Table of Contents

1. Introduction

The purpose of this project is to create a streamlined pipeline for ingesting, transforming, and visualizing sales data. The solution includes data quality checks, transformations for meaningful insights, and an interactive Power BI dashboard for decision-making. The use of Azure Data Lake Storage Gen2 ensures scalability, security, and efficient data organization for big data analytics

2. High-Level Solution Design

This section provides an overview of the architecture and workflow for the end-to-end sales data analysis pipeline.

Solution Overview

The proposed solution leverages Azure Data Lake Storage Gen2 (ADLS Gen2), Azure Data Factory (ADF), and Power BI to create a scalable, secure, and interactive analytics platform. The design focuses on ensuring data quality, applying meaningful transformations, and enabling intuitive visualizations to support business decision-making.

Components and Workflow

1. Data Ingestion

- Source: Sales data in CSV format.
- Destination: Azure Data Lake Storage Gen2 (raw data container).
- Tool Used: Python with the Azure Storage Blob SDK to upload and manage files.
- Purpose: Ensure secure, scalable, and hierarchical storage for raw data**.**

2. Data Transformation

- **Tool Used: Azure Data Factory.**
- **Steps:**
    - Data Cleaning: Handle missing values, incorrect data types, and anomalies.
    - Data Enrichment: Add calculated fields (e.g., Total Sales, Profit Margin, Customer Segment).
    - Data Aggregation: Summarize sales by location, product, and customer.
    - Security: Mask sensitive fields like Credit Card Numbers.

- **Output:** Cleaned and enriched data stored in Azure Data Lake Storage Gen2 (transformed data container).

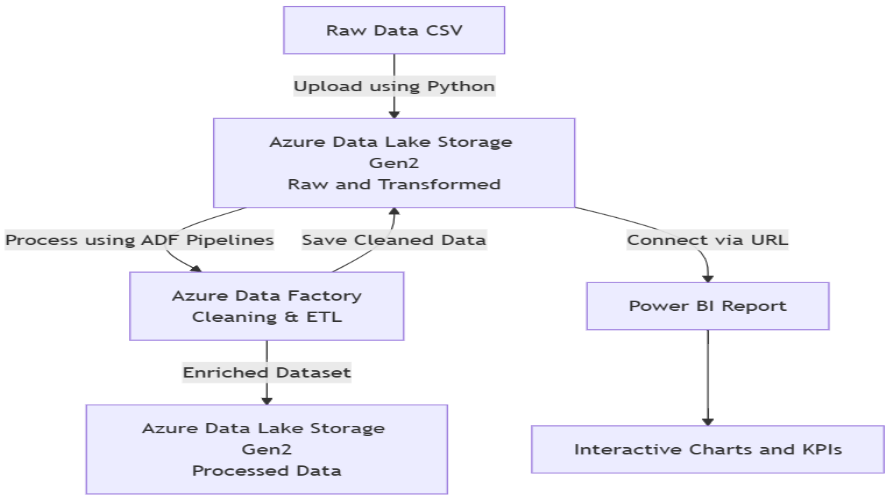
3. Data Visualization

- **Tool Used: Power BI.**
- **Steps:**
    - Connect Power BI to the transformed data using the Azure Data Lake Storage Gen2 URL.
    - Build interactive visuals to analyze sales trends, customer segmentation, and product performance.
    - Include filters for time, product, and location to enhance interactivity**.**

---

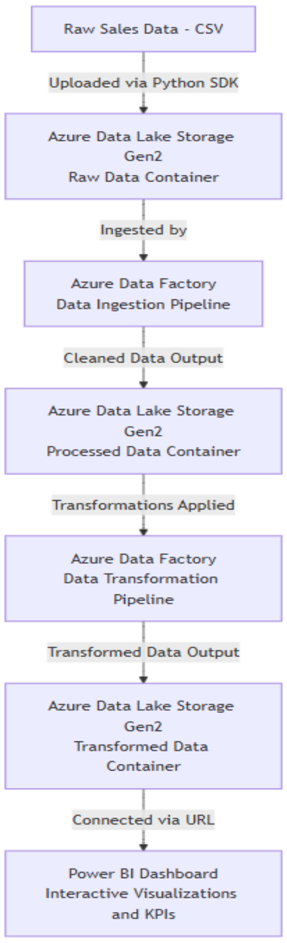
3. ARCHITECTURAL DIAGRAM

1.Diagram


Raw Data CSV
Upload using Python
Azure Data Lake Storage Gen2 Raw and Transformed
Process using ADF Pipelines
Save Cleaned Data
Connect via URL
Azure Data Factory Cleaning & ETL
Power BI Report
Enriched Dataset
Azure Data Lake Storage Gen2 Processed Data
Interactive Charts and KPIs

2 . Diagram

○


Raw Sales Data - CSV
Uploaded via Python SDK
Azure Data Lake Storage Gen2 Raw Data Container
Ingested by
Azure Data Factory Data Ingestion Pipeline
Cleaned Data Output
Azure Data Lake Storage Gen2 Processed Data Container
Transformations Applied
Azure Data Factory Data Transformation Pipeline
Transformed Data Output
Azure Data Lake Storage Gen2 Transformed Data Container
Connected via URL
Power BI Dashboard Interactive Visualizations and KPIs

○

Architectural Diagram

The flow of the data pipeline:

1. **Raw Data Upload**:
   - Raw data is uploaded to Azure Blob Storage.
2. **Data Ingestion Pipeline**:
   - Azure Data Factory reads raw data from Blob Storage.
   - Performs cleaning and schema validation.
   - Stores cleaned data back into Blob Storage.
3. **Transformation Pipeline**:
   - Additional calculations and enrichments (e.g., profit margin, customer segmentation).
   - Stores the final transformed data in Blob Storage.
4. **Visualization**:
   - Power BI connects to the transformed data URL for reporting

CHOICE OF DATA STORAGE :JUSTIFICATION FOR AZURE BLOB STORAGE

1. **Scalability**: Supports large datasets and integrates seamlessly with Azure Data Factory and Power BI.
2. **Cost-Effectiveness**: Pay-as-you-go pricing makes it ideal for dynamic workloads.
3. **Security**: Role-based access control (RBAC) ensures data security.
4. **Ease of Integration**: Supports multiple tools (Python, ADF, Power BI) for a cohesive workflow

JUSTIFICATION FOR USING AZURE DATA LAKE STORAGE GEN2

5. **Hierarchical Namespace:**
   1. Enables file/folder structure for easy organization of raw, intermediate, and transformed data.
   2. Supports file-level operations, such as renaming and moving**.**
6. **Scalability:**
   1. Handles large volumes of structured and unstructured data, making it ideal for enterprise-scale analytics.
7. **Cost-Effectiveness:**
   1. Tiered storage (Hot, Cool, Archive) optimizes cost based on data access frequency**.**

8. **Big Data Integration:**
    1. Seamlessly integrates with big data analytics tools like Azure Synapse, Databricks, and HDInsight.
9. **Security:**
    1. Provides fine-grained access control with Azure Role-Based Access Control (RBAC) and Access Control Lists (ACLs).
10. **Ease of Integration:**
    1. Works seamlessly with Azure Data Factory for data processing and Power BI for visualization.

4 CODE IMPLEMENTATION

4.Overview Flow of the Data Pipeline

1. Raw Data Upload

- Raw sales data (in CSV format) is uploaded to Azure Data Lake Storage Gen2 using Python scripts.
- The uploaded data resides in a dedicated container (e.g., /raw-data).

2. Data Ingestion Pipeline

- **Azure Data Factory (ADF)** reads raw data from ADLS Gen2.
- Performs initial cleaning and validation:
  - Removes missing or invalid values.
  - Standardizes column formats and data types (e.g., Date in yyyy-MM-dd format).
- Stores the cleaned data in a separate container/directory (/processed-data).

