Title : Sales Forecasting for E-Commerce Platform

Submitted by: Anika Thasin Mim – 0796557

And Sabrina Farzana - 0792500

1. 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.

2. Description

2.1 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.

2.2 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.

2.3 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.

2.4 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.

2.5 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.

2.6 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.

2.7 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.

3. 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:

3.1 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.

3.2 Dataset Loading and Preprocessing

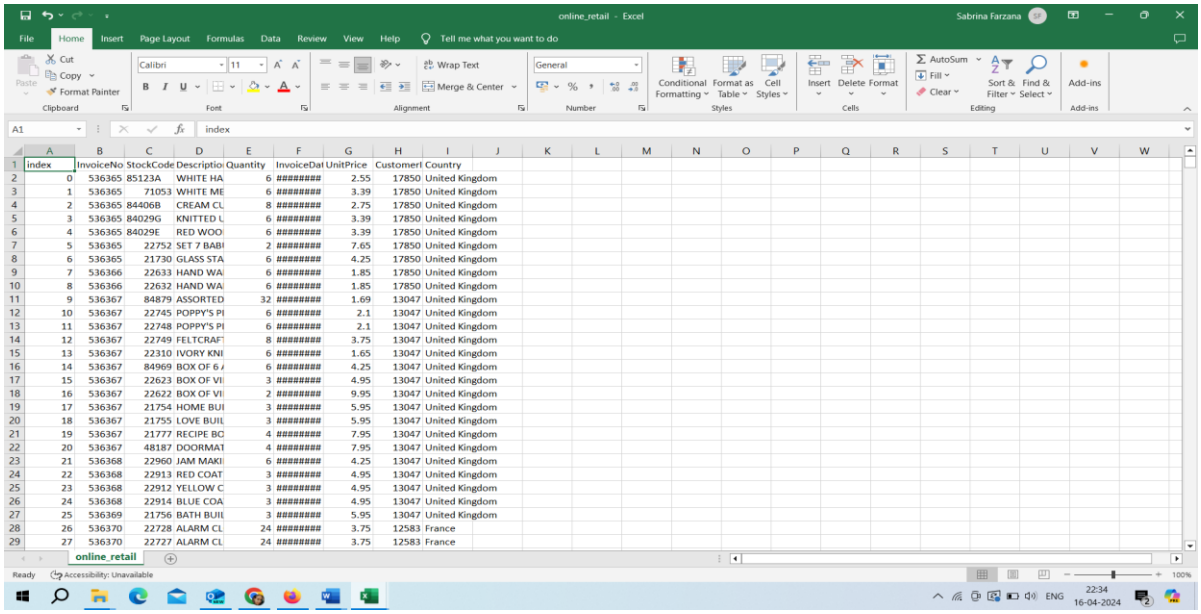
The dataset was initially loaded into Google BigQuery, a powerful and scalable data warehouse platform, to facilitate efficient data processing. Several preprocessing steps were performed to ensure data quality and integrity before analysis.

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.

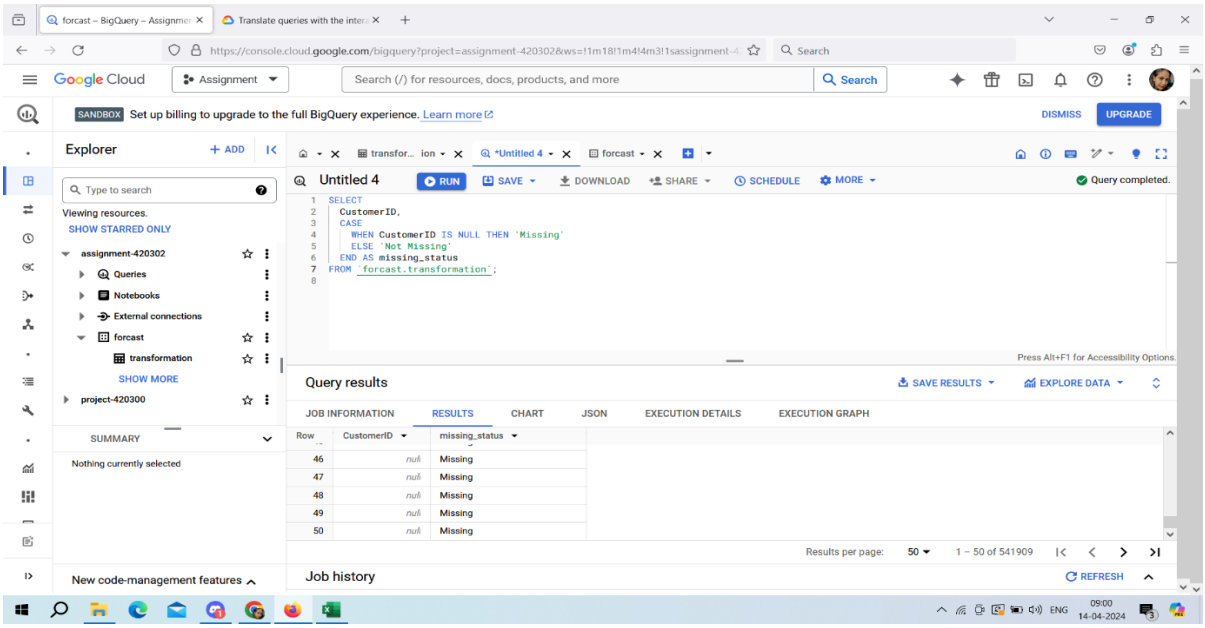
Handling Missing Values: Queries were executed to identify and remove records with missing values in critical attributes such as InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, and Country. This step ensured the completeness of the dataset and avoided potential biases in the analysis.

