Key KPIs:

1. Total Revenue: Sum of the Amount or Total_Amount column.
2. Total Transactions: Count of unique Transaction_ID.
3. Average Transaction Value: Total_Amount / Total_Purchases or Total_Amount / count of Transaction_ID.
4. Average Customer Age: Average of the Age column.
5. Customer Retention Rate: Percentage of returning customers (those with multiple Transaction_ID entries) relative to all customers.
6. Product Popularity: Count of purchases by Product_Category or Product_Brand to find the most popular items.
7. Order Fulfillment Rate: Percentage of Order_Status marked as "Completed" relative to total orders.
8. Customer Satisfaction: Average of Ratings column or analysis of the Feedback column if it's qualitative.
9. Revenue by Customer Segment: Sum of Amount for each Customer_Segment.

GitHub Link:

https://github.com/kumkumbaswal003/Cloudthat_Project.git

AZURE DATA PIPELINE

For TechRetail

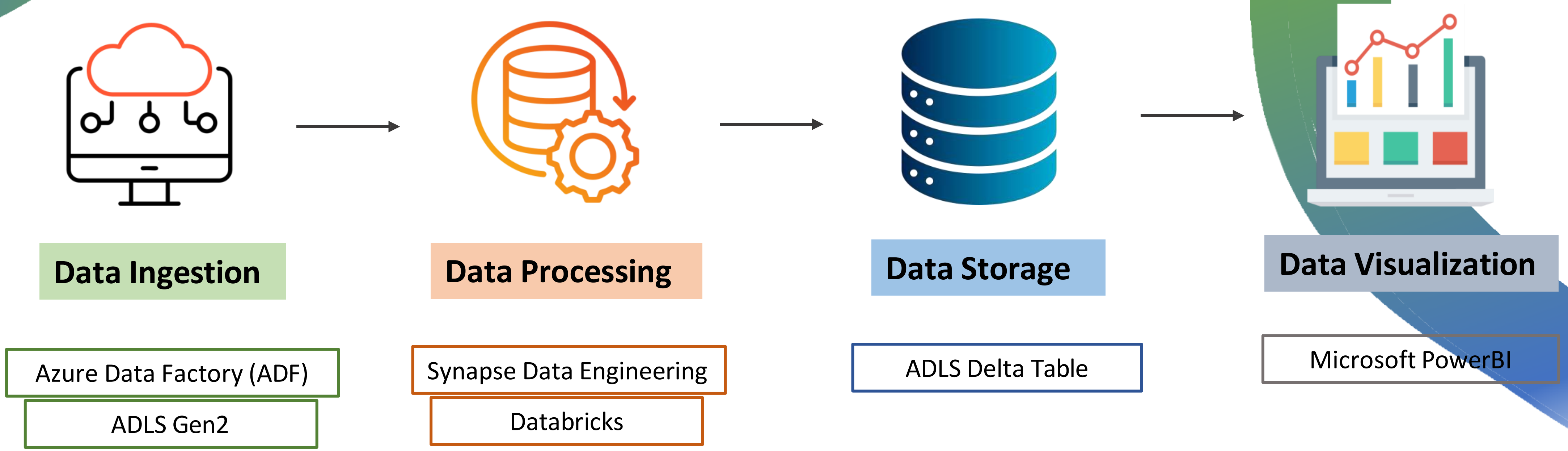
Group 4:

- Arnab Saha
- Debadrita Acharjee
- Kumkum Baswal

Background:

- **TechRetail**, a mid-sized retail company, wants to create a data pipeline retail data from various sources, process it using advanced analytics, and visualize the results in a dashboard. The goal is to gain insights into sales trends and improve decision-making. The company wants to leverage Azure Databricks for data processing and Microsoft Fabric for data integration and visualization.

