**Customer 360 Data Integration**

Bootcamp Project - 3

**Overview**

A retail business wants to build a unified Customer 360 view by integrating data from multiple sources, including online transactions, in-store purchases, customer service interactions, and loyalty programs. This project uses a mix of fact and dimension tables to ensure a clean, scalable structure.

The source table are:

* Customers
* Products
* Online transactions
* Stores
* Instore Transactions
* Agents
* Customer Service Interactions
* Loyal Accounts
* Loyal Transactions

**Architecture of the pipeline:**

**A group of blue and white logos

AI-generated content may be incorrect.**

**Data from various to Raw Storage**:

Backend data: Where all the data is stored from various sourcesA screenshot of a computer

AI-generated content may be incorrect.

**Related Linked Services:**

A screenshot of a computer

AI-generated content may be incorrect.

**Related Data sets:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**ADF:**

Using Copy data activity, Data from backend is transformed to Raw container.

A screenshot of a computer

AI-generated content may be incorrect.

Here we used Wildpath to make sure same file names appear on the raw container.

A screenshot of a computer

AI-generated content may be incorrect.

* **Raw Storage to Stagging Database:**

Cleaned data operations are performed and stored in Stagging database

Created related datasets and executed the respective pipeline.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* **Stagging to Dedicated pool:**

In this area, we are creating 4 tables as an output in SQL dB from the stagging data.

**Table 1 - Average Order Value (AOV)**

* Source Tables: Online Transactions, Instore Transactions, Products, Stores
* Logic:
  + Join **Online Transactions** with **Products** (for product and category details).
  + Join **Instore Transactions** with **Stores** (for location details).
  + Combine both transaction data.
  + Calculate **SUM(Amount) / COUNT(OrderID)** grouped by product, category, and location.

SQL script used:

CREATE TABLE AverageOrderValue (

ProductID INT,

ProductName VARCHAR(100),

Category VARCHAR(50),

Location VARCHAR(100),

TotalAmount DECIMAL(18, 2),

OrderCount INT,

AverageOrderValue DECIMAL(18, 2));

The following SQL query will join the tables, calculate totals, and compute the Average Order Value:

INSERT INTO AverageOrderValue

SELECT

COALESCE(O.ProductID, I.ProductID) AS ProductID,

P.Name AS ProductName,

P.Category,

S.Location,

SUM(COALESCE(O.Amount, I.Amount)) AS TotalAmount,

COUNT(COALESCE(O.OrderID, I.TransactionID)) AS OrderCount,

SUM(COALESCE(O.Amount, I.Amount)) /

NULLIF(COUNT(COALESCE(O.OrderID, I.TransactionID)), 0) AS AverageOrderValue

FROM OnlineTransactions O

LEFT JOIN Products P ON O.ProductID = P.ProductID

FULL OUTER JOIN InStoreTransactions I ON O.CustomerID = I.CustomerID

LEFT JOIN Stores S ON I.StoreID = S.StoreID

WHERE COALESCE(O.Amount, I.Amount) IS NOT NULL -- Null value cleanup

GROUP BY

COALESCE(O.ProductID, I.ProductID),

P.Name,

P.Category,

S.Location;

**Table 2 - Customer Segmentation**

* Source Tables: Customers, Online Transactions, Instore Transactions, Loyalty Accounts
* Logic:
  + Calculate total spend per customer by combining **Online Transactions** and **Instore Transactions.**
  + Calculate purchase frequency (count of transactions).
  + Join with **Loyalty Accounts** for tier-level information.
  + Define segmentation logic for "High-Value Customers," "One-Time Buyers," etc.

**Table 3 - Peak Activity Analysis**

* Source Tables: Online Transactions, InStoreTransactions
* Logic:
  + Extract Date Time details like day of the week and hour.
  + Aggregate by peak days and times for both in-store and online transactions.

In Synapse, developing, I have created a new script called, **creating aggregate table-3**

here I have used SQL for doing my aggregate table **Peak Activity Analysis.**

**SQL script used:**

INSERT INTO PeakActivityAnalysis

SELECT

    'Online' AS TransactionType,

    DATENAME(WEEKDAY, DateTime) AS DayOfWeek,

    DATEPART(HOUR, DateTime) AS HourOfDay,

    COUNT(\*) AS TransactionCount

FROM OnlineTransactions

WHERE DateTime IS NOT NULL  -- Null value cleanup

GROUP BY DATENAME(WEEKDAY, DateTime), DATEPART(HOUR, DateTime)

UNION ALL

SELECT

    'In-Store' AS TransactionType,

    DATENAME(WEEKDAY, DateTime) AS DayOfWeek,

    DATEPART(HOUR, DateTime) AS HourOfDay,

    COUNT(\*) AS TransactionCount

FROM InStoreTransactions

WHERE DateTime IS NOT NULL  -- Null value cleanup

GROUP BY DATENAME(WEEKDAY, DateTime), DATEPART(HOUR, DateTime);

Now my data got transferred to **Dedicated Pool** and table got executed.

A screenshot of a computer

AI-generated content may be incorrect.

**Table 4 - Agent Performance**

* Source Tables: Customer Service Interactions, Agents
* Logic:
  + Count total interactions per agent.
  + Calculate resolution success rate as:

Resolution Success Rate = COUNT (Resolution Status = 'Resolved') / COUNT (Interaction ID)

The following process is an alternative where **DATAFLOW** is used instead of creating an SQL script in Synapse.