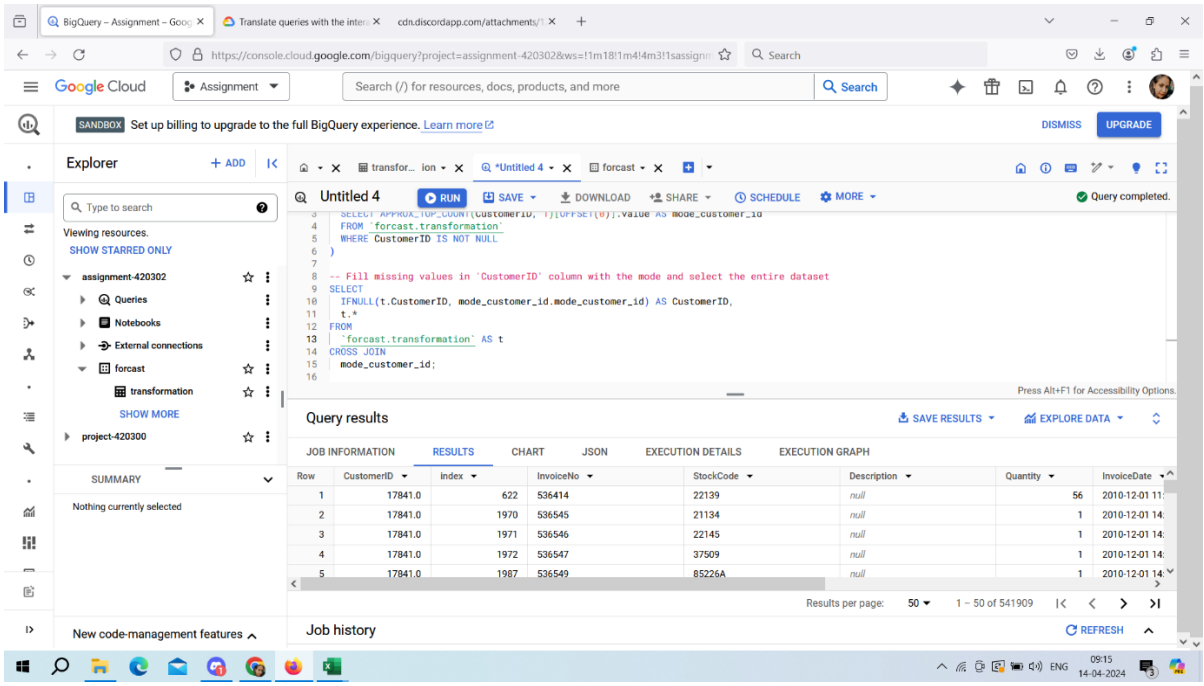
Removing Duplicate Entries: Duplicate entries were detected and eliminated using SQL queries, ensuring that each transaction was represented only once in the dataset. This helped maintain data accuracy and consistency throughout the analysis process.

