

Futurense Technologies |
13/06/2023

E-COMMERCE: CUSTOMER ANALYSIS

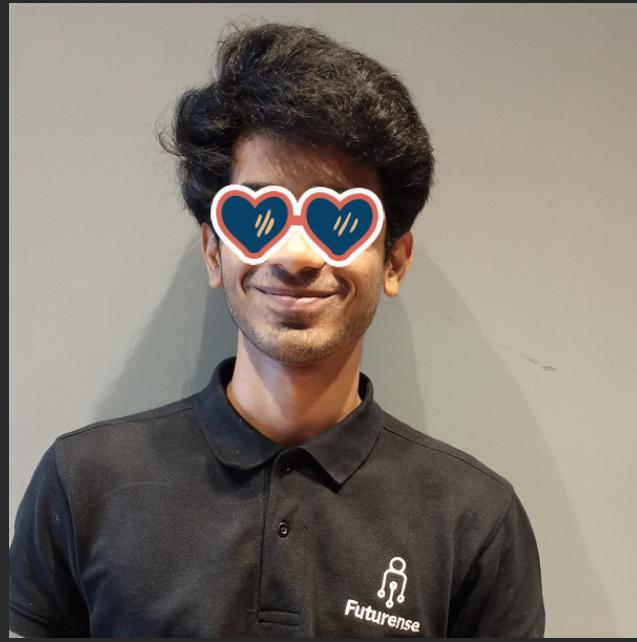
AZURE--PYSPARK-SPARK SQL-- SYNAPSE AND
MUCH MORE....



Agenda

- 
1. Team Introduction & Project Overview
 2. Data Description & Preprocessing
 3. Azure Pipelines and Analysis
 4. Visualization and Results
 5. Final Thoughts
-

Team Introduction



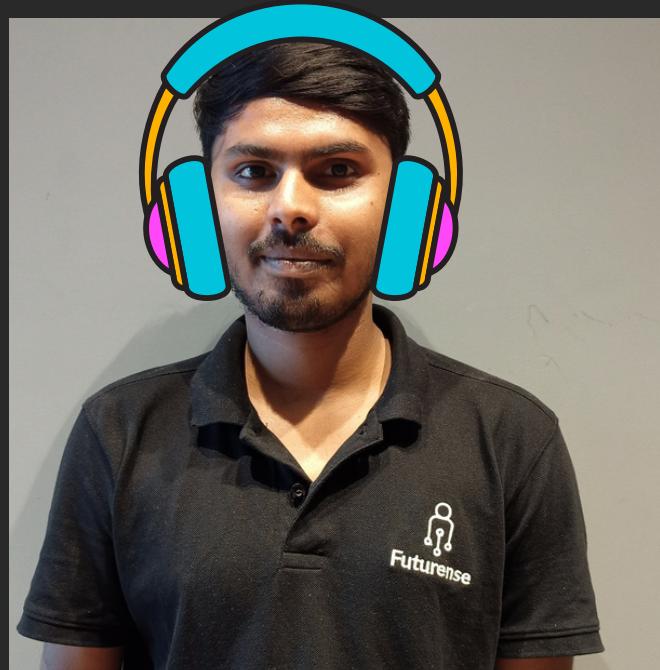
Nahush Dhavalikar



Venkat Sir DE Expert



Shriya Gandreti



Shivam Kumar



Jyoti Biswas



Sejal Singh



Deepak Mehta



Project Goals

- Analyze an e-commerce dataset to understand customer behavior
- Present the analysis results in a clear and concise manner, facilitating easy comprehension and decision-making.
- Enable the client to make informed decisions and take necessary actions to drive positive business outcomes.

Project Objectives

- Gain a comprehensive understanding of the provided e-commerce dataset.
- Identify any data quality issues, missing values, or anomalies that may impact the analysis.
- Utilize Azure platform to perform in-depth analysis of the preprocessed dataset.
- Leverage SQL queries and aggregations to derive valuable insights into client problems and potential solutions.
- Foster collaboration among team members by leveraging Azure Databricks and ADF to facilitate seamless data processing and sharing.
- Document the analysis process, methodologies, and key findings for future reference and knowledge sharing within the organization.

0110101000100111

Data Description

The e-commerce dataset consists of transactional data related to customer purchases.

Columns:

- **InvoiceNo**: A unique identifier for each invoice or transaction. It can include alphanumeric characters.
- **StockCode** : Unique identifier for the stock of the product.
- **Description**: A text description of the item or product purchased in the transaction.
- **Quantity**: The quantity or number of units purchased for a particular item in the transaction.
- **InvoiceDate**: The date and time when the invoice was generated or the transaction occurred. It includes information about both the date and time.
- **UnitPrice**: The price of a single unit of the product or item purchased. It is represented in the currency of the dataset.
- **CustomerID**: A unique identifier for each customer. It may contain alphanumeric characters or numeric IDs.
- **Country**: The country or region associated with the customer or transaction.

Data Size: 43.7 MB

Number Of Rows: 541909

Number of Columns: 8

Data Pre-processing

- Converted Date column from string to required data type, i.e. timestamp

```
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])
```

- Dropped the null values and duplicated rows

```
data=data.dropna(how='any')
data.drop_duplicates(inplace=True)
```

- Filtered the dataset to remove rows where Quantity is negative AND UnitPrice = 0.0
- Reduced memory usage by using data types that take less memory

```
data['Quantity'] = data['Quantity'].astype('int32')
data['UnitPrice'] = data['UnitPrice'].astype('float32')
data['CustomerID'] = data['CustomerID'].astype('int64')
```

Data

BEFORE

```
df.info()
```

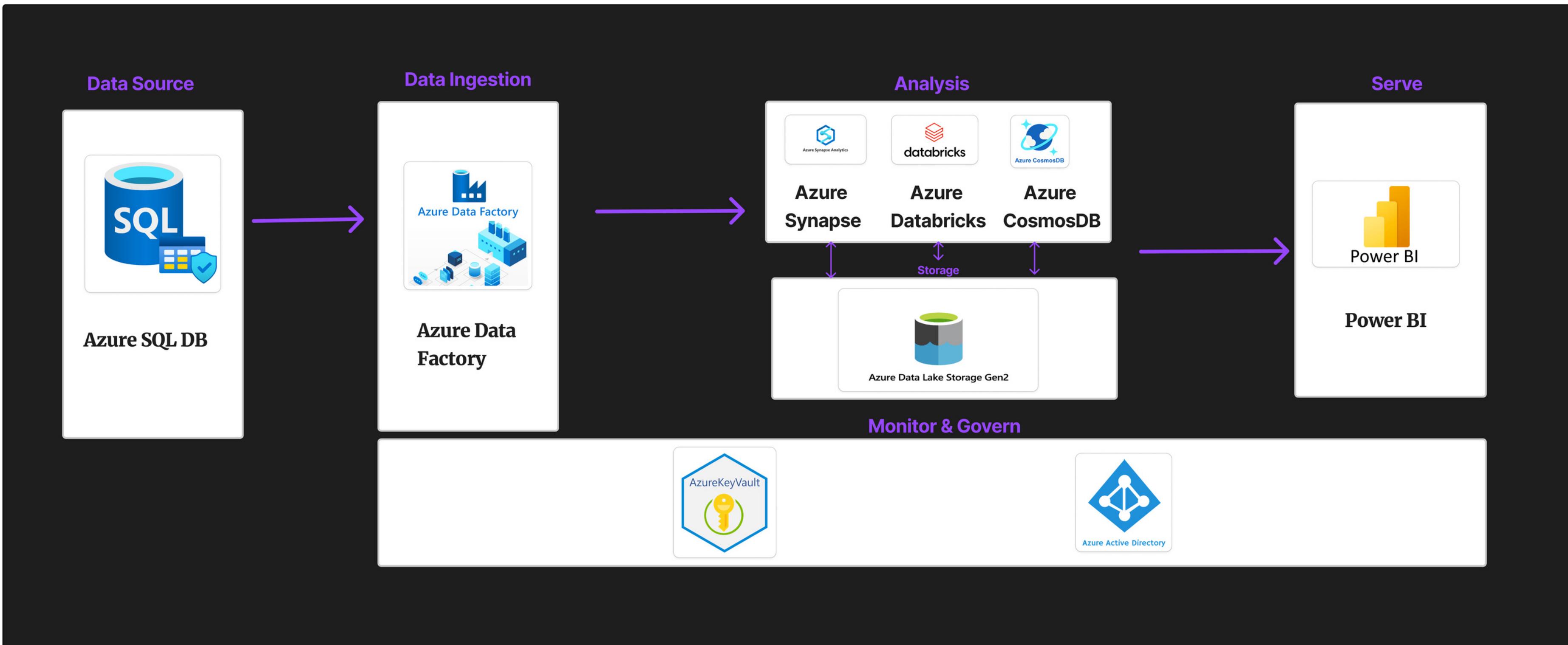
```
<class 'pandas.core.frame.DataFrame'>
Index: 397884 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   InvoiceNo    397884 non-null   object  
 1   StockCode     397884 non-null   object  
 2   Description   397884 non-null   object  
 3   Quantity      397884 non-null   float64 
 4   InvoiceDate   397884 non-null   datetime64[ns]
 5   UnitPrice     397884 non-null   float64 
 6   CustomerID    397884 non-null   float64 
 7   Country       397884 non-null   object  
dtypes: datetime64[ns](1), float64(3), object(4)
memory usage: 44+ MB
```