Objectives:



Data Ingestion

Azure Data Factory (ADF)

ADLS Gen2

Data Processing

Synapse Data Engineering

Databricks

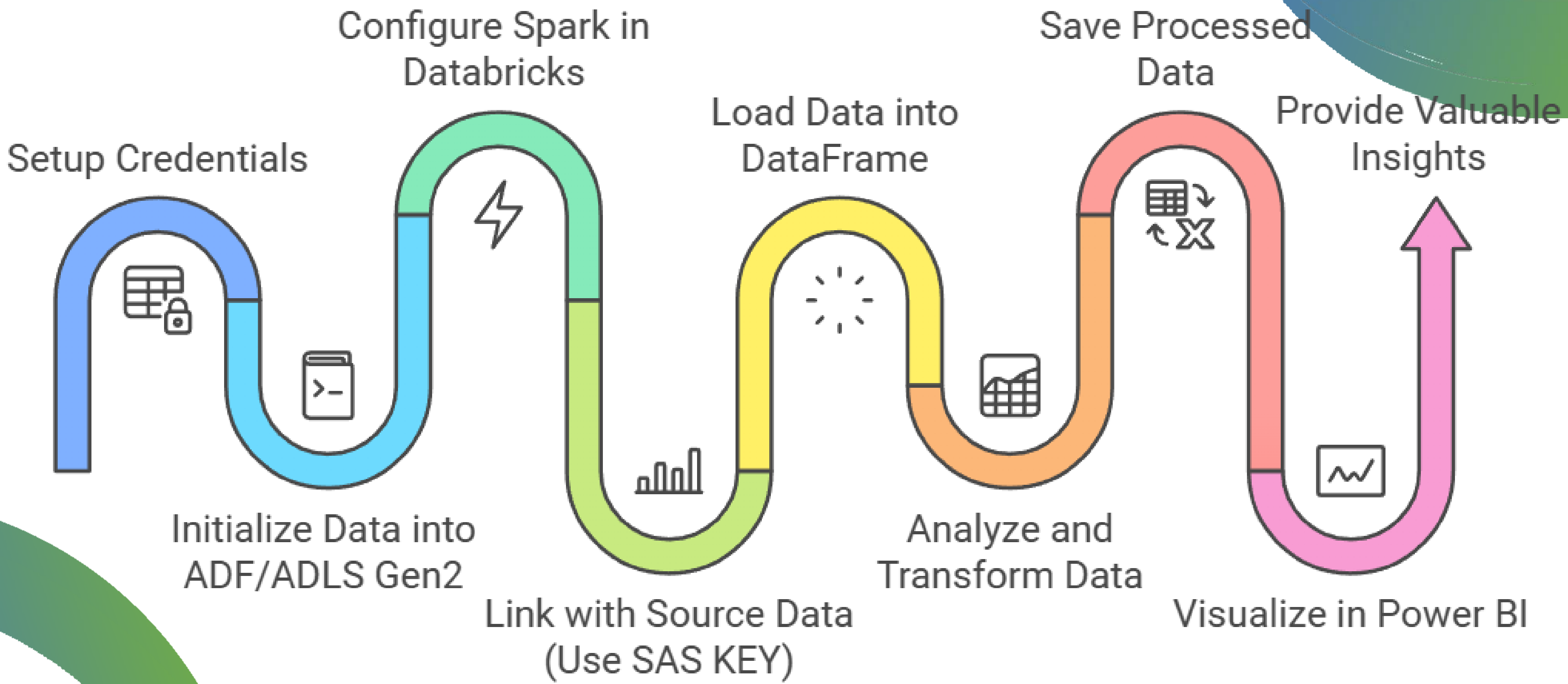
Data Storage

ADLS Delta Table

Data Visualization

Microsoft PowerBI

Architectural Process Flow:



Setup Credentials

Initialize Data into ADF/ADLS Gen2

Configure Spark in Databricks

Link with Source Data (Use SAS KEY)