3. Transformation Pipeline

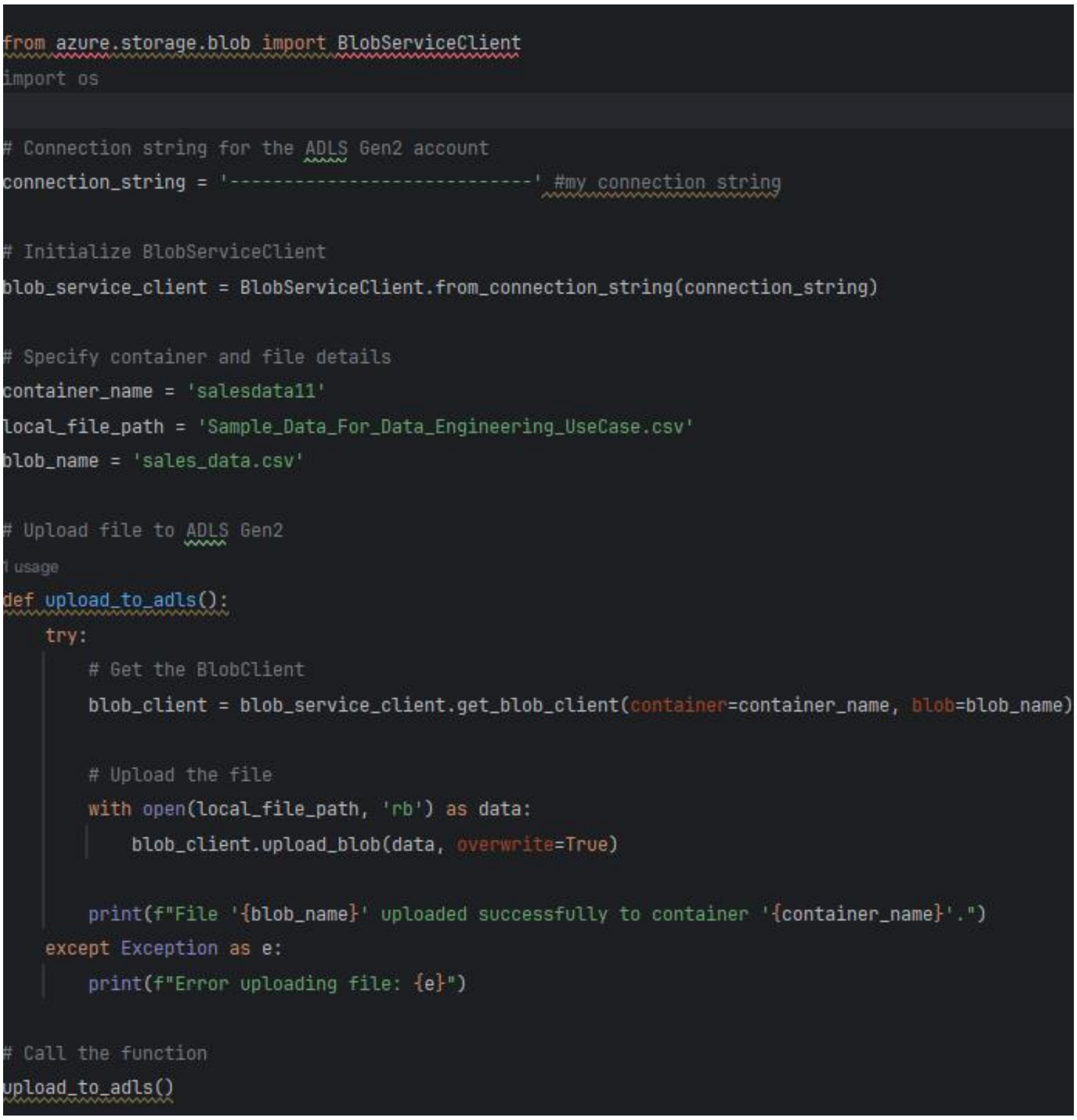
- ADF applies transformations to generate additional insights:
  - **Derived Columns**:
    - TotalSales = Quantity * Price
    - CustomerSegment = High/Medium/Low based on sales value.
  - **Aggregations**:
    - Total sales by location.
    - Count of orders per customer.
  - **Masking Sensitive Data**:
    - Masks credit card details (xxxx-xxxx-xxxx-1234).
- Writes the transformed data to a final container (/transformed-data).

4. Visualization

- Power BI connects to the transformed data using an **SAS URL** or **managed identity**.
- Interactive dashboards are created to visualize:
  - Sales trends.
  - Customer segmentation.
  - Product performance

4.1 Data Ingestion:

- To upload the raw sales data from a local CSV file into **Azure Data Lake Storage Gen2**



```
from azure.storage.blob import BlobServiceClient
import os

# Connection string for the ADLS Gen2 account
connection_string = '----------------------------' #my connection string

# Initialize BlobServiceClient
blob_service_client = BlobServiceClient.from_connection_string(connection_string)

# Specify container and file details
container_name = 'salesdata11'
local_file_path = 'Sample_Data_For_Data_Engineering_UseCase.csv'
blob_name = 'sales_data.csv'

# Upload file to ADLS Gen2
1 usage
def upload_to_adls():
    try:
        # Get the BlobClient
        blob_client = blob_service_client.get_blob_client(container=container_name, blob=blob_name)

        # Upload the file
        with open(local_file_path, 'rb') as data:
            blob_client.upload_blob(data, overwrite=True)

        print(f"File '{blob_name}' uploaded successfully to container '{container_name}'.")
    except Exception as e:
        print(f"Error uploading file: {e}")

# Call the function
upload_to_adls()
```

4.2 Cleaning Pipeline

Objective:

- To clean the raw data by handling missing values, correcting data types, and masking sensitive data.

Pipeline Steps:

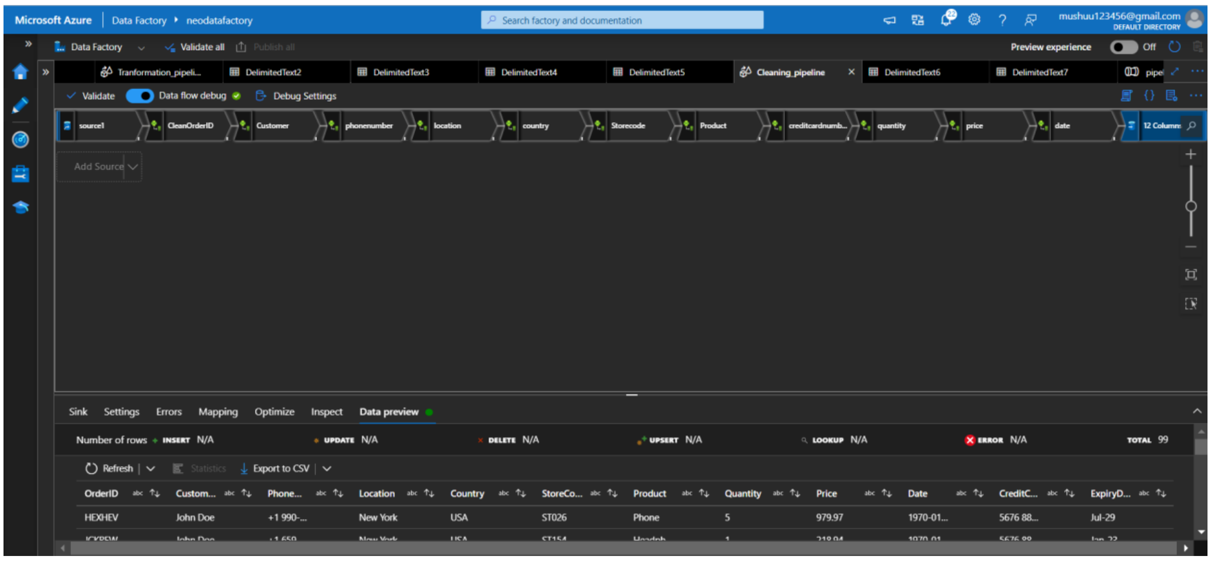
1. **Source:**
   - Input: Raw data from the ADLS Gen2 container raw-data.
2. **Transformations:**
   - Replace missing values (e.g., empty OrderID, CustomerName).
   - Convert data types (e.g., Quantity to integer, Price to decimal).
   - Mask sensitive data (CreditCardNumber**).**
3. **Sink:**
   - Save the cleaned data to the ADLS Gen2 container processed-data.

