ATLIQ
MARTS

# OPTIMIZING SUPPLY CHAIN PERFORMANCE FOR ATLIQ MART

END-TO-END DATA ENGINEERING PORJECT FOR SOLVING SERVICE LEVEL ISSUES AT ATLIQ MART


1-خالد شهام

2-بسنت ياسر

3-لبنى عيد

4-أحمد علاء

5-محمد صبري


Prepared for

**ATLIQ MART STAKEHOLDERS**

01

# Problem Statement

**AtliQ Mart, a growing FMCG manufacturer, is experiencing service level issues across three cities: Surat, Ahmedabad, and Vadodara.**

### Key Concerns:

- **Delivery Delays**: Some key customers did not renew their annual contracts due to **consistent delays in product deliveries**.
- **Incomplete Orders**: Orders were either not delivered on time or not delivered in full, leading to customer dissatisfaction.

### Business Goal:

- **AtliQ Mart's management aims to track and improve On-Time and In-Full Delivery Service Levels on a daily basis.**

### ✦ Metrics to track:

- **On-time Delivery (OT) %**
- **In-full Delivery (IF) %**
- **On-Time In-Full (OTIF) %**

# Project Overview

## ✦ Objective:

- **Address service level issues at AtliQ Mart by predicting late and incomplete deliveries.**
- **Track key metrics: On-time Delivery (OT%), In-full Delivery (IF%), and OTIF% (On Time In Full).**

## ✦ Key Deliverables:

- **A well-designed SQL database schema and populated database.**
- **Integrated Azure Data services setup.**
- **Deployed machine learning model.**
- **Final report and presentation.**

## ✦ Technologies Used:

- **SQL Server, Python (Pandas, ), azure.**

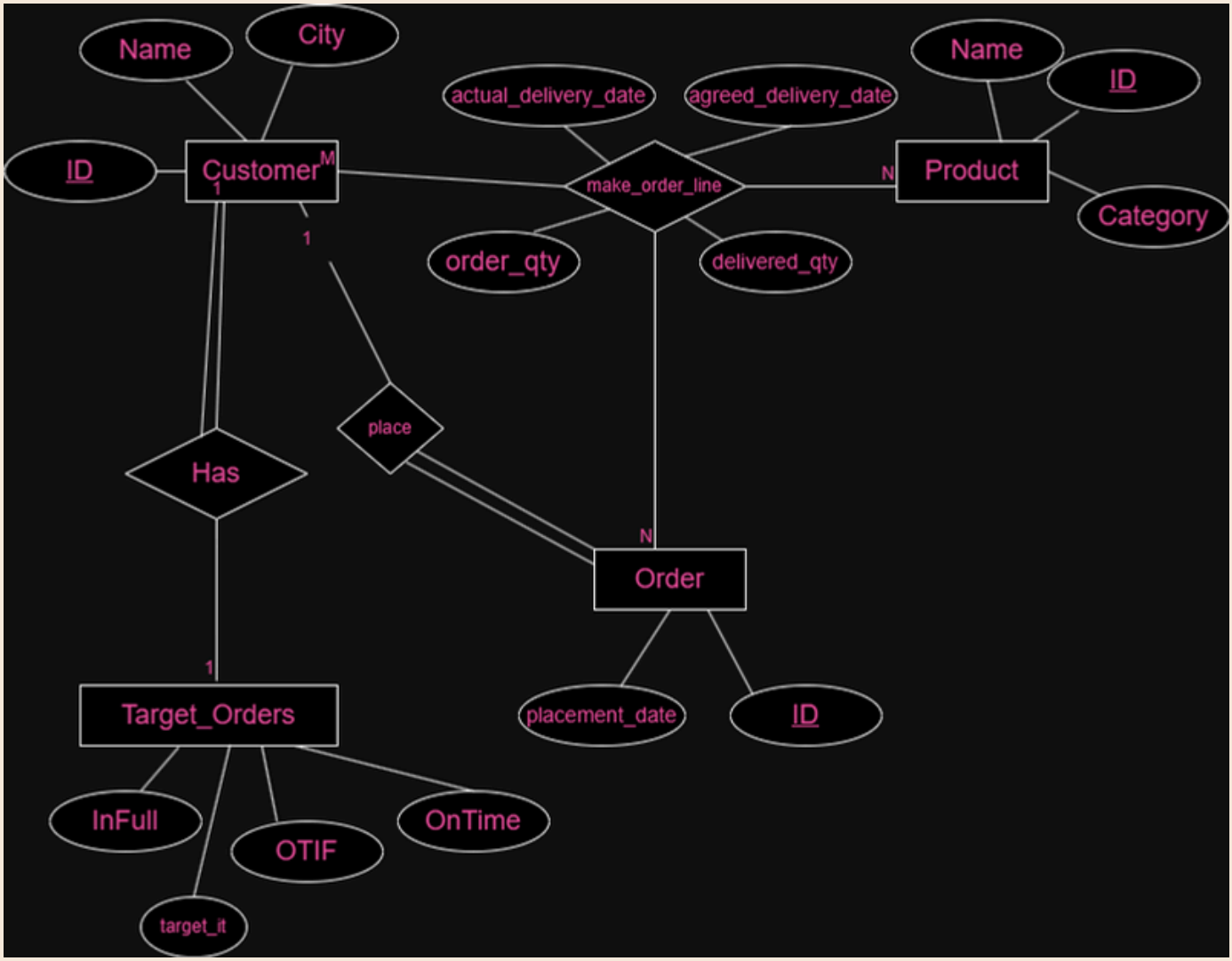
# 01 - Entity Relationship Diagram (ERD) and Data Mapping

## Objective:

- **Design a relational database schema to store and manage all relevant data for analyzing service level issues at AtliQ Mart.**

## ERD Creation Tool

- **Drawio**



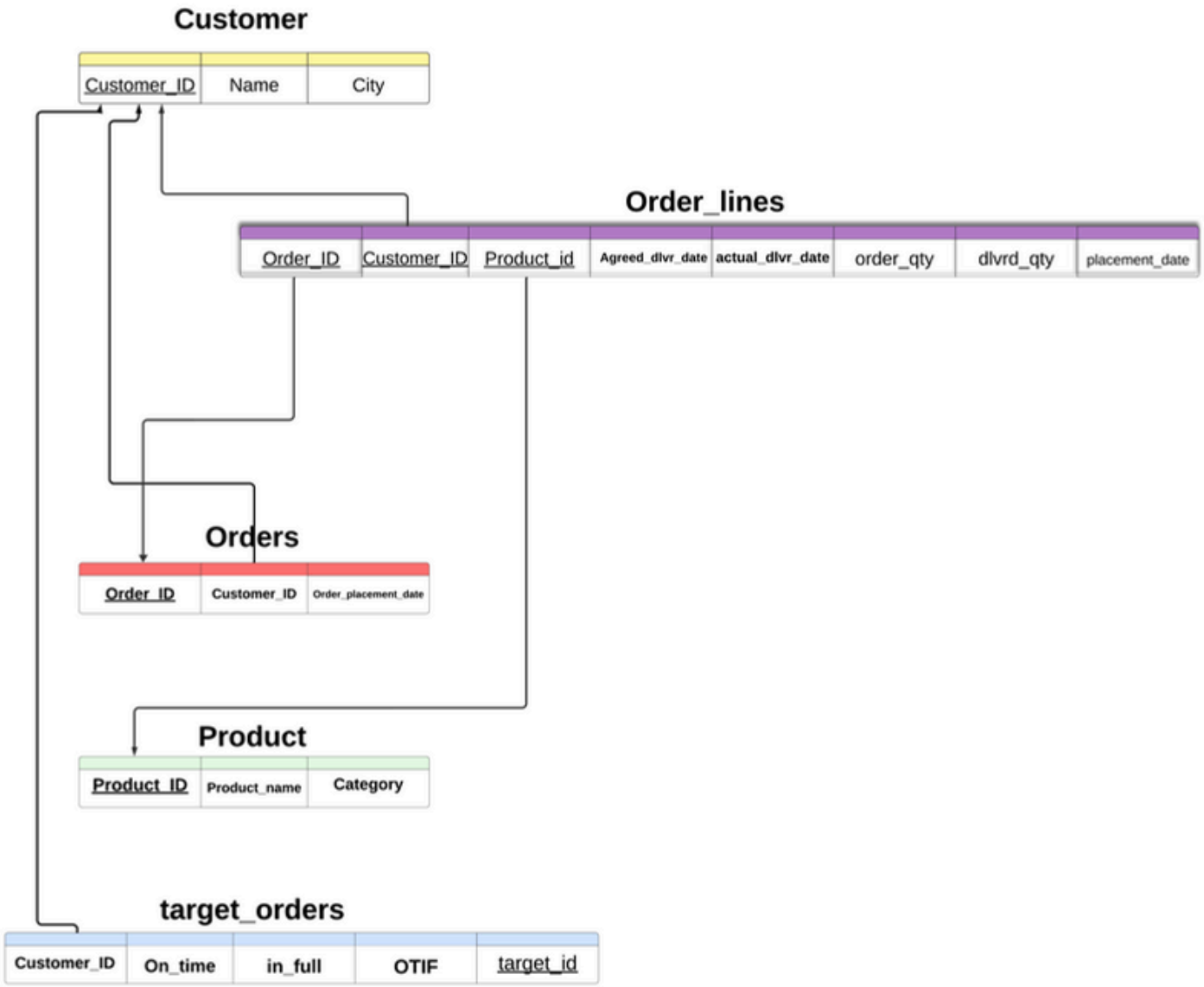
# 01 - Entity Relationship Diagram (ERD) and Data Mapping

