ETL Data Engineering with Azure & Tata Store Data

Building a Cloud-Based Data Pipeline for Advanced Analytics

A hands-on project leveraging Azure Data Factory, Databricks, Synapse Analytics & Power BI

Muhammed Sinan
DSE MBA Business Analytics

Project Overview

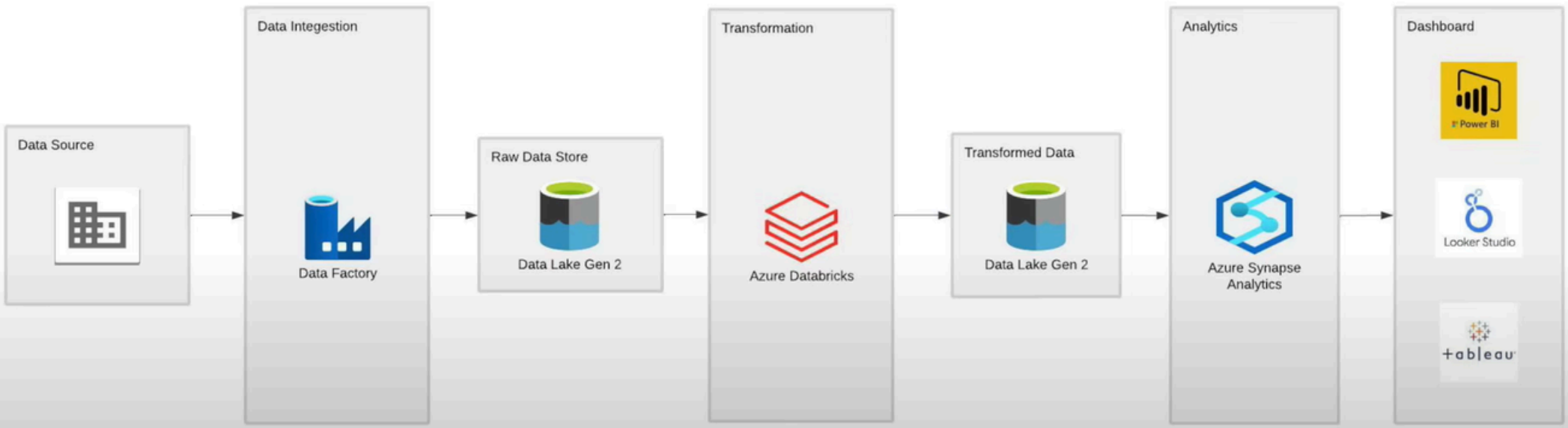
- **Dataset:** Tata Store Sales Data from Kaggle (541,910 rows, 9 columns)

- **Goal:** Build an end-to-end ETL pipeline for data transformation, warehousing, and visualization

- **Tools Used:** Azure Data Factory, Data Lake, Databricks (PySpark), Synapse Analytics, Power BI

- **Key Outputs:**
  Star Schema Data Model
  SQL-Based Business Analytics
  Interactive Power BI Dashboard

Project Workflow & Architecture



Data Ingestion → Data Storage → Data Transformation → Data Warehousing → Data Visualization

Data Ingestion & Storage

- **Source Data:** Tat Retail Dataset from Kaggle. Link
- **Ingestion:** Uploaded to GitHub → Connected to Azure Data Lake Storage via Azure Data Factory
- **Storage Management:**
  - Raw data stored in Raw Zone
  - Processed data stored in Transformed Zone

Data Transformation with Databricks (PySpark)