Load Data into DataFrame

Analyze and Transform Data

Save Processed Data

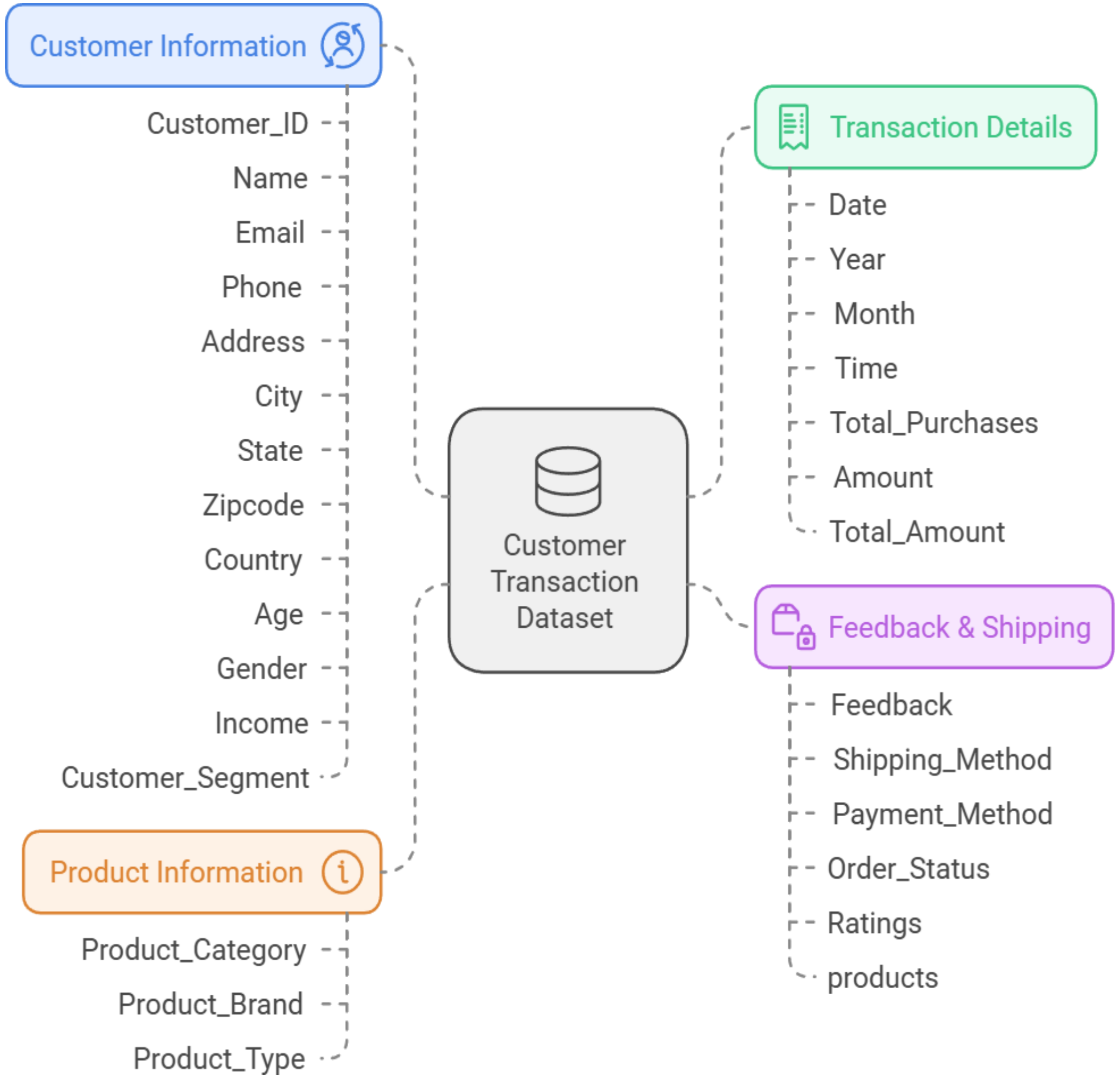
Visualize in Power BI

Provide Valuable Insights

KNOW YOUR DATA

Dataset Insights

DATASET OVERVIEW



Customer Transaction Dataset

Customer Information

- Customer_ID
- Name
- Email
- Phone
- Address
- City
- State
- Zipcode
- Country
- Age
- Gender
- Income
- Customer_Segment