Here is the Raw Data,

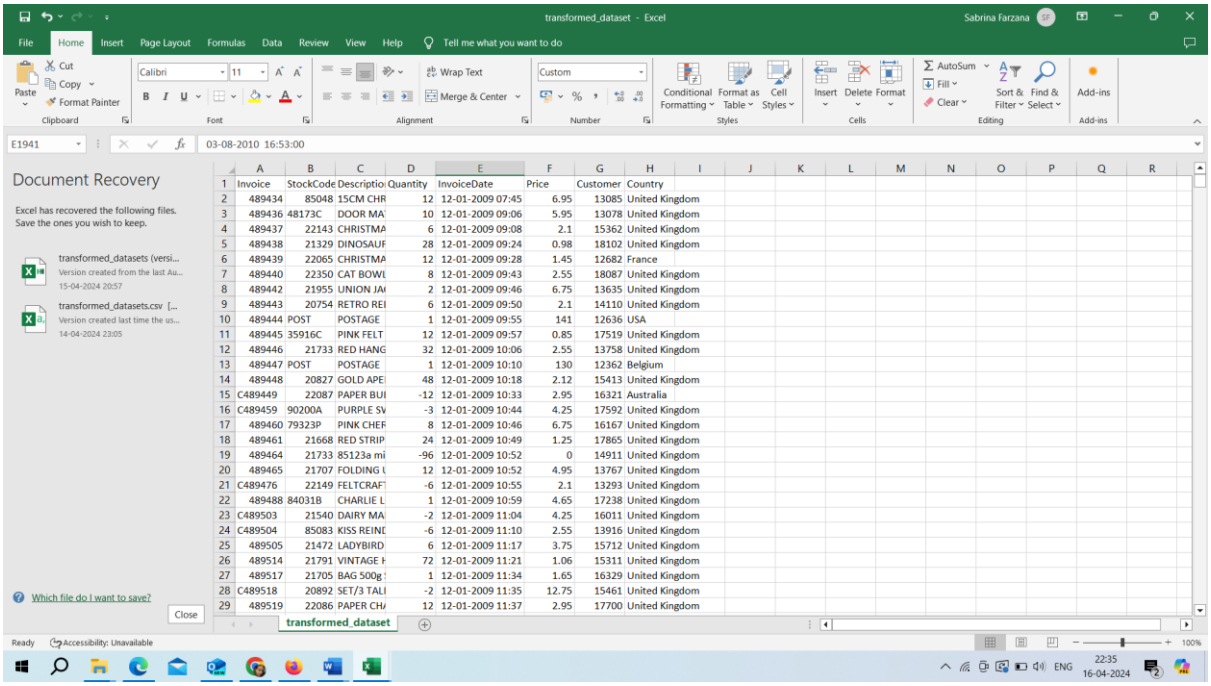


index	InvoiceNo	StockCode	Descriptio	Quantity	InvoiceDat	UnitPrice	Customerl	Country
0	536365	85123A	WHITE HA	6	########	2.55	17850	United Kingdom
1	536365	71053	WHITE ME	6	########	3.39	17850	United Kingdom
2	536365	84406B	CREAM CU	8	########	2.75	17850	United Kingdom
3	536365	84029G	KNITTED U	6	########	3.39	17850	United Kingdom
4	536365	84029E	RED WOO	6	########	3.39	17850	United Kingdom
5	536365	22752	SET 7 BAB	2	########	7.65	17850	United Kingdom
6	536365	21730	GLASS STA	6	########	4.25	17850	United Kingdom
7	536366	22633	HAND WA	6	########	1.85	17850	United Kingdom
8	536366	22632	HAND WA	6	########	1.85	17850	United Kingdom
9	536367	84879	ASSORTED	32	########	1.69	13047	United Kingdom
10	536367	22745	POPPY'S PI	6	########	2.1	13047	United Kingdom
11	536367	22748	POPPY'S PI	6	########	2.1	13047	United Kingdom
12	536367	22749	FELTCRAFT	8	########	3.75	13047	United Kingdom
13	536367	22310	IVORY KNI	6	########	1.65	13047	United Kingdom
14	536367	84969	BOX OF 6 A	6	########	4.25	13047	United Kingdom
15	536367	22623	BOX OF VI	3	########	4.95	13047	United Kingdom
16	536367	22622	BOX OF VI	2	########	9.95	13047	United Kingdom
17	536367	21754	HOME BUI	3	########	5.95	13047	United Kingdom
18	536367	21755	LOVE BUIL	3	########	5.95	13047	United Kingdom
19	536367	21777	RECIPE BO	4	########	7.95	13047	United Kingdom
20	536367	48187	DOORMAT	4	########	7.95	13047	United Kingdom
21	536368	22960	JAM MAKI	6	########	4.25	13047	United Kingdom
22	536368	22913	RED COAT	3	########	4.95	13047	United Kingdom
23	536368	22912	YELLOW C	3	########	4.95	13047	United Kingdom
24	536368	22914	BLUE COA	3	########	4.95	13047	United Kingdom
25	536369	21756	BATH BUIL	3	########	5.95	13047	United Kingdom
26	536370	22728	ALARM CL	24	########	3.75	12583	France
27	536370	22727	ALARM CL	24	########	3.75	12583	France





