

## **Data Engineering Project 3:**

### **Customer 360 Data Integration**

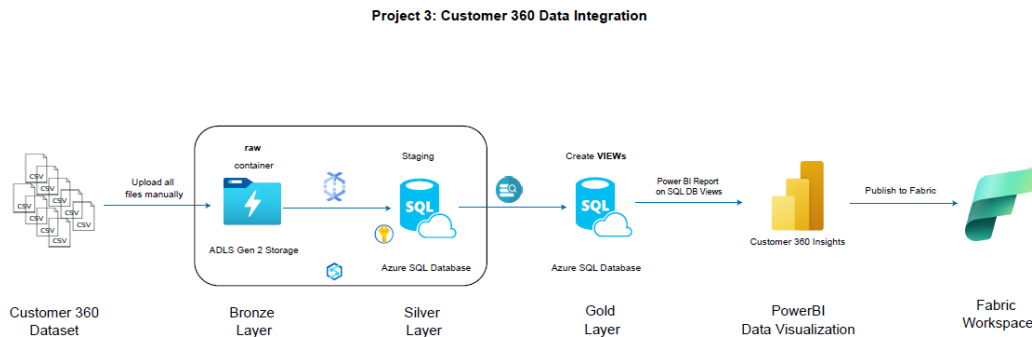
## **Table of Contents**

Objective: .....	2
Architecture Diagram: .....	2
Step 1: Data Ingestion .....	2
Step 2: Check the data into SSMS .....	13
Step 3: Create Views for Gold Layer .....	13
Step 4: Power BI Dashboard/Report .....	19
Data Connectivity Mode .....	20
Step 5: Publish to Fabric Workspace .....	22
Points to Remember .....	23

## Objective:

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.

## Architecture Diagram:



Dataset used: <https://www.kaggle.com/datasets/varunkumari/customer-360-data>

## Step 1: Data Ingestion

Use Azure Synapse Analytics pipelines to ingest data from multiple sources (online, in-store, customer service, loyalty programs) into the raw container in ADLS.

**Step 1.1:** Create a folder called **bronze** in ADLS Gen 2 storage account and upload all the csv files which you have downloaded from source dataset URL.

Authentication method: Access key (Switch to Microsoft Entra user account)  
Location: project3 / bronze

Search blobs by prefix (case-sensitive)

☐ Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size
<input type="checkbox"/> Agents.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	3.82 KiB
<input type="checkbox"/> Customers.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	8.72 KiB
<input type="checkbox"/> CustomerServiceInteractions.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	5 KiB
<input type="checkbox"/> InStoreTransactions.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	4.62 KiB
<input type="checkbox"/> LoyaltyAccounts.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	2.93 KiB
<input type="checkbox"/> LoyaltyTransactions.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	3.67 KiB
<input type="checkbox"/> OnlineTransactions.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	5.43 KiB
<input type="checkbox"/> Products.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	2.57 KiB
<input type="checkbox"/> Stores.csv	4/28/2025, 12:15:27 PM	Hot (Inferred)		Block blob	4.84 KiB

**Step 1.2** Open SSMS and Connect the Azure SQL Database

Create these table in this sequence (as they have constraints)

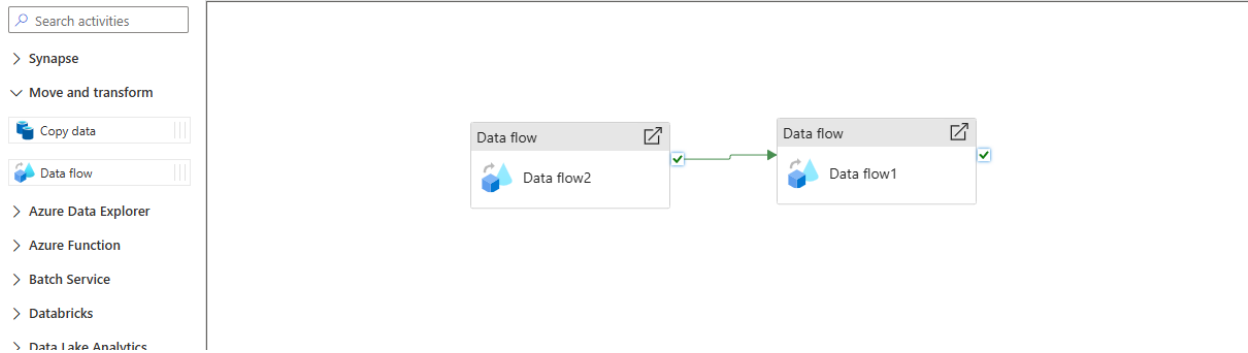
```

1  -----Parent Table-----
2  CREATE TABLE Customers (
3      CustomerID INT PRIMARY KEY,
4      Name VARCHAR(100),
5      Email VARCHAR(100),
6      Address VARCHAR(255)
7  );
8
9  -----Parent Table-----
10 CREATE TABLE Products (
11     ProductID INT PRIMARY KEY,
12     Name VARCHAR(100),
13     Category VARCHAR(50),
14     Price DECIMAL(10, 2)
15 );
16
17 -----Dependent Table-----
18 CREATE TABLE OnlineTransactions (
19     OrderID INT PRIMARY KEY,
20     CustomerID INT,
21     ProductID INT,
22     DateTime DATETIME,
23     PaymentMethod VARCHAR(50),
24     Amount DECIMAL(10, 2),
25     Status VARCHAR(20),
26     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
27     FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
28 );
29
30 -----Parent Table-----
31 CREATE TABLE Stores (
32     StoreID INT PRIMARY KEY,
33     Location VARCHAR(100),
34     Manager VARCHAR(100),
35     OpenHours VARCHAR(50)
36 );
37
38 -----Dependent Table-----
39 CREATE TABLE InStoreTransactions (
40     TransactionID INT PRIMARY KEY,
41     CustomerID INT,
42     StoreID INT,
43     DateTime DATETIME,
44     Amount DECIMAL(10, 2),
45     PaymentMethod VARCHAR(50),
46     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
47     FOREIGN KEY (StoreID) REFERENCES Stores(StoreID)
48 );
49
50 -----Parent Table-----
51 CREATE TABLE Agents (
52     AgentID INT PRIMARY KEY,
53     Name VARCHAR(100),
54     Department VARCHAR(50),
55     Shift VARCHAR(50)
56 );
57
58 -----Dependent Table-----
59 CREATE TABLE CustomerServiceInteractions (
60     InteractionID INT PRIMARY KEY,
61     CustomerID INT,
62     DateTime DATETIME,
63     AgentID INT,
64     IssueType VARCHAR(50),
65     ResolutionStatus VARCHAR(50),
66     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
67     FOREIGN KEY (AgentID) REFERENCES Agents(AgentID)
68 );
69
70 -----Dependent Table-----
71 CREATE TABLE LoyaltyAccounts (
72     LoyaltyID INT PRIMARY KEY,
73     CustomerID INT,
74     PointsEarned INT,
75     TierLevel VARCHAR(20),
76     JoinDate DATE,
77     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
78 );
79
80 -----Dependent Table-----
81 CREATE TABLE LoyaltyTransactions (
82     LoyaltyID INT,
83     DateTime DATETIME,
84     PointsChange INT,
85     Reason VARCHAR(100),
86     PRIMARY KEY (LoyaltyID, DateTime),
87     FOREIGN KEY (LoyaltyID) REFERENCES LoyaltyAccounts(LoyaltyID)
88 );

```

**Step 1.3:** Open Synapse Workspace, Create a new pipeline

Drag and drop two dataflow activities in canvas area.