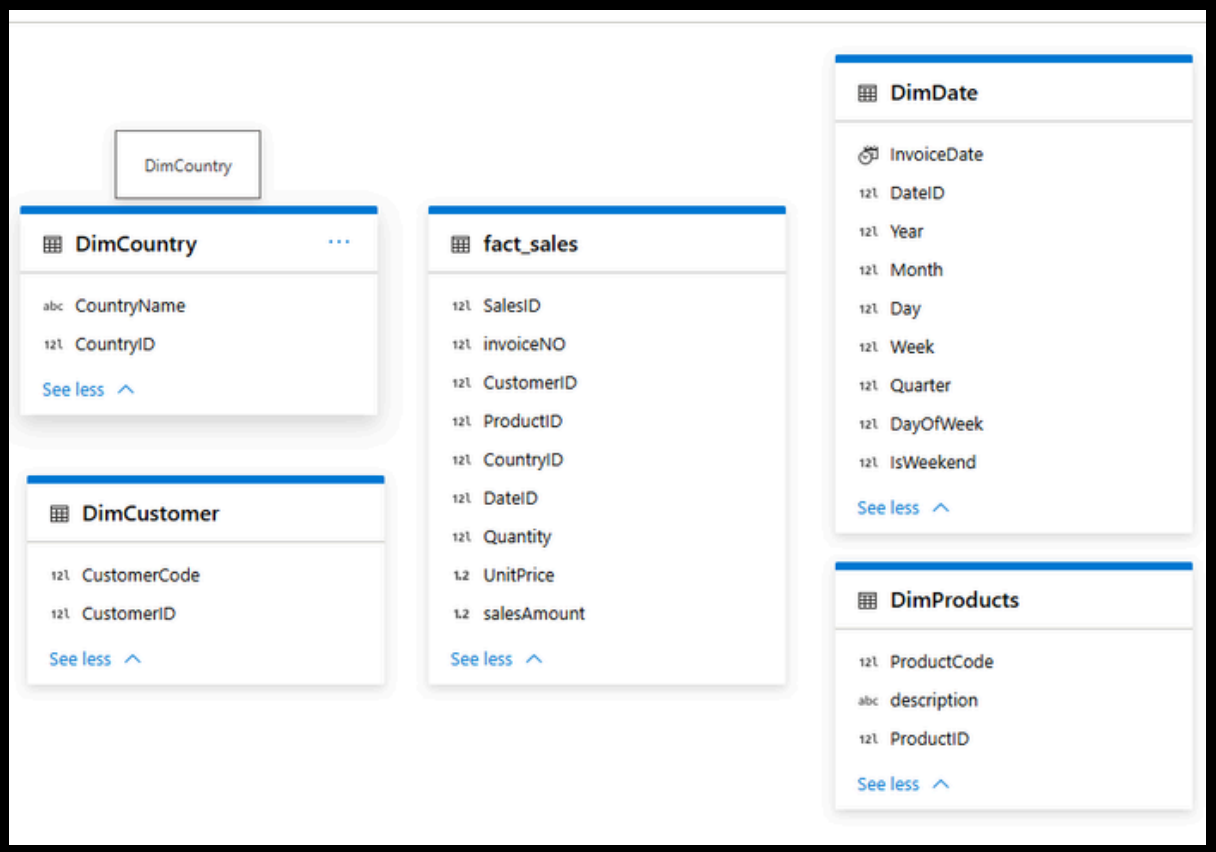
Goal: Convert flat data into a star schema model

Process:

Data Cleaning, Normalization, and Transformation using PySpark in Databricks

Star Schema Tables:

- Fact_Sales
- Dim_Date
- Dim_Product
- Dim_Customer
- Dim_Country



2 days ago (<1s) 1 Python

```python
from pyspark.sql.functions import col,to_timestamp,year,month,dayofmonth, weekofyear,regexp_replace,substring,monotonically_increasing_id,
round,when,dayofweek,quarter
```

2 days ago (24s) 2 Python

```python
%python
# Check if the directory is already mounted
if any(mount.mountPoint == '/mnt/tata-etl-data' for mount in dbutils.fs.mounts()):
    dbutils.fs.unmount('/mnt/tata-etl-data')

# Define the configurations
configs = {
    "fs.azure.account.auth.type": "OAuth",
    "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
    "fs.azure.account.oauth2.client.id": "901623e1-dc7c-41c2-92e1-8e9010c250ec",
    "fs.azure.account.oauth2.client.secret": 'BBT8Q~618OA6n3MSjfXqPYHLDfpCUmKFHu--Qas9',
    "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/b74ecb77-a71e-4179-b670-f60bd5ee677c/oauth2/token"
}

# Mount the directory
dbutils.fs.mount(
    source="abfss://tata-etl-data@tataetldata2.dfs.core.windows.net",  # container@storageacc
    mount_point="/mnt/tata-etl-data",
    extra_configs=configs
```