Product Information

- Product_Category
- Product_Brand
- Product_Type

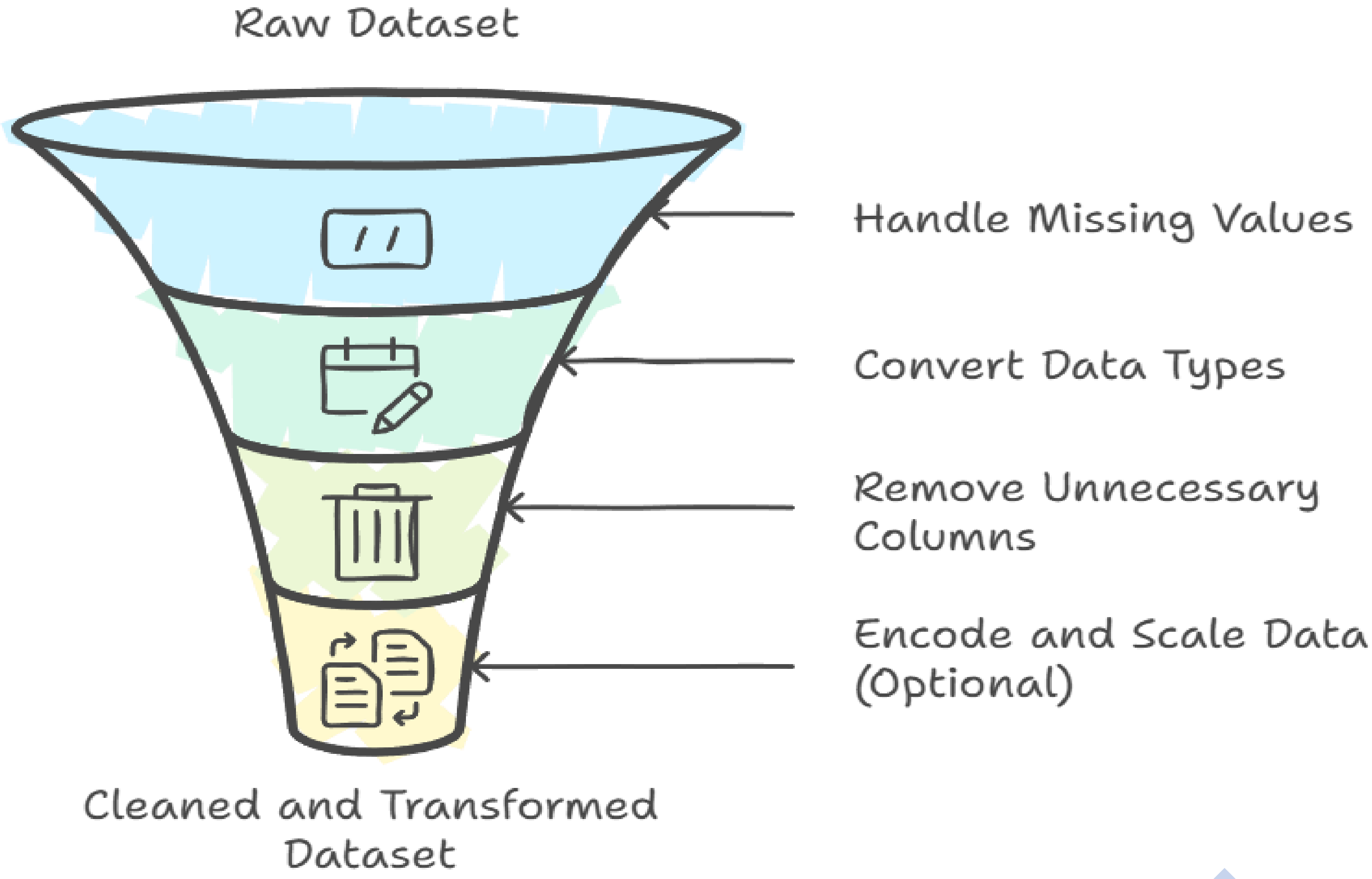
Transaction Details

- Date
- Year
- Month
- Time
- Total_Purchases
- Amount
- Total_Amount

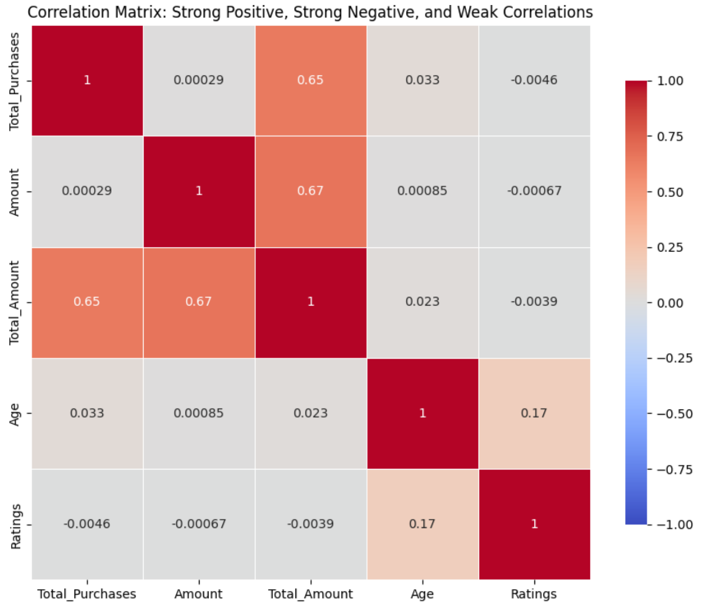
Feedback & Shipping

- Feedback
- Shipping_Method
- Payment_Method
- Order_Status
- Ratings
- products

Data Cleaning and Transformation Process

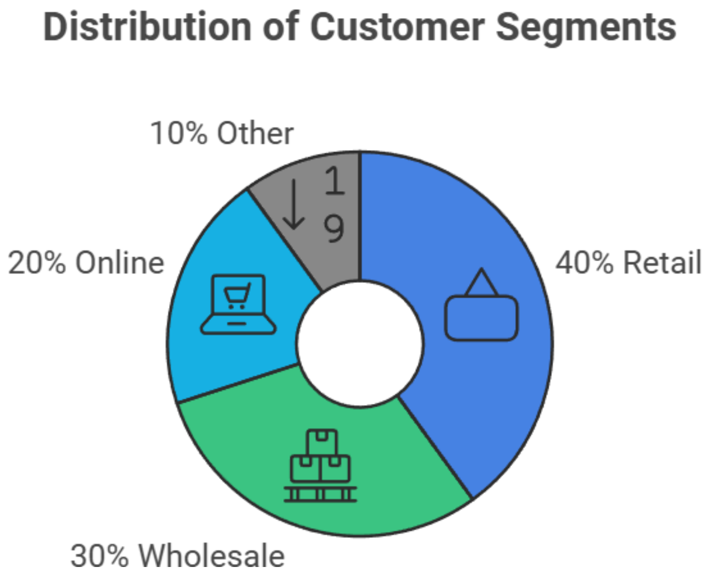

Raw Dataset
Handle Missing Values
Convert Data Types