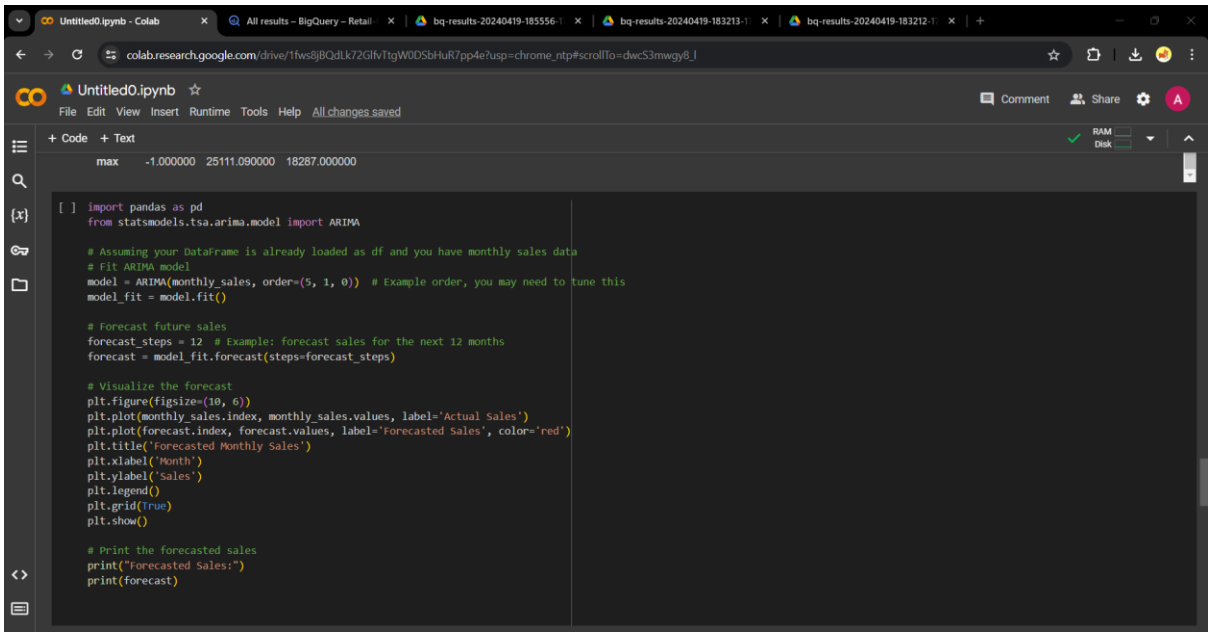
After Cleaning the dataset,



Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer	Country
489434	85048	15CM CHR	12	12-01-2009 07:45	6.95	13085	United Kingdom
489436	48173C	DOOR MA	10	12-01-2009 09:06	5.95	13078	United Kingdom
489437	22143	CHRISTMA	6	12-01-2009 09:08	2.1	15362	United Kingdom
489438	21329	DINOSAUF	28	12-01-2009 09:24	0.98	18102	United Kingdom
489439	22065	CHRISTMA	12	12-01-2009 09:28	1.45	12682	France
489440	22350	CAT BOWL	8	12-01-2009 09:43	2.55	18087	United Kingdom
489442	21955	UNION JA	2	12-01-2009 09:46	6.75	13635	United Kingdom
489443	20754	RETRO RE	6	12-01-2009 09:50	2.1	14110	United Kingdom
489444	POST	POSTAGE	1	12-01-2009 09:55	141	12636	USA
489445	35916C	PINK FELT	12	12-01-2009 09:57	0.85	17519	United Kingdom
489446	21733	RED HANG	32	12-01-2009 10:06	2.55	13758	United Kingdom
489447	POST	POSTAGE	1	12-01-2009 10:10	130	12362	Belgium
489448	20827	GOLD APE	48	12-01-2009 10:18	2.12	15413	United Kingdom
C489449	22087	PAPER BU	-12	12-01-2009 10:33	2.95	16321	Australia
C489459	90200A	PURPLE SV	-3	12-01-2009 10:44	4.25	17592	United Kingdom
489460	79323P	PINK CHE	8	12-01-2009 10:46	6.75	16167	United Kingdom
489461	21668	RED STRIP	24	12-01-2009 10:49	1.25	17865	United Kingdom
489464	21733	85123a mi	-96	12-01-2009 10:52	0	14911	United Kingdom
489465	21707	FOLDING	12	12-01-2009 10:52	4.95	13767	United Kingdom
C489476	22149	FELTCRAF	-6	12-01-2009 10:55	2.1	13293	United Kingdom
489488	84031B	CHARLIE L	1	12-01-2009 10:59	4.65	17238	United Kingdom
C489503	21540	DAIRY MA	-2	12-01-2009 11:04	4.25	16011	United Kingdom
C489504	85083	KISS REIND	-6	12-01-2009 11:10	2.55	13916	United Kingdom
489505	21472	LADYBIRD	6	12-01-2009 11:17	3.75	15712	United Kingdom
489514	21791	VINTAGE H	72	12-01-2009 11:21	1.06	15311	United Kingdom
489517	21705	BAG 500g	1	12-01-2009 11:34	1.65	16329	United Kingdom
C489518	20892	SET/3 TALI	-2	12-01-2009 11:35	12.75	15461	United Kingdom
489519	22086	PAPER CH	12	12-01-2009 11:37	2.95	17700	United Kingdom

3.3 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.



```
max     -1.000000  25111.090000  18287.000000
```

```python
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
# 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)
```

3.4 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.

3.5 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.