## Objective:

- **Design a relational database schema to store and manage all relevant data for analyzing service level issues at AtliQ Mart.**

## Mapping creation tool:

- **lucidchart**



# 02 - Data Management and SQL Database Setup

✦ **Database Setup:**

- SQL Server schema design for tracking customer orders and deliveries.

✦ **Key Tables:**

- dim_customers || dim_products || dim_date
- || dim_targets_orders || fact_order_lines || fact_orders_aggregate.

Tasks:

- Creating The Database.
- Data extraction from CSV files.
- Calculating new fields: on_time, in_full, otif.

# 02 - Data Management and SQL Database Setup

## ✦ Creating The Database:

Supply_final_demo....emo1 (khalid (64))

```sql
USE master;


CREATE DATABASE Supply_Chain_1;


USE Supply_Chain_1;

-- Create the dimension tables for customers
CREATE TABLE dim_customers (
    customer_id int PRIMARY KEY,
    customer_name VARCHAR(255) NOT NULL,
    city VARCHAR(255) NOT NULL
);

-- Create the dimension table for products
CREATE TABLE dim_products (
    product_name VARCHAR(255) NOT NULL,
    product_id INT PRIMARY KEY,
    category VARCHAR(255) NOT NULL
);

-- Create the dimension table for dates
CREATE TABLE dim_date (
    date DATE PRIMARY KEY,
    mmm_yy date NOT NULL,
    week_no varchar(15) NOT NULL
```

# 02 - Data Management and SQL Database Setup

## ✦ Creating The Database:

```sql
-- Create the dimension table for order targets
CREATE TABLE dim_targets_orders (
    customer_id int,
    ontime_target DECIMAL(10,2),
    infull_target DECIMAL(10,2),
    otif_target DECIMAL(10,2),
    PRIMARY KEY (customer_id),
    FOREIGN KEY (customer_id) REFERENCES dim_customers(customer_id)
);

-- Create the fact table for order lines
CREATE TABLE fact_order_lines (
    order_id varchar(50),
    order_placement_date varchar(255),
    customer_id int,
    product_id INT,
    order_qty int,
    agreed_delivery_date varchar(255),
    actual_delivery_date varchar(255),
    delivered_qty int,
    PRIMARY KEY (order_id, customer_id, product_id),
    FOREIGN KEY (customer_id) REFERENCES dim_customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES dim_products(product_id)
);
```

# 02 - Data Management and SQL Database Setup

## ✦ Creating The Database:

```sql
-- Create the fact table for aggregated orders
CREATE TABLE fact_orders_aggregate (
    order_id varchar(250),
    customer_id int,
    order_placement_date varchar(250),
    on_time INT,
    in_full INT,
    otif INT,
    PRIMARY KEY (order_id, customer_id),
    FOREIGN KEY (customer_id) REFERENCES dim_customers(customer_id)
);
```

# 02 - Data Management and SQL Database Setup

## Data extraction from CSV files.

```
-- Bulk insert data into the dim_customers table from a CSV file
BULK INSERT dim_customers
FROM 'C:\Users\PC\Downloads\dim_customers.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);

-- Preview the first 10 records of dim_customers
SELECT TOP 10 * FROM dim_customers;

-- Bulk insert data into the dim_products table from a CSV file
BULK INSERT dim_products
FROM 'C:\Users\PC\Downloads\dim_products.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);

-- Preview the first 10 records of dim_products
SELECT TOP 10 * FROM dim_products;

-- Bulk insert data into the dim_date table from a CSV file
BULK INSERT dim_date
```

# 02 - Data Management and SQL Database Setup

## Data extraction from CSV files.

```
-- Bulk insert data into the dim_date table from a CSV file
BULK INSERT dim_date
FROM 'C:\Users\PC\Downloads\dim_date.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);

-- Preview the first 10 records of dim_date
SELECT TOP 10 * FROM dim_date;

-- Bulk insert data into the dim_targets_orders table from a CSV file
BULK INSERT dim_targets_orders
FROM 'C:\Users\PC\Downloads\dim_targets_orders.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);

-- Preview the first 10 records of dim_targets_orders
SELECT TOP 10 * FROM dim_targets_orders;

-- Bulk insert data into the fact_order_lines table from a CSV file
BULK INSERT fact_order_lines
FROM 'C:\Users\PC\Downloads\oreder_line_3.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
```

110 %

Connected. (1/1)

# 02 - Data Management and SQL Database Setup

## Data extraction from CSV files.

```
-- Bulk insert data into the fact_orders_aggregate table f
BULK INSERT fact_orders_aggregate
FROM 'C:\Users\PC\Downloads\fact_orders_aggregate_1.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);
```

# 02 - Data Management and SQL Database Setup

## changing data type

```sql
-- Clean up order_placement_date by trimming spaces
UPDATE fact_order_lines
SET order_placement_date = LTRIM(RTRIM(order_placement_date));


-- Add a new column to store the converted order placement date
ALTER TABLE fact_order_lines
ADD order_placement_date_converted DATE;


-- Convert order_placement_date to DATE format and store it in the new column
UPDATE fact_order_lines
SET order_placement_date_converted = TRY_CONVERT(DATE, LTRIM(RTRIM(order_placement_date)), 103);


-- Drop the original VARCHAR order_placement_date column
ALTER TABLE fact_order_lines
DROP COLUMN order_placement_date;


-- Rename the converted date column
EXEC sp_rename 'fact_order_lines.order_placement_date_converted', 'order_placement_date', 'COLUMN';
```

# 02 - Data Management and SQL Database Setup

## changing data type

```sql
-- Clean and convert agreed_delivery_date
UPDATE fact_order_lines
SET agreed_delivery_date = LTRIM(RTRIM(agreed_delivery_date));


-- Add a temporary column to store the converted agreed delivery date
ALTER TABLE fact_order_lines
ADD agreed_delivery_date_temp DATE;


-- Convert agreed_delivery_date to DATE format and store it in the new column
UPDATE fact_order_lines
SET agreed_delivery_date_temp = TRY_CONVERT(DATE, LTRIM(RTRIM(agreed_delivery_date)), 103);


-- Drop the original VARCHAR agreed_delivery_date column
ALTER TABLE fact_order_lines
DROP COLUMN agreed_delivery_date;


-- Rename the temporary column to agreed_delivery_date
EXEC sp_rename 'fact_order_lines.agreed_delivery_date_temp', 'agreed_delivery_date', 'COLUMN';
```

# 02 - Data Management and SQL Database Setup

## changing data type

```sql
-- Alter delivered_qty column to ensure it is an INT
ALTER TABLE fact_order_lines
ALTER COLUMN delivered_qty INT;
```