PIPELINE script

```
source(output(
        OrderID as string,
        CustomerName as string,
        PhoneNumber as string,
        Location as string,
        Country as string,
        StoreCode as string,
        Product as string,
        Quantity as string,
        Price as string,
        Date as string,
        CreditCardNumber as string,
        ExpiryDate as string
    ),
    allowSchemaDrift: true,
    validateSchema: false,
    ignoreNoFilesFound: false) ~> source1
source1 derive(OrderID = iif(isNull(OrderID) || trim(OrderID) == '', 'Unknown', OrderID)) ~> CleanOrderID
CleanOrderID derive(CustomerName = iif(isNull(CustomerName) || trim(CustomerName) == '', 'Unknown', CustomerName)) ~> Customer
Customer derive(PhoneNumber = iif(isNull(PhoneNumber) || trim(PhoneNumber) == '', 'Unknown', PhoneNumber)) ~> phonenumber
phonenumber derive(Location = iif(isNull(Location) || trim(Location) == '', 'Unknown', Location)) ~> location
```

```
location derive(Country = iif(isNull(Country) || trim(Country) == '', 'Unknown', Country)) ~> country
country derive(StoreCode = iif(isNull(StoreCode) || trim(StoreCode) == '', 'Unknown', StoreCode)) ~> Storecode
Storecode derive(Product = iif(isNull(Product) || trim(Product) == '', 'Unknown', Product)) ~> Product
Product derive(CreditCardNumber = iif(isNull(CreditCardNumber) || trim(CreditCardNumber) == '', 'Unknown', CreditCardNumber)) ~> creditcardnumber
creditcardnumber derive(Quantity = iif(isNull(trim(Quantity)) || trim(Quantity) == '', '10', Quantity)) ~> quantity
quantity derive(Price = iif(isNull(trim(Price)) || trim(Price) == '', '100', Price)) ~> price
price derive(Date = iif(isNull(toDate(Date, 'yyyy-MM-dd')), toDate('1970-01-01', 'yyyy-MM-dd'), toDate(Date, 'yyyy-MM-dd'))) ~> date
date sink(allowSchemaDrift: true,
    validateSchema: false,
    input(
        OrderID as string,
        CustomerName as string,
        PhoneNumber as string,
        Location as string,
        Country as string,
        StoreCode as string,
        Product as string,
        Quantity as string,
        Price as string,
        Date as string,
        CreditCardNumber as string,
        ExpiryDate as string
    ),
    partitionFileNames:['cleanedsalesdata.csv'],
    umask: 0022,
    preCommands: [],
    postCommands: [],
    skipDuplicateMapInputs: true,
    skipDuplicateMapOutputs: true,
    partitionBy('hash', 1)) ~> sink1

```

CLEANING PIPELINE DATAFLOW



Number of rows INSERT N/A UPDATE N/A DELETE N/A UPSERT N/A LOOKUP N/A ERROR N/A TOTAL 99

OrderID	Custom...	Phone...	Location	Country	StoreCo...	Product	Quantity	Price	Date	CreditC...	ExpiryD...
HEXHEV	John Doe	+1 990-...	New York	USA	ST026	Phone	5	979.97	1970-01...	5676 88...	Jul-29

4.3 Transformation Pipeline

Objective:

To enrich the cleaned data with calculated fields, customer segmentation, and aggregated insights.

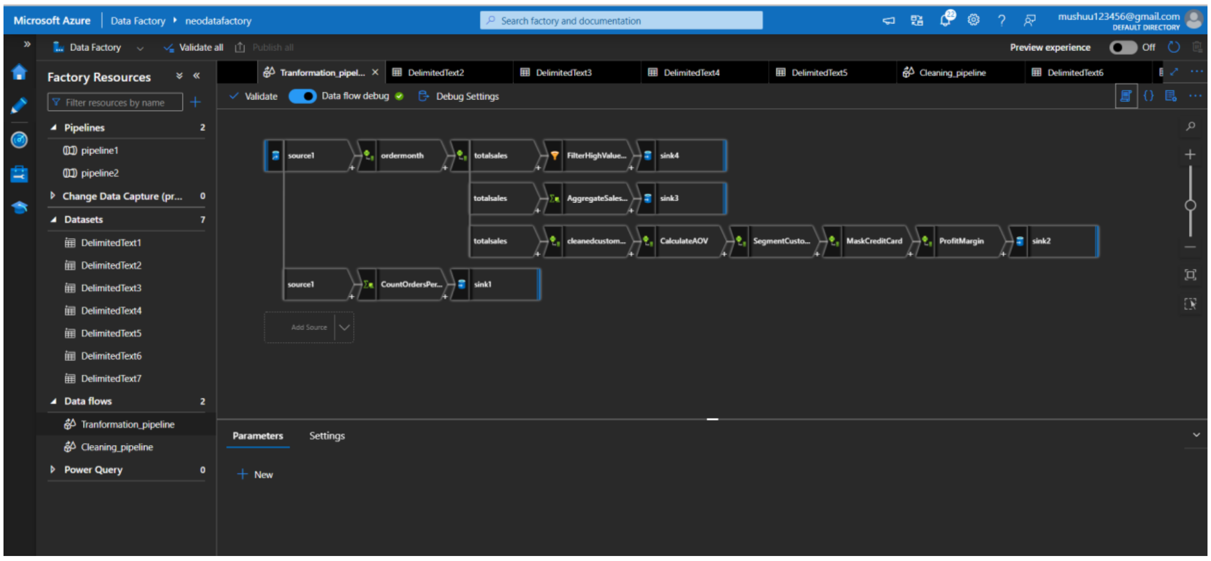
Pipeline Steps:

1. **Source:**
   - **Input**: Cleaned data from the ADLS Gen2 container processed-data**.**
2. **Transformations:**
   - **Add calculated columns:**
     - **TotalSales = Quantity * Price**
     - **AverageOrderValue = TotalSales / Quantity**
   - **Segment customers into High-value, Medium-value, and Low-value.**
   - **Aggregate sales by Location and CustomerName.**
3. **Sink:**
   - **Save transformed data to the ADLS Gen2 container transformed-data.**

