# Title : Sales Forecasting for E-Commerce Platform

# Introduction

Sales forecasting is crucial for businesses to make informed decisions and plan effectively. In this project, we aimed to explore the significance of selecting the appropriate forecasting model by leveraging Python programming and Google BigQuery. The selected dataset, spanning from December 2009 to April 2011, offers a rich source of transactional information suitable for sales forecasting analysis. With detailed records of item descriptions, quantities sold, prices, and customer details, the dataset enables a comprehensive understanding of sales patterns and customer behaviour over time by analysing historical sales data obtained from the dataset, we utilized several libraries, including pandas, matplotlib, seaborn, and statsmodels, to conduct comprehensive analyses and develop accurate forecasting models. Our methodology involved preprocessing the data, visualizing sales trends over time, identifying key insights, and implementing advanced forecasting techniques such as ARIMA (AutoRegressive Integrated Moving Average) models. Additionally, we utilized Google BigQuery to perform exploratory data analysis, extract valuable insights about transaction volumes across different countries, and compute aggregate metrics such as average transaction quantity and price. Through our analysis, we aimed to demonstrate the importance of model selection in generating reliable sales forecasts, enabling businesses to optimize their strategies, allocate resources effectively, and achieve sustainable growth in today's competitive market landscape.

# Description

## BigQuery

Google BigQuery, a serverless, highly scalable, and cost-effective data warehouse, was used for preprocessing the dataset. With its SQL-like syntax and powerful processing capabilities, BigQuery facilitated data cleaning, transformation, and aggregation tasks. The dataset, containing transactional information spanning from December 2009 to April 2011, was processed in BigQuery to extract relevant features for sales forecasting analysis. By leveraging BigQuery's efficient querying and processing capabilities, large volumes of data were processed swiftly, enabling seamless data preparation for subsequent analysis. Additionally, BigQuery's built-in functions such as REGEXP\_CONTAINS, FORMAT\_DATE, PARSE\_TIMESTAMP, GROUP BY, SUM, and ROUND were utilized

for various data manipulation and preprocessing tasks, allowing for efficient data transformation and aggregation.

## Jupyter Notebook

Jupyter Notebook, an interactive computing environment, was utilized for conducting data analysis, exploration, and documentation. With its support for various programming languages, including Python, R, and Julia, Jupyter Notebook provided a flexible platform for executing code, visualizing data, and generating insights collaboratively. Through Jupyter Notebook, code snippets, visualizations, and textual explanations were combined seamlessly, enabling the creation of reproducible and well- documented analyses. Additionally, Jupyter Notebook's integration with other libraries and tools, such as Pandas, Matplotlib, Seaborn, and Statsmodels, facilitated an integrated workflow for data analysis and modelling tasks.

## Pandas

Pandas is a powerful Python library for data manipulation and analysis. Pandas provided a rich set of data structures and functions for handling structured data effectively, including DataFrame objects for tabular data representation. Functions such as reading CSV files, cleaning missing values, filtering data, and computing descriptive statistics were performed using Pandas, facilitating comprehensive data exploration and preparation.

## Matplotlib

Matplotlib, a popular plotting library in Python, was utilized for creating insightful visualizations of sales data. Through Matplotlib, various types of plots, including line plots, bar plots, and scatter plots, were generated to visualize sales trends, distributions, and relationships between different variables. These visualizations provided a clear understanding of sales patterns over time and helped identify potential trends and anomalies within the dataset.

## Seaborn

Seaborn, built on top of Matplotlib, offered enhanced functionalities for statistical data visualization and exploratory data analysis (EDA). With Seaborn, sophisticated visualizations such as distribution plots, categorical plots, and heatmap visualizations were created to explore relationships and patterns in the sales data. Through EDA with Seaborn, insights were gained into the distribution of sales across different categories, the impact of variables on sales performance, and potential correlations between features.

## Statsmodels

Statsmodels is a Python library for estimating and analysing statistical models. It offers a wide range of statistical models, including linear regression, time series analysis, and hypothesis testing. Statsmodels provides tools for exploring data, fitting models to data, and making predictions.

## AutoRegressive Integrated Moving Average (ARIMA) model

The AutoRegressive Integrated Moving Average (ARIMA) model, a well-established and widely used method in time series analysis, was implemented using the Statsmodels library. ARIMA is renowned for its capability to capture and model complex temporal patterns present in sequential data, making it particularly suited for forecasting future values based on historical observations. The model consists of three main components: AutoRegressive (AR) terms, Integrated (I) terms, and Moving Average (MA) terms, each contributing to capturing different aspects of the data's behaviour. The AR terms capture the linear relationship between the current value and its past values, the I term deals with differencing to achieve stationarity, and the MA terms model the dependency between the current value and a stochastic term representing past forecast errors. By fitting the ARIMA model to the sales data, a comprehensive analysis of temporal trends and patterns was conducted, facilitating the generation of accurate forecasts for future sales. These forecasts provided invaluable insights for strategic decision- making in sales and inventory management, aiding businesses in optimizing their operations and maximizing profitability.

# Evaluation Plan

In this project, we aimed to evaluate several key aspects related to sales forecasting using the provided dataset. The main objectives were to:

## Dataset Description

Assess the effectiveness of data preprocessing techniques, including cleaning, transformation, and feature engineering, in preparing the dataset for forecasting analysis. The dataset used in this project consists of online retail sales and customer transaction data spanning from December 2009 to April 2011. It offers a comprehensive view of transactions, product details, and customer information documented by an online retail company based in the UK. With its rich and granular data, the dataset serves as a valuable resource for analysing sales performance, understanding customer preferences, and optimizing inventory management strategies.

**InvoiceNo:** This attribute consists of six-digit integral numbers that uniquely identify each transaction in the system. Transactions prefixed with 'c' denote cancellations, providing additional insight into purchase patterns.

**StockCode:** Stock codes are represented by five-digit integral numbers, allowing for easy identification and differentiation between products within the inventory system.

