# Bootcamp Project 3
# Customer 360 Data Integration

### Overview Of the Project

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.

### Flow of Project :

- We have given data of project in raw csv files.
- We are going to follow Medalen Architecture: storing data into 3 different containers: Bronze and Silver and Gold
- Raw CSV files are stored in Raw container.
- We have given 10 SQL table structures, which have relationships among them.
- We have to ingest the data from csv files into tables that will be created in Azure MS SQL Database. This ends up the raw layer.
- In Next phase, we are going to clean the data and transform into desired tabular structures.
- After finishing this, in next phase, we have given some insights or KPIs and we are going to create views on top of tables based on KPIs and then at last we are going to populate the processed data into Power BI dashboards.

## PHASE –1 (DATA INGESTION)

- In this phase, we are going to store raw csv files into ADLS G2 container.

**Location:** raw / project-3 / raw files

Search blobs by prefix (case-sensitive) | Show deleted objects

| Name | Modified | Access tier | Archive status | Blob type | Size | Lease state |
|---|---|---|---|---|---|---|
| [..] | | | | | | |
| Agents.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 3.82 KiB | Available |
| Customers.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 8.72 KiB | Available |
| CustomerServiceInterac... | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 5 KiB | Available |
| InStoreTransactions.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 4.62 KiB | Available |
| LoyaltyAccounts.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 2.93 KiB | Available |
| LoyaltyTransactions.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 3.67 KiB | Available |
| OnlineTransactions.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 5.43 KiB | Available |
| Products.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 2.57 KiB | Available |
| Stores.csv | 4/27/2025, 9:57:20 PM | Hot (Inferred) | | Block blob | 4.84 KiB | Available |

- Now, we are creating all 10 tables associated with files.
- Please note that tables have relations among them, we are going to create parent tables first, after that child tables.
- Tables' Name:
  - Customers
  - Products
  - Stores
  - Agents
  - OnlineTransactions
  - InStoreTransactions
  - CustomerServiceInteractions
  - LoyaltyAccounts
  - LoyaltyTransactions

## Creation of Tables queries:

```
-- Products SP
CREATE PROCEDURE project3.Cleanproductstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY productid
```

```sql
                        ORDER BY productid) AS RowNum
                FROM   project3.products)
        DELETE FROM cte
        WHERE  rownum > 1;

        --trim the records
        UPDATE project3.products
        SET    NAME = Trim(NAME);

        UPDATE project3.products
        SET    category = Trim(category);

        --default value
        UPDATE project3.products
        SET    price = COALESCE(price, 0.00)
        WHERE  price IS NULL;
    END;

-- Stores SP
CREATE PROCEDURE project3.Cleanstorestable
AS
    BEGIN
        --removing duplicate records
        WITH cte
             AS (SELECT *,
                        Row_number()
                          OVER (
                            partition BY storeid, manager
                            ORDER BY storeid) AS RowNum
                FROM   project3.stores)
        DELETE FROM cte
        WHERE  rownum > 1;

        --trim the records
        UPDATE project3.stores
        SET    manager = Trim(manager);

        UPDATE project3.stores
        SET    location = Trim(location);
    END;
```