TRANFORM PIPELINE SCRIPT

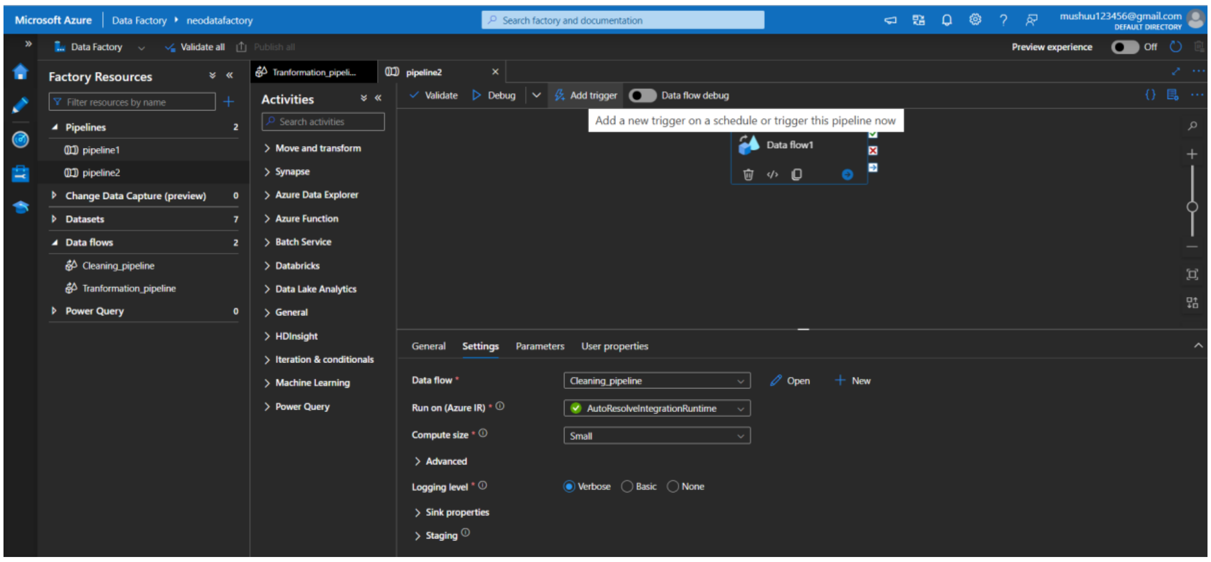
```
source(output(
        OrderID as string,
        CustomerName as string,
        PhoneNumber as string,
        Location as string,
        Country as string,
        StoreCode as string,
        Product as string,
        Quantity as string,
        Price as string,
        Date as string,
        CreditCardNumber as string,
        ExpiryDate as string
    ),
    allowSchemaDrift: true,
    validateSchema: false,
    ignoreNoFilesFound: false) ~> source1
source1 derive(ordermonth = month(toDate(Date, 'yyyy-MM-dd'))) ~> ordermonth
ordermonth derive(Totalsales = toInteger(Quantity) * toDecimal(Price)) ~>
totalsales
totalsales derive(FullCustomerName = CustomerName + ' (' + Location + ')') ~>
cleanedcustomernames
cleanedcustomernames derive(AverageOrderValue = Totalsales /
toFloat(Quantity)) ~> CalculateAOV
CalculateAOV derive(CustomerSegment = iif(Totalsales > 5000, 'High-value',
iif(Totalsales >= 200, 'Medium-value', 'Low-value'))) ~> SegmentCustomers
SegmentCustomers derive(MaskedCreditCard = iif(isNull(CreditCardNumber), '',
concat('xxxx-xxxx-xxxx-', substring(CreditCardNumber, length(CreditCardNumber)
- 4, 4)))) ~> MaskCreditCard
MaskCreditCard derive(ProfitMargin = toInteger(Price) * 0.2,
        DiscountApplied = iif(toInteger(Quantity) > 10, 'Yes', 'No')) ~>
ProfitMargin
source1 aggregate(groupBy(CustomerName),
    OrderID = count(OrderID)) ~> CountOrdersPerCustomer
totalsales filter(Totalsales > 500) ~> FilterHighValueTransactions
totalsales aggregate(groupBy(Location),
    Totalsales = sum(Totalsales)) ~> AggregateSalesByLocation
CountOrdersPerCustomer sink(allowSchemaDrift: true,
    validateSchema: false,
    input(
        OrderID as string,
        CustomerName as string,
        PhoneNumber as string,
        Location as string,
        Country as string,
        StoreCode as string,
        Product as string,
        Quantity as string,
        Price as string,
        Date as string,
        CreditCardNumber as string,
        ExpiryDate as string
    ),
    partitionFileNames:['final3.csv'],
    umask: 0022,
    preCommands: [],
    postCommands: [],
    skipDuplicateMapInputs: true,
    skipDuplicateMapOutputs: true,
    partitionBy('hash', 1)) ~> sink1
ProfitMargin sink(allowSchemaDrift: true,
    validateSchema: false,
    input(
        OrderID as string,
        CustomerName as string,
        PhoneNumber as string,
        Location as string,
        Country as string,
        StoreCode as string,
        Product as string,
        Quantity as string,
        Price as string,
        Date as string,
        CreditCardNumber as string,
        ExpiryDate as string
    ),
    partitionFileNames:['final.csv'],
    umask: 0022,
    preCommands: [],
    postCommands: [],
    skipDuplicateMapInputs: true,
    skipDuplicateMapOutputs: true,
    partitionBy('hash', 1)) ~> sink2
AggregateSalesByLocation sink(allowSchemaDrift: true,
    validateSchema: false,
    input(
        OrderID as string,
        CustomerName as string,
        PhoneNumber as string,
        Location as string,
        Country as string,
        StoreCode as string,
        Product as string,
        Quantity as string,
        Price as string,
        Date as string,
        CreditCardNumber as string,
        ExpiryDate as string
    ),
    partitionFileNames:['final2.csv'],
    umask: 0022,
    preCommands: [],
    postCommands: [],
    skipDuplicateMapInputs: true,
    skipDuplicateMapOutputs: true,
    partitionBy('hash', 1)) ~> sink3
FilterHighValueTransactions sink(allowSchemaDrift: true,
    validateSchema: false,
    input(
        OrderID as string,
        CustomerName as string,
        PhoneNumber as string,
        Location as string,
        Country as string,
        StoreCode as string,
        Product as string,
        Quantity as string,
        Price as string,
        Date as string,
        CreditCardNumber as string,
        ExpiryDate as string
    ),
    partitionFileNames:['final1.csv'],
    umask: 0022,
    preCommands: [],
    postCommands: [],
    skipDuplicateMapInputs: true,
    skipDuplicateMapOutputs: true,



    partitionBy('hash', 1)) ~> sink4
```

DATAFLOW TRANSFORMATION PIPELINE



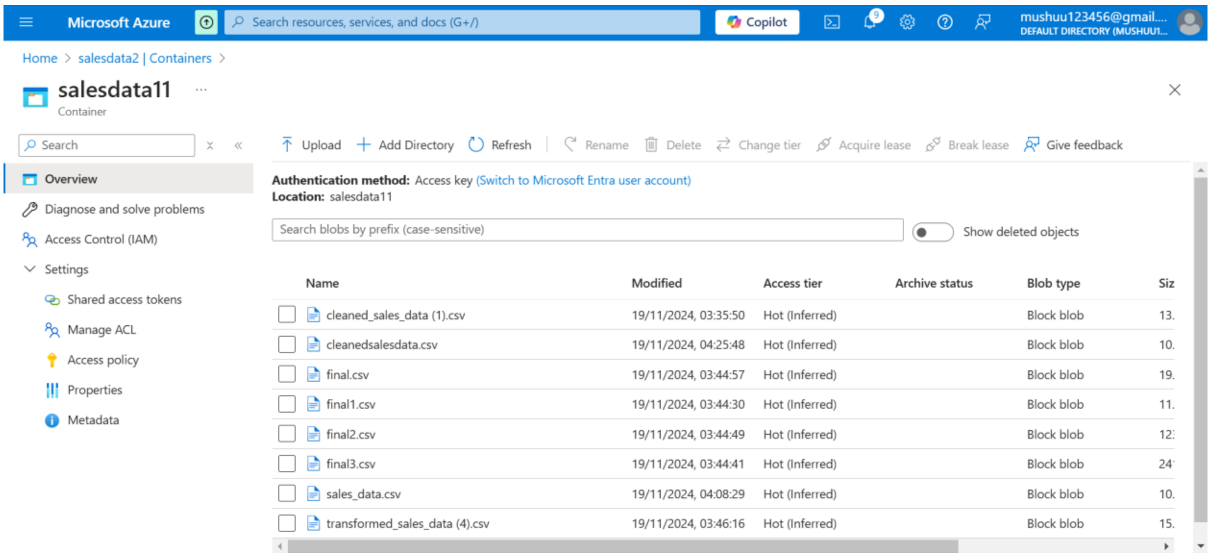
After Trigerring pipelines:

Triggering cleaned_pipeline



Data flow * Cleaning_pipeline
Run on (Azure IR) * AutoResolveIntegrationRuntime
Compute size * Small
Logging level * [✓ Verbose] Basic None

The saved files are in azure data lake storage:



Authentication method: Access key (Switch to Microsoft Entra user account)
Location: salesdata11

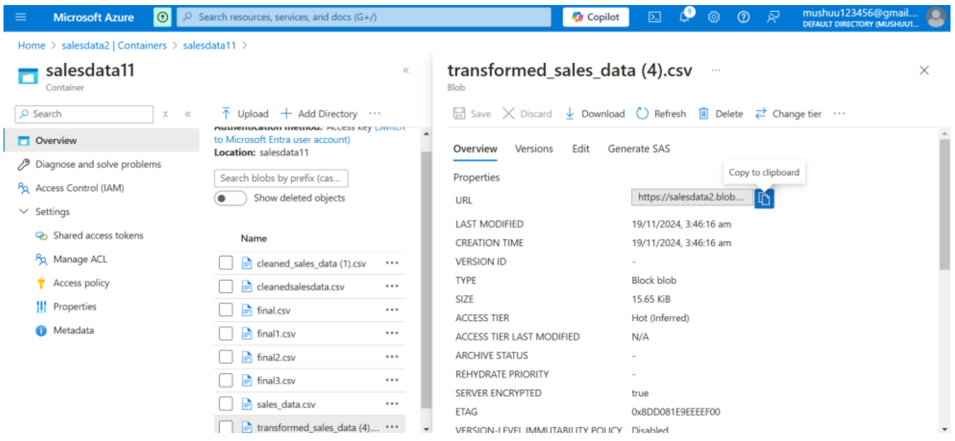
Name	Modified	Access tier	Archive status	Blob type	Siz
cleaned_sales_data (1).csv	19/11/2024, 03:35:50	Hot (Inferred)		Block blob	13.
cleanedsalesdata.csv	19/11/2024, 04:25:48	Hot (Inferred)		Block blob	10.
final.csv	19/11/2024, 03:44:57	Hot (Inferred)		Block blob	19.
final1.csv	19/11/2024, 03:44:30	Hot (Inferred)		Block blob	11.
final2.csv	19/11/2024, 03:44:49	Hot (Inferred)		Block blob	12
final3.csv	19/11/2024, 03:44:41	Hot (Inferred)		Block blob	24
sales_data.csv	19/11/2024, 04:08:29	Hot (Inferred)		Block blob	10.
transformed_sales_data (4).csv	19/11/2024, 03:46:16	Hot (Inferred)		Block blob	15.

4.4 Data Visualization

Objective:

To create a Power BI dashboard for visualizing sales trends, customer segmentation, and product performance.

Steps:

1. **Connect to Transformed Data:**
   - Use the Azure Data Lake Storage Gen2 URL to connect Power BI to transformed_sales_data.csv.
   - 

transformed_sales_data (4).csv
Blob

URL	https://salesdata2.blob....
LAST MODIFIED	19/11/2024, 3:46:16 am
CREATION TIME	19/11/2024, 3:46:16 am
VERSION ID	-
TYPE	Block blob
SIZE	15.65 KiB
ACCESS TIER	Hot (Inferred)
ACCESS TIER LAST MODIFIED	N/A
ARCHIVE STATUS	-
REHYDRATE PRIORITY	-
SERVER ENCRYPTED	true
ETAG	0x8DD081E9EEEEF00
VERSION-LEVEL IMMUTABILITY POLICY	Disabled

2. **Visuals:**
   - **Sales Trends:** Line chart to display sales over time.
   - **Customer Segmentation:** Pie chart for customer segments.
   - **Product Performance:** Bar chart to show product-wise sales.
3. **Interactivity:**
   - Add slicers for filtering by date, location, and product

4.5 Security Measures

1. **Role-Based Access Control (RBAC):**
   - Restrict access to ADLS Gen2 containers**.**
2. **Sensitive Data Masking:**
   - Mask credit card numbers in the pipeline**.**
3. **Monitoring:**
   - Enable logs in Azure Data Factory to track pipeline activities

5.DATASET DOCUMENTATION

The transformed sales dataset contains structured and cleaned data ready for analysis and visualization. Below is the detailed documentation of the dataset, including column descriptions, data types, and preprocessing steps.

5.1 Dataset Overview

The dataset consists of cleaned and enriched sales data that has undergone:

1. **Cleaning:** Addressing missing values, data type corrections, and schema validation**.**
2. **Transformation:** Addition of calculated fields, segmentation, and aggregations.
3. **Security:** Masking sensitive fields like Credit Card Numbers

5.2 Dataset Structure

Column Name	Data Type	Description
OrderID	String	Unique identifier for each order. Defaulted to Unknown if missing.
CustomerName	String	Name of the customer placing the order. Defaulted to Unknown if missing.
PhoneNumber	String	Contact number of the customer. Defaulted to Unknown if missing.
Location	String	Geographic location of the customer. Defaulted to Unknown if missing.
Country	String	Country of the customer. Defaulted to Unknown if missing.
StoreCode	String	Code of the store where the order was placed. Defaulted to Unknown if missing.
Product	String	Name of the product ordered. Defaulted to Unknown if missing.
Quantity	Integer	Number of units ordered. Defaulted to 10 if missing or invalid.
Price	Decimal	Price per unit of the product. Defaulted to 100 if missing or invalid.
Date	Date	Date when the order was placed. Defaulted to 1970-01-01 if invalid or missing.
CreditCardNumber	String	Masked credit card number (xxxx-xxxx-xxxx-XXXX) for security.
ExpiryDate	String	Expiry date of the credit card. Defaulted to Unknown if missing.
OrderMonth	Integer	Extracted month from the order date.
TotalSales	Decimal	Total sales value for the order (Quantity * Price).
AverageOrderValue	Decimal	Average value of an order (TotalSales / Quantity).
CustomerSegment	String	Customer segmentation (High-value, Medium-value, or Low-value) based on TotalSales.
MaskedCreditCard	String	Fully masked credit card number for security purposes.
ProfitMargin	Decimal	Calculated profit margin (Price * 0.2).
DiscountApplied	String	Indicates whether a discount was applied (Yes for orders with quantity > 10).
FullCustomerName	String	Combination of CustomerName and Location for enriched insights.