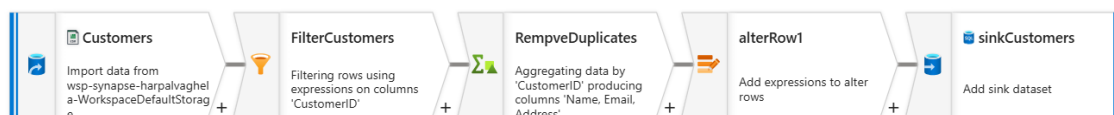


Click on the first one, and Settings -> Dataflow -> New

Blank dataflow design canvas will be opened

Here, we will design transformation step from bronze to silver (staging) layer

**Step 1.4** Overview of the customer records transformation steps:



Let's do step by step.

Select the source

**Source Settings Tab:**

Write output stream: Customers

Source type: Inline

Inline dataset type: Delimited Text

Linked Service: ADLS Gen 2

Source settings   Source options   Projection   Optimize   Inspect   Data preview

Output stream name \*  [Learn more](#)

Description  [Reset](#)

Source type \* Integration dataset **Inline** Workspace DB

Inline dataset type \*

Linked service \*  [Test connection](#) [Edit](#) [New](#)

Skip line count

Sampling \* ☐ Enable ☒ Disable

### Source Options Tab:

File path: browse and select the file from storage account as shown below

Select first row as header

Source settings   **Source options**   Projection   Optimize   Inspect   Data preview

File settings

File mode ☒ File ☐ Wildcard

File path \*  /  /  [Browse](#)

Allow no files found ☐

Change data capture ☐

Compression type

Encoding

Column delimiter

Row delimiter

Quote character

Escape character

First row as header ☒

### Projection Tab:

Click on Import Schema

### Filter Transformation

Click on plus icon on bottom of source transformation and add filter transformation

## Filter Settings Tab:

**Filter settings** | Optimize | Inspect | Data preview

Output stream name \*

FilterCustomers

[Learn more](#)

Description

Filtering rows using expressions on columns 'CustomerID'

Reset

Incoming stream \*

Customers

Filter on \*

!isNull(CustomerID)

Expression used in filter on field is: !isNull(CustomerID)

**Aggregate Transformation (To Remove Duplicates)**

Click on plus icon on bottom of filter transformation and add aggregate transformation

**Aggregate settings tab**

Select ID column (here, CustomerID) in Group by tab as shown below:

**Aggregate settings** | Optimize | Inspect | Data preview

Output stream name \*

RempveDuplicates

[Learn more](#)

Description

Aggregating data by 'CustomerID' producing columns 'Name, Email, Address'

Reset

Incoming stream \*

FilterCustomers

Group by

Aggregates

Columns

12s CustomerID

Name as

CustomerID

Click on Aggregates tab now and click on plus icon and take column pattern and delete the default field.

Group by Aggregates

Grouped by: CustomerID

+ Add Clone Delete Open expression builder

Column	Expression
<input type="checkbox"/>	Each column that matches <input type="text" value="name != 'CustomerID'"/> creates 1 column(s)
<input type="text" value="\$"/>	<input type="text" value="first(\$)"/>

Write this expression in

Each column that matches: name != 'CustomerID'

Check the data in Inspect -> Input

Aggregate settings Optimize **Inspect** Data preview

Schema **Input** Output

Number of columns **Total** 4

Order	Column	Type	Used by
1	CustomerID	128 short	CustomerID
2	Name	abc string	Name
3	Email	abc string	Email
4	Address	abc string	Address

Inspect -> Output

Aggregate settings Optimize **Inspect** Data preview

Schema **Input** Output

Number of columns **Total** 4

Order	Column	Type	Used by
1	CustomerID	128 short	CustomerID
2	Name	abc string	Name
3	Email	abc string	Email
4	Address	abc string	Address

### Alter Row Transformation

Click on plus icon on bottom of aggregate transformation and alter row sink transformation

Alter row conditions: Upsert if

Expression: 1==1

Alter row settings   Optimize   Inspect   Data preview

Output stream name \*  [Learn more](#)

Description  [Reset](#)

Incoming stream \*

Alter row conditions \* ⓘ  
  [+](#) [-](#)

**Sink Transformation:**

Click on plus icon on bottom of aggregate transformation and add sink transformation

**Sink tab:**

Output stream name: sinkCustomers

Sink type: Inline

Inline dataset type: Azure SQL Database

Linked service: Azure SQL Database

Sink   Settings   Errors   Mapping   Optimize   Inspect   Data preview

Output stream name \*  [Learn more](#)

Description  [Reset](#)

Incoming stream \*

Sink type \*  
☐ Integration dataset ☒ Inline ☐ Workspace DB ☐ Cache

Inline dataset type \*

Linked service \*  [Test connection](#) [Edit](#) [New](#)

Options  
☒ Allow schema drift ⓘ  
☐ Validate schema ⓘ

While Creating Linked Service, use Azure Key vault option to store Azure SQL server password in to key vault.



Password

Azure Key Vault

AKV linked service \* ⓘ

ls\_key\_vault

Secret name \*

azuresqldb-password

☒ Edit

Secret version ⓘ

Latest version

☐ Edit

Always encrypted ⓘ

☐

Encrypt ⓘ

Mandatory


Trust server certificate ⓘ

☐

Step to create new azure sql password into key vault:

Open **Key Vault** and click on **Generate/Import**

Dashboard > keyvault-harpal-vaghela

 **keyvault-harpal-vaghela | Secrets** ☆ ...

Key vault

Search

⌵

⏪

+ Generate/Import

🔄 Refresh

⬆️ Restore Backup

🔑 Overview

Write Secret name and Secret value (password of Azure SQL Server) and click on Create.