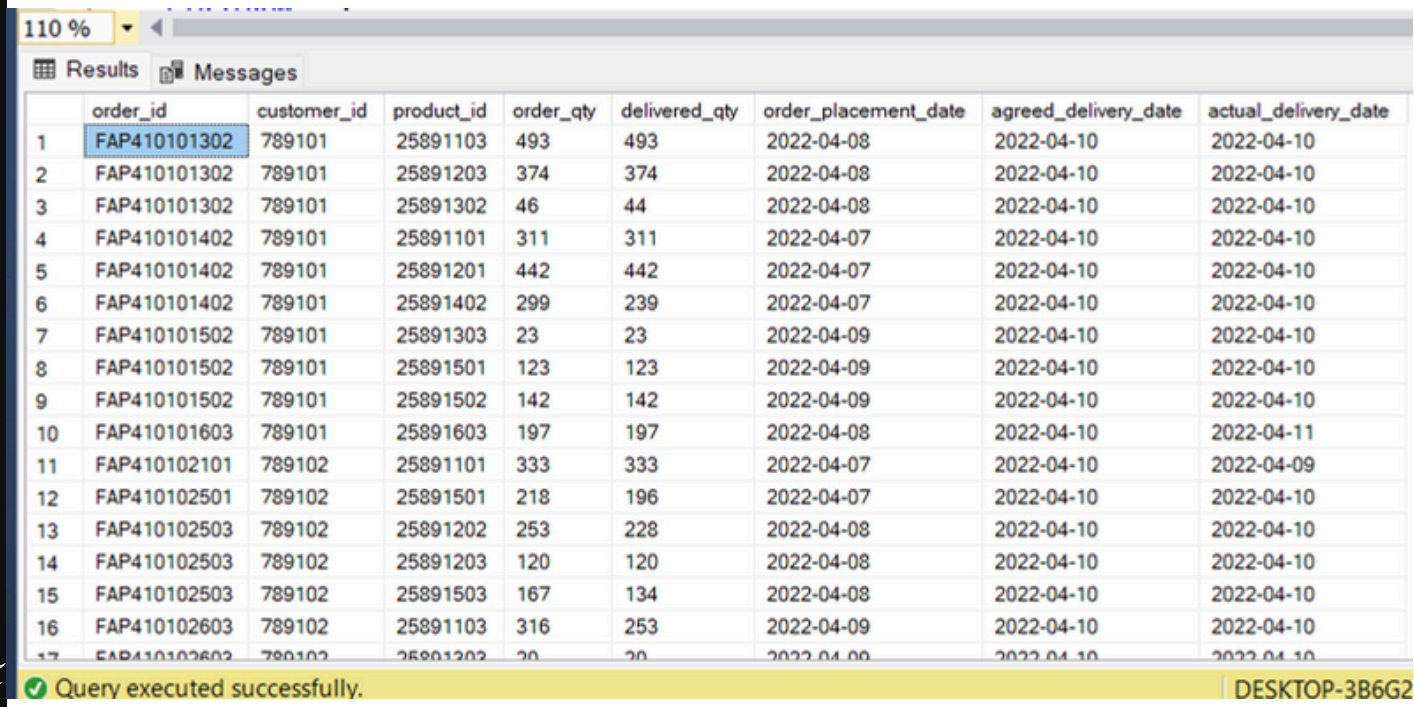
```sql
-- Add a foreign key constraint for order_placement_date in fact_order_lines
ALTER TABLE fact_order_lines
ADD CONSTRAINT FK_fact_order_lines_order_placement_date
FOREIGN KEY (order_placement_date) REFERENCES dim_date(date);


-- Identify distinct agreed_delivery_date values not present in dim_date
SELECT DISTINCT agreed_delivery_date
FROM fact_order_lines
WHERE TRY_CONVERT(DATE, agreed_delivery_date, 103) NOT IN (SELECT date FROM dim_date);



-- Insert missing date into dim_date
INSERT INTO dim_date (date, mmm_yy, week_no)
VALUES ('2022-08-31', '2022-08-01', '36');

-- Add foreign key constraint for agreed_delivery_date
ALTER TABLE fact_order_lines
ADD CONSTRAINT FK_fact_order_lines_agreed_delivery_date
FOREIGN KEY (agreed_delivery_date) REFERENCES dim_date(date);

-- Insert additional missing dates into dim_date
INSERT INTO dim_date (date, mmm_yy, week_no)
VALUES
('2022-09-01', '2022-09-01', '35'),
('2022-09-02', '2022-09-01', '35'),
('2022-09-03', '2022-09-01', '35');
```

# 02 - Data Management and SQL Database Setup

```sql
-- Identify distinct actual_delivery_date values not present in dim_date
SELECT DISTINCT actual_delivery_date
FROM fact_order_lines
WHERE TRY_CONVERT(DATE, actual_delivery_date, 103) NOT IN (SELECT date FROM dim_date);

-- Add foreign key constraint for actual_delivery_date
ALTER TABLE fact_order_lines
ADD CONSTRAINT FK_fact_order_lines_actual_delivery_date
FOREIGN KEY (actual_delivery_date) REFERENCES dim_date(date);
```

```sql
-- Add a new column for order_placement_date in fact_orders_aggregate
ALTER TABLE fact_orders_aggregate
ADD order_placement_date_converted DATE;

-- Convert order_placement_date to DATE format and store it in the new column
UPDATE fact_orders_aggregate
SET order_placement_date_converted = TRY_CONVERT(DATE, LTRIM(RTRIM(order_placement_date)), 103);

-- Check for conversion errors in the new column
SELECT COUNT(*) AS InvalidDateCount
FROM fact_orders_aggregate
WHERE order_placement_date_converted IS NULL;

-- Drop the original order_placement_date column
ALTER TABLE fact_orders_aggregate
DROP COLUMN order_placement_date;

-- Rename the converted date column
EXEC sp_rename 'fact_orders_aggregate.order_placement_date_converted', 'order_placement_date', 'COLUMN';

-- Add a foreign key constraint for order_placement_date in fact_orders_aggregate
ALTER TABLE fact_orders_aggregate
ADD CONSTRAINT FK_fact_orders_aggregate_order_placement_date
FOREIGN KEY (order_placement_date) REFERENCES dim_date(date);
```

