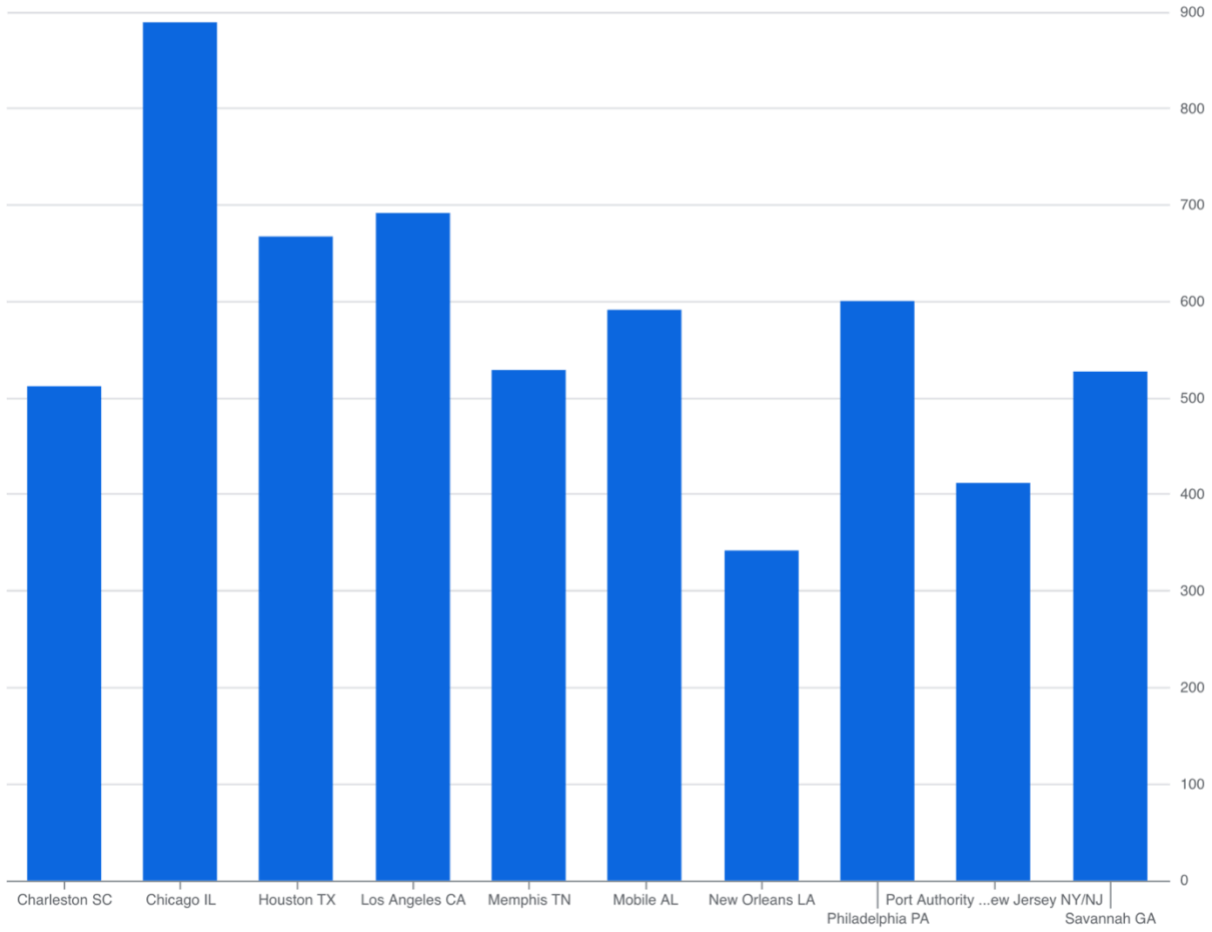
total_sales by category



total_sales by center



units_sold by center



Query 2:

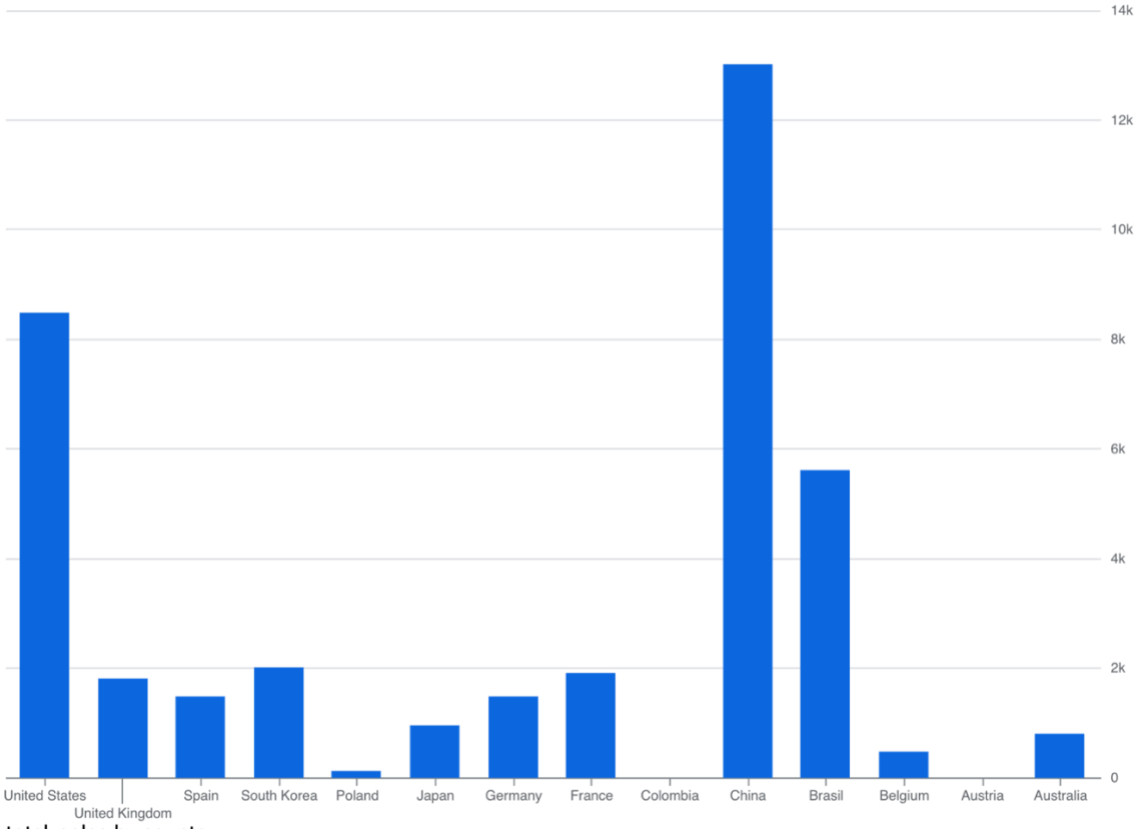
```
SELECT
u.country,
u.traffic_source,
COUNT(*) AS user_count,
SUM(oi.sale_price) AS total_sales
FROM `cc-coursework-456415.thelook.users` AS u
JOIN `cc-coursework-456415.thelook.orders` AS o
ON u.id = o.user_id
JOIN `cc-coursework-456415.thelook.order_items` AS oi
ON o.order_id = oi.order_id
WHERE o.status = 'Shipped'
GROUP BY u.country, u.traffic_source
ORDER BY country DESC;
```

1. Explanation- The query tries to analyze how different traffic source tries to contribute to the total sales revenue. We achieve this by finding out the number of users for each traffic source and group them based on country.
2. Working- We select 4 columns which will be displayed such as country, traffic source, volume of order items which is user counts and total revenue. We JOIN each user with their orders with the help of id. We JOIN the order_ids from orders to order items as well which gives us the sales price. We only consider shipped orders to filter completed orders. We GROUP the results by country and traffic source and ORDER BY country to sort the results in a reverse alphabetical order.