[Dashboard](#) > [keyvault-harpal-vaghela](#)

## keyvault-harpal-vaghela | Secrets

 [+ Generate/Import](#) [Refresh](#) [Restore Backup](#) [Manage deleted secrets](#) [View sample code](#)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Access policies
- Resource visualizer
- Events

Name	Type
appid	
appsecret	
azuresqldb-password	
onprem-outlookuser-password	
sas-token-adls	

Also, in Key Vault Access Policies give permission (Get, List for Synapse Workspace to access key vault secrets)

[Dashboard](#) > [keyvault-harpal-vaghela](#)

## keyvault-harpal-vaghela | Access policies

 [+ Create](#) [Refresh](#) [Delete](#) [Edit](#)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Access policies
- Resource visualizer
- Events
- Objects
  - Keys
  - Secrets
  - Certificates

Access policies enable you to have fine grained control over access to vault items. [Learn more](#)

 [Permissions : All](#) [Type : All](#)

Showing 1 to 4 of 4 records.

<input type="checkbox"/> Name ↑↓	Email ↑↓	Key Permissions
APPLICATION		
<input type="checkbox"/> adf-harpalvaghela		
<input type="checkbox"/> AzureDatabricks		
<input type="checkbox"/> wsp-synapse-harpalvaghela		
USER		
<input type="checkbox"/> harpal	harpal@kanaka526vishaloutlook.onmicrosoft.com	Get, List, Update, Create, Import, Delete, Recov

Settings tab:

Schema Name : dbo (select from a dropdown list)

Table Name: Customers (select from a dropdown list)

Update Method: Allow Upsert

Key Columns: CustomerID

Sink **Settings** Errors Mapping Optimize Inspect Data preview

Schema name \*  Refresh

Table name \*

Table action ☒ None ☐ Recreate table ☐ Truncate table

Update method ⓘ ☐ Allow insert ☐ Allow delete ☒ Allow upsert ☐ Allow update

Skip writing key columns ⓘ ☐

Key columns \* ⓘ ☒ List of columns ☐ Custom expression ⓘ

+

### Mapping tab:

Click on Import Schema and map the columns as shown below:

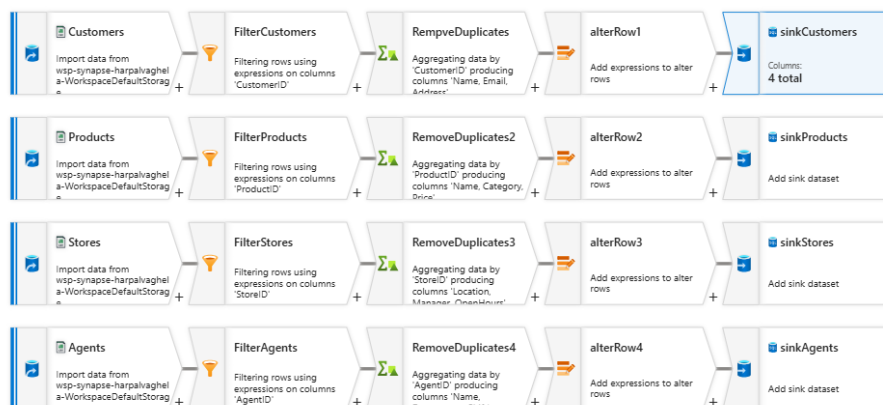
Options ☒ Skip duplicate input columns ⓘ ☒ Skip duplicate output columns ⓘ

☐ Auto mapping ⓘ + Add mapping Delete Reset Import schema View schema 4 mappings: All outputs ma

Input columns		Output columns	
<input type="checkbox"/> CustomerID	→	<input type="text" value="123 CustomerID"/>	+
<input type="checkbox"/> Name	→	<input type="text" value="abc Name"/>	+
<input type="checkbox"/> Email	→	<input type="text" value="abc Email"/>	+
<input type="checkbox"/> Address	→	<input type="text" value="abc Address"/>	+

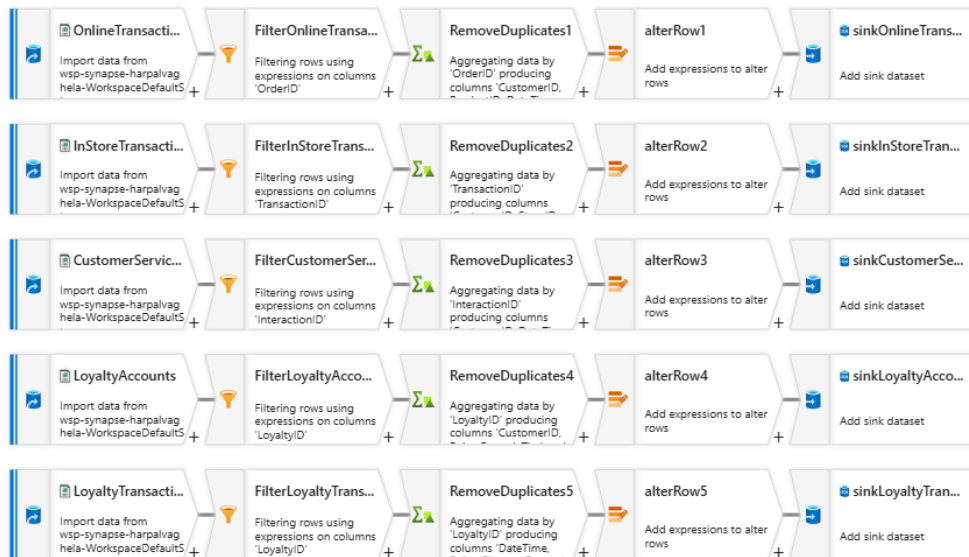
Similarly, we will do transformations steps or Products, Stores, Agents data files.

Here is an final design in dataflow1:



**Step 1.5 Design of Dataflow2:**

Similarly, we have to design for OnlineTransaction, InStoreTransactions, CustomerService, LoyaltyAccounts, LoyaltyTransactions

**Step 1.6 Run the Pipeline**

The screenshot shows the Synapse IDE interface with the pipeline 'pl\_master' executed successfully. The 'Output' tab displays the following data:

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	User prop...	Activity run ID
DF_Dependent_Tables	Succeeded	Data flow	4/28/2025, 3:59:33 PM	25s	AutoResolveIntegrationRuntime (Canada Central)		3556e3cb-621b-43e1-ae3b-b8dc976
DF_Parents_Tables	Succeeded	Data flow	4/28/2025, 3:53:36 PM	5m 36s	AutoResolveIntegrationRuntime (Canada Central)		95f9f6e-64c6-4b87-bb64-4f5aab3a

## Step 2: Check the data into SSMS

The screenshot shows the SQL Server Enterprise Manager interface with a query result displayed. The query is named 'Project\_3\_SQL\_Quer...' and it returns data from the 'project3db' database. The results are organized into several tables:

CustomerID	Name	Email	Address
1	Mrs. Crystal Carroll	karverjacob@example.org	7566 Kelly Shoals Apt. 207, Port Joanne, FM 25401
2	Debra Newman DVM	brownkerry@example.net	72713 Helmsch Lodge Suite 206, Loxford, NJ 49406
3	Thomas Mason	frankmark@example.com	3024 Riley Ferry Suite 573, Chedberg, MT 40822
4	Karla Hill	jnelson@example.org	9899 Hubbard Station, Lake Vanessa, CT 97851