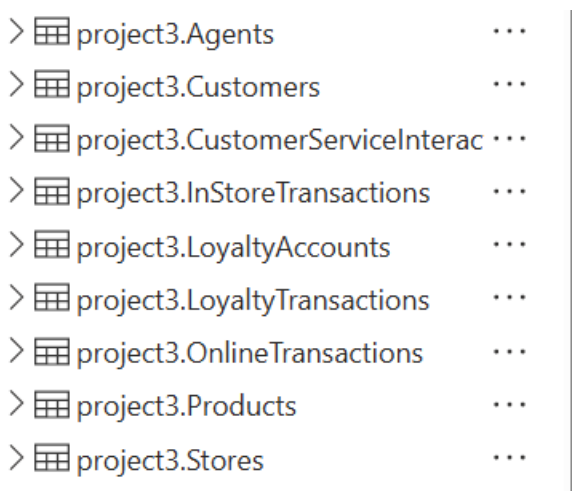
```sql
-- Agents SP
CREATE PROCEDURE project3.Cleanagentstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY agentid, NAME, department
                          ORDER BY agentid) AS RowNum
               FROM   project3.agents)
      DELETE FROM cte
      WHERE  rownum > 1;

      --trim the records
      UPDATE project3.agents
      SET    NAME = Trim(NAME);
  END;

-- Online Transactions SP
CREATE PROCEDURE project3.Cleanonlinetransactionstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY orderid
                          ORDER BY orderid) AS RowNum
               FROM   project3.onlinetransactions)
      DELETE FROM cte
      WHERE  rownum > 1;

      --trim the records
      UPDATE project3.onlinetransactions
      SET    paymentmethod = Trim(paymentmethod);
```

```sql
    --default value
    UPDATE project3.onlinetransactions
    SET    amount = COALESCE(amount, 0.00)
    WHERE  amount IS NULL;
  END;

-- InStoreTransactions SP
CREATE PROCEDURE project3.Cleaninstoretransactionstable
AS
  BEGIN
    --removing duplicate records
    WITH cte
         AS (SELECT *,
                    Row_number()
                      OVER (
                        partition BY transactionid
                        ORDER BY transactionid) AS RowNum
             FROM   project3.instoretransactions)
    DELETE FROM cte
    WHERE  rownum > 1;

    --trim the records
    UPDATE project3.instoretransactions
    SET    paymentmethod = Trim(paymentmethod);

    --default value
    UPDATE project3.instoretransactions
    SET    amount = COALESCE(amount, 0.00)
    WHERE  amount IS NULL;
  END;

-- CustomerServiceInteractions SP
CREATE PROCEDURE project3.Cleancustomerserviceinteractionstable
AS
  BEGIN
    --removing duplicate records
    WITH cte
         AS (SELECT *,
                    Row_number()
                      OVER (
```

```sql
                        partition BY interactionid
                        ORDER BY interactionid) AS RowNum
                FROM   project3.customerserviceinteractions)
        DELETE FROM cte
        WHERE  rownum > 1;

        --trim the records
        UPDATE project3.customerserviceinteractions
        SET    issuetype = Trim(issuetype);

        UPDATE project3.customerserviceinteractions
        SET    resolutionstatus = Trim(resolutionstatus);
    END;

-- LoyaltyAccounts SP
CREATE PROCEDURE project3.Cleanloyaltyaccountstable
AS
    BEGIN
        --removing duplicate records
        WITH cte
             AS (SELECT *,
                        Row_number()
                          OVER (
                            partition BY loyaltyid
                            ORDER BY loyaltyid) AS RowNum
                FROM   project3.loyaltyaccounts)
        DELETE FROM cte
        WHERE  rownum > 1;

        --trim the records
        UPDATE project3.loyaltyaccounts
        SET    tierlevel = Trim(tierlevel);

        --default value
        UPDATE project3.loyaltyaccounts
        SET    pointsearned = COALESCE(pointsearned, 0)
        WHERE  pointsearned IS NULL;
    END;

-- LoyaltyTransactions SP
```

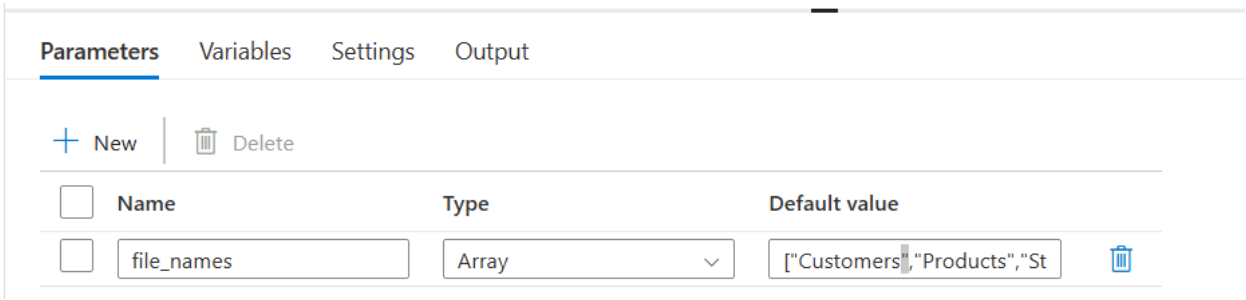
```sql
CREATE PROCEDURE project3.Cleanloyaltytransactionstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY loyaltyid
                          ORDER BY loyaltyid) AS RowNum
               FROM   project3.loyaltytransactions)
      DELETE FROM cte
      WHERE  rownum > 1;

      --trim the records
      UPDATE project3.loyaltytransactions
      SET    reason = Trim(reason);
  END;
```