2 days ago (1s) 3

```python
%fs
ls "/mnt/tata-etl-data"
```

Table +

	path	name	size	modificationTime
1	dbfs:/mnt/tata-etl-data/raw-data/	raw-data/	0	1738738868000
2	dbfs:/mnt/tata-etl-data/transformed-data/	transformed-dat...	0	1738738877000

2 rows | 1.45s runtime Refreshed 2 days ago

2 days ago (1s) 4

```python
spark
```

✓ 2 days ago (1s) 5

```python
retail_data = spark.read.format("csv").option("header","true").load("/mnt/tata-etl-data/raw-data/retailData")
retail_data.show()
```

(2) Spark Jobs

retail_data: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
|   536365|   84406B|CREAM CUPID HEART...|        8|01-12-2010 08:26|     2.75|     17850|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|        6|01-12-2010 08:26|     3.39|     17850|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|        6|01-12-2010 08:26|     3.39|     17850|United Kingdom|
|   536365|    22752|SET 7 BABUSHKA NE...|        2|01-12-2010 08:26|     7.65|     17850|United Kingdom|
|   536365|    21730|GLASS STAR FROSTE...|        6|01-12-2010 08:26|     4.25|     17850|United Kingdom|
|   536366|    22633|HAND WARMER UNION...|        6|01-12-2010 08:28|     1.85|     17850|United Kingdom|
|   536366|    22632|HAND WARMER RED P...|        6|01-12-2010 08:28|     1.85|     17850|United Kingdom|
|   536367|    84879|ASSORTED COLOUR B...|       32|01-12-2010 08:34|     1.69|     13047|United Kingdom|
|   536367|    22745|POPPY'S PLAYHOUSE...|        6|01-12-2010 08:34|      2.1|     13047|United Kingdom|
|   536367|    22748|POPPY'S PLAYHOUSE...|        6|01-12-2010 08:34|      2.1|     13047|United Kingdom|
|   536367|    22749|FELTCRAFT PRINCES...|        8|01-12-2010 08:34|     3.75|     13047|United Kingdom|
```

✓ 2 days ago (1s) 7 Python

```python
from pyspark.sql.functions import col, to_timestamp, regexp_replace

# Cast columns to appropriate data types and clean invoiceID
retail_data = retail_data.withColumn("Quantity", col("Quantity").cast("int")) \
                         .withColumn("UnitPrice", col("UnitPrice").cast("double")) \
                         .withColumn("CustomerID", col("CustomerID").cast("int")) \
                         .withColumn("InvoiceDate", to_timestamp("InvoiceDate", "MM-dd-yyyy HH:mm")) \
                         .withColumn("InvoiceNo", regexp_replace(col("InvoiceNo"), r"[^0-9]", ""))\
                         .withColumn("StockCode", regexp_replace(col("StockCode"), r"[^0-9]", ""))\
                         .withColumn("description", col("description").cast("string"))\
                         .withColumn("description", regexp_replace(col("description"), r"[^\\x00-\\x7F]", ""))

# Display updated schema
retail_data.printSchema()

# Display the data to check the changes
display(retail_data)
```

(1) Spark Jobs

retail_data: pyspark.sql.dataframe.DataFrame

```
InvoiceNo: string
StockCode: string
description: string
Quantity: integer
InvoiceDate: timestamp
```

2 days ago (1s) 9 Python

```python
DimProduct = retail_data.select("StockCode","Description")\
                .distinct()\
                .withColumn("ProductID",monotonically_increasing_id())\
                .withColumnRenamed("StockCode", "ProductCode")\
                .withColumn("description", substring("description", 1, 100))

display(DimProduct)
```

(3) Spark Jobs

DimProduct: pyspark.sql.dataframe.DataFrame = [ProductCode: string, description: string ... 1 more field]

Table

	ProductCode	description	ProductID
1	85231	ORANGESCENTEDSET9TLIGHTS	0
2	22335	HEARTDECORATIONPAINTEDZINC	1
3	22649	STRAWBERRYFAIRYCAKETEAPOT	2
4	84459	YELLOWMETALCHICKENHEART	3
5	21161	KEEPOUTBOYSDOORHANGER	4
6	21784	SHOESHINEBOX	5

2 days ago (1s) 11: h Python