4. 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:

4.1 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.

```
Descriptive Statistics for Numerical Columns:
          Quantity         Price   Customer ID
count  4383.000000   4383.000000   4383.000000
mean     20.009126     23.058937  15350.611453
std     102.872919    453.759994   1700.737934
min    -150.000000      0.000000  12346.000000
25%       2.000000      1.250000  13880.500000
50%       6.000000      2.550000  15354.000000
75%      12.000000      4.950000  16837.000000
max    4320.000000  25111.090000  18287.000000

Descriptive Statistics for Categorical Columns:
       Invoice StockCode                         Description      InvoiceDate  \
count     4383      4383                                4383             4383
unique    4383      1713                                1796             4267
top     489434    85123A  WHITE HANGING HEART T-LIGHT HOLDER  12/1/2009 12:18
freq         1        63                                  63                4

                Country
count              4383
unique               37
top     United Kingdom
freq              4035
```

Figure 1: Descriptive summary in google collab.

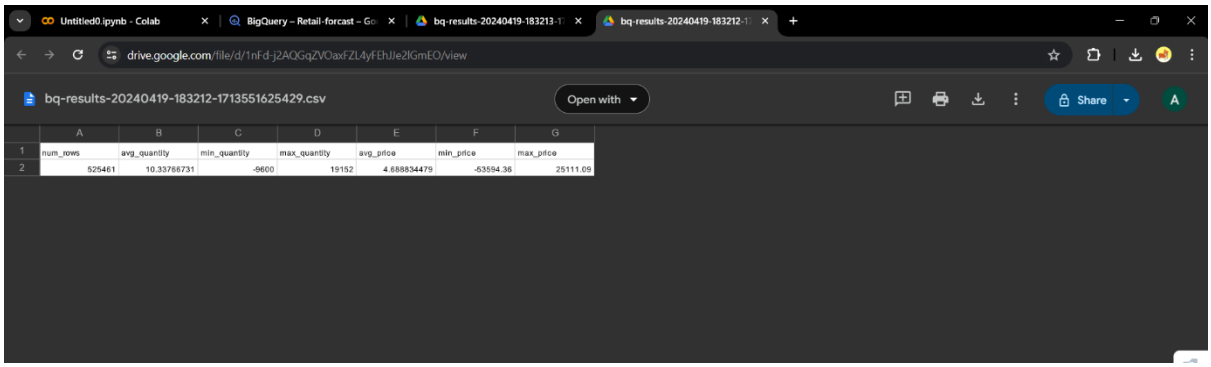


Figure 2:Descriptive summary in BigQuery.

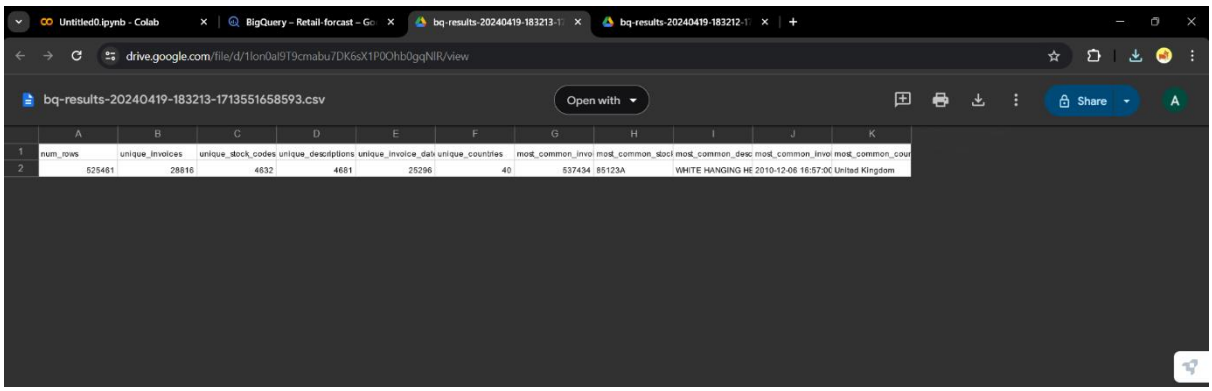


Figure 3: Descriptive summary in BigQuery.

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.

4.2 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.