- Here are the tables created in Azure MS SQL database:



# Next Phase: Ingesting Data from ADLS G2 files into Azure SQL Tables

- To ingest the data from CSV files to Azure SQL tables, we are using Synapse pipeline.
- For that, I have created pipeline in Synapse named Project-3.
- In that, I'm using for-each activity to use parameterized values for copy data of each file.
- I've created pipeline parameter as string array that will be passed to for each activity and it is being run for each file sequentially.




Pipeline parameter

- Here is the array of string(please note that order of each file) ->

  ["Customers","Products","Stores","Agents","OnlineTransactions","InStoreTransactions","CustomerServiceInteractions","LoyaltyAccounts","LoyaltyTransactions"]

- Inside the for-each activity, I'm invoking copy activity which uses parameterized values to copy data from ADLS G2 csv files to Azure SQL table data.
- Source is : delimited file
- Sink : SQL Tables.

***<u>Source :</u>***

General | Source | Sink | Mapping | Settings | User properties

Source dataset * ds_project3_csv Open New Preview data Learn more

Dataset properties

| Name | Value | Type |
|---|---|---|
| file_name | @concat(item(),'.csv') | string |

File path type: File path in dataset / Wildcard file path / List of files

DelimitedText
**ds_project3_csv**

Connection | Schema | Parameters

Linked service * ls_project3 Test connection Edit New Learn more

Integration runtime * AutoResolveIntegrationRuntime Edit

File path raw / project-3/raw files / @dataset().file_name Browse

## ***Sink :***

General | Source | Sink | Mapping | Settings | User properties

Sink dataset * ds_project3_sql Open New Learn more

Dataset properties

| Name | Value | Type |
|---|---|---|
| table_name | @item() | string |

Write behavior: Insert / Upsert / Stored procedure

Azure SQL Database
**ds_project3_sql**

Connection | Schema | Parameters

Linked service * ls_project3_sql Test connection Edit New Learn more

Integration runtime * AutoResolveIntegrationRuntime Edit

Table project3 . @dataset().table_name Preview data

Enter manually

# Next Phase: Cleaning data of SQL Tables using Stored Procedures:

- Now it's time to clean the data of these tables. I'm using different approach.
- Instead of using azure computational services, I'm using capabilities of Database only to clean the data.
- I'm creating stored procedures and cleaned the data by updating the tables.
- So I have created stored procedure in SQL database for each table and invoke each stored procedure sequentially using Pipeline created in Synapse.


Stored Procedures created in SQL dB to clean data for each table

***What activities will be performed*** :

Duplicate records

Missing (NULL) values

Inconsistent formats (e.g., dates, strings)

Invalid or outlier values

Extra spaces or special characters

- Here are the queries of each stored Procedure :

```sql
-- Products SP
CREATE PROCEDURE project3.Cleanproductstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY productid
                          ORDER BY productid) AS RowNum
               FROM   project3.products)
      DELETE FROM cte
      WHERE  rownum > 1;

      --trim the records
      UPDATE project3.products
      SET    NAME = Trim(NAME);

      UPDATE project3.products
      SET    category = Trim(category);

      --default value
      UPDATE project3.products
      SET    price = COALESCE(price, 0.00)
      WHERE  price IS NULL;
  END;

-- Stores SP
CREATE PROCEDURE project3.Cleanstorestable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY storeid, manager
                          ORDER BY storeid) AS RowNum
```