ProductID	Name	Category	Price
1	Consider	Electronics	183.51
2	At	Toys	403.28
3	Risk	Clothing	186.53
4	Serious	Clothing	275.39

OrderID	CustomerID	ProductID	DateTime	PaymentMethod	Amount	Status
1	34	91	2025-01-20 22:34:31.000	Gift Card	189.90	Completed
2	17	59	2025-02-10 20:42:13.000	Gift Card	193.25	Pending
3	13	57	2025-02-16 18:26:04.000	Debit Card	183.82	Pending

StoreID	Location	Manager	OpenHours
1	Anglemouth	Michelle Robinson	8:00 AM - 8:00 PM
2	Tannyclester	Jonathan Frey	8:00 AM - 8:00 PM
3	Vanessache	Barbara Lynch	8:00 AM - 7:00 PM

TransactionID	CustomerID	StoreID	DateTime	Amount	PaymentMethod
1	88	100	2025-03-12 06:49:37.000	135.74	Credit Card
2	60	12	2025-01-21 23:09:14.000	34.14	Credit Card
3	54	51	2025-01-29 22:12:51.000	111.42	Debit Card
4	32	94	2025-02-08 23:51:36.000	179.45	Debit Card

AgentID	Name	Department	Shift
1	Jonathan Williams	General Inquiry	Morning
2	Terry Edwards	Billing	Evening
3	Garrett Knapp	Sales	Morning
4	Daryl Benjamin	Sales	Evening
5	Matthew Long	Sales	Morning

InteractionID	CustomerID	DateTime	AgentID	IssueType	ResolutionStatus
1	55	2025-01-20 08:17:26.000	33	Technical Issue	Resolved
2	83	2025-01-16 18:52:51.000	89	Technical Issue	Escalated
3	53	2025-03-16 13:13:05.000	96	Complaint	Resolved
4	30	2025-02-27 13:45:03.000	20	Other	Pending
5	49	2025-02-16 07:56:45.000	48	Technical Issue	Pending
6	42	2025-02-13 21:09:41.000	98	Product Inquiry	Resolved
7	73	2025-03-01 13:19:50.000	38	Other	Escalated
8	50	2025-01-15 18:44:36.000	86	Other	Escalated

We have successfully loaded the cleaned data into azure SQL database.

## Step 3: Create Views for Gold Layer

### View 1 - Average Order Value (AOV) (AOV = Total Sales Amount / Total Number of Orders)

SUM(Amount) / COUNT(OrderID) per product, category, and location.

- We listed each product along with its category and the store location.
- We calculated the average amount customers spend each time they order a product.
- We connected each online sale to its product to know what was sold.
- We also tried to link customers to store locations, even for online sales.
- Finally, we grouped the results by product, category, and store to show clear average spending for each.

```

1  -- In-store Transactions
2  CREATE OR ALTER VIEW View_1_InStoreAverageOrderValue AS
3  SELECT
4      S.StoreID,
5      S.Location AS StoreLocation,
6      CAST(SUM(IST.Amount) * 1.0 / COUNT(IST.TransactionID) AS DECIMAL(10,2)) AS AverageOrderValue
7  FROM
8      InStoreTransactions IST
9  JOIN
10     Stores S ON IST.StoreID = S.StoreID
11 GROUP BY
12     S.StoreID, S.Location;
13
14
15 select * from View_1_InStoreAverageOrderValue;

```

**Output :** select \* from View\_1\_InStoreAverageOrderValue;

	StoreID	StoreLocation	AverageOrderValue
1	3	Vanessachester	175.69
2	4	Maytown	67.71
3	5	Kathleenfort	152.15
4	7	Rebeccastad	158.58
5	8	North Mariaburgh	189.17
6	10	Thompsonshire	135.21
7	11	East Justin	162.33
8	12	Jeffreyport	57.15
9	14	Josephfort	188.61
10	15	New Kimberly	101.80
11	16	Lake Phillip	123.26
12	17	East Laura	197.34
13	18	Rodriguezstad	81.02
14	19	Bakerberg	107.82
15	20	Ashleyfort	98.91
16	22	Simpsonview	121.97
17	23	South Amyfurt	41.82
18	24	Bishopview	27.89
19	25	Suttonbury	137.98

### ➤ Online Transactions

```

1  -----OnlineTransactions only - AOV for Online Only
2
3  CREATE OR ALTER VIEW View_1_OTAverageOrderValue AS
4  SELECT
5      P.ProductID,
6      P.Name AS ProductName,
7      P.Category,
8      CAST(SUM(OT.Amount) * 1.0 / COUNT(OT.OrderID) AS DECIMAL(10,2)) AS AverageOrderValue
9  FROM
10     OnlineTransactions OT
11 JOIN
12     Products P ON OT.ProductID = P.ProductID
13 GROUP BY
14     P.ProductID, P.Name, P.Category;
15
16
17 Select * from View_1_OTAverageOrderValue

```

### Output:

17 | Select \* from View\_1\_OTAverageOrderValue

	ProductID	ProductName	Category	AverageOrderValue
1	1	Consider	Electronics	49.78
2	2	At	Toys	159.13
3	6	Area	Books	114.78
4	7	Happy	Toys	174.17
5	8	Effect	Electronics	91.92
6	13	Play	Home Goods	87.52
7	14	Almost	Toys	178.32
8	15	Statement	Books	129.13
9	18	No	Electronics	153.63
10	19	Early	Electronics	175.37
11	20	Talk	Electronics	114.12
12	21	Involve	Clothing	88.09
13	23	Ahead	Home Goods	154.14
14	24	Find	Home Goods	68.88
15	25	Young	Books	63.78
16	26	Security	Books	20.81
17	28	Executive	Clothing	84.75
18	29	Seven	Home Goods	141.89
19	31	Line	Toys	152.73

Query executed successfully.

**View 2 - Segment customers based on total spend, purchase frequency, and loyalty tier (LoyaltyAccounts.TierLevel).**

**Example: "High-Value Customers" (Top 10% spenders), "One-Time Buyers," "Loyalty Champions."**

- We looked at each customer's total spending and how often they made purchases.
- We combined both online and in-store purchases to get a full view of customer activity.
- We checked if the customer belongs to a loyalty program and noted their tier level.
- We identified the top 10% spenders as "High-Value Customers."
- Customers who only purchased once are marked as "One-Time Buyers."
- Customers with high loyalty status like Gold or Platinum are called "Loyalty Champions."
- Everyone else is grouped as a "Regular Customer" for simpler analysis.