5.3 Data Preprocessing Steps

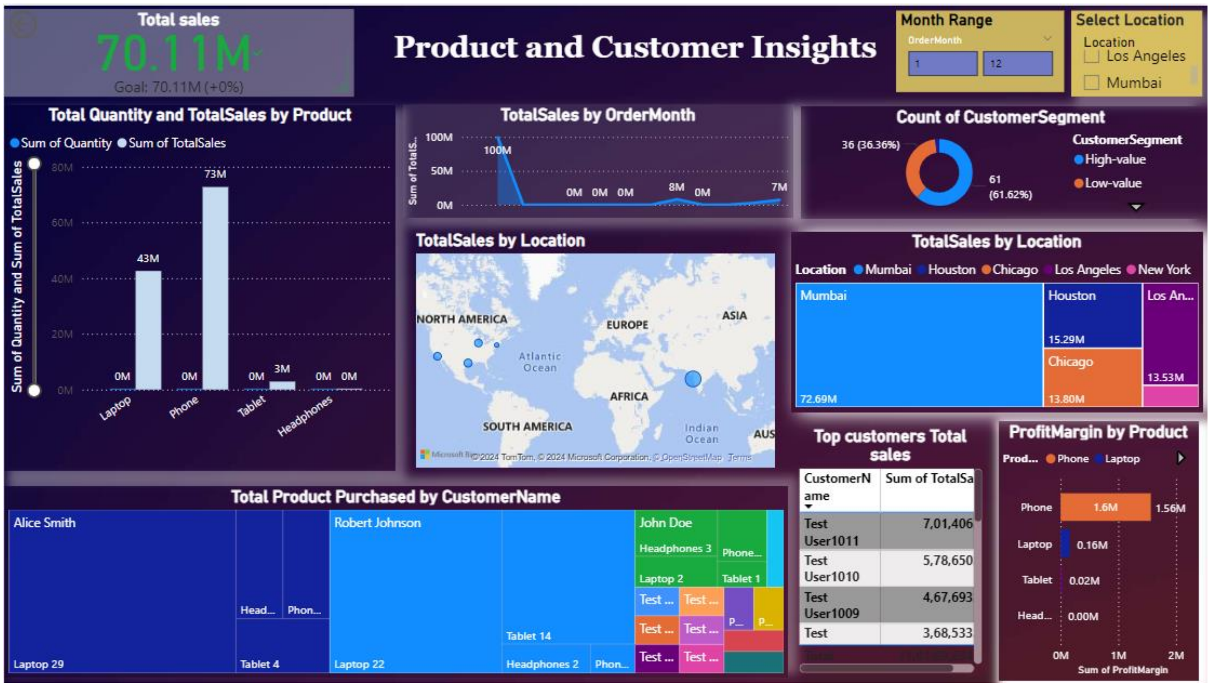
1. **Handling Missing Values:**
   - Replaced missing values in OrderID, CustomerName, PhoneNumber, Location, and Country with Unknown.
   - Defaulted missing numerical values in Quantity and Price to 10 and 100**, respectively.**
2. **Standardization:**
   - Converted Quantity to integer and Price to decimal.
   - Standardized the Date column to the format yyyy-MM-dd. Defaulted invalid or missing dates to 1970-01-01.
3. **Enrichment:**
   - Added calculated columns:
     - TotalSales = Quantity * Price.
     - AverageOrderValue = TotalSales / Quantity.
     - CustomerSegment based on TotalSales thresholds:
     - High-value: TotalSales > 5000
     - Medium-value: 200 <= TotalSales <= 5000
     - Low-value: TotalSales < 200
   - Created a MaskedCreditCard column to ensure sensitive data security.
   - Combined CustomerName and Location into FullCustomerName.
4. **Aggregations:**
   - Summarized sales data by Location and CustomerName for deeper insights**.**
5. **Security:**
   - Masked credit card numbers to display only the last four digits

6 Power BI Dashboard

I . Connection to Power BI

- The cleaned and transformed dataset stored in Azure Blob Storage is connected to Power BI via a shared URL.
- Steps:
  1. Open Power BI Desktop.
  2. Select "Get Data" → "Web" → Enter the Blob Storage URL of the dataset.
  3. Transform the data if necessary within Power BI.
  4. Build the charts and visuals.

CHART INTERPRETATION:



CustomerName	Sum of TotalSales
Test User1011	7,01,406
Test User1010	5,78,650
Test User1009	4,67,693
Test	3,68,533

Title: Product and Customer Insights Dashboard

Description:

This dashboard provides a holistic view of sales performance, customer behavior, and product profitability. Key metrics such as total sales, product-wise revenue, customer segmentation, and location-based sales trends are highlighted. The interactive filters for time and location enable deep dives into specific periods and regions, helping businesses identify high-performing products, loyal customers, and profitable markets to drive strategic decision-making

list of chart names with their respective chart types and titles:

1 KPI Chart: "Total Sales KPI Chart"

- **Chart Type: Key Performance Indicator (KPI)**

2 Bar Chart: "Total Quantity and Total Sales by Product"

- **Chart Type: Clustered Bar Chart**

3 Line Chart: "Monthly Sales Trends"

- **Chart Type: Line Chart**

4 Donut Chart: "Customer Segment Distribution"

- **Chart Type: Donut Chart**

5 Treemap: "Total Sales by Location (Treemap)"

- **Chart Type: Treemap**

6 Map View: "Geographic Distribution of Total Sales (Map View)"

- **Chart Type: Filled Map**

7 Table Chart: "Top Customers Total Sales"

- **Chart Type: Table**

8 Treemap: "Total Product Purchased by Customer"

- **Chart Type: Treemap**

9 Bar Chart: "Profit Margin by Product"

- **Chart Type: Stacked Bar Chart**

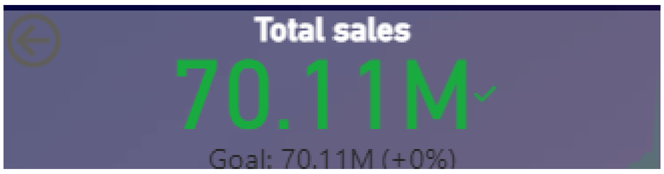
10 Slicer: "Interactive Month-Based Sales Analysis"

- **Chart Type: Slicer (with drill-through enabled)**

11 Slicer: "Location-Based Filtering for Sales Analysis"

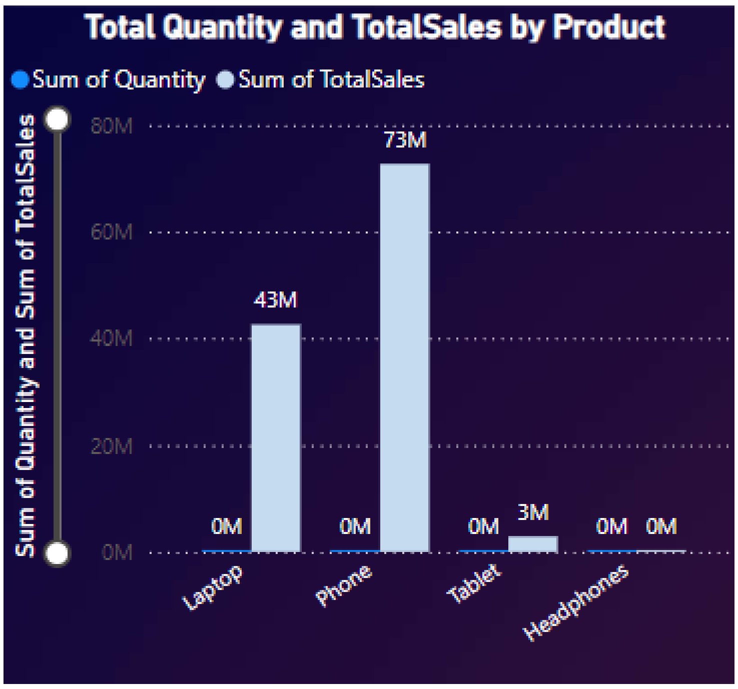
- **Chart Type: Slicer (with drill-through enabled)**

1.TOTAL SALES



- **Total Sales Achieved**: The organization successfully achieved its sales target of **70.11M**, reflecting strong performance.
- **Goal Comparison**: The exact match of target and achievement highlights precise planning and execution.
- **Positive Indicator**: The green checkmark confirms the target was met with no shortfall.
- **Stability in Performance**: Accurate projections demonstrate effective forecasting and operational consistency.
- **Future Focus**: Set higher sales goals to drive growth and challenge the team for better performance.