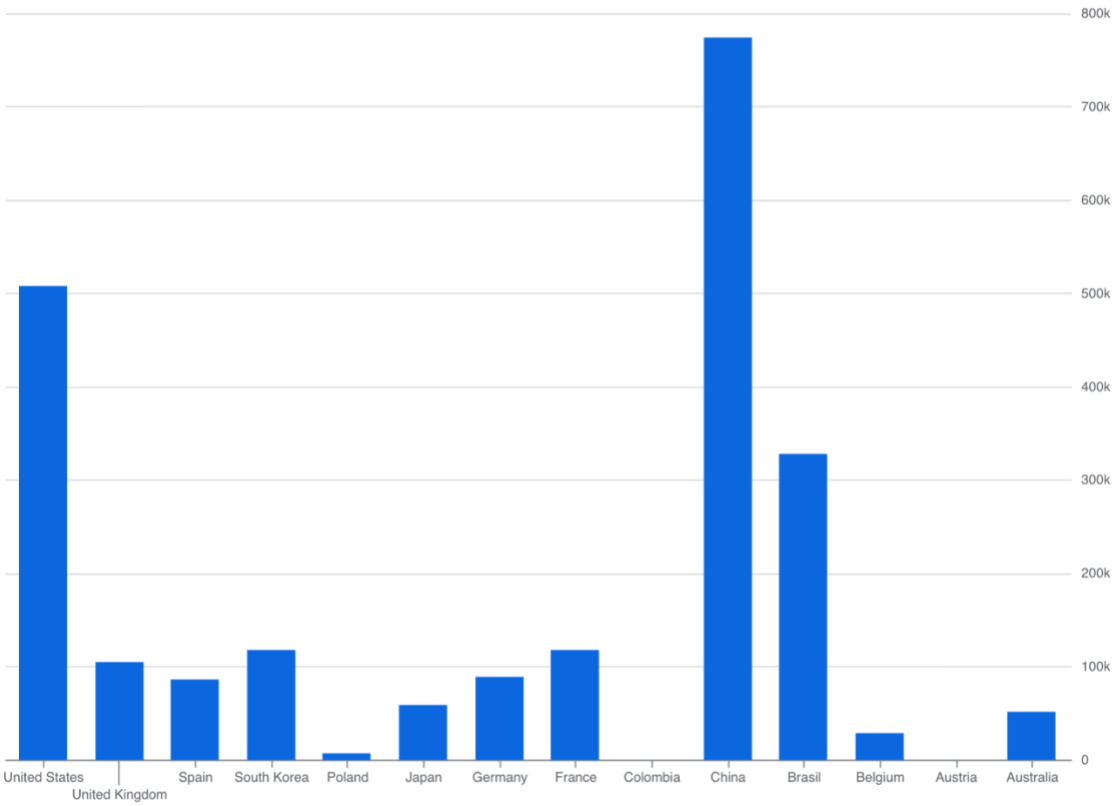
Result 2: Query 2 result

Highest sale was brought in by users from China, United States and Brazil via the ‘search’ traffic source. These 3 countries had the highest user count as well. Countries with lowest user count and sales were Colombia, Austria and Poland. Search leads as the top source in bringing users with organic leading after it.