**Description:** Product names are included in this attribute, providing qualitative information about the items frequently bought and sold.

**Quantity:** The quantity attribute indicates the volume of each product involved in a transaction, offering valuable insights into purchasing trends.

**InvoiceDate:** Invoice dates are timestamped, recording the precise time when each transaction occurred. This information is essential for conducting time-based trend analysis and segmentation studies.

**UnitPrice:** Unit prices represent the retail price of each unit of a product, facilitating revenue calculations and cost-related analyses.

**Country:** This attribute indicates the geographical location of each customer, enabling geographical segmentation for further data investigation and analysis.

## Dataset Loading and Preprocessing

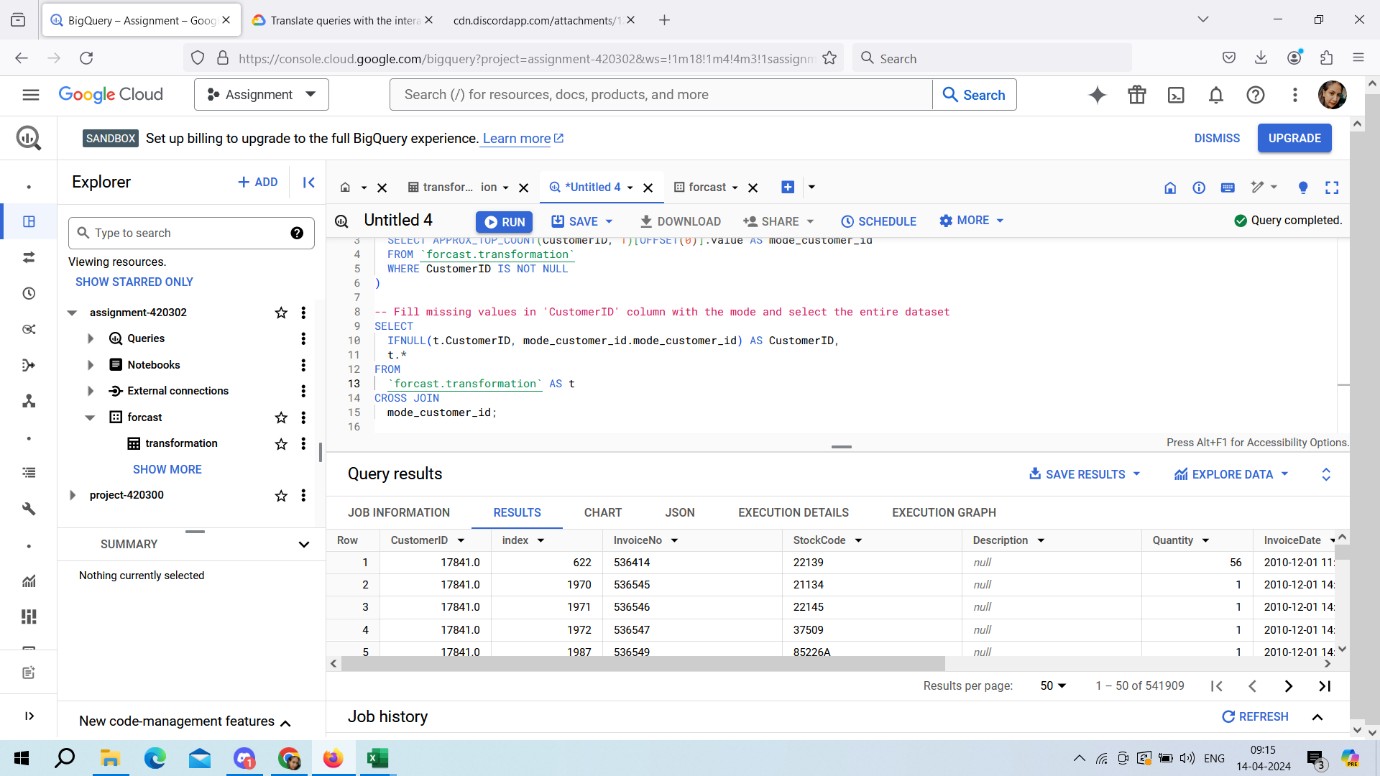
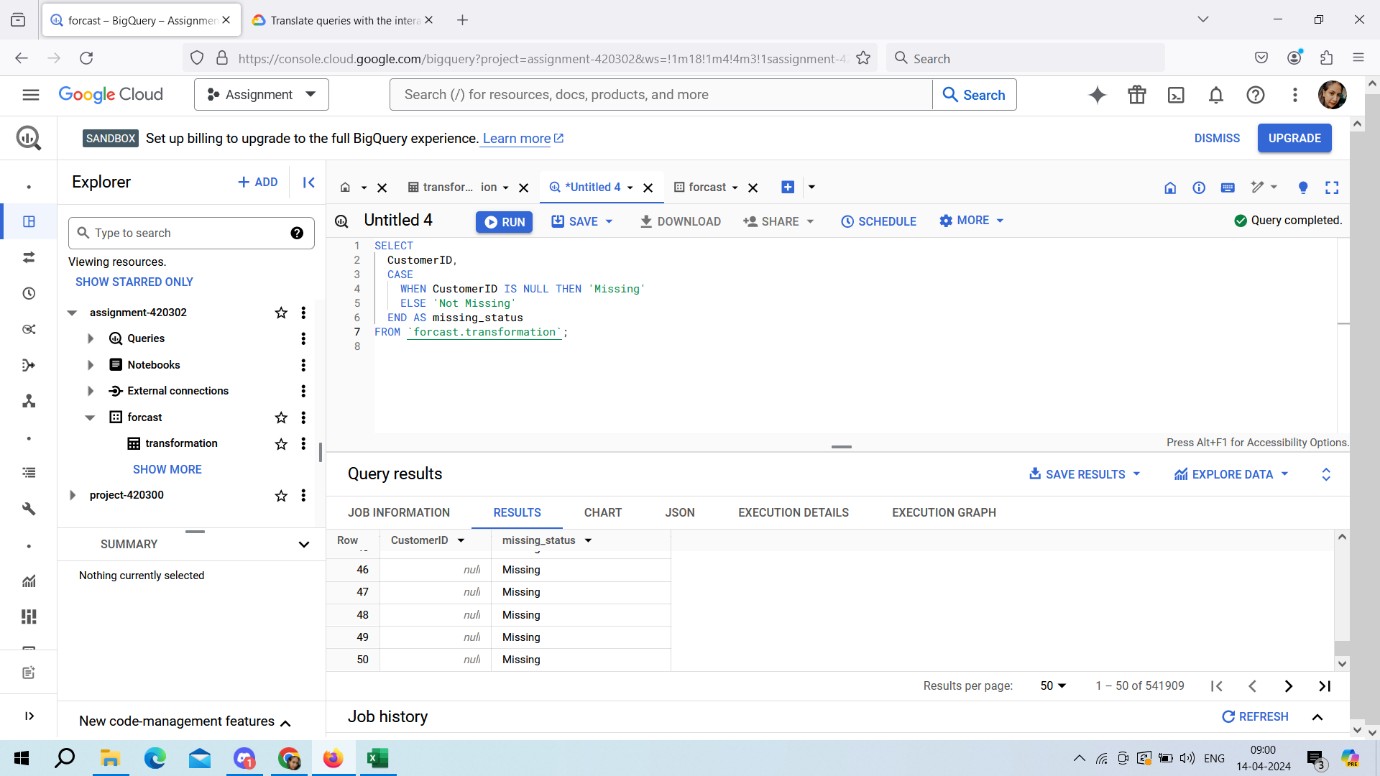
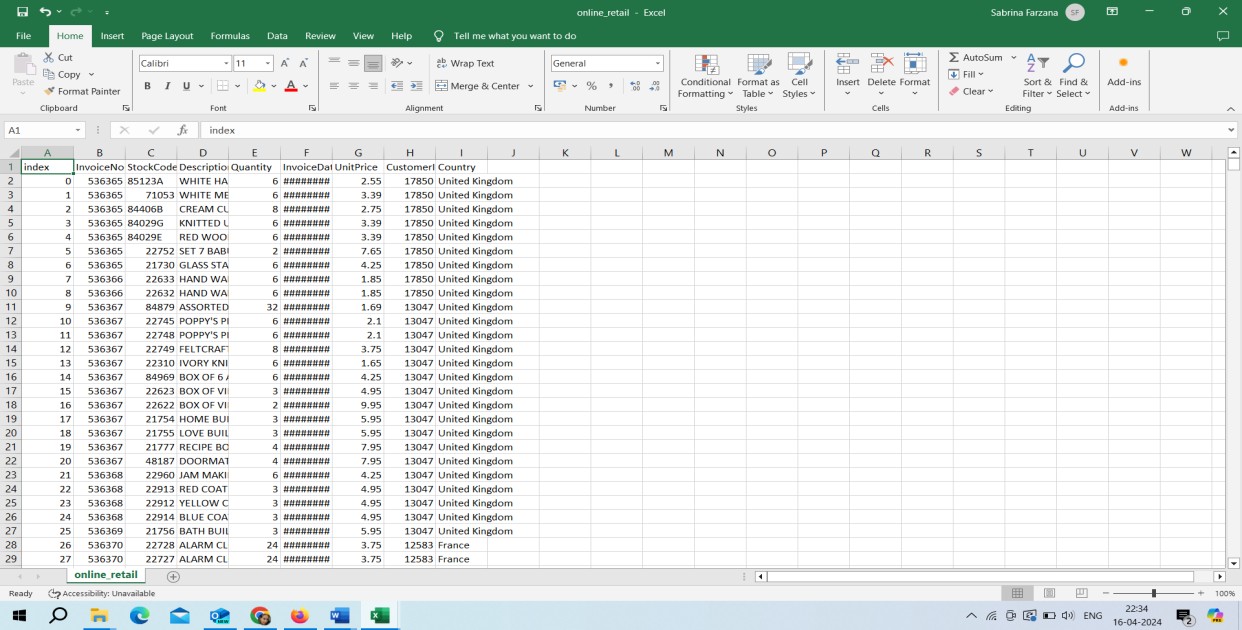
The dataset was initially loaded into Google BigQuery, a powerful and scalable data warehouse platform, to facilitate efficient data processing. Several preprocessing steps were performed to ensure data quality and integrity before analysis.