AFTER

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 392692 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   InvoiceNo    392692 non-null   object  
 1   StockCode     392692 non-null   object  
 2   Description   392692 non-null   object  
 3   Quantity      392692 non-null   int32  
 4   InvoiceDate   392692 non-null   datetime64[ns]
 5   UnitPrice     392692 non-null   float32 
 6   CustomerID    392692 non-null   int64  
 7   Country       392692 non-null   object  
dtypes: datetime64[ns](1), float32(1), int32(1), int64(1), object(4)
memory usage: 24.0+ MB
```

ARCHITECTURE



Azure Databricks



databricks

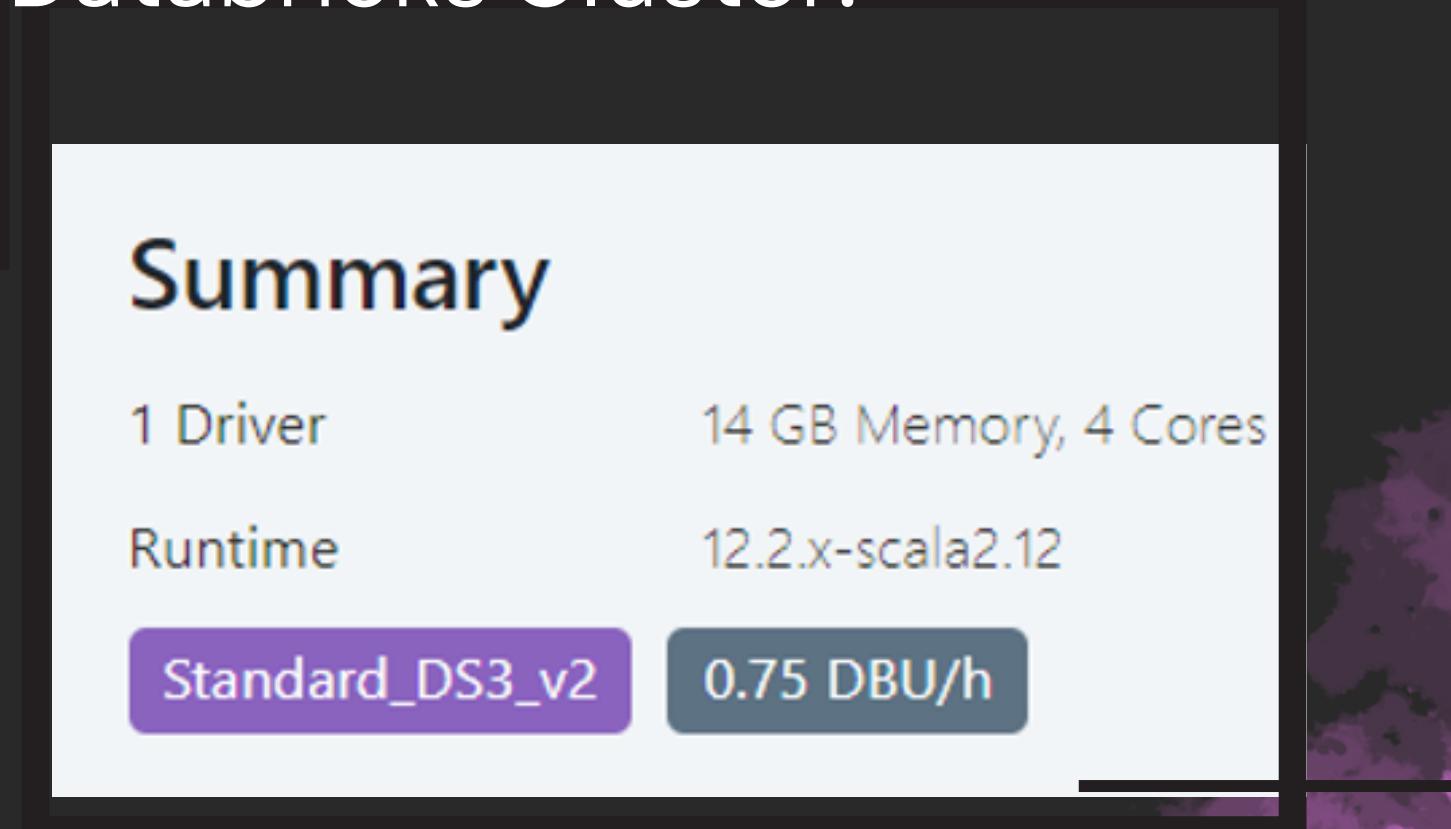
Analysis using Azure Databricks

Pipeline Used :

Azure SQL DB -----> Azure Databricks -----> ADLS Gen 2

SOURCE : Azure SQL DB
ANALYSIS PLATFORM : Azure Databricks
DESTINATION : ADLS Gen 2

Databricks Cluster:



STEPS

1. Created ADLS Storage account



2. Created Azure SQL DB Server



3. Created table in Azure SQL to store my dataset

A screenshot of the Azure portal's Query editor (preview) interface. The title bar shows "capstoneDB (sejal-capstone-server/capstoneDB) | Query editor (preview)". The left sidebar lists "Databases", "Tables", "Activity log", "Groups and roles", "Problems", "Diag started", and "Query editor (preview)". The main area shows a database named "capstoneDB (myadmin)" with a message: "Showing limited object explorer here. For full capability please click here to open Azure Data Studio." Below this is a "Tables" section with a "ecom" table. The "ecom" table has the following schema:

```
CREATE TABLE ecom(InvoiceID INT, StockCode VARCHAR(50), Description VARCHAR(500), InvoiceDate VARCHAR(25), UnitPrice FLOAT, CustomerID INT, Country VARCHAR(100));--drop table ecom
```

The "Query 1" tab is active, and the "Run" button is visible in the toolbar.

4. Created Azure Databricks Cluster inside Azure Databricks workspace

The screenshot shows the 'Create a cluster' interface in the Azure Databricks workspace. The left sidebar has a green header bar with the text 'Completed'. The main area displays the configuration for 'Shivam kumar's Cluster'. The 'Configuration' tab is selected, showing the following details:

- Policy:** Unrestricted
- Access mode:** Single user account (Shivam kumar (jhavakumar15499@...))
- Node type:** Standard_DS3_v2 (16 GB Memory, 4 Cores)
- Termination settings:** Terminate after 120 minutes of inactivity
- Tags:** No custom tags

The right side of the screen shows a summary of the cluster configuration:

Summary
1 Driver 16 GB Memory, 4 Cores
Runtime 12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12)
Standard_DS3_v2 0.75 DBUs/h

4. Created Azure Databricks Cluster inside Azure Databricks workspace

The screenshot shows the 'Create a cluster' interface in the Azure Databricks workspace. The cluster is named 'Shivam kumar's Cluster' and is in a 'Completed' state. The configuration includes:

- Policy:** Unrestricted
- Access mode:** Single user account (Shivam kumar (jhavakumar15499@...))
- Performance:** Databricks Runtime Version: 12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12). Option to 'Use Photon Acceleration' is available.
- Node type:** Standard_DS3_v2 (14 GiB Memory, 4 Cores)
- Termination settings:** Set to terminate after 120 minutes of inactivity.
- Tags:** No custom tags, with an option to automatically add tags.

A 'Summary' section on the right provides details: 1 Driver, 14 GiB Memory, 4 Cores, Runtime 12.2 LTS, Standard_DS3_v2, and 0.75 DBUs/h.

5. Used Azure Key Vault



to access Azure SQL



5. Used Azure Key Vault to access Azure SQL

a. Created Azure Key Vault

b. Created Secrets in Key Vault to contain Azure SQL Password, user name, host name etc.

The screenshot shows the 'Secrets' page of an Azure Key Vault named 'azurekvforCapstone'. A success message at the top states: 'The secret 'azure-sql-pw' has been successfully created.' The table below lists the secret details:

Name	Type	Status	Expiration date
azure-sql-pw	same as myadmin	✓ Enabled	

c. Created Databricks scoped credential in Databricks workspace

The screenshot shows the 'Create a secret scope' dialog box in the Databricks workspace. A confirmation message says: 'The secret scope named azurekv-scope has been added.' Below it, instructions say: 'Manage secrets in this scope in Azure KeyVault with manage principal = creator'. The dialog has an 'OK' button at the bottom right.

STEP 6:

Analysis in Databricks Notebook

a. Connecting Azure SQL with Databricks