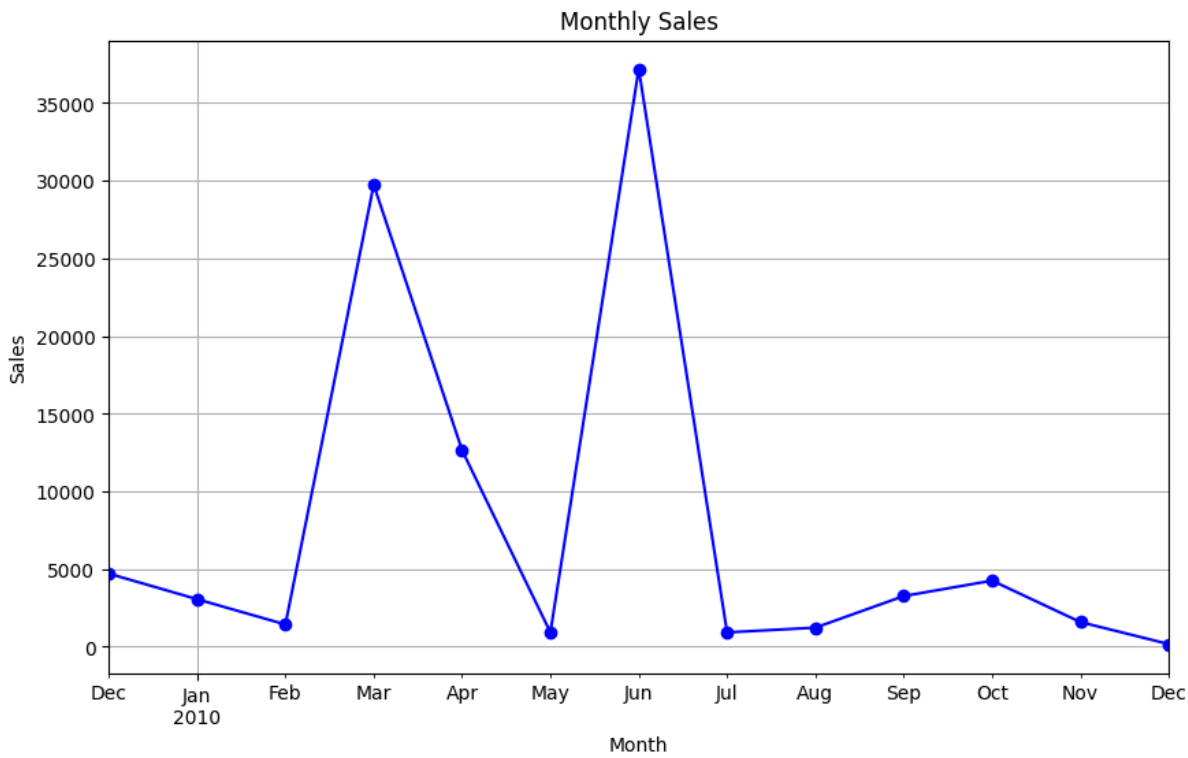
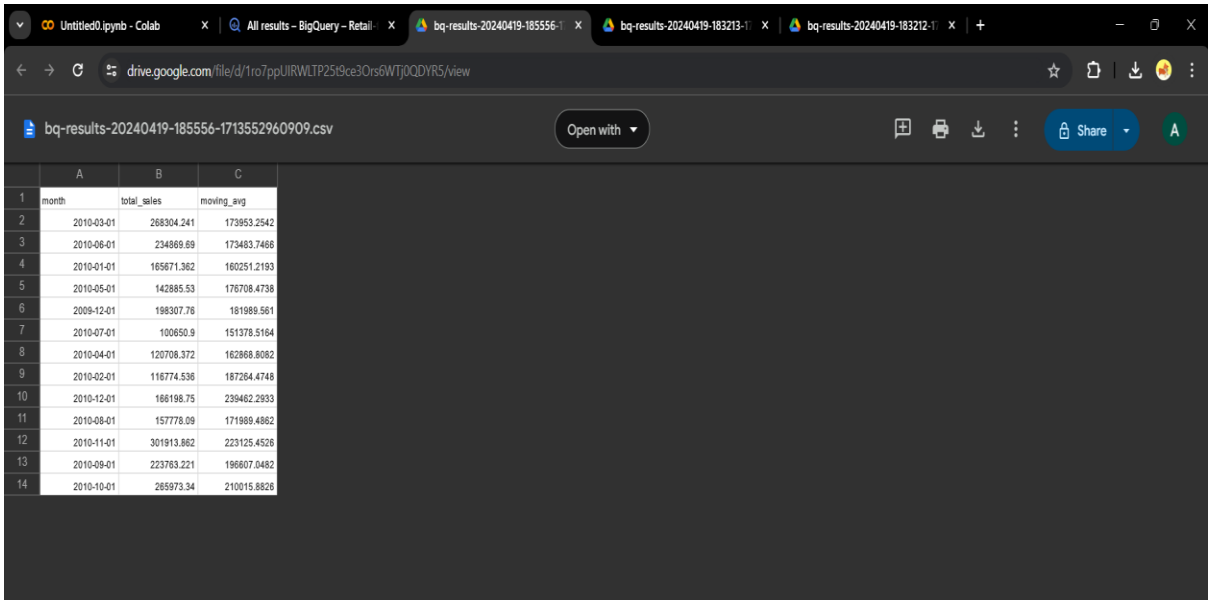


Figure 4: Monthly sales in Goggle collab.