```python
DimCountry = retail_data.select("Country") \
                .distinct() \
                .withColumnRenamed("Country", "Countryname") \
                .withColumn("CountryID",monotonically_increasing_id())

display(DimCountry)
```

(2) Spark Jobs

DimCountry: pyspark.sql.dataframe.DataFrame = [Countryname: string, CountryID: long]

Table

	Countryname	CountryID
1	Sweden	0
2	Singapore	1
3	Germany	2
4	France	3
5	Greece	4
6	Belgium	5
7	Finland	6

2 days ago (4s) 10

```python
DimDate = retail_data.select("InvoiceDate") \
            .distinct() \
            .withColumn("DateID",monotonically_increasing_id())\
            .withColumn("Year", year("InvoiceDate")) \
            .withColumn("Month", month("InvoiceDate")) \
            .withColumn("Day", dayofmonth("InvoiceDate")) \
            .withColumn("Week", weekofyear("InvoiceDate"))\
            .withColumn("Quarter", quarter("InvoiceDate")) \
            .withColumn("DayOfWeek", dayofweek("InvoiceDate")) \
            .withColumn("IsWeekend", when(col("DayOfWeek").isin([6, 7]), 1).otherwise(0))

display(DimDate)
```

(2) Spark Jobs

DimDate: pyspark.sql.dataframe.DataFrame = [InvoiceDate: timestamp, DateID: long ... 7 more fields]

Table

	InvoiceDate	DateID	Year	Month	Day	Week	Quarter	DayOfWeek	IsWeeke
1	2010-01-12T11:41:00.000+00:...	0	2010	1	12	2	1	3	
2	2010-02-12T11:45:00.000+00:...	1	2010	2	12	6	1	6	
3	2010-02-12T12:59:00.000+00:...	2	2010	2	12	6	1	6	
4	2010-06-12T15:25:00.000+00:...	3	2010	6	12	23	2	7	
5	2010-07-12T09:28:00.000+00:...	4	2010	7	12	28	3	2	

2 days ago (1s) 8 Python

```python
DimCustomer = retail_data.select("CustomerID") \
            .distinct() \
            .withColumnRenamed("CustomerID","CustomerCode")\
            .withColumn("CustomerID",monotonically_increasing_id())

display(DimCustomer)
```

(2) Spark Jobs

DimCustomer: pyspark.sql.dataframe.DataFrame = [CustomerCode: integer, CustomerID: long]

Table

	CustomerCode	CustomerID
1	17420	0
2	16861	1
3	16503	2
4	15727	3
5	17389	4
6	15447	5

```python
from pyspark.sql.functions import monotonically_increasing_id, round

# Create the fact_sales table with a unique SalesID and round the salesAmount
fact_sales = retail_data.select(
    monotonically_increasing_id().alias("SalesID"),
    col("InvoiceNo").alias("invoiceNO"),
    col("StockCode").alias("productCode"),
    col("CustomerID"),
    col("InvoiceDate"),
    col("Quantity"),
    col("UnitPrice"),
    round((col("Quantity") * col("UnitPrice")), 1).alias("salesAmount"),  # Round to 1 decimal
    col("Country")
)

# Join fact_sales with the dimension tables to fetch the corresponding foreign keys
fact_sales = fact_sales.join(DimCustomer, fact_sales.CustomerID == DimCustomer.CustomerCode, "left") \
                       .join(DimProduct, fact_sales.productCode == DimProduct.ProductCode, "left") \
                       .join(DimCountry, fact_sales.Country == DimCountry.Countryname, "left") \
                       .join(DimDate, fact_sales.InvoiceDate == DimDate.InvoiceDate, "left")

# Remove any duplicates after the join (if any) and reset SalesID
fact_sales = fact_sales.distinct()

# Reassign SalesID to ensure uniqueness
```

✓ 2 days ago (12s) 12

```python
# Select the required columns with correct foreign key references
fact_sales = fact_sales.select(
    "SalesID",  # Unique SalesID
    "invoiceNO",  # Corrected column name
    DimCustomer.CustomerID.alias("CustomerID"),  # Foreign Key for Customer
    DimProduct.ProductID.alias("ProductID"),  # Foreign Key for Product
    DimCountry.CountryID.alias("CountryID"),  # Foreign Key for Country
    DimDate.DateID.alias("DateID"),  # Foreign Key for Date
    "Quantity",
    "UnitPrice",
    "salesAmount"
)

fact_sales = fact_sales.filter(
    (fact_sales.Quantity.isNotNull()) &
    (fact_sales.UnitPrice.isNotNull()) &
    (fact_sales.CustomerID.isNotNull())
)