```
jdbcHost = "sejal-capstone-server.database.windows.net"
jdbcPort = 1433
jbbcdB = "capstoneDB"
jdbcUser = "myadmin"
jdbcPW = dbutils.secrets.get("azurekv-scope", "azure-sql-pw")
jdbcDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver"

jdbcUrl = f"jdbc:sqlserver://{{jdbcHost}}:{{jdbcPort}};databaseName={{jbbcdB}};user={{jdbcUser}};password={{jdbcPW}}"
```

b. Loading data

```
ecom = spark.read.format("jdbc").option("url", jdbcUrl).option("dbtable", tableName).load()
```

STEP 6: (c)

Analysis in Databricks Notebook

PROBLEM STATEMENTS:

1. What is the total purchase bill for each customer?
2. Who are the customers from United Kingdom who have made purchases in July?
3. How many among customers are from France?
4. What products have Unit price of more than 10?

Analysis in Databricks Notebook

User story 1 solution:

```
# 1.Find the total price of the customer based on the customer id given. (price = quantity*Unit price )  
  
job1DF = spark.sql('SELECT CustomerID, ROUND(sum(Quantity * UnitPrice),3) as Total_Price FROM ecom \  
GROUP BY CustomerID ORDER BY ROUND(sum(Quantity * UnitPrice),3) DESC ')
```

CustomerID	Country	Year	Month
12748	United Kingdom	2011	7
12830	United Kingdom	2011	7
12833	United Kingdom	2011	7
12836	United Kingdom	2011	7
12839	United Kingdom	2011	7
12840	United Kingdom	2011	7
12841	United Kingdom	2011	7
12843	United Kingdom	2011	7
12853	United Kingdom	2011	7
12864	United Kingdom	2011	7
12875	United Kingdom	2011	7
12901	United Kingdom	2011	7
12910	United Kingdom	2011	7
12915	United Kingdom	2011	7
12916	United Kingdom	2011	7
12919	United Kingdom	2011	7
12921	United Kingdom	2011	7
12922	United Kingdom	2011	7
12928	United Kingdom	2011	7
12935	United Kingdom	2011	7

CustomerID	Total_Price
14646	280206.02
18102	259657.3
17450	194390.79
16446	168472.5
14911	143711.17
12415	124914.53
14156	117210.08
17511	91062.38
16029	80850.84
12346	77183.6
16684	66653.56
14096	65164.79
13694	65039.62
15311	60632.75
13089	58762.08
17949	58510.48
15769	56252.72
15061	54534.14
14298	51527.3
14088	50491.81

only showing top 20 rows

User story 2 solution:

```
# 2.Find the customer ID who are from country United Kingdom and has Invoice date at July month  
job2DF = spark.sql('SELECT DISTINCT CustomerID, Country, SUBSTRING(InvoiceDate,1,4) AS Year,\nSUBSTRING(InvoiceDate,6,2) AS Month \  
FROM ecom WHERE Country = "United Kingdom" AND SUBSTRING(InvoiceDate,6,2) = 7\  
ORDER BY CustomerID')
```

Analysis in Databricks Notebook

User story 3 solution:

```
# 3. Find the total number of people from France.  
job3DF = spark.sql("SELECT COUNT(CustomerID) AS Total_People_from_France FROM ecom \  
WHERE Country = 'France' ")
```

Total_People_from_France
87

```
+-----+-----+  
|StockCode|UnitPrice|  
+-----+-----+  
| 22120 | 10.13 |  
| 23314 | 10.39 |  
| 23143 | 10.4  |  
| 23104 | 10.4  |  
| 23085 | 10.4  |  
| 23142 | 10.4  |  
| 23427 | 10.4  |  
| 23111 | 10.4  |  
| 47570B | 10.65 |  
| 85141 | 10.75 |  
| 90035A | 10.75 |  
| 90035C | 10.75 |  
| 22832 | 10.75 |  
| 22165 | 10.75 |  
| 21216 | 10.79 |  
| 85174 | 10.79 |  
| 23245 | 10.79 |  
| 47566 | 10.79 |  
| 22476 | 10.79 |  
| 22139 | 10.79 |  
+-----+-----+  
only showing top 20 rows
```

User story 4 solution:

```
# 4. Find out the stock code which has unit price more than 10  
job4DF = spark.sql("SELECT DISTINCT StockCode, UnitPrice FROM ecom \  
WHERE UnitPrice > 10 ORDER BY UnitPrice")
```

STEP 7:

ADF Pipeline

The screenshot shows the Azure Data Factory (ADF) pipeline editor interface. The pipeline is named "sqlDB_to_databricks_to_ADLS".

Pipeline Structure:

```
graph LR; A[Set variable output_location] --> B[Set variable tableName]; B --> C[Notebook capstone_ecom]
```

Properties Panel:

- Name: sqlDB_to_databricks_to_ADLS
- Description: (empty)
- Annotations: (empty)

Output Tab:

Pipeline run ID: f6c91b12-cf5d-4139-9174-bc78984e8e57

Activity name	Status	Activity type	Run start	Duration	Log
capstone_ecom	Succeeded	Notebook	6/6/2023, 2:58:52 AM	00:00:34	
tableName	Succeeded	Set variable	6/6/2023, 2:58:51 AM	00:00:01	
output_location	Succeeded	Set variable	6/6/2023, 2:58:50 AM	00:00:01	

File Explorer:

- Authentication method: Access key (Switch to Azure AD User Account)
- Location: output-ecom
- Search blobs by prefix (case-sensitive):
- Items listed: op1, op2, op3, op4, op5, op6

Standalone Analysis

PySpark RDD

Importing the CSV into RDD

```
# 1. Creating RDD for ecommerce dataset
```

```
ecomRDD = sc.textFile('data_processed.csv')
type(ecomRDD)
ecomRDD.count()
```

```
# 2. Removing headers
```

```
header = ecomRDD.first()
ecomRDD = ecomRDD.filter(lambda row: row!=header)
ecomRDD.take(2)
```

```
# 3. Splitting records using comma -> " , " as delimiter
```

```
ecomRDD_final = ecomRDD.map(lambda x : x.split(','))
ecomRDD_final.take(1)
```

```
# 5. Converting Data types for required columns using function
```

```
def change_dataType(x):
    return [x[0], x[1], x[2], int(x[3]), x[4], float(x[5]), x[6], x[7]]
```

```
ecomRDD_final = ecomRDD_final.map(change_dataType)
ecomRDD_final.take(2)
```

Final RDD

```
ecom_final.take(2)
```

```
[['536365',
  '85123A',
  'WHITE HANGING HEART T-LIGHT HOLDER',
  6,
  '01-12-2010 08:26',
  2.55,
  '17850',
  'United Kingdom'],
 ['536365',
  '71053',
  'WHITE METAL LANTERN',
  6,
  '01-12-2010 08:26',
  3.39,
  '17850',
  'United Kingdom']]
```

User Story : How many customers from France?

```
def is_from_France(row):  
    return row[7] == 'France'
```

```
ecom_fromFrance = ecomRDD_final.filter(is_from_France)
```

```
custCount_fromFrance = ecom_fromFrance.map(lambda r: r[6]).distinct().count()
```

Total No. of Customers from France : -----> 87

User Story : Find out the stock code which has unit price more than 10 ?