# 02 - Data Management and SQL Database Setup



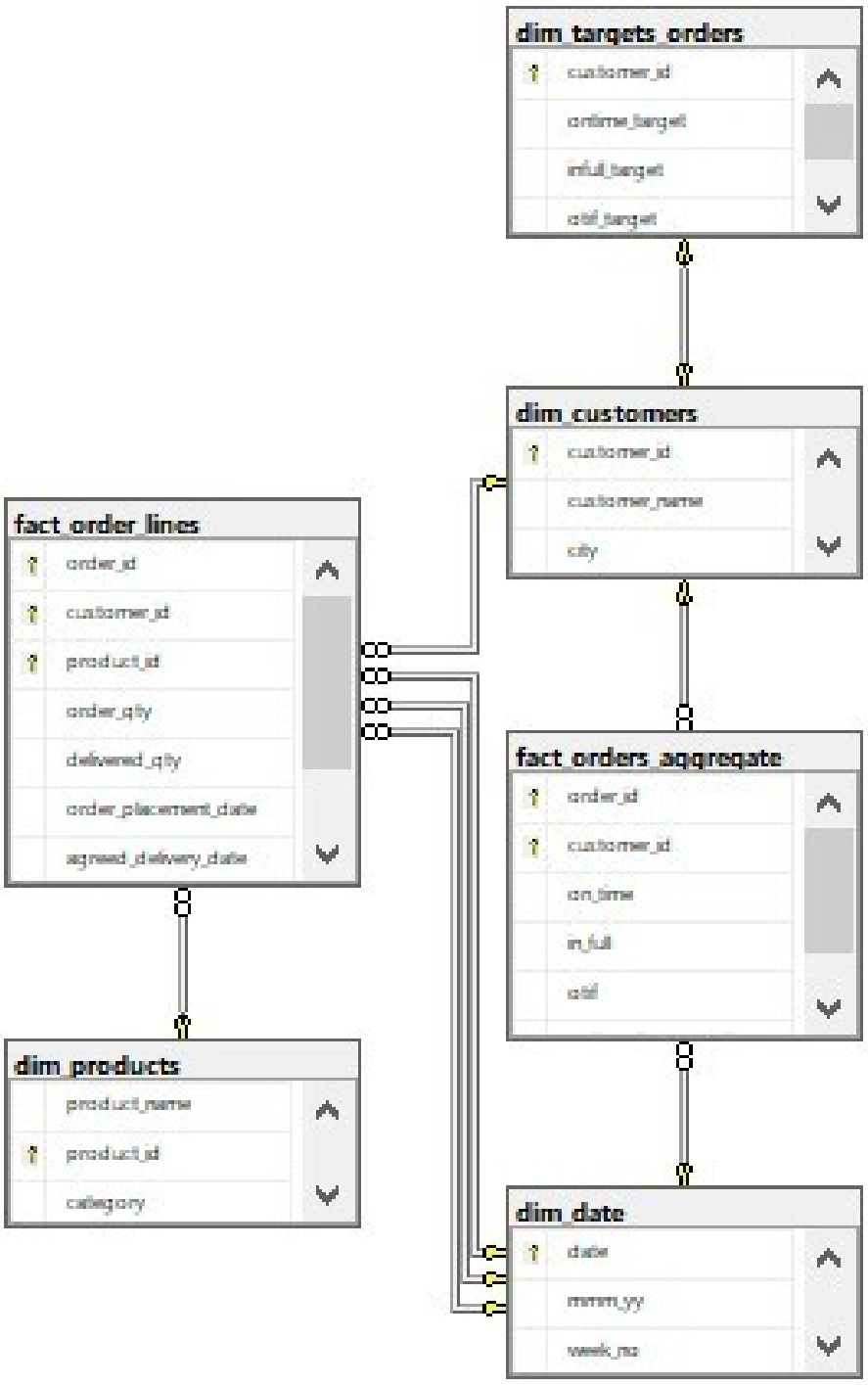
110 %

Results | Messages

| | order_id | customer_id | product_id | order_qty | delivered_qty | order_placement_date | agreed_delivery_date | actual_delivery_date |
|---|---|---|---|---|---|---|---|---|
| 1 | FAP410101302 | 789101 | 25891103 | 493 | 493 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 2 | FAP410101302 | 789101 | 25891203 | 374 | 374 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 3 | FAP410101302 | 789101 | 25891302 | 46 | 44 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 4 | FAP410101402 | 789101 | 25891101 | 311 | 311 | 2022-04-07 | 2022-04-10 | 2022-04-10 |
| 5 | FAP410101402 | 789101 | 25891201 | 442 | 442 | 2022-04-07 | 2022-04-10 | 2022-04-10 |
| 6 | FAP410101402 | 789101 | 25891402 | 299 | 239 | 2022-04-07 | 2022-04-10 | 2022-04-10 |
| 7 | FAP410101502 | 789101 | 25891303 | 23 | 23 | 2022-04-09 | 2022-04-10 | 2022-04-10 |
| 8 | FAP410101502 | 789101 | 25891501 | 123 | 123 | 2022-04-09 | 2022-04-10 | 2022-04-10 |
| 9 | FAP410101502 | 789101 | 25891502 | 142 | 142 | 2022-04-09 | 2022-04-10 | 2022-04-10 |
| 10 | FAP410101603 | 789101 | 25891603 | 197 | 197 | 2022-04-08 | 2022-04-10 | 2022-04-11 |
| 11 | FAP410102101 | 789102 | 25891101 | 333 | 333 | 2022-04-07 | 2022-04-10 | 2022-04-09 |
| 12 | FAP410102501 | 789102 | 25891501 | 218 | 196 | 2022-04-07 | 2022-04-10 | 2022-04-10 |
| 13 | FAP410102503 | 789102 | 25891202 | 253 | 228 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 14 | FAP410102503 | 789102 | 25891203 | 120 | 120 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 15 | FAP410102503 | 789102 | 25891503 | 167 | 134 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 16 | FAP410102603 | 789102 | 25891103 | 316 | 253 | 2022-04-09 | 2022-04-10 | 2022-04-10 |