We can analyse these details from this view2:

- Total money spent (online + in-store combined)
- Number of purchases made.
- Their loyalty program tier (if they have one)
- A customer segment label:  
(High-Value Customer, One-Time Buyer, Loyalty Champion, or Regular Customer).
- Customers who are in the top 10% by total spending (high-priority for special offers or retention programs).
- Customers who purchased only once, it could be targeted for re-engagement campaigns.
- Customers who are highly engaged in the loyalty program (Gold/Platinum tier).
- Tailor promotions, discounts, and communications specifically for each group.

```

WITH LoyaltyRanked AS (
    SELECT *,
        ROW_NUMBER() OVER (PARTITION BY CustomerID
                            ORDER BY
                                CASE TierLevel
                                    WHEN 'Platinum' THEN 4
                                    WHEN 'Gold' THEN 3
                                    WHEN 'Silver' THEN 2
                                    WHEN 'Bronze' THEN 1
                                    ELSE 0
                                END DESC) AS rn
    FROM LoyaltyAccounts
),
BestLoyalty AS (
    SELECT CustomerID, TierLevel
    FROM LoyaltyRanked
    WHERE rn = 1
),
CustomerSpend AS (
    SELECT
        C.CustomerID,
        C.Name AS CustomerName,
        SUM(COALESCE(OT.Amount, 0) + COALESCE(IST.Amount, 0)) AS TotalSpend,
        COUNT(DISTINCT OT.OrderID) + COUNT(DISTINCT IST.TransactionID) AS PurchaseFrequency,
        COALESCE(BL.TierLevel, 'None') AS TierLevel
    FROM
        Customers C
    LEFT JOIN
        OnlineTransactions OT ON C.CustomerID = OT.CustomerID
    LEFT JOIN
        InStoreTransactions IST ON C.CustomerID = IST.CustomerID
    LEFT JOIN
        BestLoyalty BL ON C.CustomerID = BL.CustomerID
    GROUP BY
        C.CustomerID, C.Name, BL.TierLevel
),

38 Percentile AS (
39     SELECT
40         PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY TotalSpend)
41         OVER () AS Spend90Percentile
42     FROM CustomerSpend
43 )
44 SELECT
45     CS.CustomerID,
46     CS.CustomerName,
47     CS.TotalSpend,
48     CS.PurchaseFrequency,
49     CS.TierLevel,
50     CASE
51         WHEN CS.TotalSpend >= (SELECT TOP 1 Spend90Percentile FROM Percentile) THEN 'High-Value Customer'
52         WHEN CS.PurchaseFrequency = 1 THEN 'One-Time Buyer'
53         WHEN CS.TierLevel IN ('Gold', 'Platinum') AND CS.TotalSpend > 0 THEN 'Loyalty Champion'
54         ELSE 'Regular Customer'
55     END AS CustomerSegment
56
57 FROM
58     CustomerSpend CS;
59
60 Select * from View_2_CustomerSegmentation;

```

**Output:** Select \* from View\_2\_CustomerSegmentation;



101 %

60 | Select \* from View\_2\_CustomerSegmentation;

Results Messages

	CustomerID	CustomerName	TotalSpend	PurchaseFrequency	TierLevel	CustomerSegment
1	1	Mrs. Crystal Carroll	299.56	3	Platinum	Loyalty Champion
2	2	Debra Newman DVM	47.20	1	None	One-Time Buyer
3	3	Thomas Mason	0.00	0	Gold	Regular Customer
4	4	Karla Hill	0.00	0	None	Regular Customer
5	5	Jeffrey Underwood	175.38	2	Gold	Loyalty Champion
6	6	Justin Lowe	94.31	1	Gold	One-Time Buyer
7	7	Jason Daniels	58.81	1	None	One-Time Buyer
8	8	Jennifer Hill	90.37	1	Gold	One-Time Buyer
9	9	Stephanie Richardson	352.96	3	Platinum	Loyalty Champion
10	10	Jo Zimmerman	0.00	0	Gold	Regular Customer
11	11	Andrew Williams	534.29	3	Platinum	Loyalty Champion
12	12	Robin Odum	402.86	3	Bronze	Regular Customer
13	13	Gina Anderson	693.24	3	Gold	Loyalty Champion
14	14	Davin Smith	50.29	1	Platinum	One-Time Buyer
15	15	Sarah Roberts	0.00	0	None	Regular Customer
16	16	Andre Wright	160.07	2	Silver	Regular Customer
17	17	Karen Roberts	292.74	2	Gold	Loyalty Champion
18	18	Alisha Phillips	1191.46	4	Silver	High-Value Customer
19	19	Brandon Armstrong	266.40	2	None	Regular Customer
20	20	Luis Jones	897.08	4	Gold	High-Value Customer
21	21	Anna Williams	239.86	2	Silver	Regular Customer
22	22	Evelyn Burton	214.25	2	Silver	Regular Customer
23	23	Donald Jacobs	293.88	2	Platinum	Loyalty Champion
24	24	Anthony Davis	141.67	2	None	Regular Customer
25	25	Molly Bridges	248.03	2	Silver	Regular Customer
26	26	Michelle Wood	1825.12	5	Gold	High-Value Customer
27	27	Kristen Romero	145.70	2	Gold	Loyalty Champion
28	28	Vanessa Anderson	0.00	0	Bronze	Regular Customer

### View 3 - Analyze DateTime to find peak days and times in-store vs. online.

- We combined online and in-store transaction data into a single view.
- We extracted the day of the week (like Monday, Tuesday) and the hour of the day from each transaction.
- We counted how many transactions happened in each day-hour slot for both channels.
- We grouped the results by day, hour, and channel (Online or In-Store).
- This helps identify when customers shop the most, both by day and time.

```

1 CREATE OR ALTER VIEW View3_PeakTimes AS
2 WITH BaseData AS (
3     SELECT
4         'Online' AS Channel,
5         DATENAME(WEEKDAY, OT.DateTime) AS DayOfWeek,
6         DATEPART(HOUR, OT.DateTime) AS HourOfDay,
7         COUNT(*) AS TransactionCount
8     FROM OnlineTransactions OT
9     GROUP BY
10         DATENAME(WEEKDAY, OT.DateTime),
11         DATEPART(HOUR, OT.DateTime)
12
13     UNION ALL
14
15     SELECT
16         'InStore' AS Channel,
17         DATENAME(WEEKDAY, IST.DateTime) AS DayOfWeek,
18         DATEPART(HOUR, IST.DateTime) AS HourOfDay,
19         COUNT(*) AS TransactionCount
20     FROM InStoreTransactions IST
21     GROUP BY
22         DATENAME(WEEKDAY, IST.DateTime),
23         DATEPART(HOUR, IST.DateTime)
24 ),
25 RankedData AS (
26     SELECT *,
27         RANK() OVER (PARTITION BY Channel, DayOfWeek ORDER BY TransactionCount DESC) AS HourRank
28     FROM BaseData
29 )
30 SELECT
31     Channel,
32     DayOfWeek,
33     HourOfDay,
34     TransactionCount
35 FROM RankedData
36 WHERE HourRank = 1;
37

```

We can analyse these details from this view3:

- Find out which weekdays (Monday, Friday, Saturday, etc.) are busiest for both online and in-store sales.
- See which hours (morning, afternoon, evening) have the highest number of purchases.
- Understand if customers prefer shopping online during specific times and visiting stores at different times.
- For in-store, schedule more employees during peak hours.
- For online, run targeted ads or flash sales during high-traffic periods.
- Reduce wait times, improve service quality, and plan better based on when customers are most active.