# Check the schema and display the result
fact_sales.printSchema()
display(fact_sales)
```

2 days ago (25s) 13

```
fact_sales.repartition(1).write.mode("overwrite").option("header","true").csv("/mnt/tata-etl-data/transformed-data/fact_sales/")
DimCustomer.repartition(1).write.mode("overwrite").option("header","true").csv("/mnt/tata-etl-data/transformed-data/DimCustomer/")
DimProduct.repartition(1).write.mode("overwrite").option("header","true").csv("/mnt/tata-etl-data/transformed-data/DimProduct/")
DimCountry.repartition(1).write.mode("overwrite").option("header","true").csv("/mnt/tata-etl-data/transformed-data/DimCountry/")
DimDate.repartition(1).write.mode("overwrite").option("header","true").csv("/mnt/tata-etl-data/transformed-data/DimDate/")
```

(26) Spark Jobs

All the PySpark code will process and transform the raw data, resulting in the structured dataset shown below.

Home > Storage accounts > tataetldata2 | Containers >

tata-etl-data
Container

Search

- Overview
- Diagnose and solve problems
- Access Control (IAM)
- Settings
  - Shared access tokens
  - Manage ACL
  - Access policy
  - Properties
  - Metadata

Upload | Add Directory | Refresh | Rename | Delete | Change tier | Acquire lease | Break lea

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: tata-etl-data / transformed-data

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archive status
[..]			
DimCountry	2/6/2025, 11:02:29 AM		
DimCustomer	2/6/2025, 11:02:25 AM		
DimDate	2/6/2025, 11:02:33 AM		
DimProduct	2/6/2025, 11:02:27 AM		
fact_sales	2/6/2025, 11:02:19 AM		

Data Warehousing with Azure Synapse Analytics

Goal: Store structured data & run SQL-based analytics

Key SQL Queries & Insights:

- Retention Rate & Churn Rate Analysis
- Country-wise Sales Analytics
- Customer Segmentation (New vs. Returning Customers)
- Fast-Moving Products Analysis

SQL Query for
1- Basic overview of sales performance

TataRetailDataBase ● Basic Sales Summary ×

Run Undo Publish Query plan Connect to Built-in Use database TataRetailDataBase

```sql
/* This query provides a high-level overview of sales performance */

SELECT
    COUNT(DISTINCT invoiceNO) AS Unique_Invoices,
    SUM(Quantity) AS Total_Quantity_Sold,
    ROUND(sum(salesAmount),1) AS Total_Sales_Amount
FROM fact_sales;
```

Result

Unique_Invoices	Total_Quantity_Sold	Total_Sales_Amount
22190	8168719	14637117.3

SQL Query for 2-Peak Sales Time Analysis

```sql
/* Peak Sales Time Analysis */

SELECT
    d.DayOfWeek,
    COUNT(DISTINCT f.invoiceNO) AS TotalTransactions,
    ROUND(SUM(f.salesAmount),1) AS TotalRevenue
FROM fact_sales f
JOIN DimDate d ON f.DateID = d.DateID
GROUP BY d.DayOfWeek
ORDER BY TotalRevenue DESC
```

Result

DayOfWeek	TotalTransactions	TotalRevenue
3	1541	1271992.5
6	1597	1050076.6
1	1435	959291
2	1204	741499.1

SQL Query for 3-Average Spending By Customer

```sql
SELECT
    d.CountryName as country,
    COUNT(DISTINCT f.invoiceNO) as UniqueInvoices,
    COUNT(DISTINCT f.customerID) as UniqueCustomers,
    ROUND(sum(f.salesAmount),1) as TotalRevenue,
    ROUND(sum(f.salesAmount),1)/COUNT(DISTINCT f.customerID) AS AverageSpendingByCustomer
FROM fact_sales f
JOIN DimCountry d on f.countryID = d.countryID
GROUP BY d.CountryName
ORDER BY TotalRevenue DESC
```

s Messages

Result

country	UniqueInvoices	UniqueCustomers	TotalRevenue	AverageSpendingByCustomer
United Kingdom	19857	3950	11294742.3	2859.428430379747
Germany	603	95	599383.9	6309.304210526316
France	458	87	491241	5646.448275862069
Netherlands	101	9	484468.4	53829.822222222225

SQL Query for
4-Cross sell and up-sell Oppertunities