```
# Defining a function to get rows only where Unit price > 10  
  
def unitPrice_moreThan10(row):  
    return row[5] > 10
```

```
ecom_price10 = ecomRDD_final.filter(unitPrice_moreThan10)  
ecom_price10.take(2)
```

Stock Codes where Unit Price is greater than 10:

```
[('22120', 10.13),  
 ('23314', 10.39),  
 ('23427', 10.4),  
 ('23142', 10.4),  
 ('23143', 10.4),  
 ('23111', 10.4),  
 ('23085', 10.4),  
 ('23104', 10.4),  
 ('47570B', 10.65),  
 ('22832', 10.75),  
 ('90035A', 10.75),  
 ('22165', 10.75),  
 ('85141', 10.75),
```

Total No. of products with unit price more than 10

```
StockCode_price10.count()
```

504

Do You Understand ?



Venkat Sir DE Expert

Azure

Pipeline 2 - CosmosDB



Why CosmosDB?

1. Global Distribution:

- a. Cosmos DB provides global distribution capabilities, allowing you to replicate your data across multiple regions worldwide. This enables low-latency access to data for users located in different regions, ensuring a better user experience.

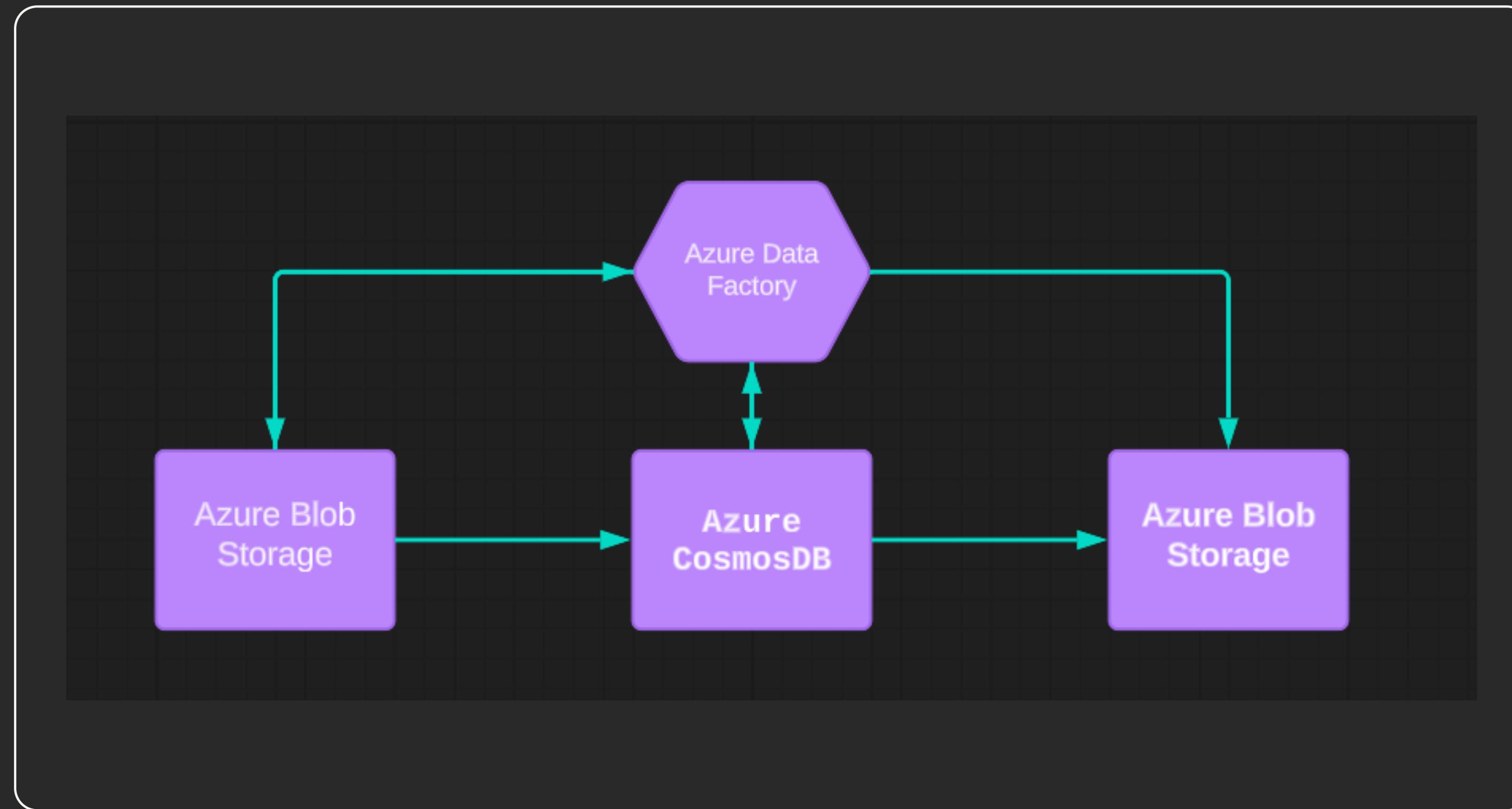
2. Scalability:

- a. Cosmos DB is designed to scale seamlessly and handle massive amounts of data. It can handle both read and write-heavy workloads, and it automatically scales throughput and storage based on demand. This elasticity makes it suitable for applications with unpredictable or rapidly changing workloads.

3. Multi-Model Database:

- a. Cosmos DB supports multiple data models, including key-value, column-family, document, graph, and time-series. This flexibility allows you to choose the most appropriate data model for your application without the need for separate databases, reducing complexity and cost.

ADF Pipeline Architecture -



Analysis in CosmosDB

PROBLEM STATEMENTS:

1. Filter out the column where quantity between 4 and 9?

```
> SELECT * FROM ecom WHERE Quantity BETWEEN 4 AND 9 ORDER BY InvoiceNo;
```

2. Filter out the column where customer ID 15862?

```
> SELECT DISTINCT InvoiceNo, InvoiceDate, StockCode, Description, Quantity,  
UnitPrice FROM ecom WHERE CustomerID = 15862;
```

3. Filter out the unit Price which has more than 5?