**Output:** Select \* from View\_3\_PeakTimes

	Channel	DayOfWeek	HourOfDay	TransactionCount
1	InStore	Friday	21	3
2	InStore	Monday	19	3
3	InStore	Saturday	3	2
4	InStore	Saturday	20	2
5	InStore	Saturday	14	2
6	InStore	Sunday	2	3
7	InStore	Thursday	20	3
8	InStore	Tuesday	22	2
9	InStore	Tuesday	4	2
10	InStore	Tuesday	1	2
11	InStore	Tuesday	11	2
12	InStore	Wednesday	21	2
13	InStore	Wednesday	20	2
14	InStore	Wednesday	7	2
15	InStore	Wednesday	6	2
16	Online	Friday	5	3
17	Online	Monday	16	3
18	Online	Saturday	8	2
19	Online	Saturday	3	2

#### View 4 - Number of interactions and resolution success rates per agent (ResolutionStatus)

- We listed each agent along with the total number of customer interactions they handled.
- We counted how many of those interactions were successfully resolved.
- We calculated each agent's resolution success rate as a percentage (with 2 decimal places).
- We used a left join to make sure even agents with zero interactions are included.
- This view helps measure and compare agent performance based on how effectively they resolve customer issues.

```

105
106 CREATE VIEW View_AgentPerformance AS
107 SELECT
108     A.AgentID,
109     A.Name AS AgentName,
110     COUNT(CSI.InteractionID) AS TotalInteractions,
111     SUM(CASE WHEN CSI.ResolutionStatus = 'Resolved' THEN 1 ELSE 0 END) AS ResolvedInteractions,
112     CASE
113         WHEN COUNT(CSI.InteractionID) = 0 THEN NULL
114         ELSE CAST((SUM(CASE WHEN CSI.ResolutionStatus = 'Resolved' THEN 1 ELSE 0 END) * 1.0 / COUNT(CSI.InteractionID)) * 100 AS DECIMAL(10,2))
115     END AS ResolutionSuccessRate
116 FROM
117     Agents A
118 LEFT JOIN
119     CustomerServiceInteractions CSI ON A.AgentID = CSI.AgentID
120 GROUP BY
121     A.AgentID, A.Name;
122
123 -----
124 Select * from View_AgentPerformance;

```

#### We can analyze these details from this view4:

- Identify agents who have the highest success rates in resolving customer issues.
- Spot agents who have a low resolution success rate and might need additional coaching.
- See which agents are handling more customer interactions — this helps in balancing team workloads.
- Track how well the overall support team is resolving issues, and how it improves over time.
- Set performance goals or KPIs (Key Performance Indicators) based on actual agent success rates.