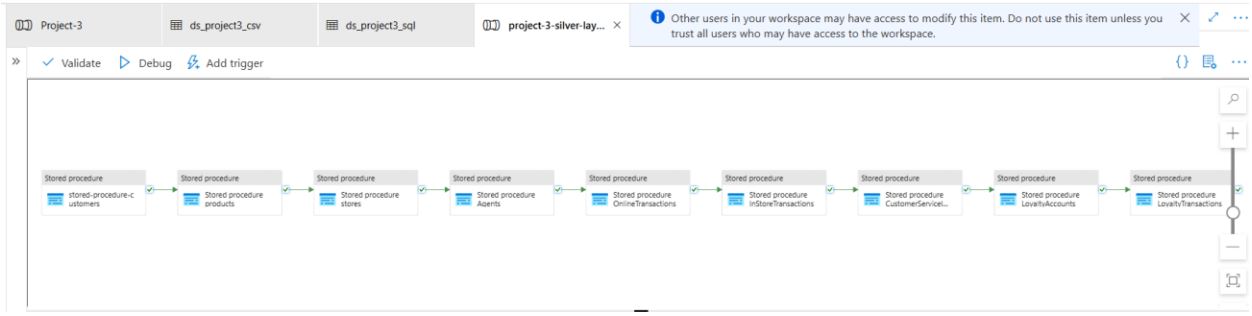
```
                FROM   project3.stores)
        DELETE FROM cte
        WHERE  rownum > 1;

        --trim the records
        UPDATE project3.stores
        SET    manager = Trim(manager);

        UPDATE project3.stores
        SET    location = Trim(location);
    END;

-- Agents SP
CREATE PROCEDURE project3.Cleanagentstable
AS
    BEGIN
        --removing duplicate records
        WITH cte
             AS (SELECT *,
                        Row_number()
                          OVER (
                            partition BY agentid, NAME, department
                            ORDER BY agentid) AS RowNum
                FROM   project3.agents)
        DELETE FROM cte
        WHERE  rownum > 1;

        --trim the records
        UPDATE project3.agents
        SET    NAME = Trim(NAME);
    END;

-- Online Transactions SP
CREATE PROCEDURE project3.Cleanonlinetransactionstable
AS
    BEGIN
        --removing duplicate records
        WITH cte
             AS (SELECT *,
                        Row_number()
```

```
                    OVER (
                      partition BY orderid
                      ORDER BY orderid) AS RowNum
            FROM   project3.onlinetransactions)
      DELETE FROM cte
      WHERE  rownum > 1;

      --trim the records
      UPDATE project3.onlinetransactions
      SET    paymentmethod = Trim(paymentmethod);

      --default value
      UPDATE project3.onlinetransactions
      SET    amount = COALESCE(amount, 0.00)
      WHERE  amount IS NULL;
  END;

-- InStoreTransactions SP
CREATE PROCEDURE project3.Cleaninstoretransactionstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                    OVER (
                      partition BY transactionid
                      ORDER BY transactionid) AS RowNum
            FROM   project3.instoretransactions)
      DELETE FROM cte
      WHERE  rownum > 1;

      --trim the records
      UPDATE project3.instoretransactions
      SET    paymentmethod = Trim(paymentmethod);

      --default value
      UPDATE project3.instoretransactions
      SET    amount = COALESCE(amount, 0.00)
      WHERE  amount IS NULL;
```

```
  END;

-- CustomerServiceInteractions SP
CREATE PROCEDURE project3.Cleancustomerserviceinteractionstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY interactionid
                          ORDER BY interactionid) AS RowNum
               FROM   project3.customerserviceinteractions)
      DELETE FROM cte
      WHERE  rownum > 1;

      --trim the records
      UPDATE project3.customerserviceinteractions
      SET    issuetype = Trim(issuetype);

      UPDATE project3.customerserviceinteractions
      SET    resolutionstatus = Trim(resolutionstatus);
  END;

-- LoyaltyAccounts SP
CREATE PROCEDURE project3.Cleanloyaltyaccountstable
AS
  BEGIN
      --removing duplicate records
      WITH cte
           AS (SELECT *,
                      Row_number()
                        OVER (
                          partition BY loyaltyid
                          ORDER BY loyaltyid) AS RowNum
               FROM   project3.loyaltyaccounts)
      DELETE FROM cte
      WHERE  rownum > 1;
```

```sql
    --trim the records
    UPDATE project3.loyaltyaccounts
    SET    tierlevel = Trim(tierlevel);

    --default value
    UPDATE project3.loyaltyaccounts
    SET    pointsearned = COALESCE(pointsearned, 0)
    WHERE  pointsearned IS NULL;
  END;

-- LoyaltyTransactions SP
CREATE PROCEDURE project3.Cleanloyaltytransactionstable
AS
  BEGIN
    --removing duplicate records
    WITH cte
         AS (SELECT *,
                    Row_number()
                      OVER (
                        partition BY loyaltyid
                        ORDER BY loyaltyid) AS RowNum
             FROM   project3.loyaltytransactions)
    DELETE FROM cte
    WHERE  rownum > 1;

    --trim the records
    UPDATE project3.loyaltytransactions
    SET    reason = Trim(reason);
  END;
```