```
> SELECT * FROM ecom WHERE UnitPrice > 5 ORDER BY InvoiceNo;
```

Azure

Pipeline 3 - Synapse

- 1. Data Integration:** Azure Synapse brings together data from different sources for seamless analysis.
- 2. Data Warehousing:** It provides a high-performance, scalable data warehousing solution for structured data.
- 3. Big Data Analytics:** Synapse supports Apache Spark and Hadoop for analyzing large volumes of unstructured data.
- 4. Advanced Analytics:** Integrated with Azure Machine Learning, it enables the development and deployment of machine learning models.
- 5. Real-Time Data Processing:** Synapse supports real-time data ingestion, processing, and monitoring.
- 6. Security and Governance:** It offers robust security features and integrates with Azure Purview for data governance.
- 7. Scalability and Flexibility:** Synapse leverages Azure's cloud infrastructure for unlimited scalability and cost efficiency.

Architecture of Synapse

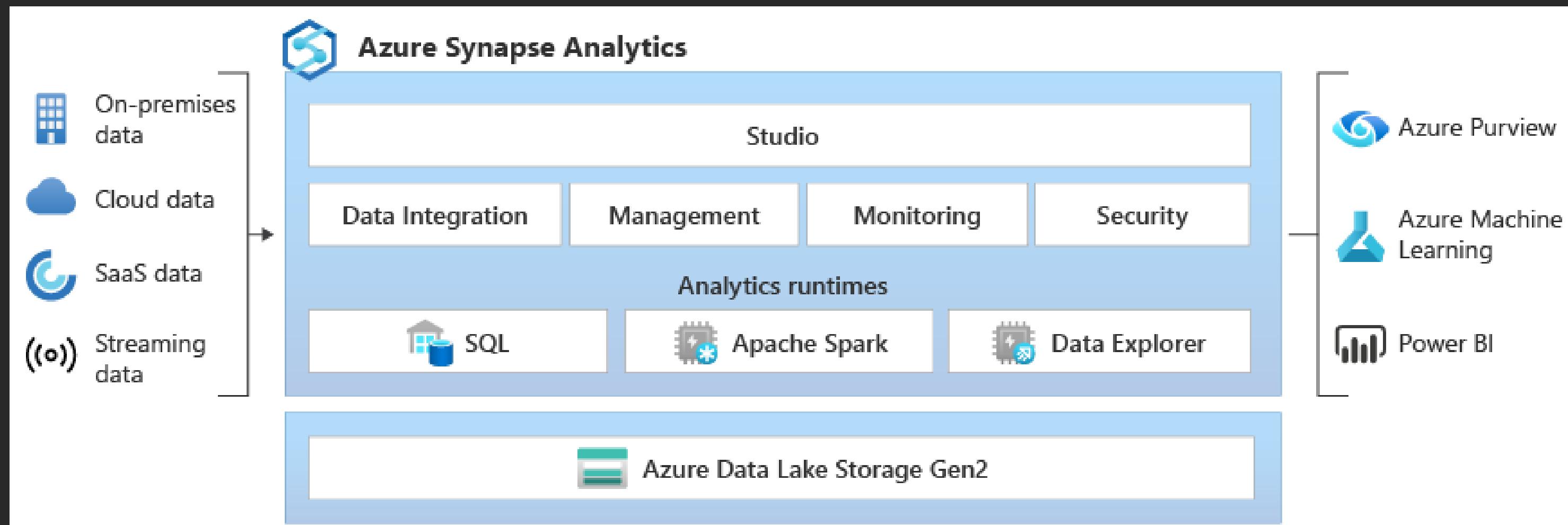


Table Creation and Data Insertion

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar is titled 'Data' and shows a 'Linked' workspace named 'biswas'. The main area has tabs for 'TABLE CREATION' and 'LOAD DATA INTO TA...'. The 'TABLE CREATION' tab is active, showing a SQL script for creating a table 'ecommerce' with columns: InvoiceNo (VARCHAR(255)), StockCode (VARCHAR(255)), Description (VARCHAR(255)), Quantity (INT), InvoiceDate (VARCHAR(255)), UnitPrice (FLOAT), CustomerID (VARCHAR(255)), and Country (VARCHAR(255)). Below the script is a 'SELECT * FROM ecommerce;' query. On the right, the 'Properties' panel is open for the 'TABLE CREATION' object, showing 'General' tab selected with 'Name' set to 'TABLE CREATION' and 'Type' set to 'sql script'.

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar is titled 'Data' and shows a 'Linked' workspace named 'biswas'. The main area has tabs for 'TABLE CREATION' and 'LOAD DATA INTO T...'. The 'LOAD DATA INTO T...' tab is active, showing a T-SQL script for loading data from a CSV file into the 'dbo.ecommerce' table. The script includes a 'WITH' clause for the BULK INSERT command, specifying the file type as 'CSV', max errors as 0, and error file location. Below the script is a 'SELECT TOP 100 * FROM dbo.ecommerce' query. On the right, the 'Properties' panel is open for the 'LOAD DATA INTO TABLE' object, showing 'General' tab selected with 'Name' set to 'LOAD DATA INTO TABLE' and 'Type' set to '.sql script'. The 'Results' tab is selected, showing a table with two rows of data:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
539470	21823	PAINTED META...	4	19-12-2010 14:...	1.45	14472
540372	22936	BAKING MOUL...	6	06-01-2011 16:...	3.25	13081



Synapse Queries

Microsoft Azure | Synapse Analytics > biswas

Search

Synapse live | Validate all | Publish all 2

Data

Workspace | Linked

Filter resources by name

Azure Data Lake Storage Gen2 2

- biswas (Primary - g2biswas) ...
- biswas (Primary) ...
- (Attached Containers) ...

biswas | Queries 7_8_9 | LOAD DATA INTO TA...

Run | Undo | Publish | Query plan | Connect to Ecomm | Use database Ecomm