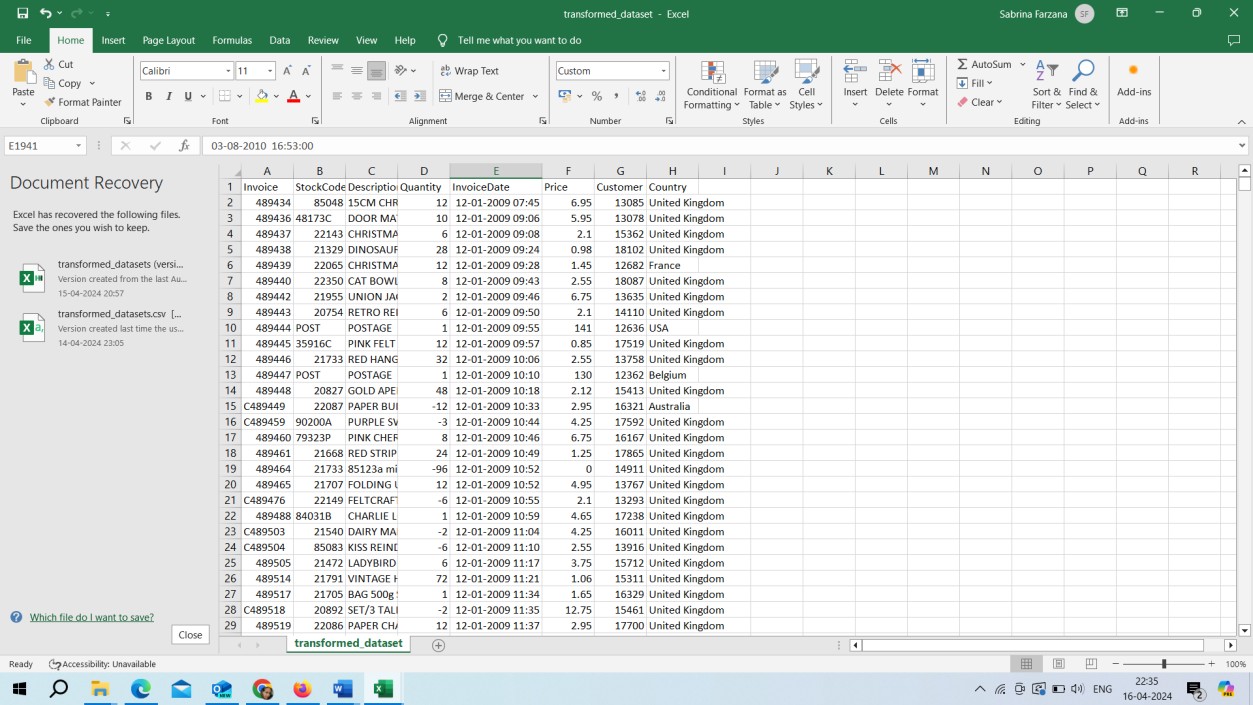
**Loading Data into BigQuery:** The dataset was imported into BigQuery, leveraging its SQL-like syntax and robust processing capabilities for handling large volumes of transactional data.