- Then Through Synapse Pipeline, I'm invoking each stored procedure using Stored Procedure activity available in Synapse Pipeline:




# Next Phase: Creating Views on top of tables for Insights:

- Few KPIs or Insights are :
  - **Average Order Value**
  - Segment customers based on total spend, purchase frequency, and loyalty tier
  - Analyze **`DateTime`** to find peak days and times in-store vs. Online
  - Number of interactions and resolution success rates per agent.

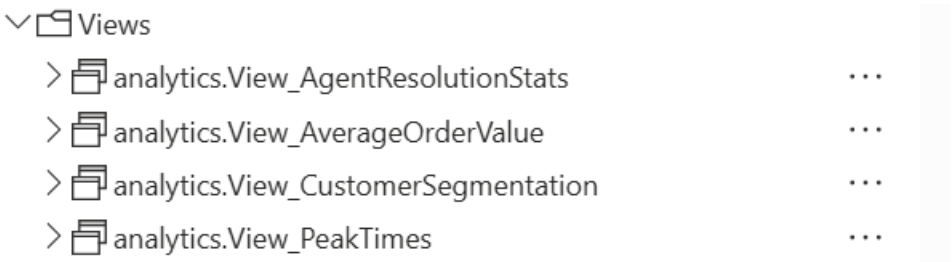
## ***Views created for each KPI:***

```
CREATE VIEW analytics.view_averageordervalue AS

SELECT    p.productid,

          p.NAME AS ProductName,

          p.category,

          s.storeid,

          s.location,

          Sum(t.amount) / Count(t.orderid) AS AverageOrderValue

FROM      project3.onlinetransactions t

INNER JOIN project3.products p

ON        t.productid = p.productid

LEFT JOIN  project3.stores s
```

```
ON      s.storeid IS NOT NULL -- since online might not have a store directly, kept flexible
GROUP BY  p.productid,
        p.NAME,
        p.category,
        s.storeid,
        s.location;

----------------------------------CREATE VIEW analytics.view_customersegmentation AS
        WITH customerspending          AS
        (
            SELECT   c.customerid,
                c.NAME,
                Sum(t.amount)   AS totalspend,
                Count(t.orderid) AS purchasefrequency,
                l.tierlevel
            FROM     project3.customers c
            LEFT JOIN project3.onlinetransactions t
            ON      c.customerid = t.customerid
            LEFT JOIN project3.loyaltyaccounts l
            ON      c.customerid = l.customerid
            GROUP BY  c.customerid,
                c.NAME,
                l.tierlevel
        )
 SELECT customerid,
      NAME,
```

```
    totalspend,
    purchasefrequency,
    tierlevel,
    CASE
        WHEN totalspend >=
            (
                SELECT   Percentile_cont(0.9) within GROUP (ORDER BY totalspend) OVER
()) THEN 'High-Value Customer'
        WHEN purchasefrequency = 1 THEN 'One-Time Buyer'
        WHEN tierlevel = 'Gold'
        OR    tierlevel = 'Platinum' THEN 'Loyalty Champion'
        ELSE 'Regular Customer'
    END AS customersegment
FROM   customerspending;
```

- ***Views created for each KPI:***




## Next Phase: Populating Dashboards on the data of Views:

TOTAL NUMBER OF AGENTS

100

TOTAL NO. OF CUSTOMERS

123

TOTAL NUMBER OF STORES

100