```
1 -- 7.find which customerid who has description related to water bottle
2 SELECT CustomerID, SUM(Quantity) AS WaterBottleCount FROM ecommerce
3 WHERE Description like '%water%' and Description like '%bottle%'
4 GROUP BY CustomerID;
5 -- 8.find out the customerid who have more than 500 InvoiceNo
6 SELECT CustomerID, COUNT(InvoiceNo) AS InvCount FROM ecommerce
7 GROUP BY CustomerID
8 HAVING COUNT(InvoiceNo) > 500
9 ORDER BY COUNT(InvoiceNo) DESC;
10 -- 9.find out customerid who has stockcode 22752
11 SELECT CustomerID, COUNT(Quantity) AS QuantityCount22752 FROM ecommerce
12 WHERE StockCode = '22752'
13 GROUP BY CustomerID
14 ORDER BY COUNT(Quantity) DESC;
15
```

Properties

General | Related (0)

Name *: Queries 7_8_9

Description:

Type: .sql script

Size: 0 bytes

Results settings per query

- First 5000 rows (default)
- All rows

Results

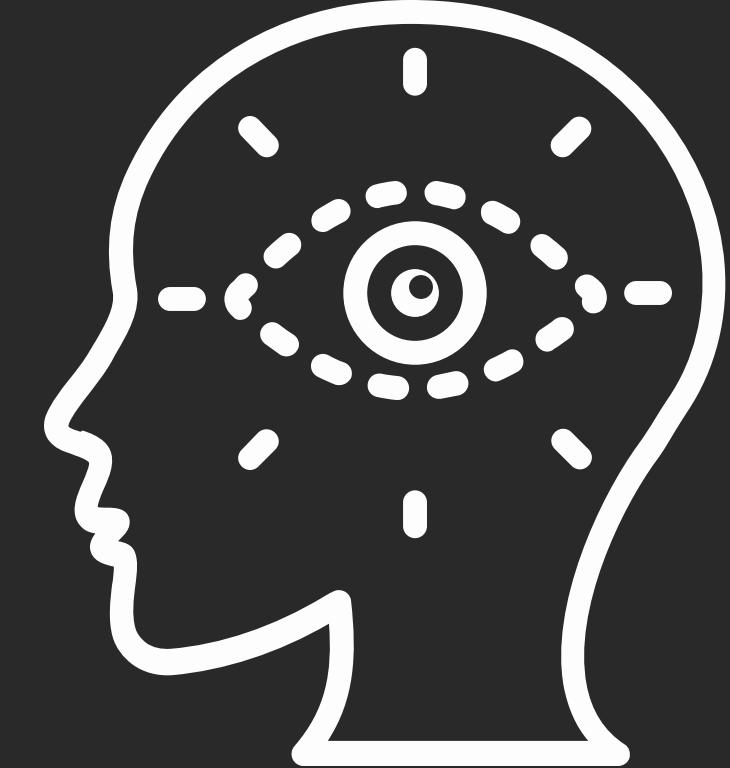
Messages

Select | Query 1 | View | Table | Chart | Export results

CustomerID | InvCount

CustomerID	InvCount
17841	7847
14911	5675
14096	5111

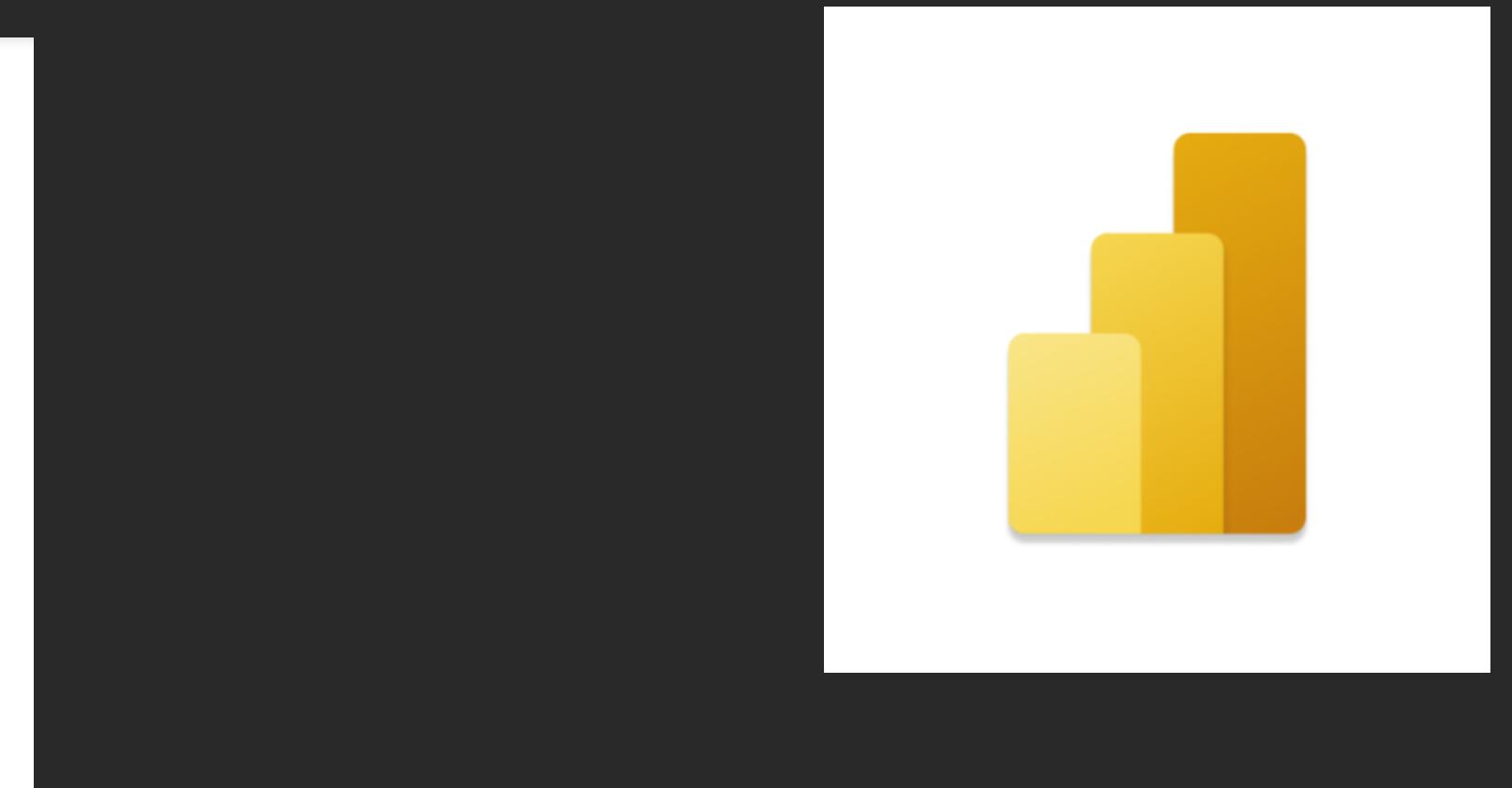
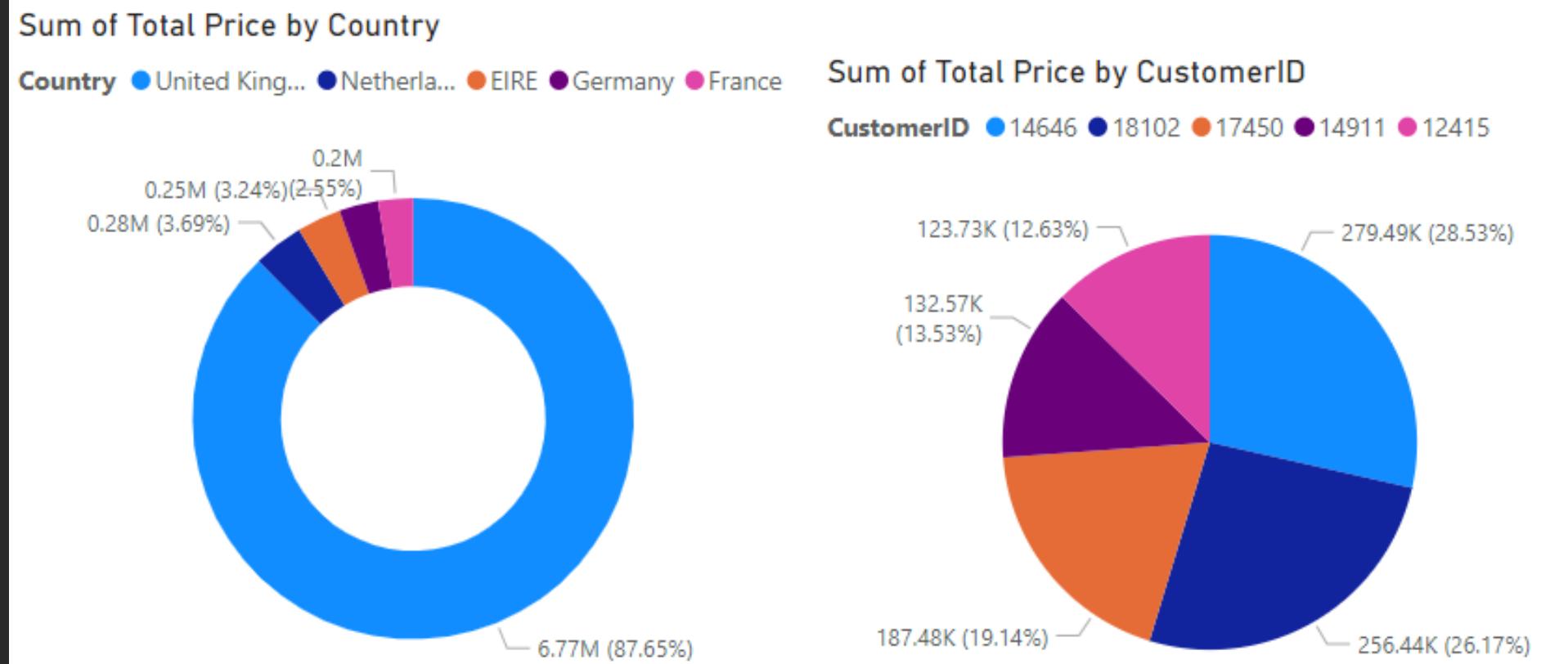
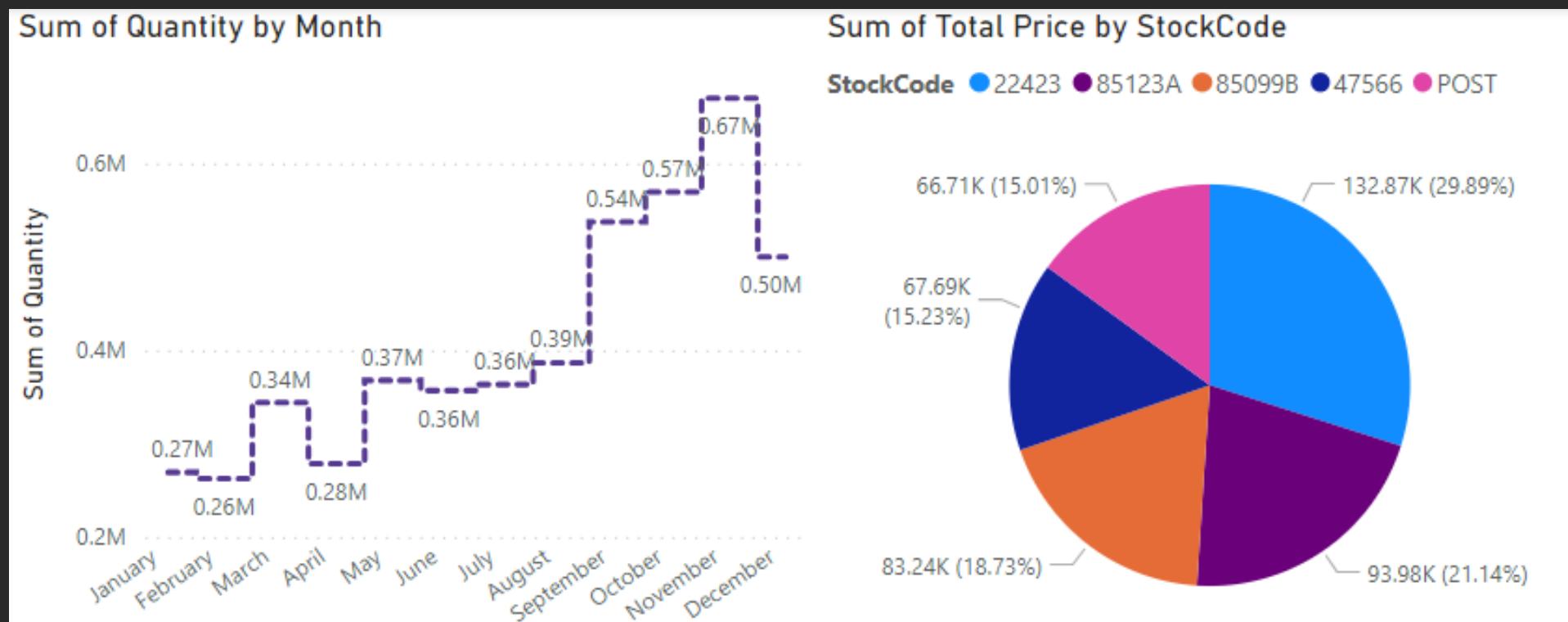
00:00:04 Query executed successfully.



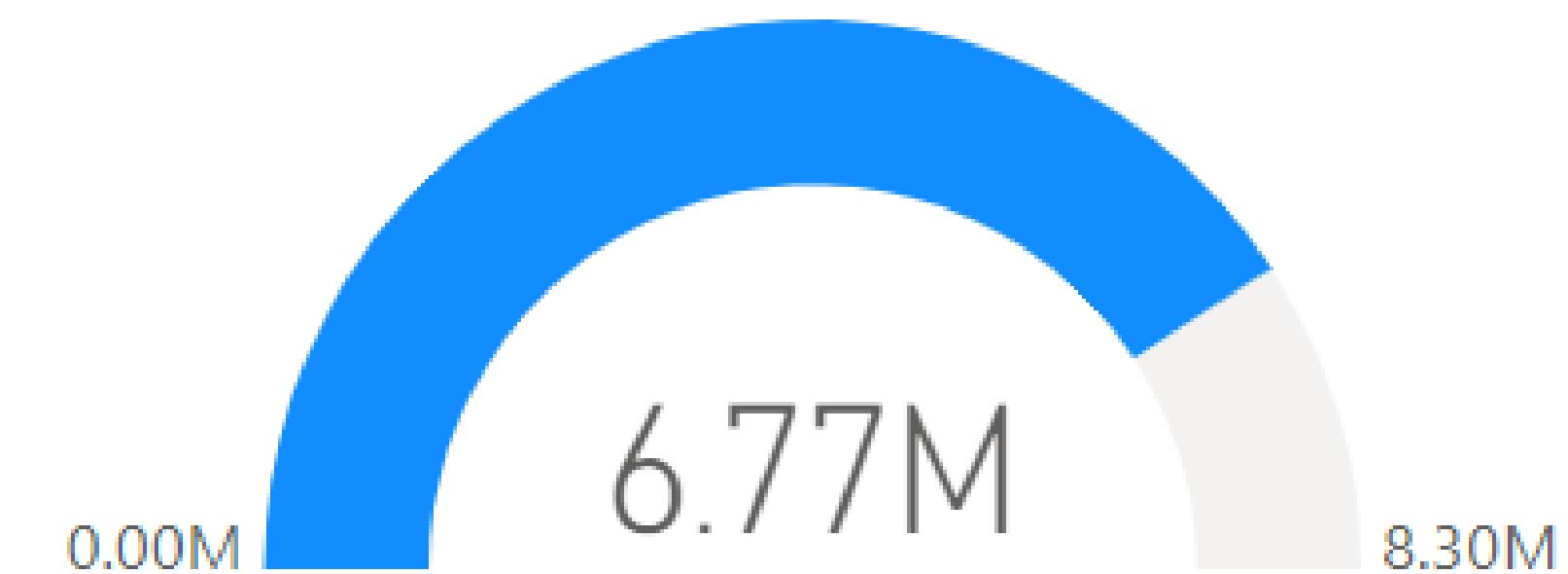
Visualization



Visualization



Country Highest Revenue Value and Total Revenue



Visualization



Singapore

Country Highest Avg Unit Price

United Kingdom

Country Highest Revenue

22423

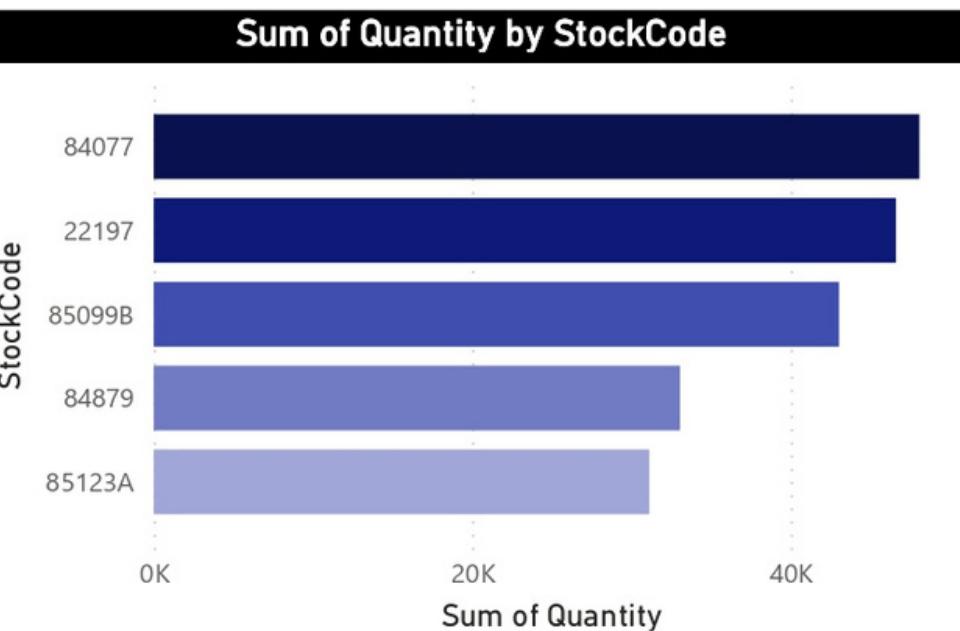
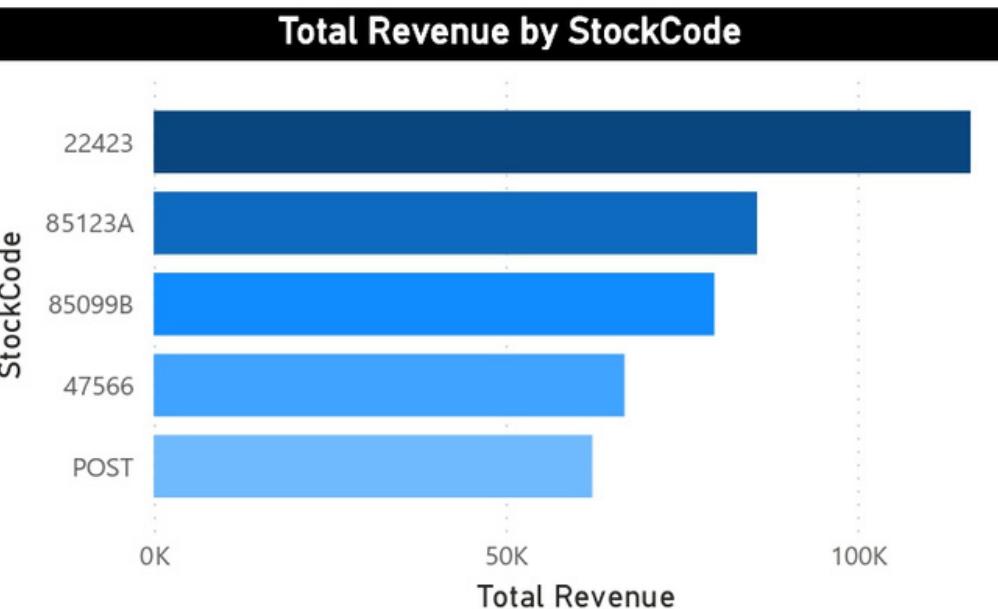
Stock Code Maximum Revenue

512

Average Number of Unique Stock Codes