**Output:** Select \* from View\_4\_AgentPerformance;

22 | `Select * from View_4_AgentPerformance;`

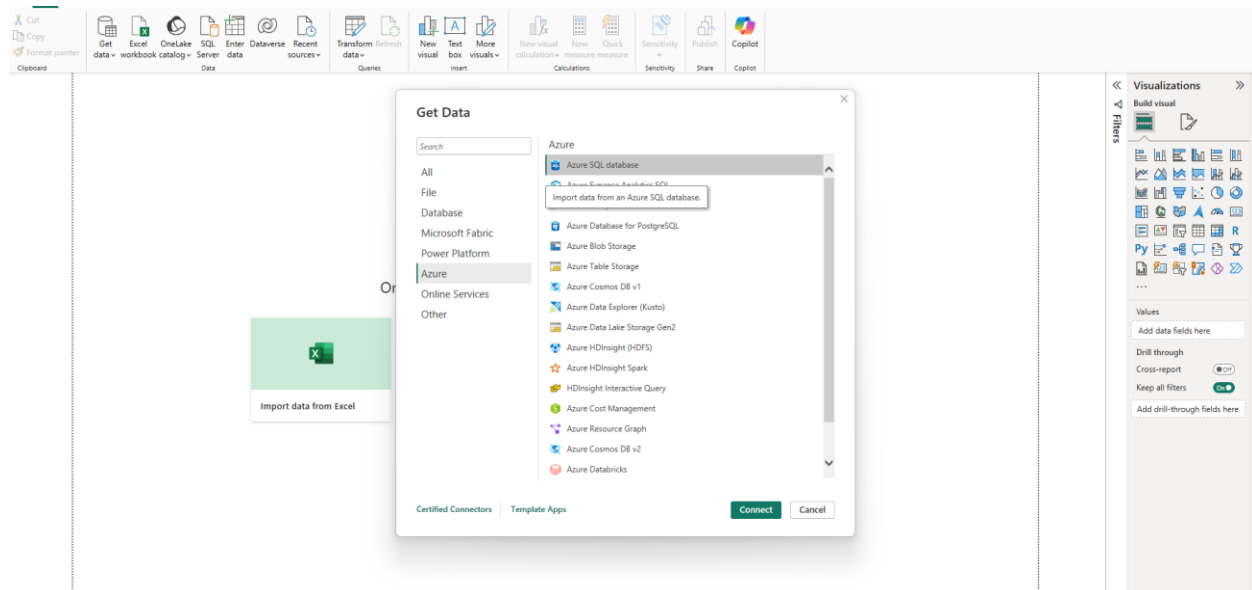
101 %

Results Messages

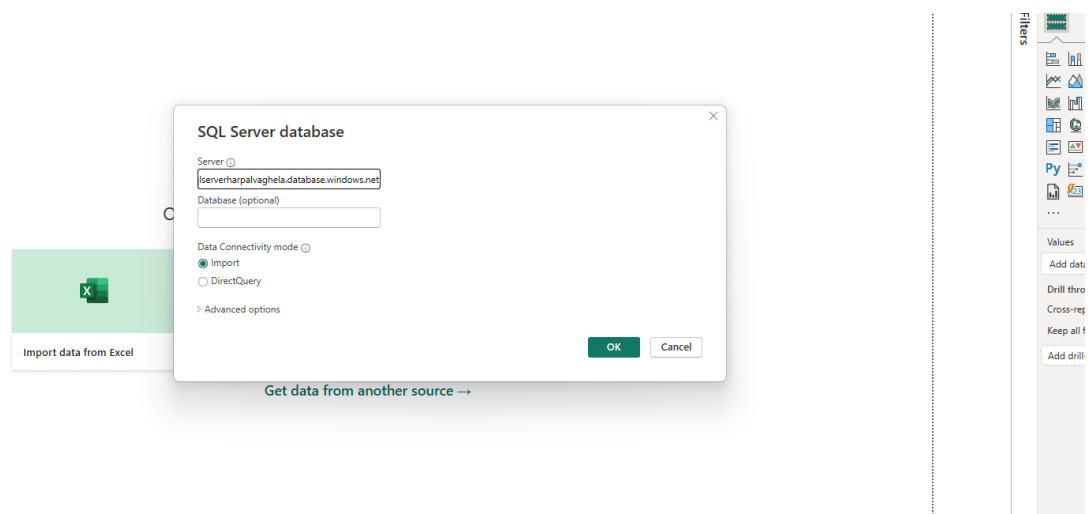
	AgentID	AgentName	TotalInteractions	ResolvedInteractions	ResolutionSuccessRate
19	19	Eddie Pierce	1	0	0.00
20	20	Julia Owens	1	0	0.00
21	21	Cindy Gomez	3	2	66.67
22	22	Greg Smith	0	0	0.00
23	23	Shane Hernandez	0	0	0.00
24	24	Gregory Chavez	0	0	0.00
25	25	Elizabeth Casey	0	0	0.00
26	26	Doris Knight	0	0	0.00
27	27	Shawn Gill	2	1	50.00
28	28	Ricky Davenport	1	0	0.00
29	29	Jessica Mora	2	1	50.00
30	30	Vanessa Henson	0	0	0.00
31	31	Kimberly Chamb...	2	0	0.00
32	32	Debbie Stewart	2	1	50.00
33	33	Jimmy Weber	1	1	100.00
34	34	Heather Harrison	4	1	25.00
35	35	Julia Moon	2	2	100.00
36	36	Thomas Rivers	0	0	0.00
37	37	Michael Casey	2	0	0.00
38	38	Sabrina Miller	2	0	0.00
39	39	Jessica Keller	0	0	0.00
40	40	Barbara Craig	0	0	0.00
41	41	Nicole Ortega	1	0	0.00
42	42	Jeff Ray	0	0	0.00
43	43	Ronnie Williams...	2	1	50.00
44	44	Nicholas Richar...	1	1	100.00
45	45	Robert Davis	1	0	0.00
46	46	Stephen Dickson	1	0	0.00

#### Step 4: Power BI Dashboard/Report

Select Azure -> Azure SQL Database and click on connect



Enter Server URL from Azure Portal (Azure SQL Server-> Overview tab)



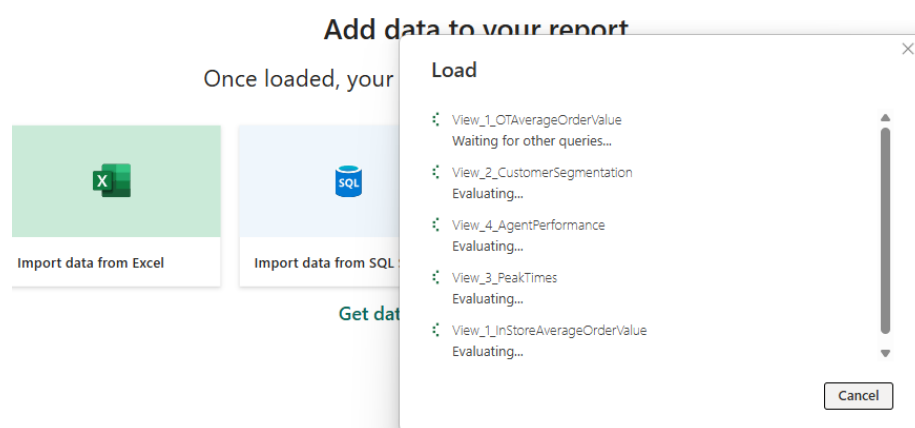
## Data Connectivity Mode

- **Import Mode:** Power BI copies the full data from your source (SQL Database) into Power BI's internal storage (in-memory).
  - Fastest performance for visuals because data is preloaded.
  - Best for small to medium datasets.
  - You need to refresh manually or schedule refreshes when source data changes.
  - Report is disconnected from live source after load.
  - Good when performance is critical and data changes aren't real-time.
- **DirectQuery Mode:** Power BI does NOT import data. Instead, every time you open a report or interact with a visual (like filter or click), it queries the database live.
  - Always live data, no refresh needed.
  - Slower performance (depends on database speed and network).

- Useful for very large datasets where importing is impractical.
- Database needs to be highly optimized (indexes, partitions, etc.) to support fast querying.
- Good when you need real-time reporting or when datasets are very large.

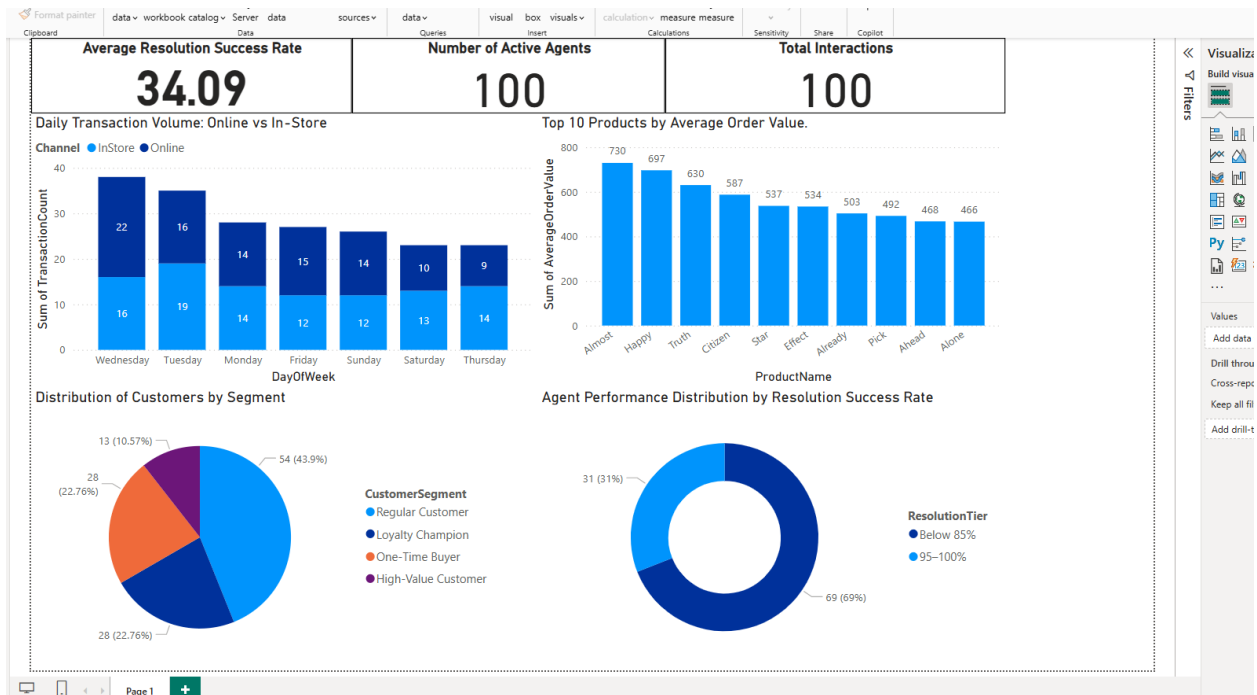
Click on Ok.