Remove Unnecessary Columns
Encode and Scale Data (Optional)
Cleaned and Transformed Dataset

Dataset Insights Visualization

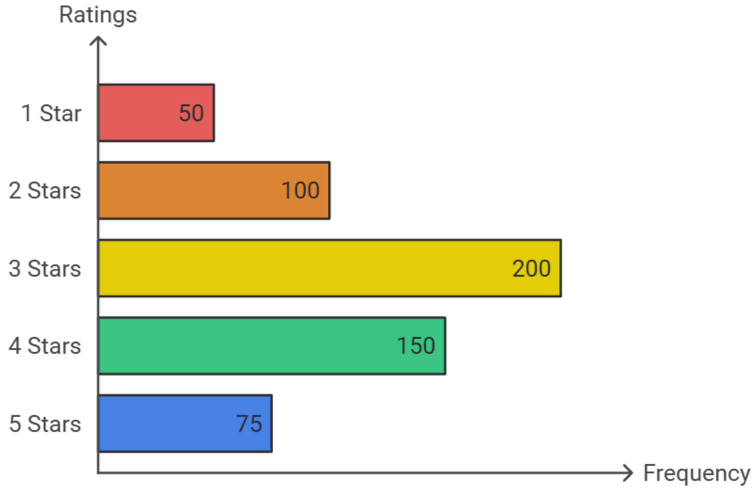


Correlation Matrix: Strong Positive, Strong Negative, and Weak Correlations

	Total_Purchases	Amount	Total_Amount	Age	Ratings
Total_Purchases	1	0.00029	0.65	0.033	-0.0046
Amount	0.00029	1	0.67	0.00085	-0.00067
Total_Amount	0.65	0.67	1	0.023	-0.0039
Age	0.033	0.00085	0.023	1	0.17
Ratings	-0.0046	-0.00067	-0.0039	0.17	1