Query executed successfully. DESKTOP-3B6G2

# 02 - Data Management and SQL Database Setup

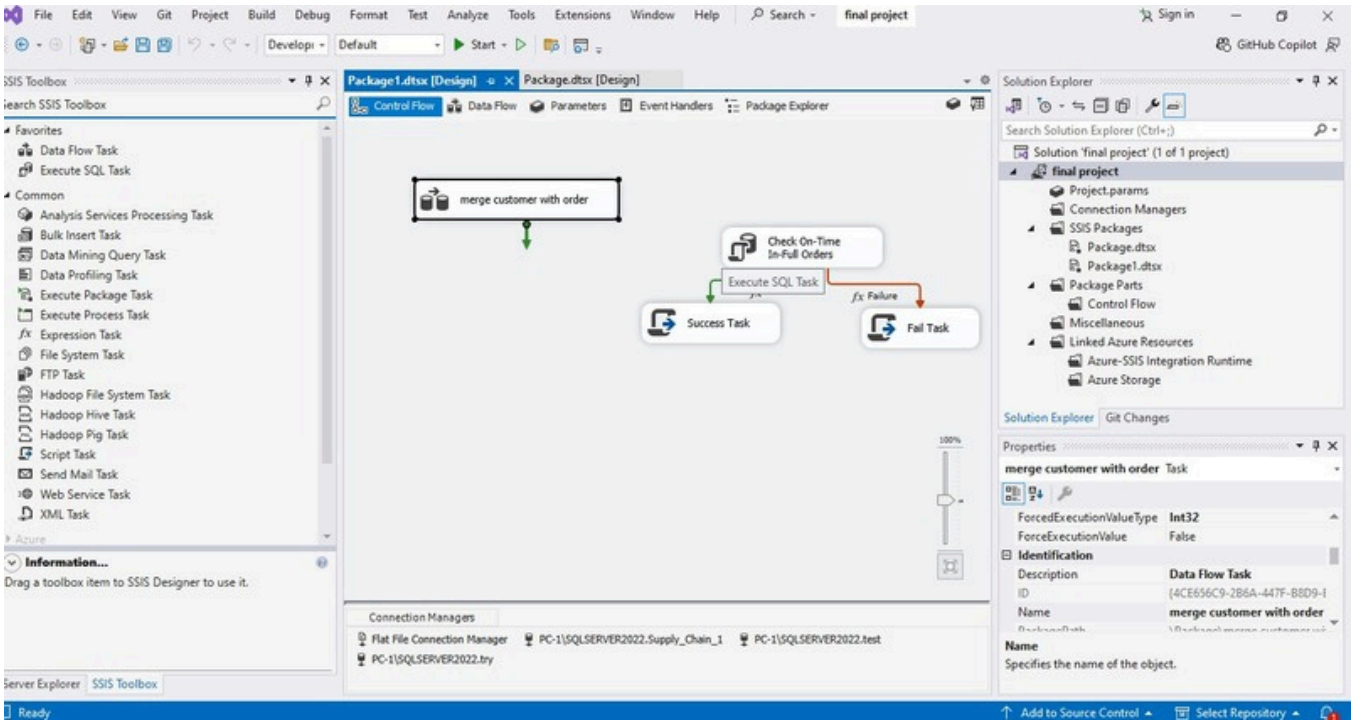


03

# Data Warehousing and Python Programming

## Data Warehouse Implementation

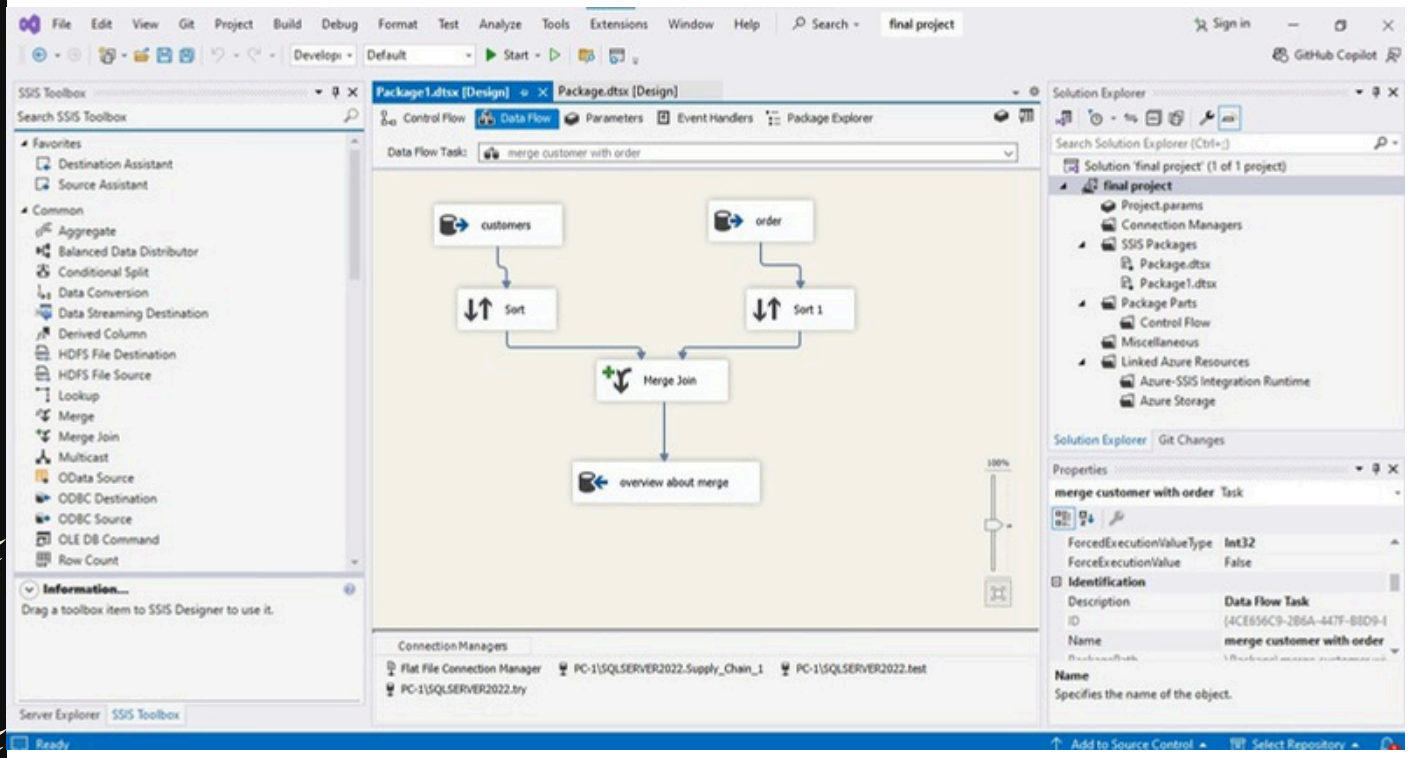
## Tools used: SSIS , control flow, data flow, execute SQL