**Handling Missing Values:** Queries were executed to identify and remove records with missing values in critical attributes such as InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, and Country. This step ensured the completeness of the dataset and avoided potential biases in the analysis.

**Removing Duplicate Entries:** Duplicate entries were detected and eliminated using SQL queries, ensuring that each transaction was represented only once in the dataset. This helped maintain data accuracy and consistency throughout the analysis process.

Here is the Raw Data,

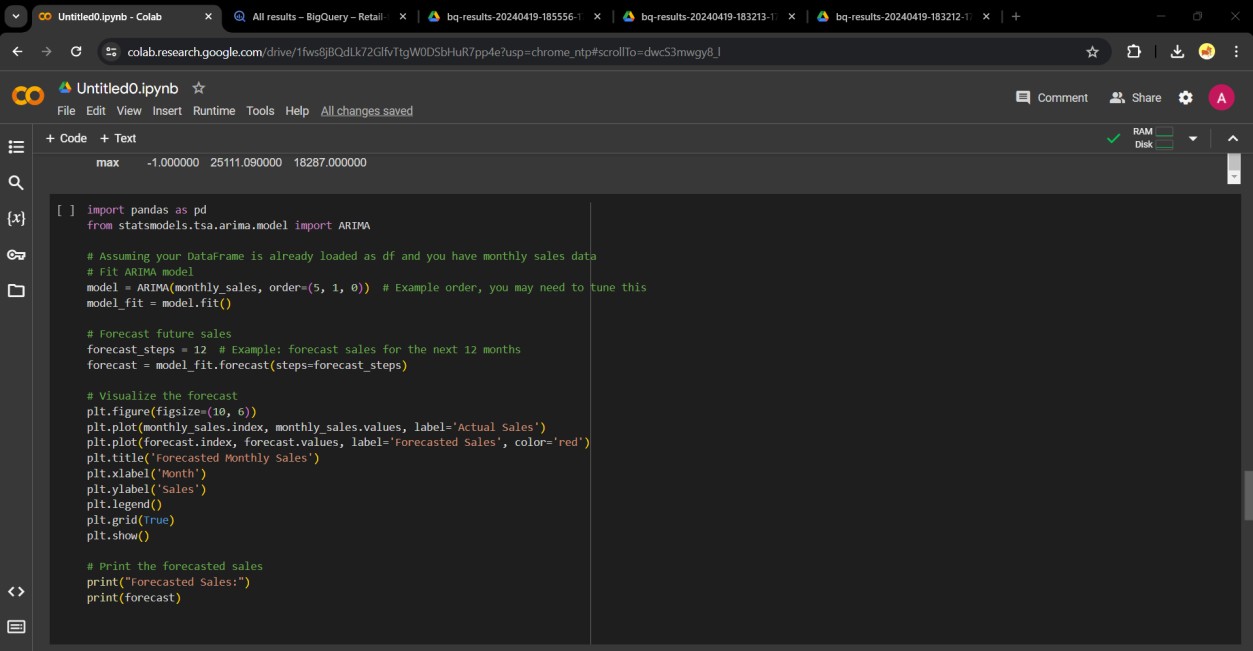


After Cleaning the dataset,

**4. Time Series Analysis:**

## Time Series Analysis

Evaluate the suitability and performance of time series analysis methods, particularly the ARIMA model, in capturing temporal patterns and generating accurate sales forecasts.



## Visualization Techniques:

Various visualization techniques were employed using Matplotlib and Seaborn to explore and analyse the sales data visually. The experiment included:

* + - Creating line plots to visualize sales trends over time and identify seasonal patterns.
    - Generating bar plots to compare sales across different categories or regions.
    - Using heatmaps to visualize correlations between sales and other variables.
    - Assessing the effectiveness of each visualization technique in conveying insights and facilitating data-driven decision-making.

## Experiment Design

**For data retrieval and exploration,** we'll run queries to display the first few rows of the dataset, count total rows, and check for null values. Visual inspection and analysis of the output will ensure the completeness and correctness of the data.

**For aggregation and summarization**, we'll execute queries to aggregate monthly sales, calculate summary statistics, and identify patterns in the data. Results will be validated against pre-calculated values or expected trends.

**For time series forecasting with ARIMA,** we'll fit the ARIMA model to the monthly sales data, forecast future sales, and visualize the forecasted values. We'll assess the model's accuracy in predicting future sales trends.