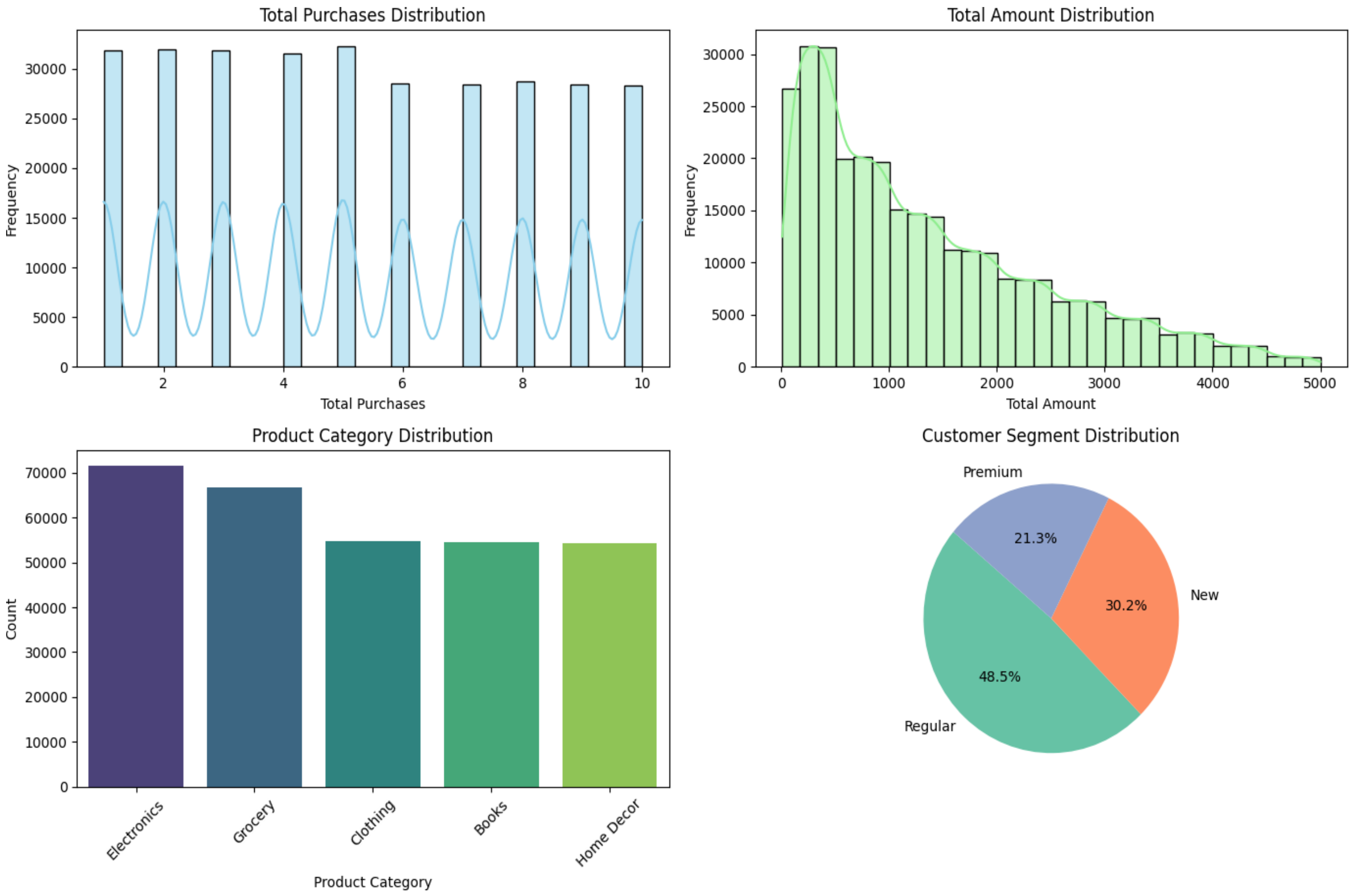


Distribution of Customer Segments



Distribution of Customer Ratings

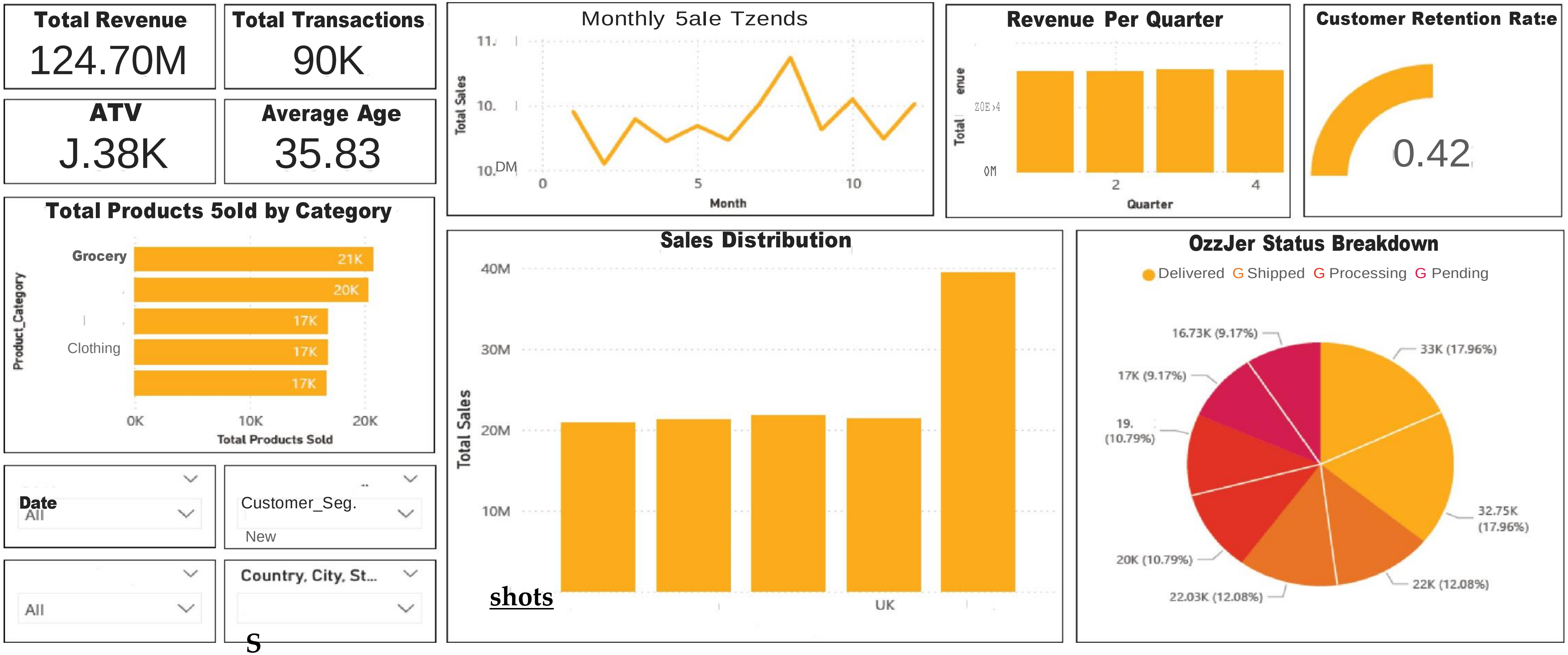
Key Distributions In The Data



TechRetail Sales Analytics Dashboard

TechRetaiI Sales Analysis

Key Metrics for Business Insights



Total Revenue
124.70M

Total Transactions
90K

ATV
J.38K

Average Age
35.83

Date
All

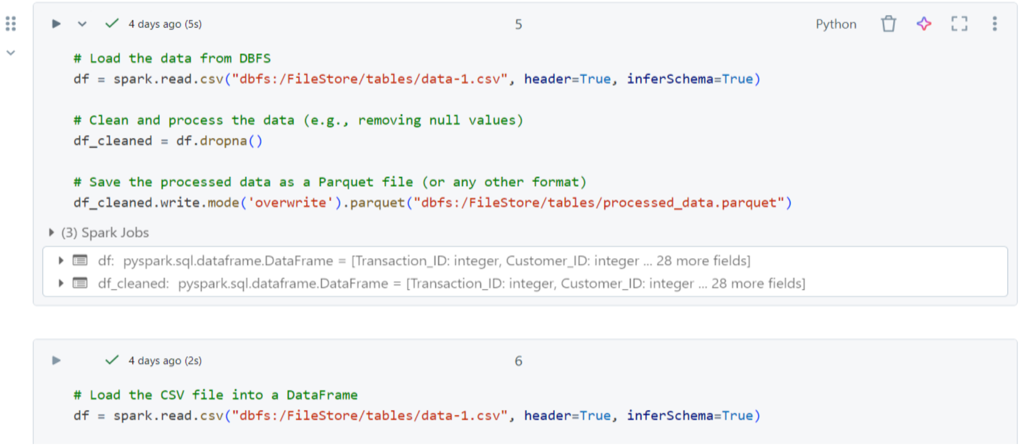
Customer_Seg.
New

All

Country, City, St...

shots

S
n
a
p



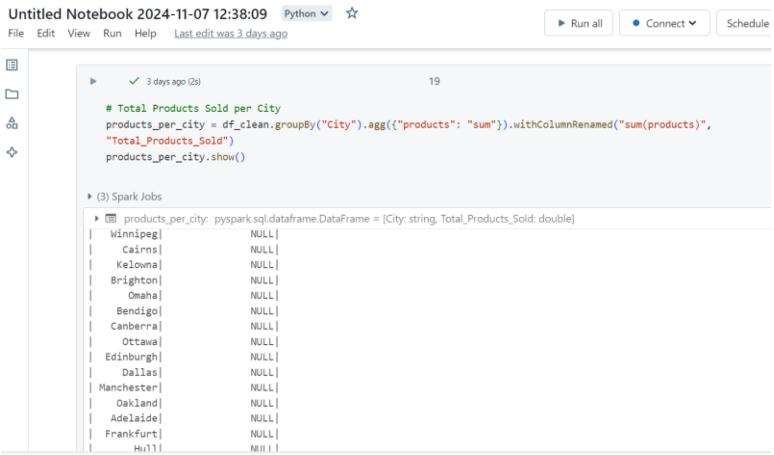
```
# Load the data from DBFS
df = spark.read.csv("dbfs:/FileStore/tables/data-1.csv", header=True, inferSchema=True)

# Clean and process the data (e.g., removing null values)
df_cleaned = df.dropna()