```sql
/* Cross-Sell and Up-sell Oppertunities */

SELECT
    a.ProductID AS ProductA,
    b.ProductID As ProductB,
    COUNT(*) AS PairCount
FROM fact_sales a
JOIN fact_sales b ON a.invoiceNO = b.invoiceNO AND a.ProductID < b.ProductID
GROUP BY a.ProductID,b.ProductID
ORDER BY PairCount DESC
```

Result

View Table Chart Export results

Search

ProductA	ProductB	PairCount
1158	2940	5158
1158	3828	5158
1158	3180	5158
3180	3828	5158

SQL Query for 5-Fast-moving Products

```sql
/* Fast-Moving Products */

SELECT Top 10
    d.ProductCode,
    COUNT(DISTINCT f.SalesID) AS TotalSales,
    ROUND(SUM(f.salesAmount), 1) AS TotalRevenue
FROM fact_sales f
JOIN DimProducts d ON f.productID = d.productID
WHERE d.ProductCode IS NOT NULL
Group By d.ProductCode
ORDER By TotalSales DESC
```

Result

View Table Chart Export results

Search

ProductCode	TotalSales	TotalRevenue
85049	15026	134290.2
85099	12048	546434

SQL Query for 6-Customer Purchase Frequency

```sql
/* Customer Purchase Frequency */

SELECT TOP 10
    f.CustomerID,
    COUNT(DISTINCT f.invoiceNO) as PurchaseFrequency,
    ROUND(SUM(f.salesAmount),1) as TotalRevenue
FROM fact_sales f
GROUP BY f.customerID
ORDER By TotalRevenue DESC

/* This query
- Helps identify high-value customers (HVCs) who contribute the most revenue.
- Shows which customers are frequent buyers and which ones make one-time purchases.
- Can be used for loyalty programs or targeted marketing campaigns. */
```

Result

CustomerID	PurchaseFrequency	TotalRevenue
1119	77	470899.2
339	62	334332.4
1334	55	310508.1

00:00:01 Query executed successfully.

SQL Query for 7-New Vs Retaining Customer Analysis

```sql
WITH FirstPurchase AS (
    SELECT
        customerID,
        MIN(d.DateID) As FirstPurchaseDateID
    FROM fact_sales f
    JOIN DimDate d on f.DateID = d.DateID
    Group By customerID
)
SELECT
    d.Year,
    d.Month,
    COUNT(DISTINCT CASE WHEN f.DateID = fp.FirstPurchaseDateId THEN f.customerID END) AS NewCustomers,
    COUNT(DISTINCT CASE WHEN f.DateID > fp.FirstPurchaseDateId THEN f.customerID END) AS ReturningCustomer
FROM fact_sales f
JOIN DimDate d ON f.DateID = d.DateID
JOIN FirstPurchase fp ON f.customerID = fp.customerID
GROUP By d.Year, d.Month
ORDER By d.Year DESC, d.Month DESC
```

Result

Year	Month	NewCustomers	ReturningCustomer
2011	12	170	284
2011	11	229	372
2011	10	237	329
2011	9	196	323
2011	8	217	390
2011	7	243	412
2011	6	268	426

SQL Query for 8-Retention and Churn Rates

TataRetailDataBase | Basic Sales Summary | Cross-Sell And Up-S... | Customer Loyalty Se... | Customer Purchase F... | Customer Retention...

Run | Undo | Publish | Query plan | Connect to: Built-in | Use database: TataRetailDataBase