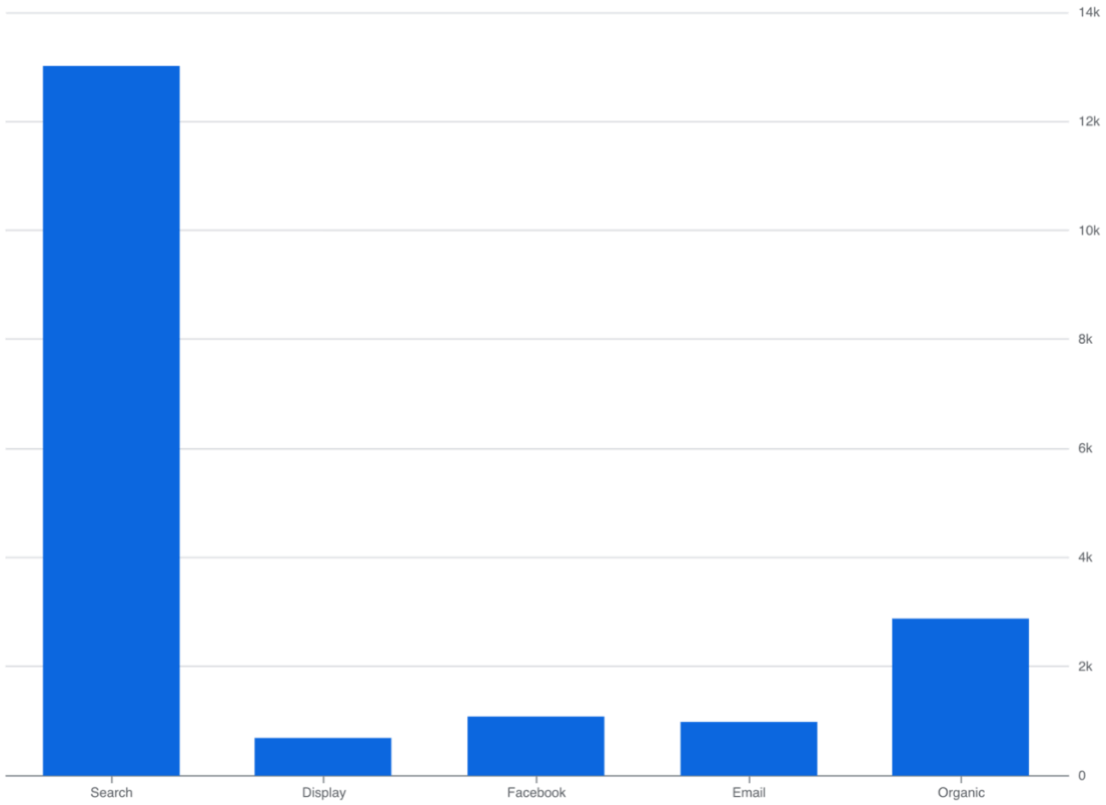
user_count by country



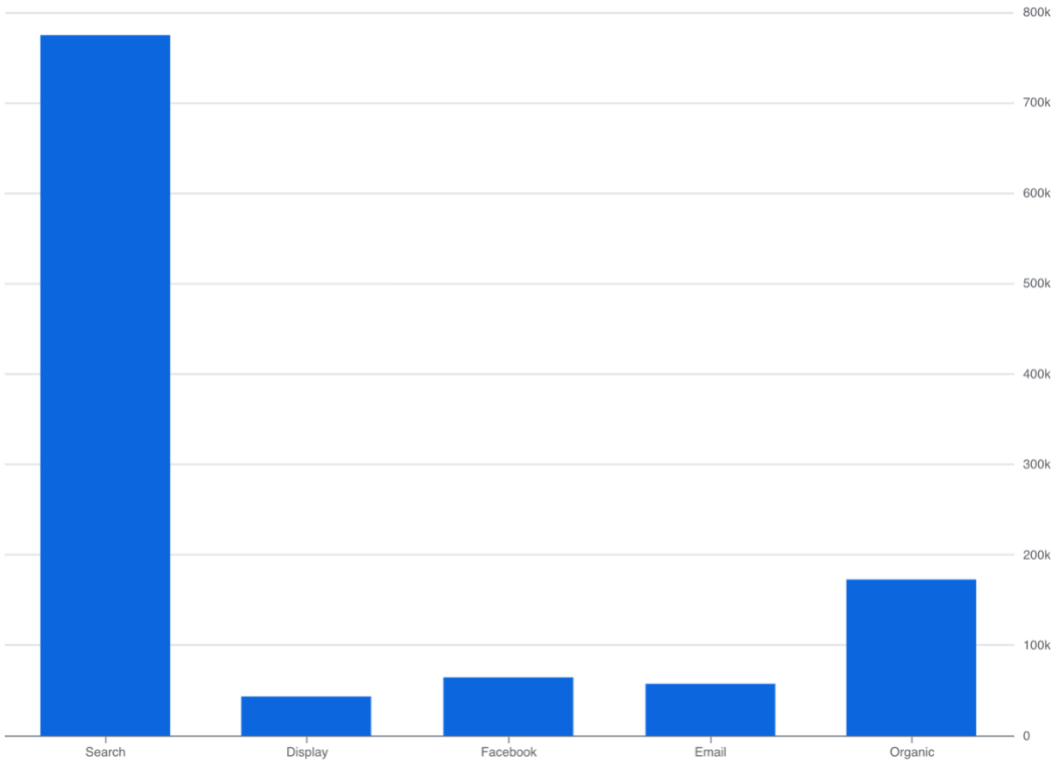
total_sales by country



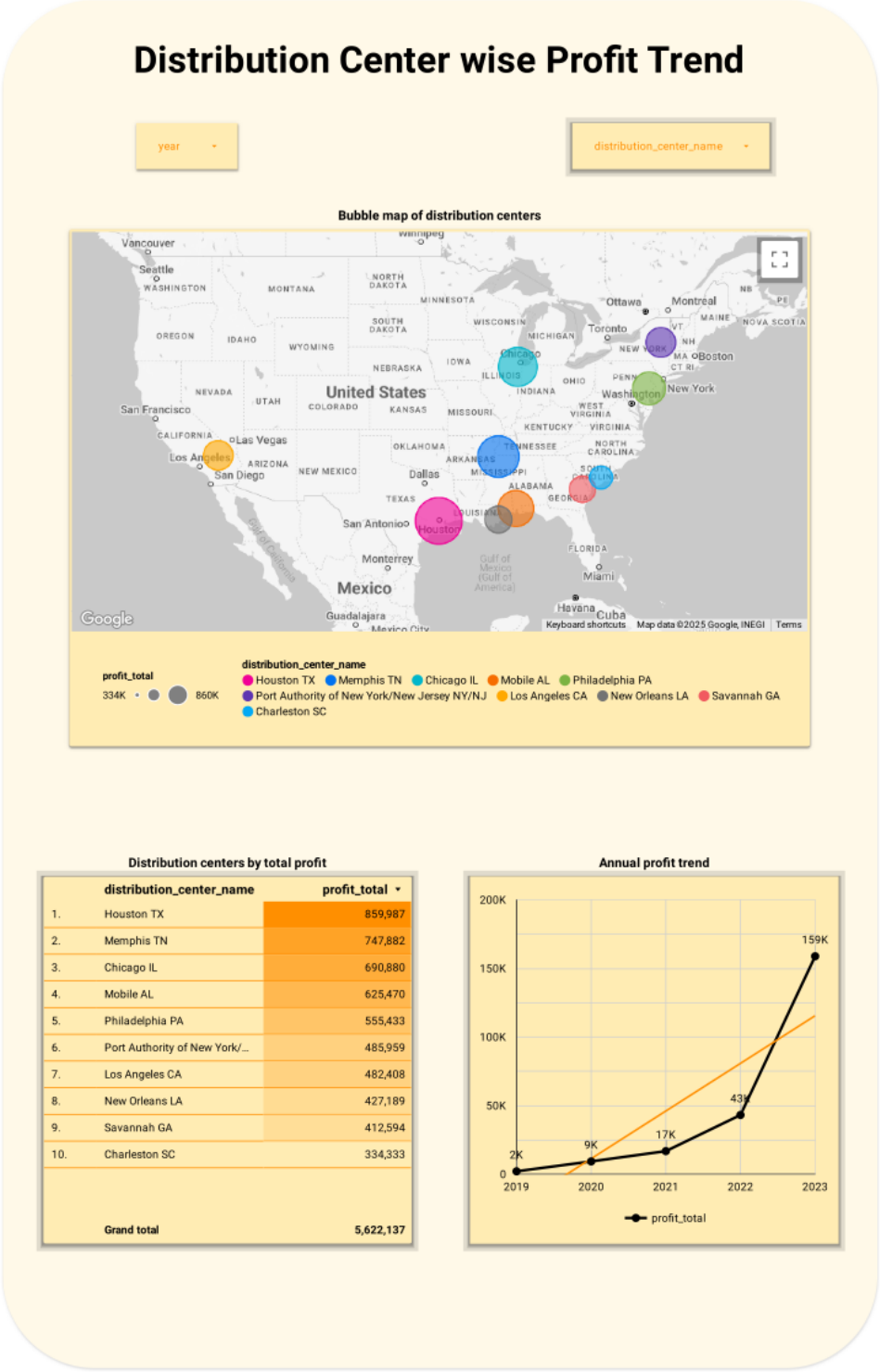
user_count by traffic_source



total_sales by traffic_source



E. Update the Dashboard



The new page in the dashboard is titled "Distribution Center Wise Profit Trend" and it provides a total view of profitability across all the center locations within the business. This page was designed to display operational information and strategy by combining location specific profit with understanding about yearly trend. This table shows which visualizations were chosen and why:

Visualisation type and name	Function	Reason for choice
Table with Heatmap: Distribution centers by total profit	Ranks each center by total profit over multiple years. Helps identify best and worst performing locations.	This visualisation uses a Heatmap to highlight total profit contribution from each distribution center. It lists centers in a descending order of profit generated, making it easy to evaluate center performance and spot underperforming locations.
Time Series chart: Annual Profit Trend	Tracks profit performance for given years. Provides insights on growth or decline of the company over time.	This visualisation is helpful for strategic planning and understanding the company's stand in the market by looking at the profit. A linear trendline was added to the graph which illustrates the direction of profit growth.
Bubble map: Bubble map of distribution centers	Displays total profit achieved by location. Size of the bubble indicates amount of profit.	A bubble map geographically displays differences with the help of different types of shapes. Over here I chose a bubble size to corelate with the amount of profit from that location.

The dashboard reveals that Houston, Memphis, and Chicago are the most profitable distribution centers. A consistent upward profit trend is observed from 2019 to 2023, with the bubble map highlighting Houston, Memphis, and Chicago bringing the most profit.