# Save the processed data as a Parquet file (or any other format)
df_cleaned.write.mode('overwrite').parquet("dbfs:/FileStore/tables/processed_data.parquet")
```

```
(3) Spark Jobs
df:  pyspark.sql.dataframe.DataFrame = [Transaction_ID: integer, Customer_ID: integer ... 28 more fields]
df_cleaned:  pyspark.sql.dataframe.DataFrame = [Transaction_ID: integer, Customer_ID: integer ... 28 more fields]
```

```
# Load the CSV file into a DataFrame
df = spark.read.csv("dbfs:/FileStore/tables/data-1.csv", header=True, inferSchema=True)
```



```
# Total Products Sold per City
products_per_city = df_clean.groupBy("City").agg({"products": "sum"}).withColumnRenamed("sum(products)",
"Total_Products_Sold")
products_per_city.show()
```

```
(3) Spark Jobs
products_per_city:  pyspark.sql.dataframe.DataFrame = [City: string, Total_Products_Sold: double]
|   Winnipeg|               NULL|
|     Cairns|               NULL|
|    Kelowna|               NULL|
|   Brighton|               NULL|
|      Omaha|               NULL|
|    Bendigo|               NULL|
|   Canberra|               NULL|
|     Ottawa|               NULL|
|  Edinburgh|               NULL|
|     Dallas|               NULL|
| Manchester|               NULL|
|    Oakland|               NULL|
|   Adelaide|               NULL|
|  Frankfurt|               NULL|
|       Hull|               NULL|
```

THANK YOU