bq-results-20240419-185556-1713552960909.csv

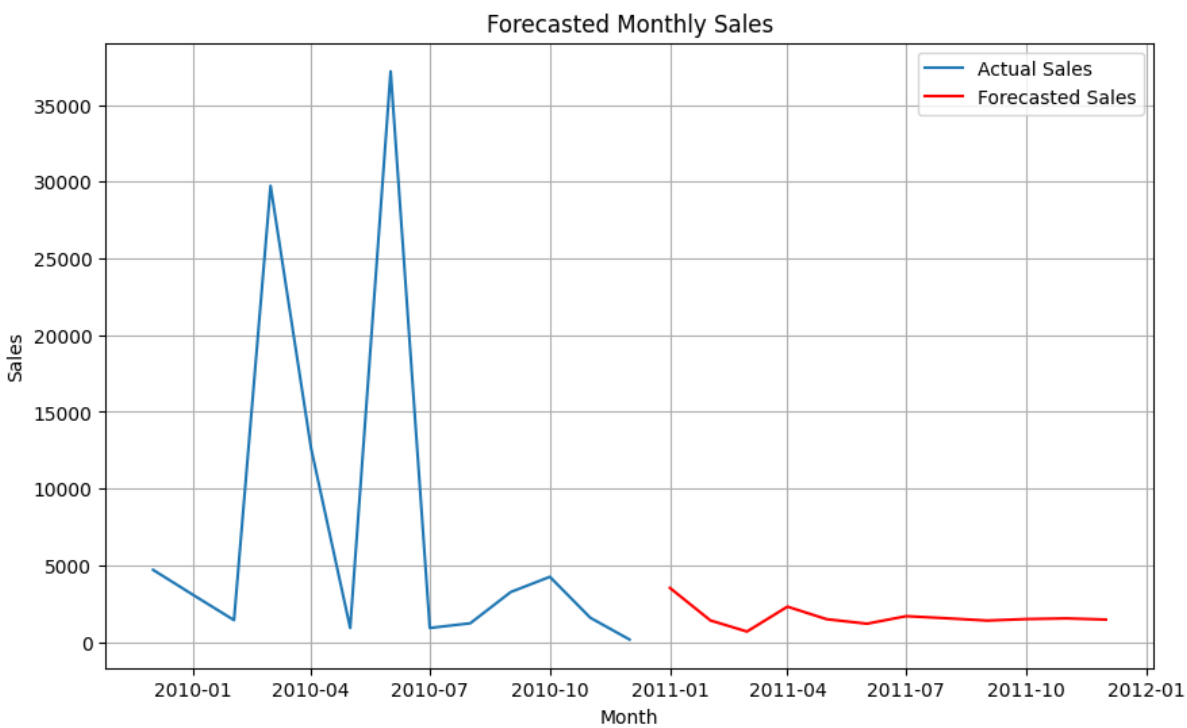
	A	B	C
1	month	total_sales	moving_avg
2	2010-03-01	268304.241	173953.2542
3	2010-06-01	234869.69	173483.7466
4	2010-01-01	165671.362	160251.2193
5	2010-05-01	142885.53	176708.4738
6	2009-12-01	198307.76	181989.561
7	2010-07-01	100650.9	151378.5164
8	2010-04-01	120708.372	162868.8082
9	2010-02-01	116774.536	187264.4748
10	2010-12-01	166198.75	239462.2933
11	2010-08-01	157778.09	171989.4862
12	2010-11-01	301913.862	223125.4526
13	2010-09-01	223763.221	196607.0482
14	2010-10-01	265973.34	210015.8826

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.

4.3 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.

5. 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.

6. References

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

7. 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

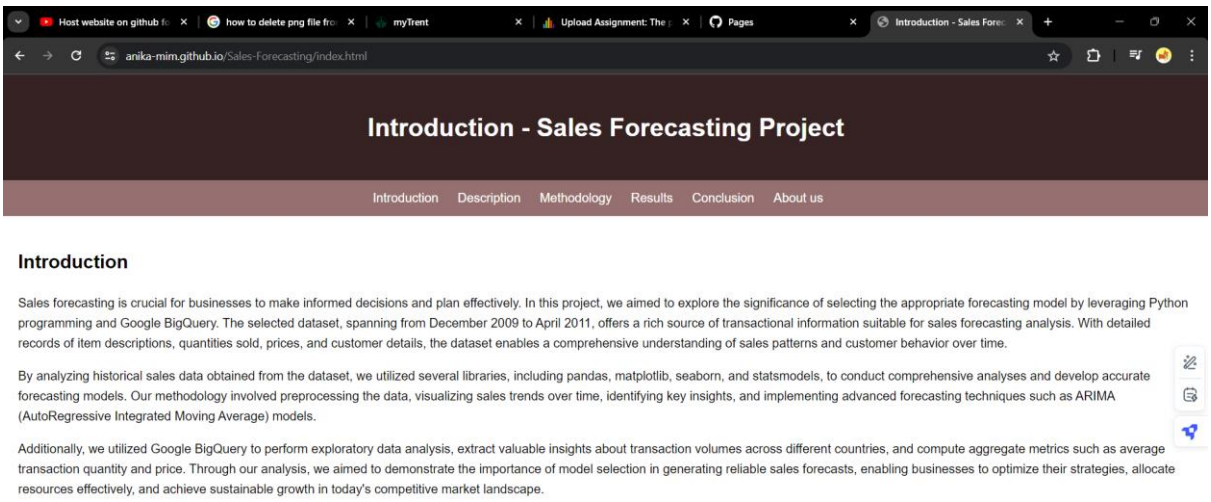
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Read the CSV file into a DataFrame
```