03

# Data Warehousing and Python Programming

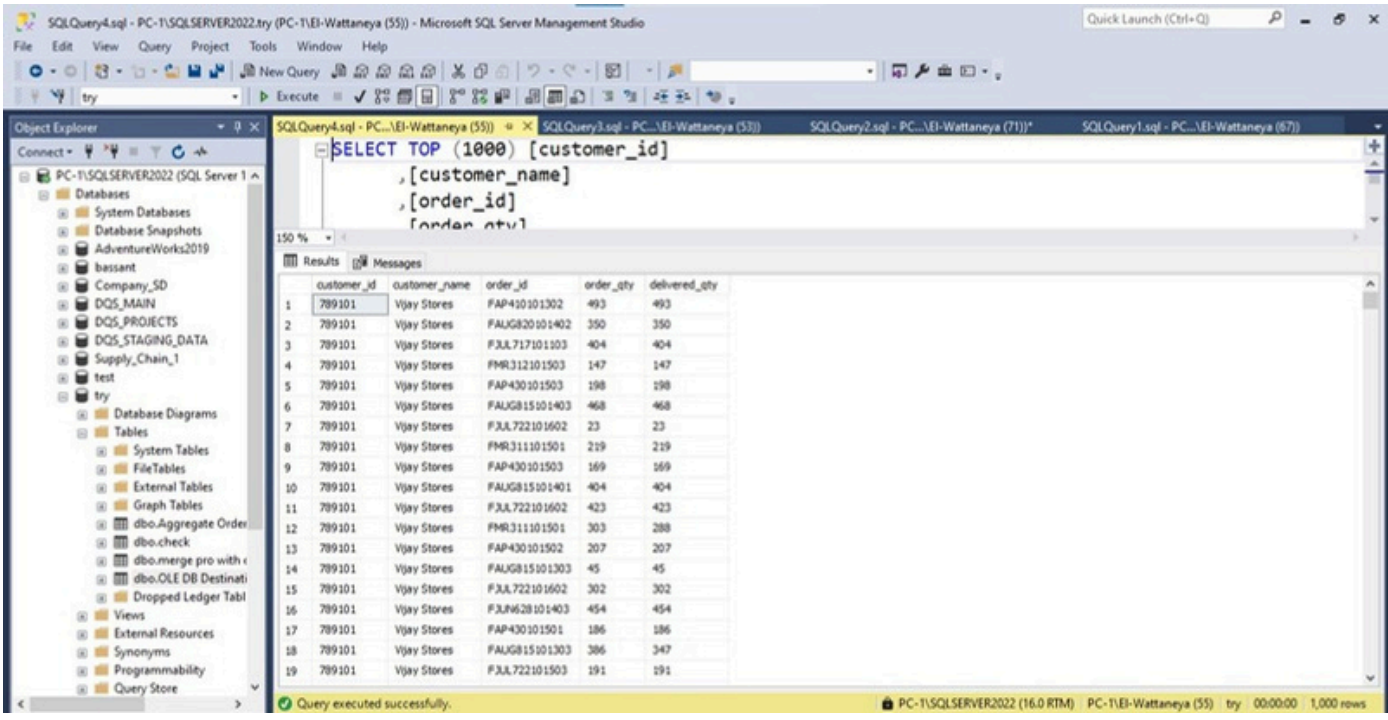
## Data Warehouse Implementation



03

# Data Warehousing and Python Programming

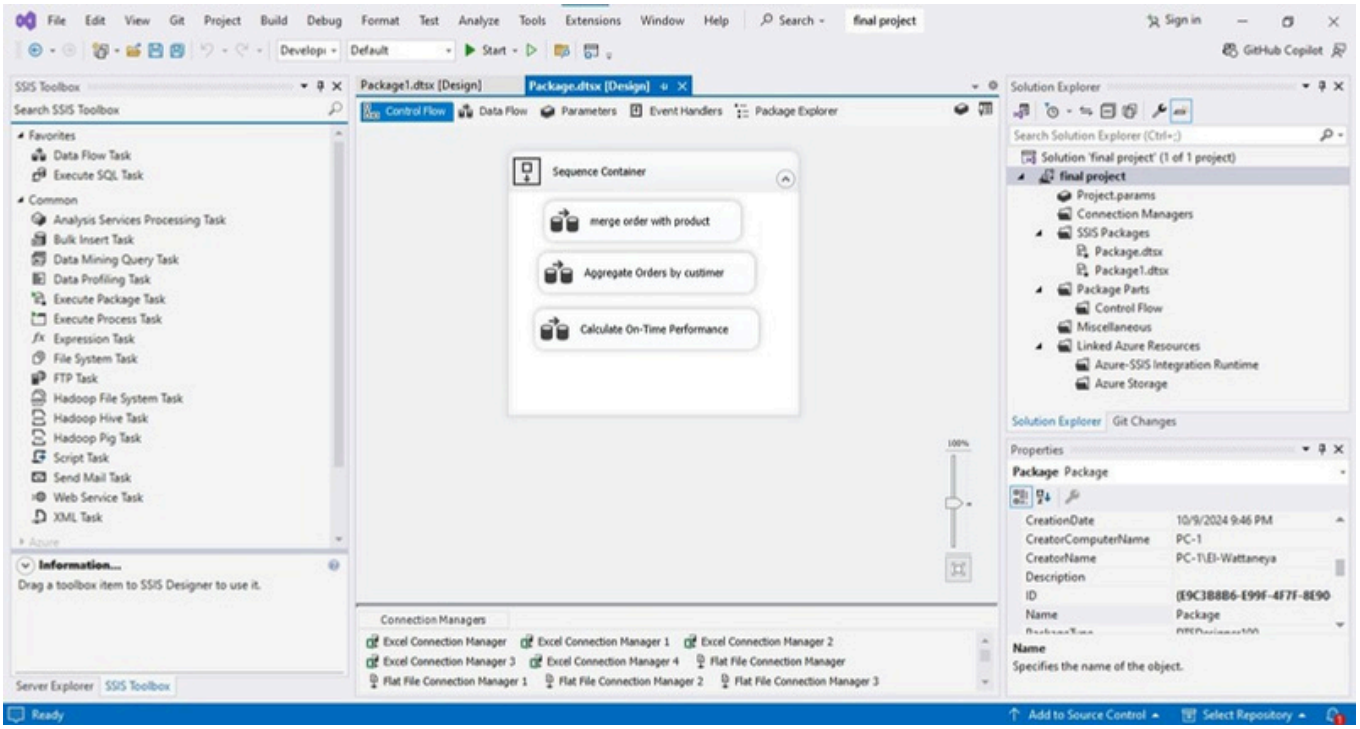
## Data Warehouse Implementation



03

# Data Warehousing and Python Programming

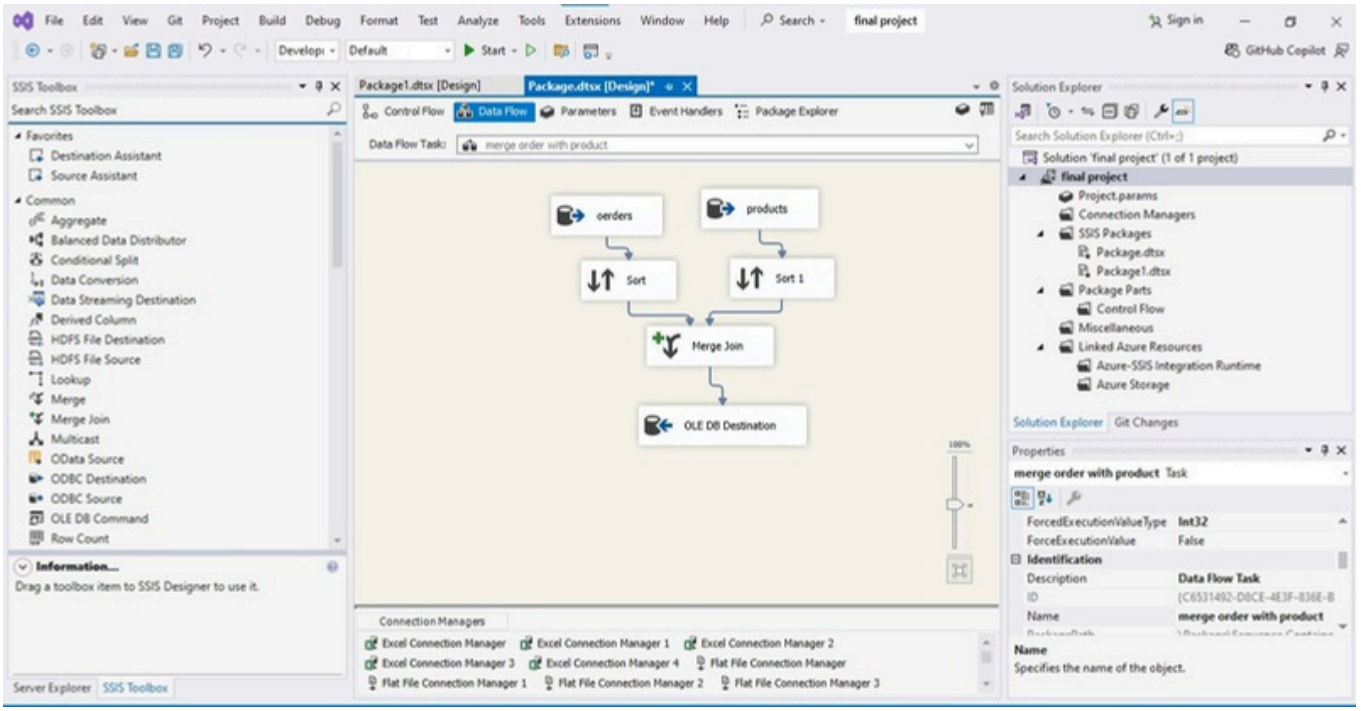
## Data Warehouse Implementation



03