**Step 1: Create the Peak Activity Analysis Table**

In your Synapse dedicated pool, create the new table with appropriate schema:

sql

CopyEdit

CREATE TABLE PeakActivityAnalysis (

TransactionType VARCHAR(20), -- 'Online' or 'In-Store'

DayOfWeek VARCHAR(20),

HourOfDay INT,

TransactionCount INT

);

**Step 2: Develop the Data Flow Logic in Azure Data Factory (ADF)**

1. **Source Transformation:**
   * Add sources for the OnlineTransactions and InStoreTransactions tables.
2. **Derived Column Transformation:**
   * Extract DayOfWeek and HourOfDay from the DateTime column:

sql

CopyEdit

DATENAME(WEEKDAY, DateTime) AS DayOfWeek,

DATEPART(HOUR, DateTime) AS HourOfDay

1. **Union Transformation:**
   * Combine both transaction types and tag them with identifiers:

sql

CopyEdit

SELECT

'Online' AS TransactionType,

DATENAME(WEEKDAY, DateTime) AS DayOfWeek,

DATEPART(HOUR, DateTime) AS HourOfDay

FROM OnlineTransactions

UNION ALL

SELECT

'In-Store' AS TransactionType,

DATENAME(WEEKDAY, DateTime) AS DayOfWeek,

DATEPART(HOUR, DateTime) AS HourOfDay

FROM InStoreTransactions

1. **Aggregate Transformation:**
   * Group by TransactionType, DayOfWeek, and HourOfDay, and count transactions:

sql

CopyEdit

SELECT

TransactionType,

DayOfWeek,

HourOfDay,

COUNT(\*) AS TransactionCount

FROM CombinedTransactions

GROUP BY TransactionType, DayOfWeek, HourOfDay;

1. **Null Value Cleanup (if needed):**
   * Use ISNULL() or COALESCE() to handle nulls:

sql

CopyEdit

ISNULL(DayOfWeek, 'Unknown') AS DayOfWeek,

ISNULL(HourOfDay, 0) AS HourOfDay

1. **Sink Transformation:**
   * Connect this to the PeakActivityAnalysis table in your Synapse dedicated pool.

**Dedicated pool to SQL Db:**

Here I connected my Synapse Dedicated pool to my SQL Server Management Studio (SSMS) and executed the table **Peak Activity Analysis.**

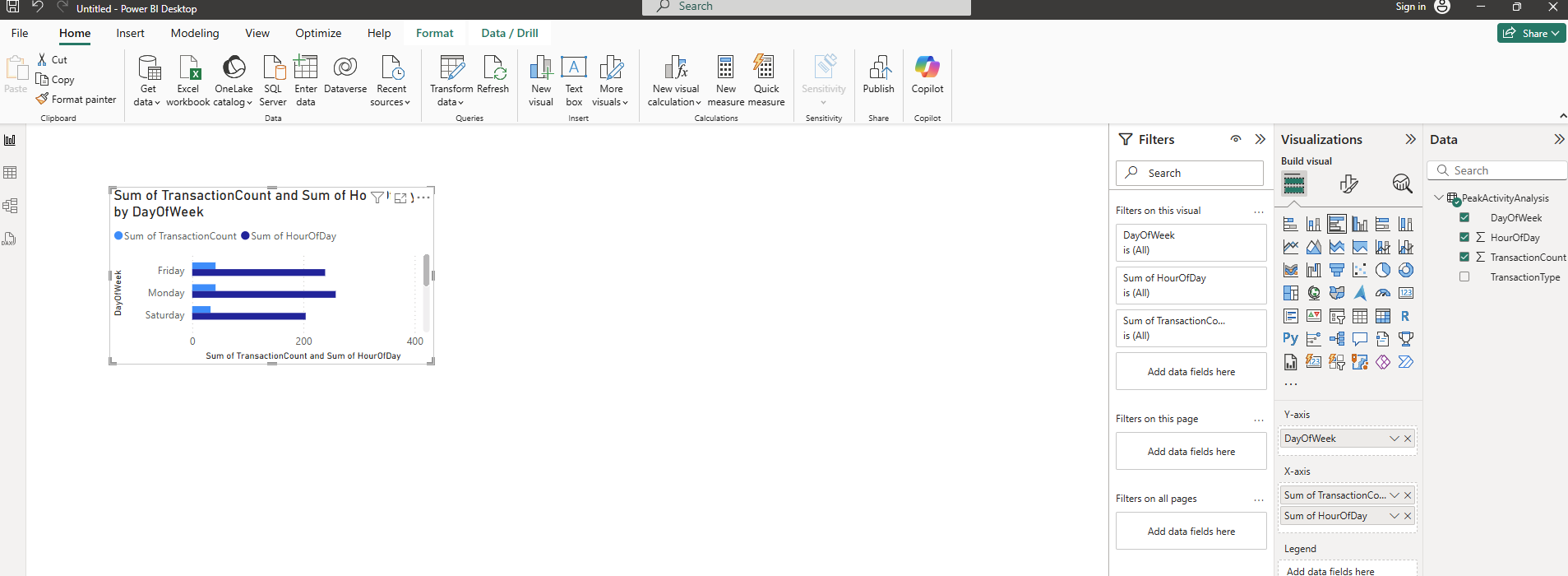
A screenshot of a computer

AI-generated content may be incorrect.

**SQL DB to Power Bi:**

I have connected my Power Bi to SQL DB, followed by loading the data of my table Peak Activity Analysis.

* + 1. **Line chart**



* + 1. **Matrix view**

A screenshot of a computer

AI-generated content may be incorrect.