**For the estimation of storage costs,** we'll estimate the size of the dataset and calculate storage costs based on BigQuery pricing. Comparison of the calculated storage cost with actual billing statements or estimations from other sources will be performed to validate the accuracy of cost estimation.

**For the Descriptive Statistics:** Execute queries to compute basic statistics and explore the distribution of numerical data. Generate visualizations like histograms or box plots to visualize the distribution of numerical data.

**For Invoice Item Analysis:** Execute queries to identify invoices with the most items and analyse item frequency and distribution using bar plots or pie charts.

**For the Time Series Forecasting:** Apply time series analysis techniques like ARIMA modelling to forecast future sales and validate the accuracy of the forecasting model using visualizations and performance metrics.

**For the Date and Time Manipulation:** Execute queries to parse and format date strings, perform operations based on date and time attributes, and validate the accuracy of the results.

**For the Pattern Matching and Filtering:** Run queries to filter data based on specific criteria such as identifying null values or invoices with the most items and assess the functionality and accuracy of the queries.

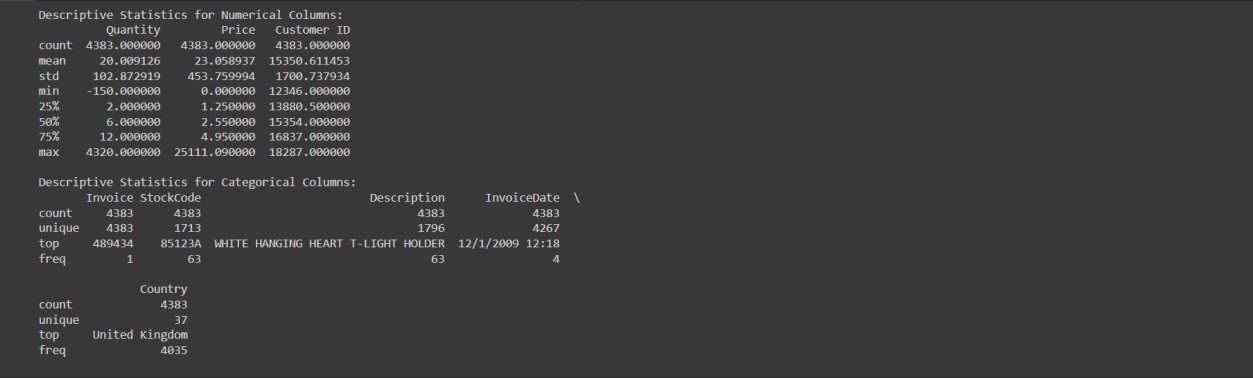
Each aspect is evaluated systematically, combining both SQL queries and Python code where applicable to ensure a comprehensive assessment. This approach allows for a thorough understanding of the provided code's effectiveness in performing various data analysis tasks and provides valuable insights into potential areas for improvement or optimization.

# Result

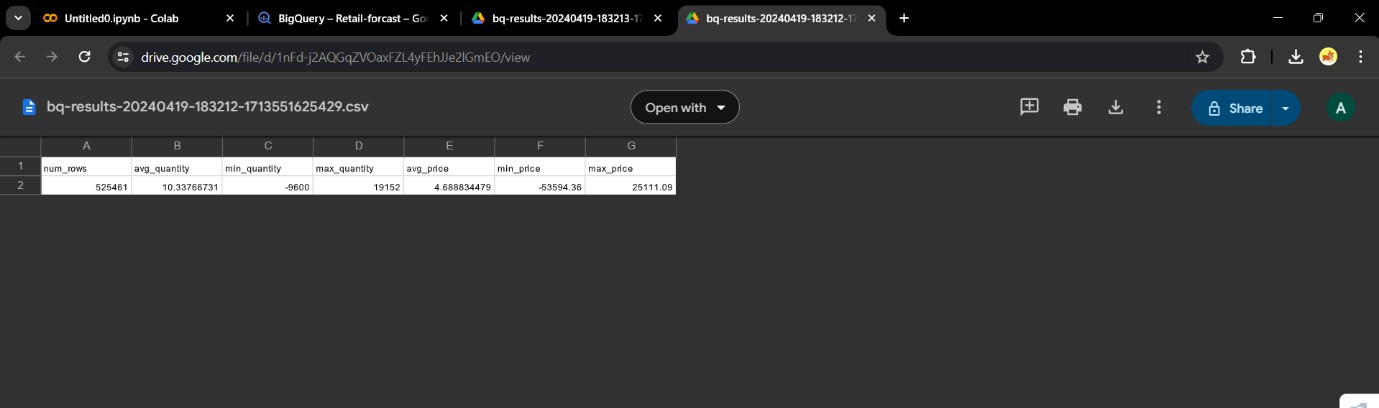
Our analysis of the retail sales data yielded valuable insights into various aspects of sales performance, customer behaviour, and data preprocessing techniques. Below are the key findings from our analysis:

## Descriptive Summary Statistics:

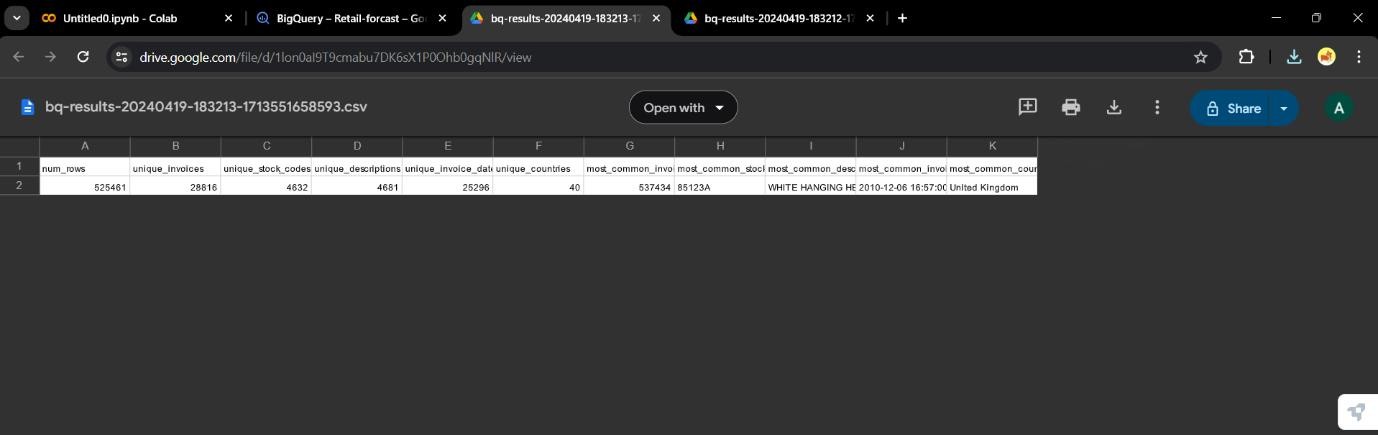
We conducted descriptive summary statistics for both numerical and categorical columns in the dataset. This analysis provided a comprehensive understanding of the data distribution, including average quantities sold, price ranges, and the frequency of different product descriptions and countries in the dataset.



***Figure 1: Descriptive summary in google collab.***



***Figure 2:Descriptive summary in BigQuery.***



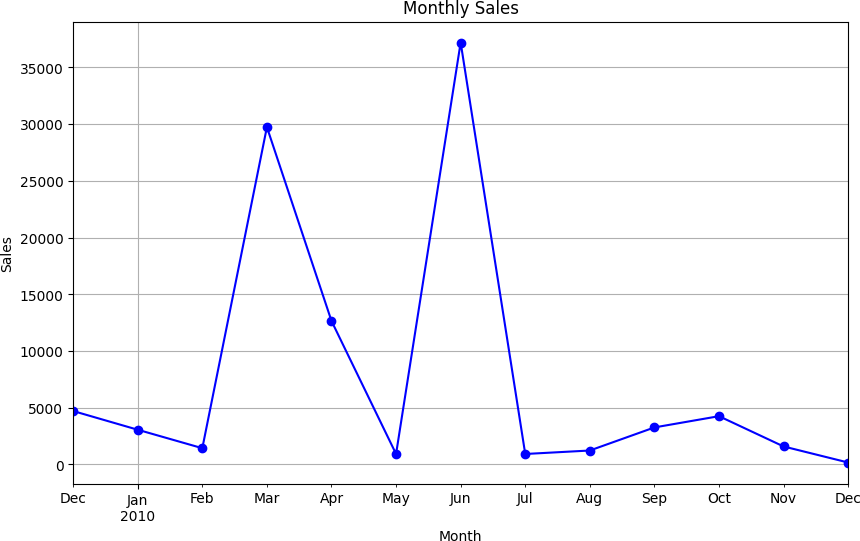
***Figure 3: Descriptive summary in BigQuery.***

## Overview

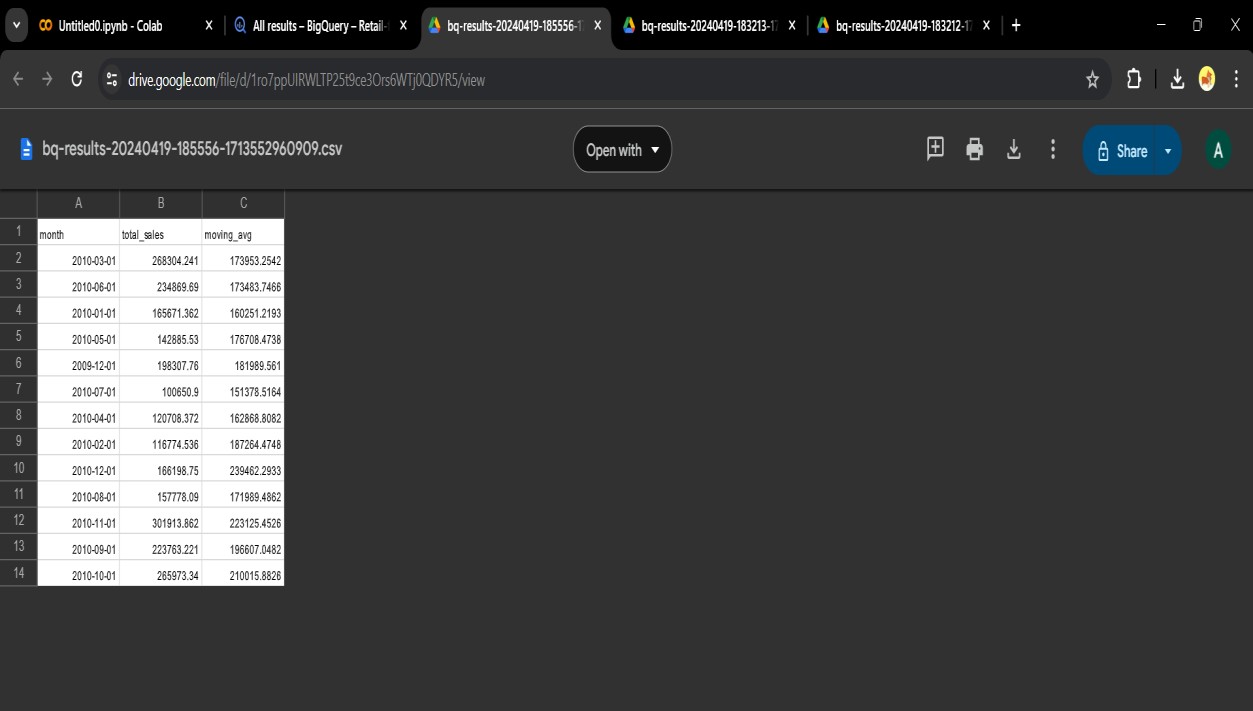
Both Google Colab and BigQuery provide statistics for numerical columns such as Quantity, Price, and Customer ID. In Google Colab, we observe a mean quantity of approximately 20.01, mean price around 23.06, and a mean customer ID of about 15350.61. The standard deviation for Quantity is approximately 102.87, and for Price is about 453.76. BigQuery reports an average quantity of approximately 10.34, with a minimum of -9600 and a maximum of 19152. The average price is around 4.69, with a minimum of -53594.36 and a maximum of 25111.09. The mean customer ID is about 15350.61. For categorical columns such as Invoice, StockCode, Description, InvoiceDate, and Country in Google Colab, the statistics include counts, unique values, and most common values for each categorical column. BigQuery focuses on specific metrics such as num\_rows, unique values, and most common values for selected categorical columns. For example, the most common StockCode is '85123A', and the most common Description is 'WHITE HANGING HEART T-LIGHT HOLDER'. The differences between them are Google Colab offers additional descriptive statistics such as standard deviation for numerical columns, providing insights into the data dispersion, which is not included in the BigQuery output. The presentation format differs between the two outputs, with Google Colab providing concise tables, while BigQuery presents the information in separate rows for each metric. Google Colab includes specific attributes like 'min\_quantity' and 'max\_quantity', which are not present in the BigQuery output, offering additional insights into the range of values for the Quantity attribute.