# Data Warehousing and Python Programming

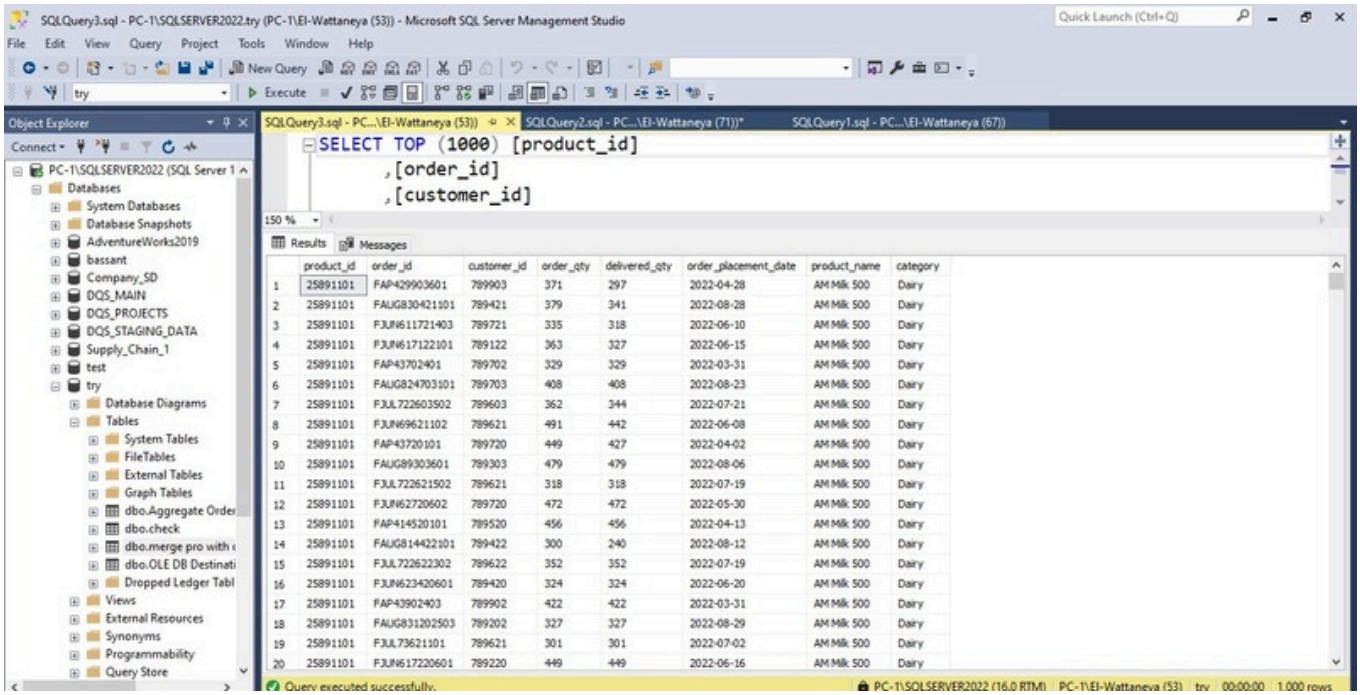
## Data Warehouse Implementation



03

# Data Warehousing and Python Programming

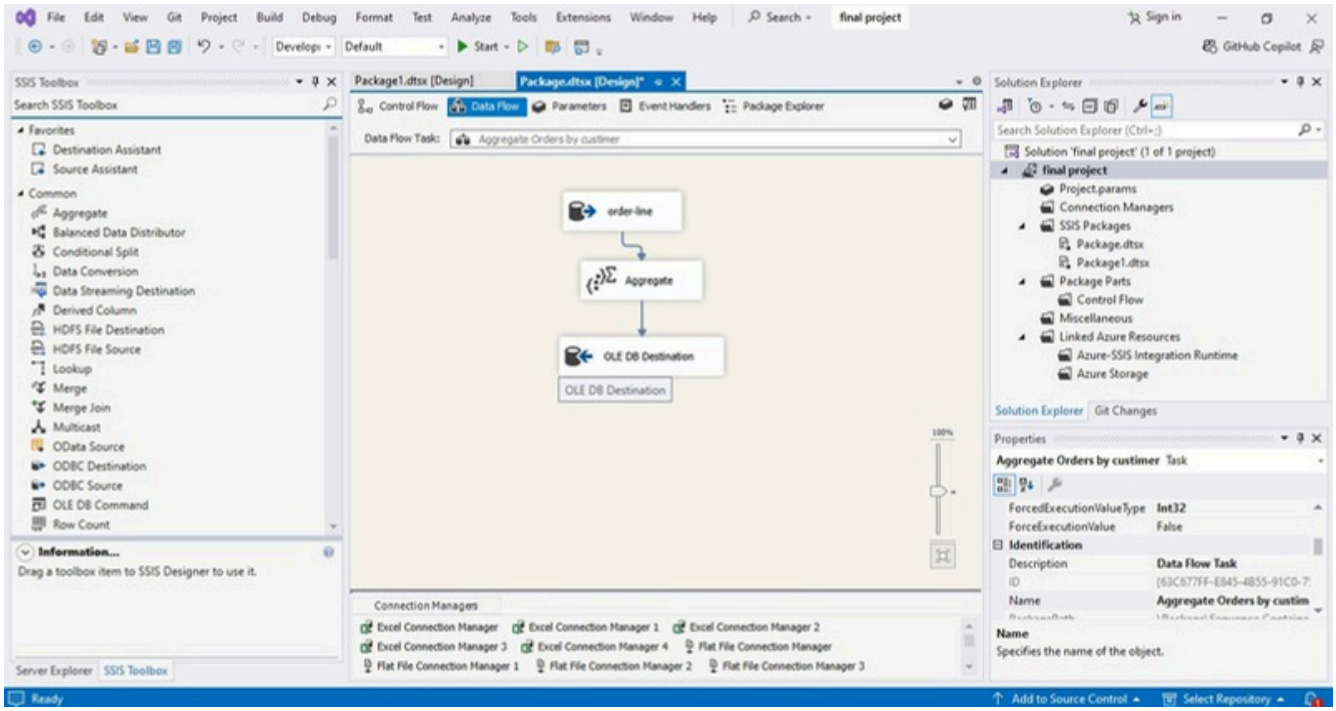
## Data Warehouse Implementation



03

# Data Warehousing and Python Programming

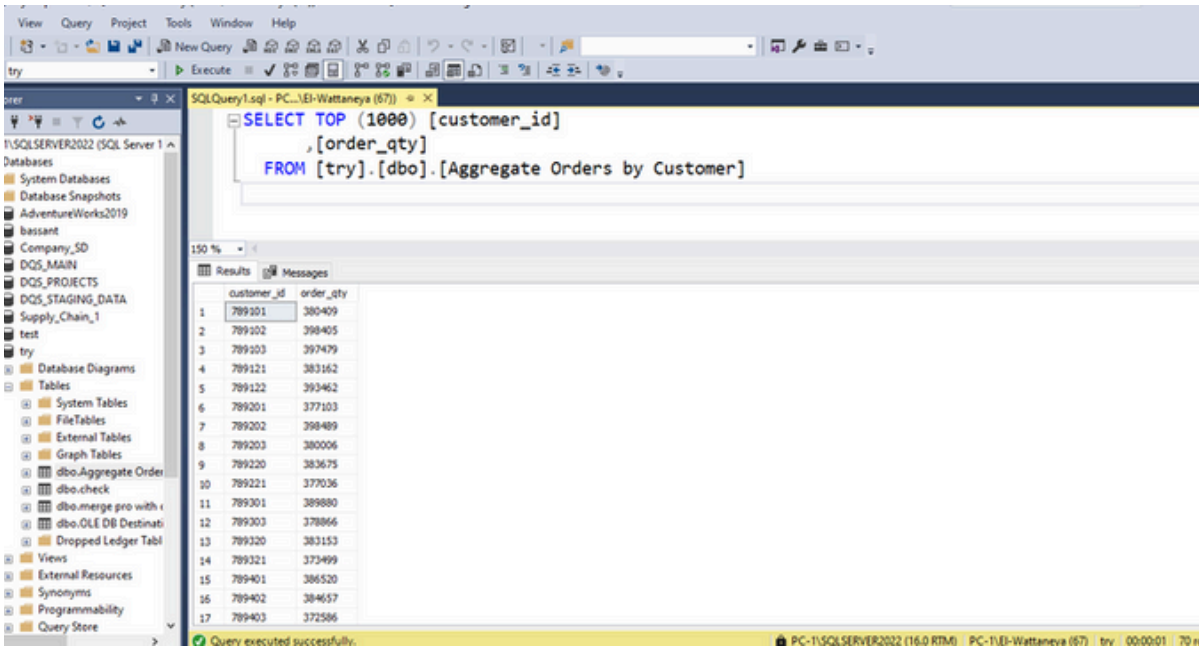
## Data Warehouse Implementation



03

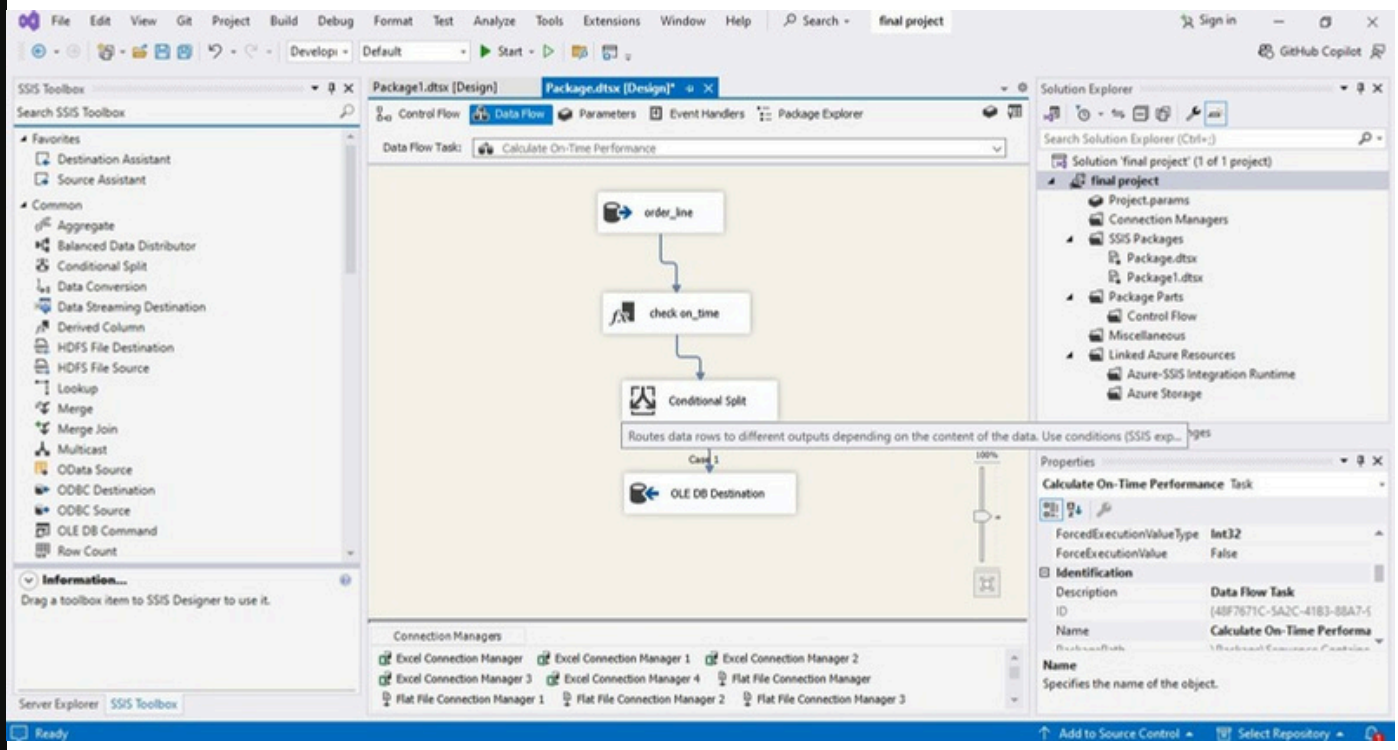
# Data Warehousing and Python Programming