```sql
WITH monthlySales AS (
    SELECT
        f.customerID,
        d.Year,
        d.Month,
        COUNT(DISTINCT f.invoiceNO) As purchaseCount
    FROM fact_sales f
    JOIN DimDate d on (f.DateID = d.DateID)
    GROUP By f.customerID,d.Year,d.Month
),
Retention as (
    SELECT
        ms.Year,
        ms.Month,
        COUNT(DISTINCT ms.customerID) AS ActiveCustomers,
        COUNT(DISTINCT CASE WHEN prev_ms.customerID IS NOT NULL THEN ms.customerID END) AS ReturningCustomers
    FROM monthlySales ms
    LEFT JOIN monthlySales prev_ms
        ON ms.customerID = prev_ms.customerID
        AND ms.Year = prev_ms.Year
        AND ms.Month = prev_ms.Month+1
    GROUP By ms.Year,ms.Month
)
SELECT
    Year,
    Month,
    ActiveCustomers,
    ReturningCustomers,
    ROUND(100 * ReturningCustomers  / NULLIF(ActiveCustomers,0) , 2) As RetentionRate,
```

Result

Year	Month	ActiveCustomers	ReturningCustomers	RetentionRate	ChurnRate
2011	12	428	116	27	73
2011	11	568	131	23	77
2011	10	537	111	20	80
2011	9	494	129	26	74
2011	8	572	159	27	73

00:00:03 Query executed successfully.

SQL Query for 9-Customer Segmentation

```sql
with customerActivity AS (
    SELECT
        f.customerID,
        COUNT(Distinct f.invoiceNO) AS Purchasefrequency,
        DATEDIFF(DAY, Max(d.invoiceDate), GETDATE()) AS Recency,
        ROUND(SUM(f.salesAmount),1) AS TotalSpending
    FROm fact_sales f
    JOIN DimDate d on f.DateID = d.DateID
    GROUP By f.CustomerID),
CustomerSegments AS(
    SELECT
    ca.customerID,
    ca.Purchasefrequency,
    ca.TotalSpending,
    ca.Recency,
    CASE
        WHEN ca.PurchaseFrequency > 100 AND ca.TotalSpending >5000 THEN 'VIP Customer'
        WHEN ca.PurchaseFrequency BETWEEN 50 AND 100 THEN 'Loyal Customer'
        WHEN ca.PurchaseFrequency  = 1 THEN 'New Customer'
        WHEN ca.Recency BETWEEN 10 AND 100 THEN 'At-Risk Customer'
        ELSE 'Churned Customer'
    END AS CustomerSegment
FROM customerActivity ca)
SELECT
    CustomerSegment,
    COUNT(*) AS CustomerCount
FROM CustomerSegments cs
GROUP By cs.CustomerSegment
ORDER By CustomerCount DESC
```

Result

CustomerSegment	CustomerCount
Churned Customer	1742
New Customer	1378
Loyal Customer	4
VIP Customer	1

SQL Query for 10-Country wise revenue Analysis

```sql
/* This query helps compare each country’s revenue contribution to the total sales.*/