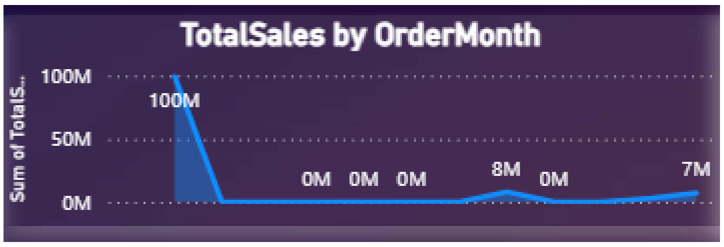
2. Product Sales Performance



- **Phones Lead in Revenue**: Despite a moderate quantity sold (**834 units**), phones generate the highest revenue at **72.71M**, indicating a high price point and strong demand.

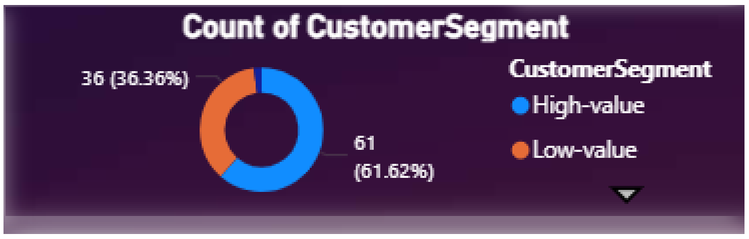
- **Laptops Drive Volume**: Laptops sold the most in quantity (**2674 units**) and contributed significantly to revenue (**42.57M**), suggesting they are a consistent revenue driver.
- **Low Impact of Headphones**: Headphones have minimal revenue (**23K**) and higher quantity sold (**56 units**), showing low pricing and limited market traction.
- **Tablet Underperformance**: Tablets sold **680 units** but generated only **2.88M**, indicating potential pricing issues or a declining market share.
- **Focus Areas**: Prioritize phones and laptops as primary revenue contributors while revisiting pricing and strategies for headphones and tablets.

3. Monthly Sales Trends



- **Peak in January**: January saw a dramatic sales spike at **100M**, likely due to seasonal or promotional factors.
- **Sales Decline**: After January, sales plummeted significantly, staying near zero for several months.
- **Small Recovery in Later Months**: Modest sales were observed in March (**7.86K**) and May (**7.64K**), showing slight recovery.
- **Seasonal Dependence**: The data suggests sales rely heavily on specific periods, indicating a need for consistent promotional efforts year-round.
- **Focus Area**: Investigate what drove the January sales surge and apply similar strategies to other months to stabilize performance.

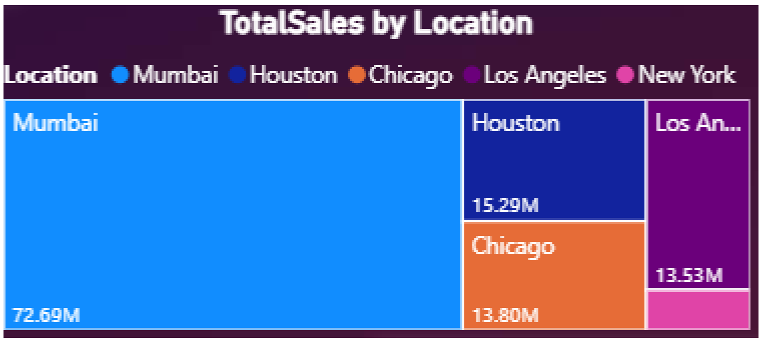
4.Customer segmentation to high or low based on their total purchase



- **High-Value Customers Dominate: The majority of customers (61.62%) belong to the high-value segment, indicating a significant focus on premium buyers.**
- **Low-Value Customers Follow: Low-value customers make up 36.36%, suggesting a smaller but notable market segment.**
- **Negligible Medium-Value Presence: The medium-value segment is almost non-existent (2 customers), highlighting a potential gap in mid-tier offerings.**

- **Strategic Opportunity**: Strengthen engagement with high-value customers while exploring growth opportunities in the medium-value segment.

5 Total Sales by Location



Interpretation:

1. **Mumbai Leads in Sales**: Mumbai dominates with **72.69M** in total sales, making it the top-performing location.
2. **Houston and Chicago Perform Well**: Houston (**15.29M**) and Chicago (**13.80M**) contribute significantly to overall sales.
3. **Low Contribution from New York**: New York has the lowest sales at **2.89M**, highlighting an underperforming market.
4. **Strategic Focus Needed**: Prioritize Mumbai for sustained growth while exploring strategies to boost sales in New York and other regions.

6. Geographic Distribution of Total Sales

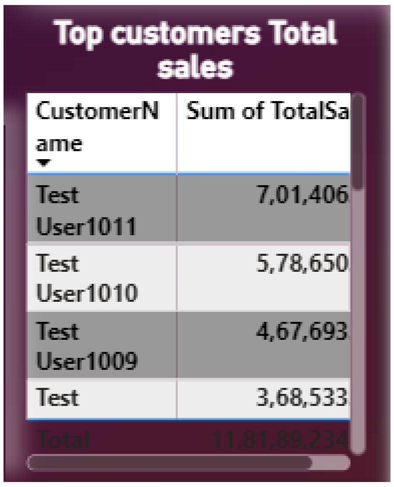
TotalSales by Location
NORTH AMERICA
EUROPE
ASIA
Atlantic Ocean
AFRICA
SOUTH AMERICA
Indian Ocean
AUS
© 2024 TomTom, © 2024 Microsoft Corporation, © OpenStreetMap Terms

Interpretation:

- **Mumbai as a Sales Hub**: The largest marker on the map indicates Mumbai as the leading contributor to total sales globally.

- **North American Presence**: Moderate sales are observed in locations like Houston, Chicago, Los Angeles, and New York, showing a consistent market in the U.S.
- **Underrepresented Regions**: Europe, Africa, and Australia lack representation, highlighting potential untapped markets.
- **Strategic Growth**: Focus on expanding successful strategies in North America to underrepresented regions while strengthening dominance in Mumbai.