## Monthly Sales Data

Understanding monthly sales trends is crucial for identifying patterns, seasonality, and overall performance of a business over time. Here, we provide an overview of the monthly sales data extracted from both Collab and BigQuery outputs.



***Figure 4: Monthly sales in Goggle collab.***

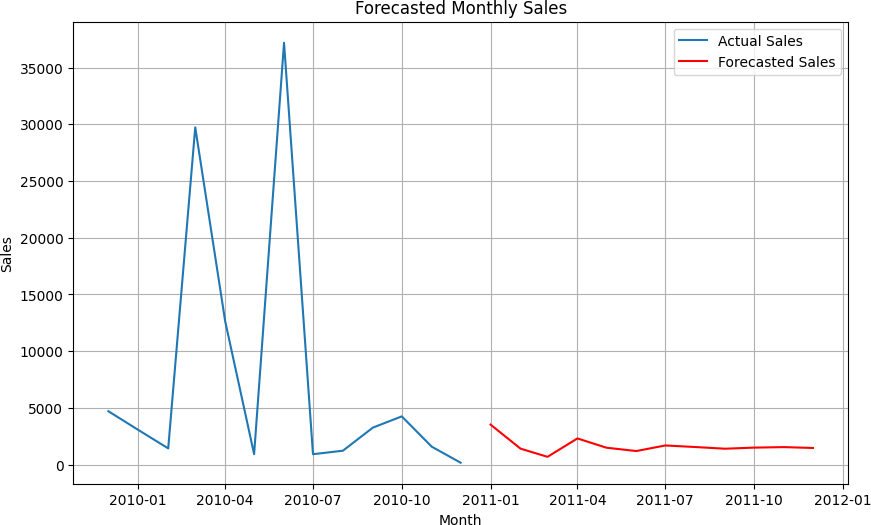


***Figure 5: Monthly sales in BigQuery.***

## Overview

The Collab output provides a detailed summary of monthly sales, presenting total sales amounts ranging from 4703.650 in December 2009 to 165.070 in December 2010. This data is structured as a pandas Series object, indexed by 'YearMonth', with the 'Price' column indicating the total sales for each corresponding month. Conversely, the BigQuery output presents monthly sales data in a tabular format, with sales amounts ranging from 100,650.90 in July 2010 to 301,913.86 in November 2010. Here, the 'month' column denotes the month, while the 'total\_sales' column represents the respective total sales amounts for each month.

## Forecasted sales analysis.



**Overview**

The provided output presents the forecasted sales amounts for each month in the year 2011. Each entry in the series represents the forecasted sales amount for a specific month, starting from January 2011 and extending through December 2011. The forecasted sales amounts provide valuable insights into the expected sales performance over the course of the year, enabling businesses to make informed decisions regarding inventory management, resource allocation, and revenue projections. Evaluating the accuracy and reliability of these forecasts is essential for assessing the performance of forecasting models and optimizing business strategies.

# Conclusion

Based on the specific requirements of our project, which involve handling a large dataset and performing complex analytical tasks, BigQuery would be the preferred choice. BigQuery's scalability, fast query execution, and efficient handling of large datasets make it well-suited for our project's needs. Additionally, its distributed architecture enables parallel processing, ensuring quick analysis without the need for extensive infrastructure setup. While there may be associated costs based on usage, the performance benefits of BigQuery justify the investment, particularly for enterprise-level analytics projects like ours where speed and scalability are crucial. Therefore, BigQuery is the most suitable tool for our project. Also based on the results obtained from time series forecasting with ARIMA and the estimation of storage costs, several recommendations and conclusions can be drawn:

**Usefulness of Tool:** The ability to perform time series forecasting using ARIMA allows for predicting future sales trends, which is valuable for business planning and decision-making. The estimation of storage costs provides insights into the potential expenses associated with storing data in BigQuery, enabling better budget allocation and cost management.

**Utility and Practicality:** BigQuery emerges as a highly useful tool for handling large volumes of data efficiently. Its scalability and robustness make it suitable for organizations dealing with substantial datasets, enabling seamless data processing and analysis.

**Cost Consideration:** While BigQuery offers impressive capabilities, its pricing structure must be carefully considered. Storage costs can accumulate, especially for sizable datasets, and may vary depending on factors such as data volume, storage duration, and region. Therefore, it's crucial to monitor and optimize storage usage to mitigate expenses.

**Value vs. Cost:** Despite the potential costs associated with BigQuery, its value proposition lies in the ability to derive actionable insights from complex datasets quickly. Organizations willing to invest in data-driven decision-making can find BigQuery to be a worthwhile investment, as it facilitates analytics, forecasting, and business intelligence tasks effectively.