SELECT
    d.CountryName as country,
    COUNT(DISTINCT f.invoiceNO) as UniqueInvoices,
    COUNT(DISTINCT f.customerID) as UniqueCustomers,
    ROUND(sum(f.salesAmount),1) as TotalRevenue,
    ROUND(sum(f.salesAmount),1)/COUNT(DISTINCT f.customerID) AS AverageSpendingByCustomer
FROM fact_sales f
JOIN DimCountry d on f.countryID = d.countryID
GROUP BY d.CountryName
ORDER BY TotalRevenue DESC
```

Messages

Result

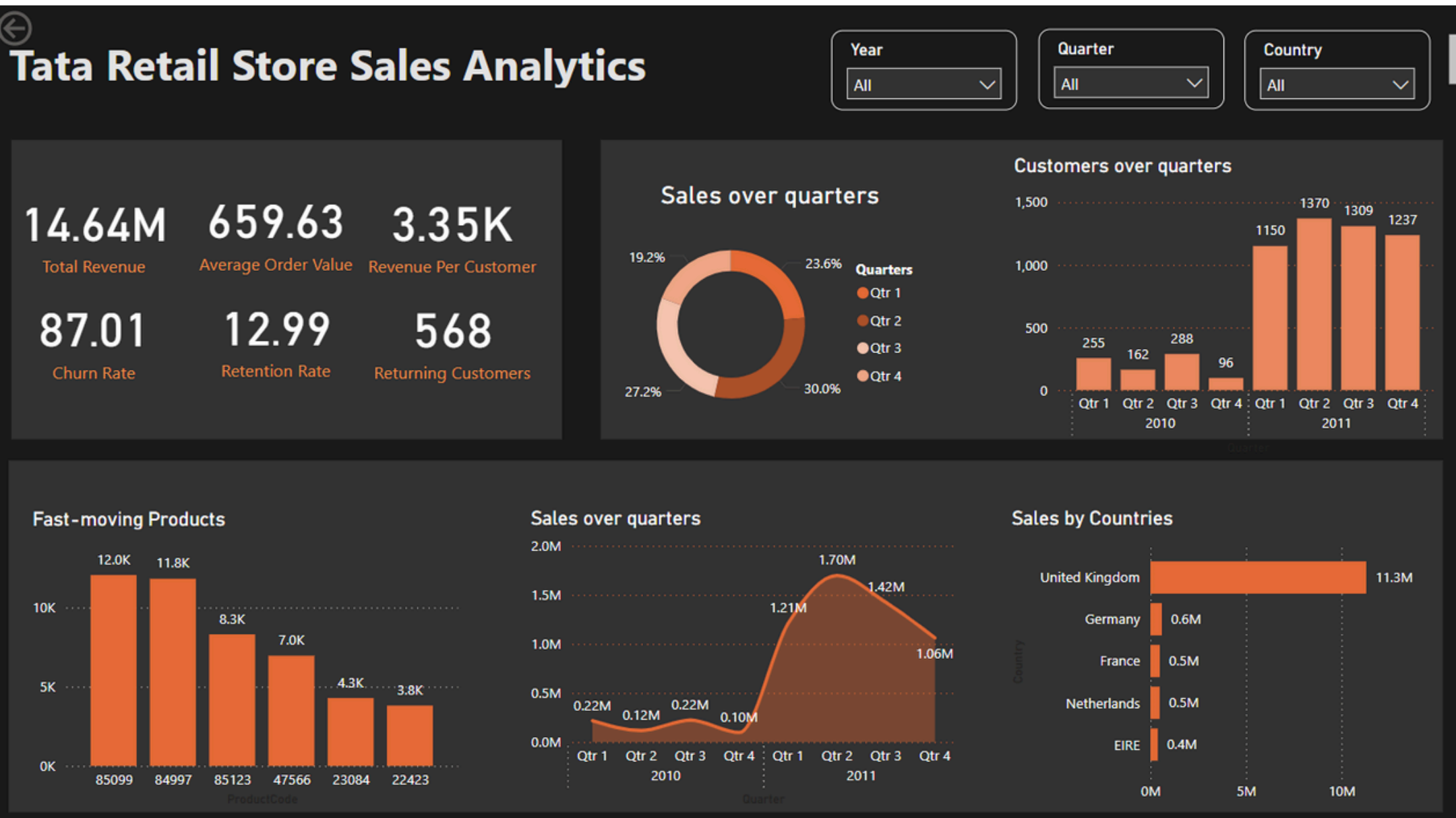
country	UniqueInvoices	UniqueCustomers	TotalRevenue	AverageSpendingByCustom
United Kingdom	19857	3950	11294742.3	2859.428430379747
Germany	603	95	599383.9	6309.304210526316
France	458	87	491241	5646.448275862069
Netherlands	101	9	484468.4	53829.822222222225

Data Visualization with Power BI

Goal: Build an interactive decision-support dashboard

Key Visuals in Power BI Dashboard:

- Revenue Trends & Growth Analysis 📊
- Customer Behavior: New vs. Returning Customers 👥
- Product Performance Analysis 📦
- Sales Breakdown by Country 🌍



Key Learnings & Takeaways

- Skills Gained:
- ✅ Cloud-Based Data Engineering with Azure
- ✅ Data Transformation with PySpark in Databricks
- ✅ Data Warehousing with Azure Synapse Analytics
- ✅ Data Visualization with Power BI

- Challenges Faced & Solutions:
  - Handling large datasets efficiently
  - Optimizing SQL queries for performance
  - Data schema design for better reporting
  - Data Cleaning

Thank You & Let's Connect!

"Data is the new oil. But refining it makes all the difference!"

LinkedIn | GitHub | sinankmuriyanal@gmail.com

Open to feedback, discussions, and collaboration!