7. Top Customers Driving Total Sales



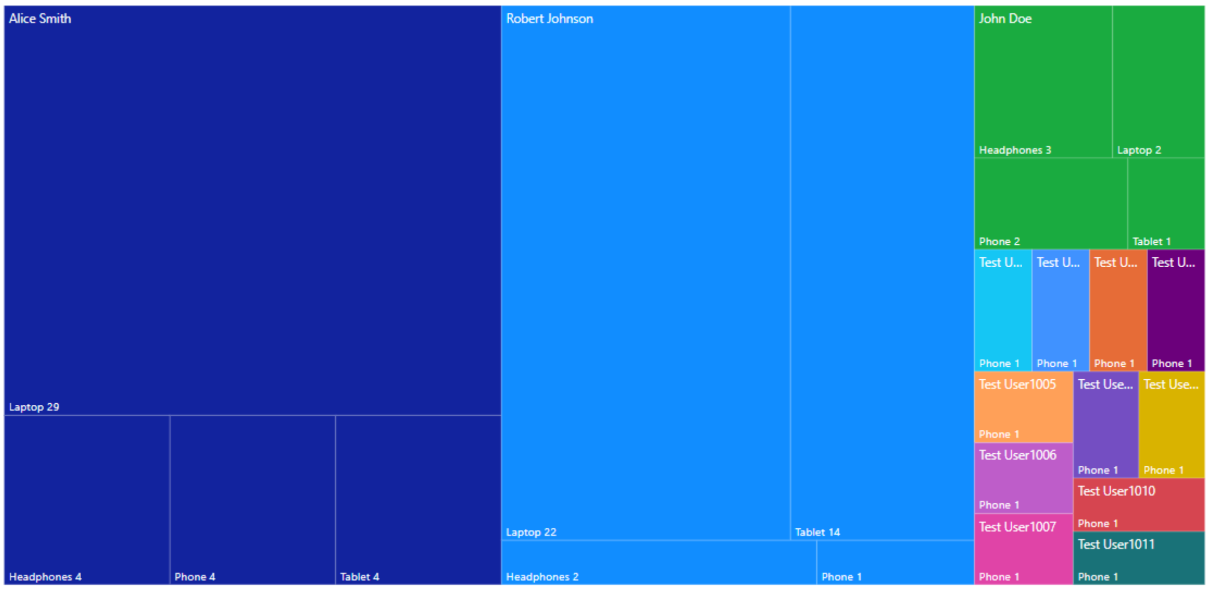
Top customers Total sales

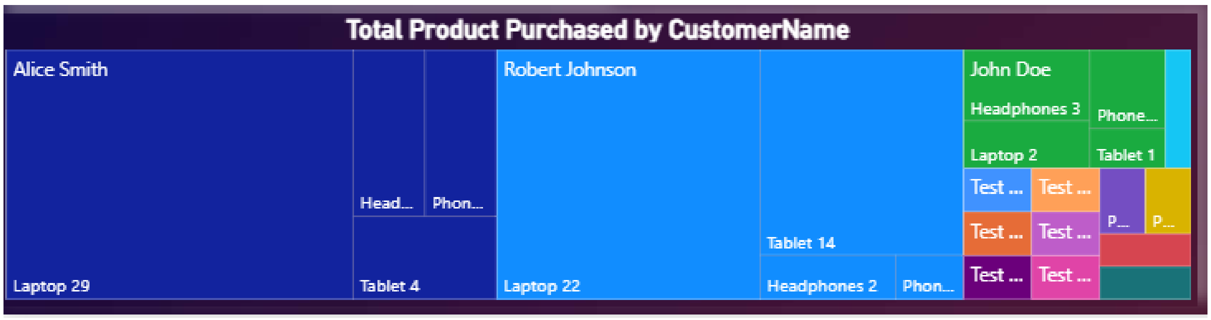
CustomerName	Sum of TotalSa
Test User1011	7,01,406
Test User1010	5,78,650
Test User1009	4,67,693
Test	3,68,533
Total	11,81,89,234

Interpretation:

- **Major Contributor**: "Test User1001" is the top customer, contributing a significant **70.11M** to total sales.
- **Diverse Customer Base**: Other notable contributors, such as "Robert Johnson" and "Alice Smith," contribute **24.48M** and **20.99M**, respectively.
- **Smaller Impact Customers**: Test users like "User1011" and "User1010" contribute significantly less, reflecting their smaller purchasing capacity.
- **Focus Area**: Retain and prioritize top customers while identifying strategies to increase sales from mid-tier contributors

8 Product Purchases by Customer

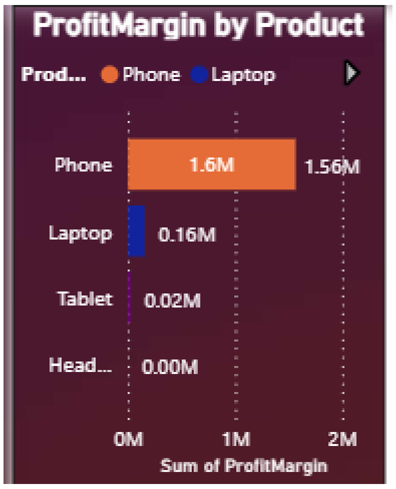




Interpretation:

1. **Alice Smith Leads**: Alice Smith is the top buyer, purchasing **29 laptops** and a small number of other products, reflecting a preference for high-value items.
2. **Diverse Buying Patterns**: Robert Johnson shows a more balanced purchase pattern, buying **22 laptops** and **14 tablets**, indicating interest in multiple product categories.
3. **Smaller Contributions**: Customers like John Doe and test users make smaller purchases, focusing on lower quantities and varied products.
4. **Insights for Strategy**: Focus on retaining high-value customers like Alice Smith and Robert Johnson while encouraging more diverse purchasing patterns among smaller buyers.

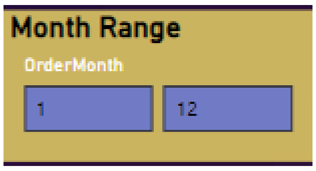
9 Title: Profit Margin by Product



Interpretation:

1. **Phones Lead in Profitability**: Phones generate the highest profit margin at **1.56M**, showcasing their significant contribution to overall profitability.
2. **Laptops as Secondary Contributors**: Laptops follow with a profit margin of **0.16M**, indicating steady but lower profitability compared to phones.
3. **Minimal Profits from Tablets and Headphones**: Tablets and headphones contribute negligible profit margins (**23K** and **790**, respectively), indicating limited revenue impact.
4. **Strategic Focus**: Prioritize phones for profitability and explore opportunities to improve margins for laptops, tablets, and headphones

10: Interactive Month-Based Sales Analysis

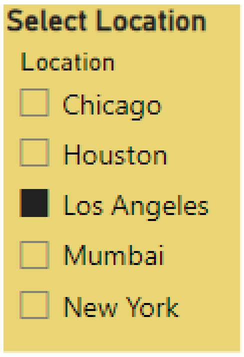


Interpretation:

1. **Drill-through Feature**: The chart allows users to drill through and filter sales data dynamically by selecting specific months, enabling detailed month-based insights.
2. **Custom Month Range**: The "Month Range" slicer allows users to focus on specific periods (e.g., January to December) to analyze sales trends for tailored reporting.

3. **Enhanced Flexibility**: Keeping all filters ensures consistency when navigating between reports, making it easier to compare and interpret seasonal trends.
4. **Strategic Use**: Utilize this feature to identify peak sales months and design targeted promotional strategies accordingly.

11 Location-Based Filtering for Sales Analysis


Select Location

Location

- ☐ Chicago
- ☐ Houston
- ■ Los Angeles
- ☐ Mumbai
- ☐ New York

Interpretation:

1. **Interactive Location Selection**: Users can filter data by selecting specific locations (e.g., Los Angeles), providing a focused view of sales performance in a chosen area.
2. **Drill-through Enabled**: The drill-through feature ensures that location-specific insights carry over across related reports, maintaining consistent filtering.
3. **Targeted Insights**: This flexibility allows businesses to analyze regional sales trends and identify underperforming or high-performing locations.
4. **Strategic Use**: Use this tool to tailor marketing efforts and resource allocation based on location-specific performance.

7. Security Considerations

7.1 Data Storage Security

- **RBAC:** Restricts access to authorized users only.
- **SAS Tokens:** Time-limited links for secure Power BI access.
- **Encryption:** Data is encrypted at rest and in transit.

7.2 Data Factory Security

- **Data Masking:** Hides sensitive fields like credit card numbers.
- **Secure Connections:** Uses secure identities for authentication.
- **Monitoring:** Logs and alerts track pipeline issues.

7.3 Power BI Security

- **Row-Level Security:** Limits data visibility by user roles.
- **Read-Only Access:** Ensures no edits to sensitive data.
- **Compliance:** Protects PII and meets GDPR standards.

7.4 Operational Security

- **Backups:** Prevents data loss with scheduled backups.
- **Firewall Rules:** Blocks unauthorized network access.
- **Audits:** Regular reviews ensure access is secure.

8. Conclusion

This project successfully demonstrates an end-to-end solution for managing, transforming, and visualizing sales data using Azure Data Lake Storage Gen2, Azure Data Factory, and Power BI. Key highlights include:

- **Scalable and Secure Architecture:** The solution ensures seamless handling of large datasets with robust security controls.
- **Clean and Enriched Data:** Pipelines efficiently clean and transform raw data into actionable insights.
- **Interactive Dashboards:** Power BI provides user-friendly, dynamic visuals for better decision-making.
- **Compliance and Security:** Data privacy and compliance with regulations like GDPR are prioritized.