F. Querying through Cloud Run

Flask web-app: https://queries-7xrnyfweoa-uc.a.run.app/

I developed and deployed a simple Flask-based application using Google Cloud Run which executes 2 different SQL queries on a BigQuery table and presents the results through a web-based user interface.

Query r	Name	Functionality
1	Distribution Centers	Displays all the distribution centers names along with their ic from the BigQuery dataset thelook.
2	Products cost by category	Aggregates the product costs and provides statistics on minir maximum and average cost of product categories.

The application was built using Flask and Google's Cloud Shell editor. The user can select from the 2 queries which is provided to them via a web-interface from index.html. The user selects the query which turns into a POST request and is sent to the Flask backend - main.py. The selected query gets executed and the result is converted into a Pandas dataframe and passed back to the HTML template. Jinja2 dynamically generates a table that displays the queried result.

Deployment was done using Cloud Shell Editor on Google Cloud Platform (GCP). The application was containerised using a Dockerfile that installs dependencies, exposes port 8080 and runs the app. The Docker image was built using Cloud Shell Editor status bar using 'Deploy to Cloud Run'. The container is deployed to Cloud Run, where it runs without a server.

When a user accesses the service, Cloud Run provides a container instance and then routes the HTTP request to the Flask app. The Flask app queries BigQuery based on user input and then retrieves the result which is sent as a response back to the browser within seconds.

G. Research Different Platform

To replicate this pipeline implemented on Google Cloud Platform (GCP) on another cloud platform, I chose Amazon Web Services (AWS). AWS offers tools and services similar to GCP's Cloud Storage, BigQuery, Cloud Run, Vertex AI and Looker Studio.

Google Cloud Platform tools and services	**Amazon Web Services equivalent**
Cloud Storage	Amazon S3 (Simple Storage Service)
BigQuery	Amazon Redshift
Vertex AI	Amazon SageMaker
Looker Studio	Amazon QuickSight
Cloud Run	AWS Lambda or AWS Fargate

Approximate cost: $30.41 per month

To build this platform in AWS, I will have to start by uploading the dataset (thelook-ecommerce) into an S3 bucket. I will set up a Redshift cluster to query the dataset. I will have to load the parquet data from S3 into the Redshift tables using COPY commands. For analysis, I will use Redshift Query editor and define the tables to match with thelook-ecommerce schema. Here, I will write and run custom queries just like in BigQuery (GCP) using SQL. To build a dashboard for data visualization, I will connect QuickSight to my Redshift data source. This will enable me to create and share any visuals I create. AWS Lambda can package my Flask app and create an API gateway that routes the HTTP requests. AWS Lambda is simpler to set up but if I want to take a Containerized path then I can use AWS Fargate which allows me to run a Docker image of my Flask app. This Docker image will be run in ECS and to give public access an API gateway can be attached. To train or add machine learning models just like Vertex AI in GCP, the AWS alternative is SageMaker Studio.

Challenges:

One of the main challenges in implementing the above solution would be complexity. AWS has a very complex setup unlike GCP. GCP has ‘one-click’ solutions and AWS often requires human intervention in configuring services such as IAM roles, S3 storage, QuickSight and RedShift. A lot of models pricing can be harder to estimate such as RedShift. Amazon QuickSight is very powerful but has a steep learning curve and lesser customization options as compared to GCP’s Looker Studio. Managing data permission can be a tedious task due to AWS’s policies. While AWS offers flexibility, its learning curve and service separation can be time-consuming for new users.

Advantages and trade-offs:

In AWS some aspects might work better, while others may be less efficient compared to GCP. Amazon S3 offers excellent durability, probably giving us better efficiency and functionalities compared to GCP’s Cloud Storage. AWS Lambda can also be more effective for serverless computing since it is more rapid and highly scalable. RedShift allows querying data directly from S3, similar to a data warehouse querying from a data lake approach. However, RedShift can also be less intuitive than BigQuery for beginners. RedShift also requires manual performance tuning, unlike GCP’s BigQuery which handles this automatically. As I mentioned in Challenges, QuickSight lacks a polished user experience as that of Looker Studio. It also doesn’t have any real-time dashboard interactivity that the Looker Studio provides. Cost estimation and control in AWS is also difficult due to its pricing.

After working through this coursework, I found GCP much easier to get comfortable with. The tools felt more interactive, and I was able to pick things up easily. I also liked how straightforward it was to deploy services since almost everything was automated and needed no manual configuration changes.

H. Video Demonstration

CC Coursework Report.docx