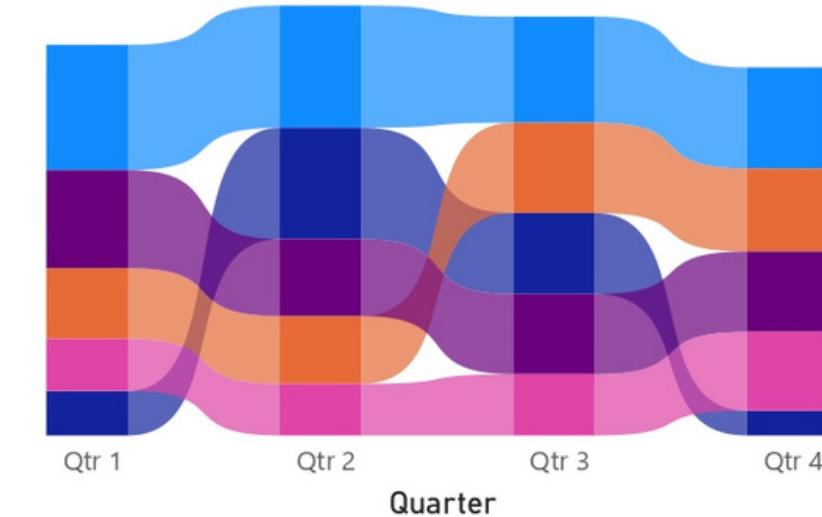
Stock Code with Highest Quantity Sold

84077



Total Revenue by Quarter and StockCode

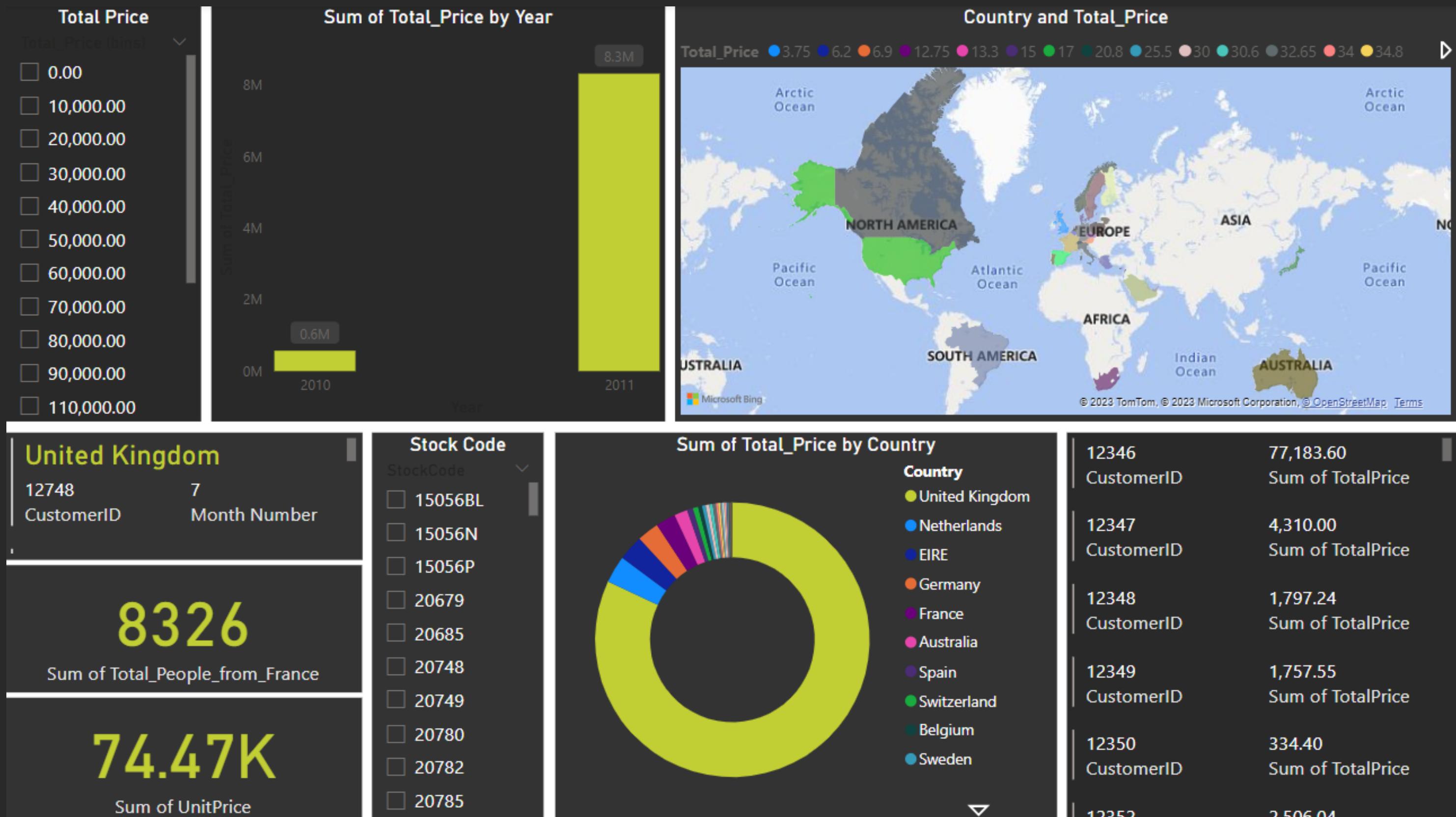
StockCode ● 22423 ● 47566 ● 85099B ● 85123A ● POST



Average Invoice Value by Country

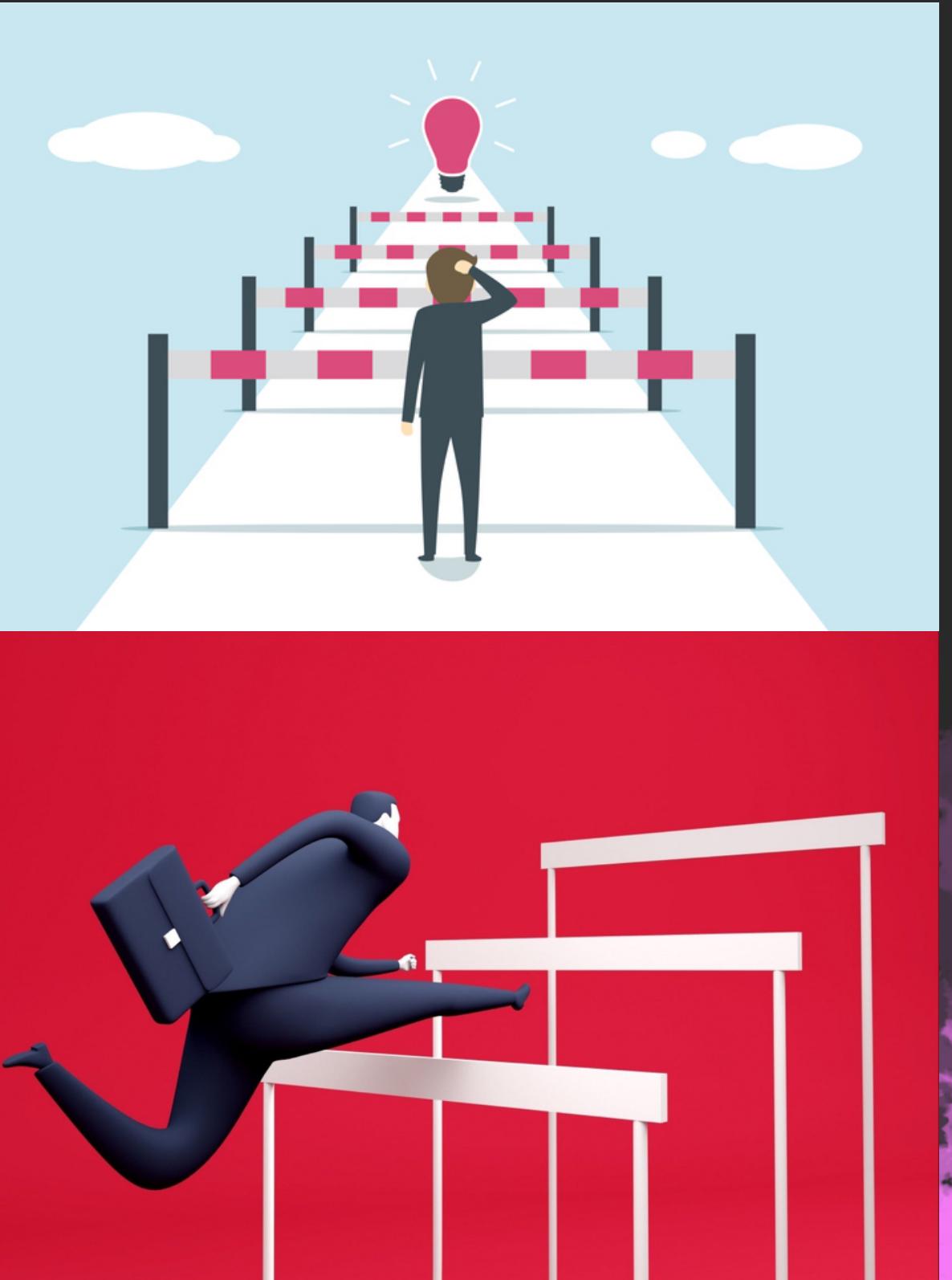


Visualization



Challenges

- Data Pre-Processing
- Choosing the right tools for Processing and Analyzing
- Building efficient Pipeline
- Methods for Evaluating and Improving Performance



Final Thoughts



The background features a dark gray gradient. On the left, there's a teal circle at the bottom and a purple circle at the top. On the right, there's a large silver-colored metal ring. In the center, there are two smaller, interlocking silver-colored metal rings.

Thank you
for participating!