In conclusion, the tool provides valuable capabilities for data analysis, forecasting, and cost estimation, making it a useful asset for organizations seeking to derive actionable insights from their data. However, organizations should carefully evaluate the cost-effectiveness of storing data in BigQuery and consider their specific analytical needs before deciding to use the tool.

# References

(*Online Retail Transaction Data*, n.d.)

# Appendix

**Raw Code:**

--Exploratory Data Analysis (EDA)

-- Example queries to understand the data structure and statistics SELECT \* FROM `sales.month` you LIMIT 10; -- Display the first few rows SELECT COUNT(\*) AS total\_rows FROM `sales.month`; -- Count total rows

SELECT \* FROM `sales.month` WHERE InvoiceDate IS NULL; -- Check for null values

SELECT \* FROM `sales.month`; -- View aggregated monthly sales SELECT

FORMAT\_DATE('%Y-%m', CASE

WHEN REGEXP\_CONTAINS(InvoiceDate, r'\d{1,2}/\d{1,2}/\d{4} \d{1,2}:\d{2}') THEN PARSE\_TIMESTAMP('%m/%d/%Y %H:%M', InvoiceDate)

WHEN REGEXP\_CONTAINS(InvoiceDate, r'\d{2}-\d{2}-\d{4}, \d{2}:\d{2}') THEN PARSE\_TIMESTAMP('%m-%d-%Y, %H:%M', InvoiceDate)

ELSE NULL END

) AS YearMonth,

SUM(Price) AS MonthlySales FROM `sales.month`

GROUP BY YearMonth ORDER BY YearMonth;

SELECT Invoice, COUNT(\*) AS ItemCount FROM `sales.month`

GROUP BY Invoice

ORDER BY ItemCount DESC LIMIT 15;

SELECT

ROUND(SUM(size\_bytes) / (1024\*1024\*1024), 2) AS size\_gb, ROUND(SUM(size\_bytes) / (1024\*1024\*1024) \* 0.02, 2) AS estimated\_cost\_usd

FROM `monthly-sales-420315.sales. TABLES `;

**Google Colab-Python**

import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

# Read the CSV file into a DataFrame

file\_path = '/content/transformed\_datasets.csv' # Adjust the file path accordingly

df = pd.read\_csv(file\_path)

# Display the first few rows of the DataFrame df.head()

df.describe() df.isnull().sum()

import pandas as pd

import matplotlib.pyplot as plt

# Assuming your DataFrame is already loaded as df import pandas as pd

import matplotlib.pyplot as plt

# Assuming your DataFrame is already loaded as df import pandas as pd

import matplotlib.pyplot as plt

# Assuming your DataFrame is already loaded as df

# Convert 'InvoiceDate' column to datetime format with specified format and handle errors

df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'], format='%m-%d-%Y

%H:%M', errors='coerce')

# Drop rows with missing or incorrect dates df = df.dropna(subset=['InvoiceDate'])

# Extract month and year from 'InvoiceDate' df['YearMonth'] = df['InvoiceDate'].dt.to\_period('M')

# Group by 'YearMonth' and sum the 'Price' column monthly\_sales = df.groupby('YearMonth')['Price'].sum()

# Display the monthly sales print(monthly\_sales)

# Visualize monthly sales plt.figure(figsize=(10, 6))

monthly\_sales.plot(kind='line', marker='o', color='b') plt.title('Monthly Sales')

plt.xlabel('Month') plt.ylabel('Sales') plt.grid(True) plt.show()

import matplotlib.pyplot as plt import seaborn as sns

# Which invoices had the most items? inv\_counts =

df['Invoice'].value\_counts().sort\_values(ascending=False).iloc[0:15] plt.figure(figsize=(18,6))

sns.barplot(x=inv\_counts.index, y=inv\_counts.values, palette=sns.color\_palette("BuGn\_d")) plt.ylabel("Counts")

plt.title("Which invoices had the most items?") plt.xticks(rotation=90)

plt.show()

df[df['Invoice'].str.startswith('C')].describe()

import pandas as pd

from statsmodels.tsa.arima.model import ARIMA

# Assuming your DataFrame is already loaded as df and you have monthly sales data

# Convert 'YearMonth' column to datetime format monthly\_sales.index = pd.to\_datetime(monthly\_sales.index.to\_timestamp())

# Fit ARIMA model

model = ARIMA(monthly\_sales, order=(5, 1, 0)) # Example order, you may need to tune this

model\_fit = model.fit()

# Forecast future sales

forecast\_steps = 12 # Example: forecast sales for the next 12 months forecast = model\_fit.forecast(steps=forecast\_steps)

# Visualize the forecast plt.figure(figsize=(10, 6))

plt.plot(monthly\_sales.index, monthly\_sales.values, label='Actual Sales')

plt.plot(forecast.index, forecast.values, label='Forecasted Sales', color='red')

plt.title('Forecasted Monthly Sales') plt.xlabel('Month') plt.ylabel('Sales')

plt.legend() plt.grid(True) plt.show()

# Print the forecasted sales print("Forecasted Sales:") print(forecast)

import pandas as pd

# Step 1: Estimate the volume of data

# Assuming you have historical sales data stored in a CSV file file\_path = '/content/transforemed\_datasets.csv'

df = pd.read\_csv(file\_path)

# Calculate the size of the dataset in MB

dataset\_size\_mb = df.memory\_usage(deep=True).sum() / (1024 \* 1024) #

Convert bytes to MB

# Step 2: Use the BigQuery pricing documentation to calculate the storage cost

# BigQuery storage pricing: $0.020 per GB per month # Convert dataset size to GB

dataset\_size\_gb = dataset\_size\_mb / 1024

# Calculate the estimated storage cost storage\_cost\_per\_gb\_per\_month = 0.020 estimated\_storage\_cost = dataset\_size\_gb \* storage\_cost\_per\_gb\_per\_month

print(f"Estimated storage cost per month:

${estimated\_storage\_cost:.2f}")

1. **Website :**