```
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}")
```

8. Website Link : https://anika-mim.github.io/Sales-Forecasting/about.html



anika-mim.github.io/Sales-Forecasting/des.html

Description - Sales Forecasting Project

Introduction Description Methodology Results Conclusion About Us

Description

The project utilized a combination of Google BigQuery, Jupyter Notebook, Pandas, Matplotlib, Seaborn, Statsmodels, and the AutoRegressive Integrated Moving Average (ARIMA) model to conduct sales forecasting analysis. Google BigQuery was employed for preprocessing the dataset, leveraging its SQL-like syntax and powerful processing capabilities for data cleaning, transformation, and aggregation tasks. Jupyter Notebook served as an interactive computing environment for data analysis, exploration, and documentation, facilitating collaborative code execution and visualization. Pandas provided robust data manipulation and analysis functionalities, while Matplotlib and Seaborn were utilized for creating insightful visualizations of sales data. Statsmodels enabled the estimation and analysis of statistical models, with focus on implementing the ARIMA model for forecasting future sales based on historical observations. Overall, the project aimed to optimize sales forecasting through a comprehensive analysis of sales data, leveraging a diverse set of tools and techniques for data manipulation, visualization, and modeling.

anika-mim.github.io/Sales-Forecasting/meth.html

Methodology - Sales Forecasting Project

Introduction Description Methodology Results Conclusion About us

Methodology

The methodology involves leveraging Google BigQuery for dataset loading and preprocessing, followed by systematic evaluation of various aspects related to sales forecasting. The dataset is initially loaded into BigQuery, where preprocessing steps such as handling missing values and removing duplicates are performed to ensure data quality. Subsequently, a series of experiments are designed to evaluate different aspects of the dataset and analysis techniques. SQL queries and Python code are used to conduct data retrieval, exploration, aggregation, summarization, time series forecasting, estimation of storage costs, descriptive statistics, invoice item analysis, date and time manipulation, and pattern matching and filtering. Through this methodology, the project aims to optimize sales forecasting and provide valuable insights for strategic decision-making in sales and inventory management.

anika-mim.github.io/Sales-Forecasting/res.html

Results - Sales Forecasting Project

Introduction Description Methodology Results Conclusion About us

Results

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. Overview The Google Colab 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,

anika-mim.github.io/Sales-Forecasting/con.html

Sales Forecasting Project

Introduction Description Methodology Results Conclusion About us

Conclusion

Based on the specific requirements of our project, which involve handling a large dataset and performing complex analytical tasks, BigQuery emerges as the preferred choice. Its 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.

Furthermore, 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.

anika-mim.github.io/Sales-Forecasting/about.html

About Us - Sales Forecasting Project

Introduction Description Methodology Results Conclusion About Us

About Us

We are a team of data enthusiasts embarking on a transformative journey through the realm of sales forecasting in the e-commerce domain.Our project, centered around the intricate art of predicting future sales, harnesses the power of Python programming and Google BigQuery to unravel the mysteries hidden within vast troves of transactional data.

In the rapidly evolving landscape of e-commerce, where every click and transaction holds the key to success, our project stands as a beacon of innovation and insight. By delving deep into the nuances of sales forecasting, we empower businesses to navigate the turbulent waters of the digital marketplace with confidence and clarity, paving the way for sustained growth and prosperity in an ever-changing world.

Contributors:

- Anika Thasin Mim - 0796557 (LinkedIn)
- Sabrina Farzana - 07925 (LinkedIn)

github.com/anika-mim/Sales-Forecasting

anika-mim / Sales-Forecasting

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Sales-Forecasting Public

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags Go to file Add file Code

anika-mim Add files via upload dd126ee · 11 minutes ago 5 Commits

README.md	Initial commit	2 hours ago
_con.html	Add files via upload	47 minutes ago
about.html	Add files via upload	47 minutes ago
con.html	Add files via upload	11 minutes ago
des.html	Add files via upload	47 minutes ago
forecasting.png	Add files via upload	33 minutes ago
index.html	Add files via upload	47 minutes ago
meth.html	Add files via upload	47 minutes ago
res.html	Add files via upload	47 minutes ago
time series.png	Add files via upload	43 minutes ago

README

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Deployments 5

github-pages 11 minutes ago