Select the views which we have created.



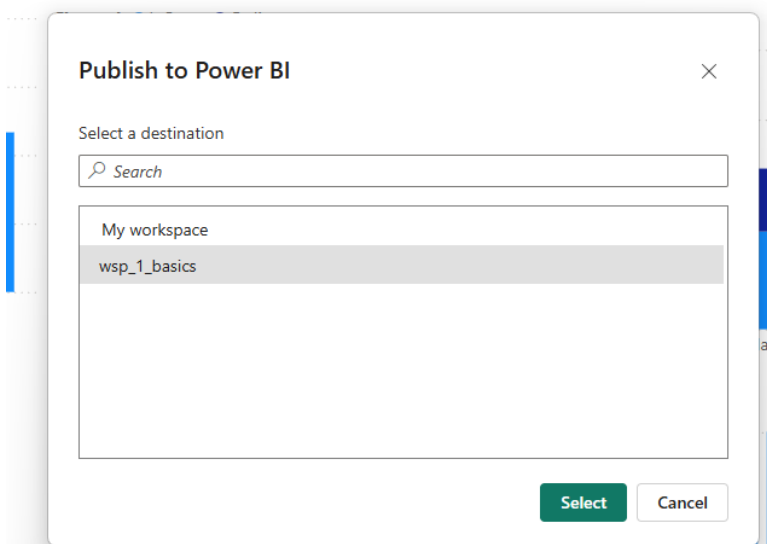
Click on Load.

We have created this dashboard in PowerBI



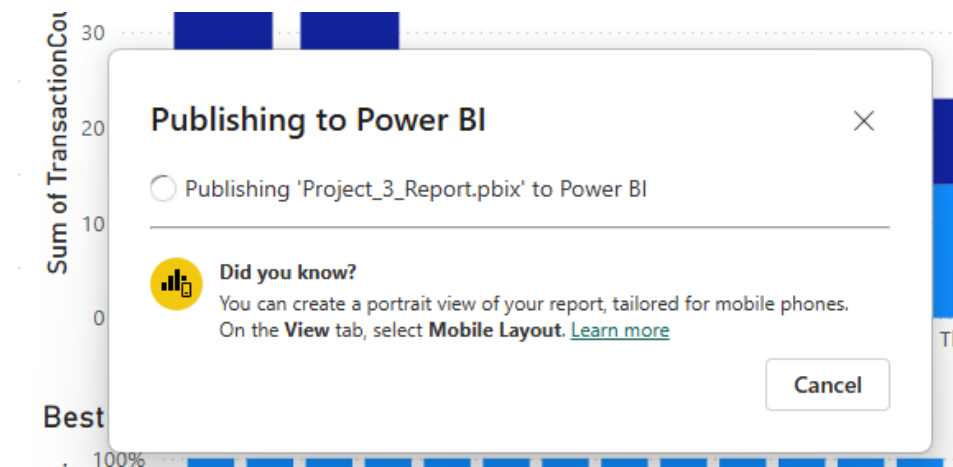
## Step 5: Publish to Fabric Workspace

Now, let's publish it into Fabric Workspace.

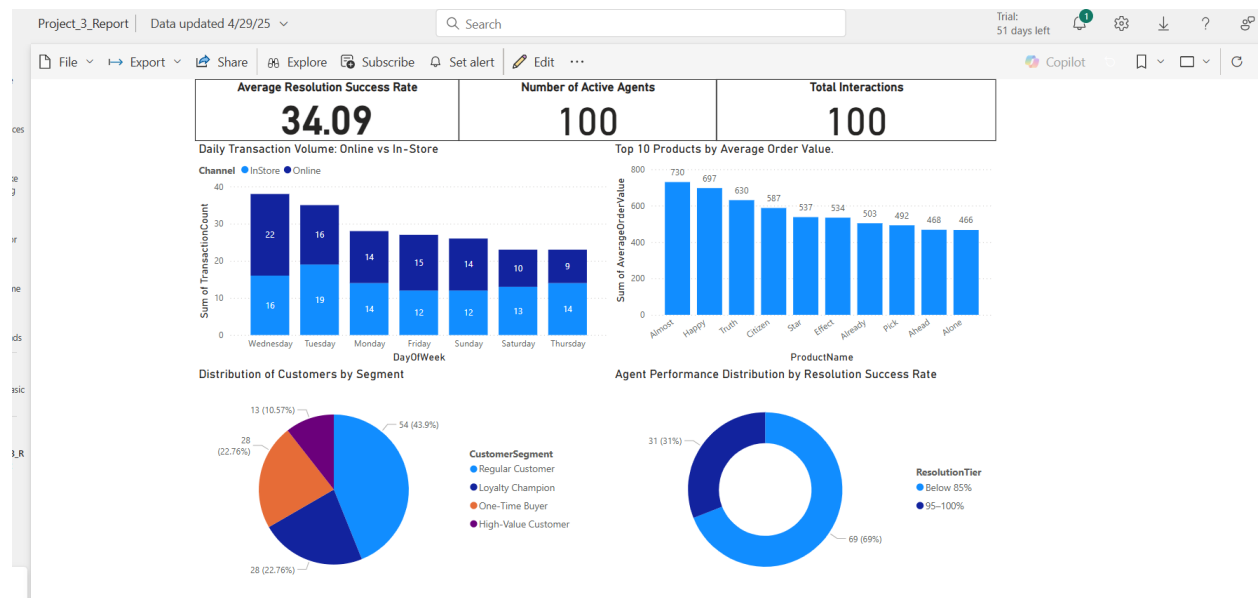


Click on Publish icon in Home -> Publish

Select the Fabric Workspace



Check the report in to fabric workspace.



## Points to Remember

- Ingest data properly into the Bronze layer (ADLS folders organized upfront).
- Create Parent tables first (Customers, Products) to avoid constraint errors.
- Filter null records early and remove duplicates before moving to the Silver layer.
- Use Alter Row in dataflows for correct upsert operations into SQL tables.
- Secure passwords via Azure Key Vault instead of hardcoding credentials.
- Gold Layer views simplify business insights (AOV, Customer Segments, Peak Times, Agent Performance).
- Import Mode = better performance, DirectQuery = live data, slower.
- Apply Top N filters in Power BI visuals (Top 10 products, Top 10 agents) for clarity.
- The Gold Layer is for business reporting, making it clean, aggregated, and ready for analysis.