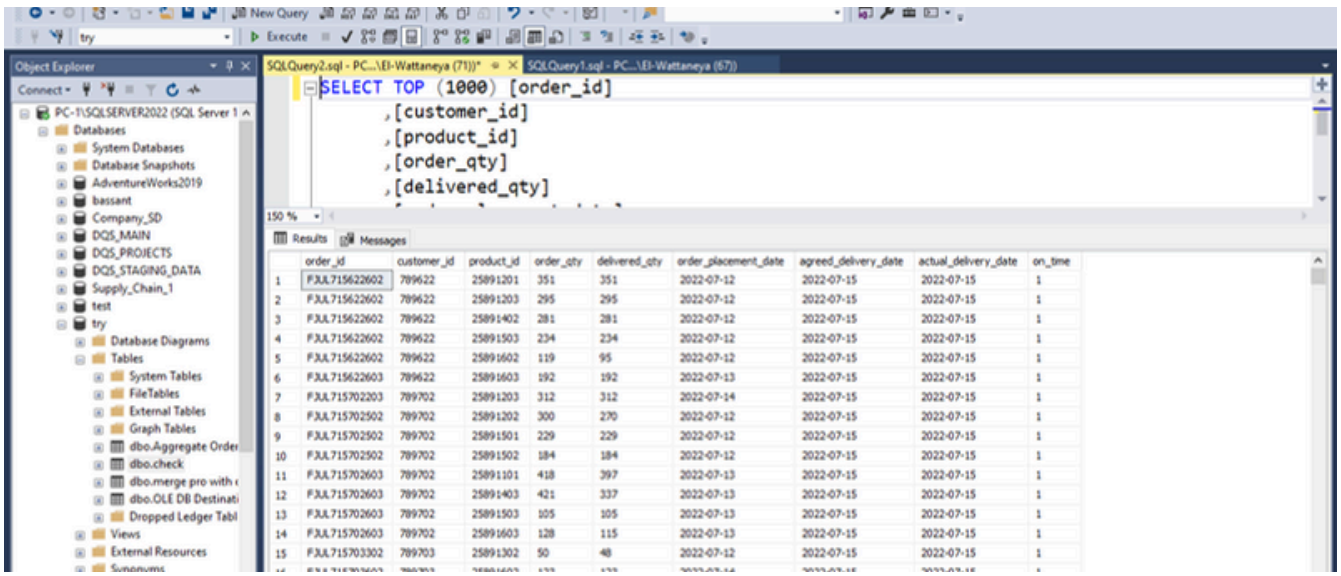
## Data Warehouse Implementation



# Data Warehousing and Python Programming

## Data Warehouse Implementation

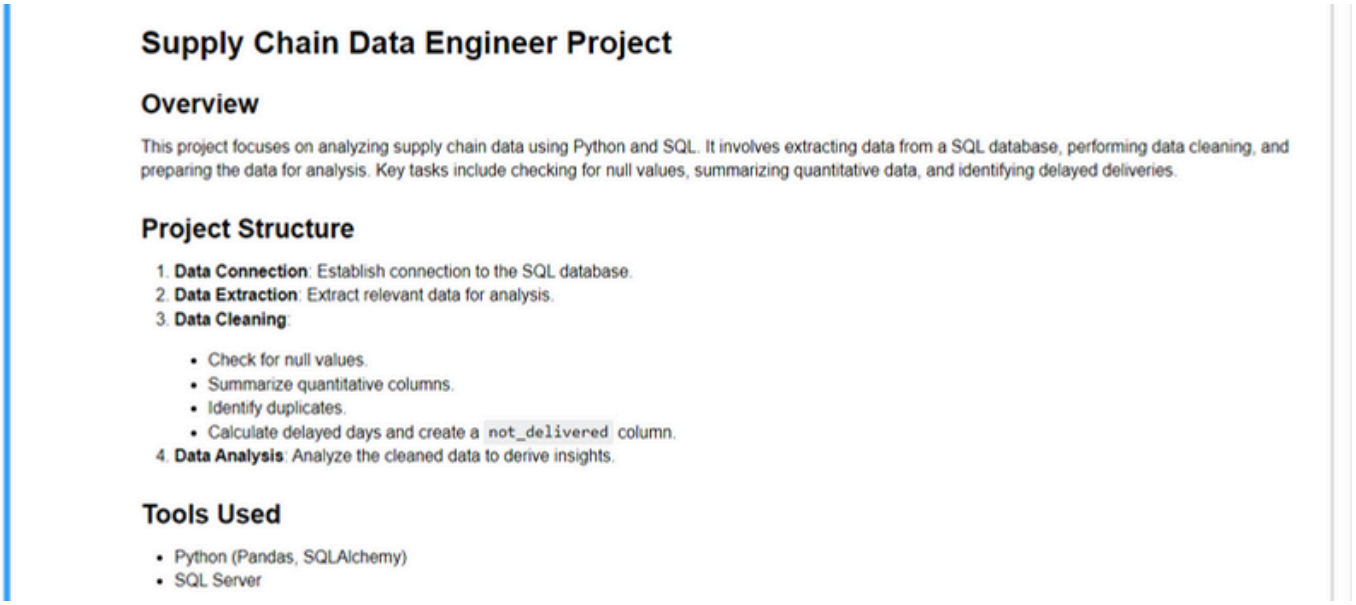


03

# Data Warehousing and Python Programming

## DATA CLEANING



### Supply Chain Data Engineer Project

#### Overview

This project focuses on analyzing supply chain data using Python and SQL. It involves extracting data from a SQL database, performing data cleaning, and preparing the data for analysis. Key tasks include checking for null values, summarizing quantitative data, and identifying delayed deliveries.

#### Project Structure

1. **Data Connection**: Establish connection to the SQL database.
2. **Data Extraction**: Extract relevant data for analysis.
3. **Data Cleaning**:
   - Check for null values.
   - Summarize quantitative columns.
   - Identify duplicates.
   - Calculate delayed days and create a `not_delivered` column.
4. **Data Analysis**: Analyze the cleaned data to derive insights.

#### Tools Used

- Python (Pandas, SQLAlchemy)
- SQL Server

# Data Warehousing and Python Programming

## DATA CLEANING

### 1. Data Connection

In [2]:
```
# Import necessary libraries
import pandas as pd
import pyodbc as odbc
```

In [3]:
```
# Establish connection to the SQL database
sql_conn = odbc.connect('DRIVER={ODBC Driver 17 for SQL Server};'
                        'SERVER=DESKTOP-3B6G2FC\MSSQLSERVER01;'
                        'DATABASE=supply_chain_1;'
                        'UID=Khalid;'
                        'PWD=246248246;')
```

# Data Warehousing and Python Programming

## DATA CLEANING

### Load data from the SQL database into a DataFrame

In [5]:
```
query = "SELECT * FROM fact_order_lines"
df = pd.read_sql(query, sql_conn)

df
```

```
C:\Users\PC\AppData\Local\Temp\ipykernel_26344\2050240598.py:2: UserWarning: pandas only supports SQLAlchemy connectable (engin
e/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using S
QLAlchemy.
  df = pd.read_sql(query, sql_conn)
```

Out[5]:

| | order_id | customer_id | product_id | order_qty | delivered_qty | order_placement_date | agreed_delivery_date | actual_delivery_date |
|---|---|---|---|---|---|---|---|---|
| 0 | FAP410101302 | 789101 | 25891103 | 493 | 493 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 1 | FAP410101302 | 789101 | 25891203 | 374 | 374 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 2 | FAP410101302 | 789101 | 25891302 | 46 | 44 | 2022-04-08 | 2022-04-10 | 2022-04-10 |
| 3 | FAP410101402 | 789101 | 25891101 | 311 | 311 | 2022-04-07 | 2022-04-10 | 2022-04-10 |
| 4 | FAP410101402 | 789101 | 25891201 | 442 | 442 | 2022-04-07 | 2022-04-10 | 2022-04-10 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 57091 | FMY59903502 | 789903 | 25891502 | 105 | 100 | 2022-05-07 | 2022-05-09 | 2022-05-09 |
| 57092 | FMY59903503 | 789903 | 25891102 | 327 | 294 | 2022-05-08 | 2022-05-09 | 2022-05-09 |
| 57093 | FMY59903503 | 789903 | 25891503 | 127 | 121 | 2022-05-08 | 2022-05-09 | 2022-05-09 |
| 57094 | FMY59903601 | 789903 | 25891601 | 91 | 86 | 2022-05-08 | 2022-05-09 | 2022-05-12 |

# Data Warehousing and Python Programming

## DATA CLEANING

### Check for null values in the DataFrame

In [11]:
```
null_values = df.isnull().sum()
print("Null values in each column:")
print(null_values)
```

```
Null values in each column:
order_id                0
customer_id             0
product_id              0
order_qty               0
delivered_qty           0
order_placement_date    0
agreed_delivery_date    0
actual_delivery_date    0
dtype: int64
```

# Data Warehousing and Python Programming

## DATA CLEANING

### Check for Duplicates

In [12]:
```
duplicate_rows = df.duplicated().sum()
print(f"\nNumber of duplicate rows in the dataset: {duplicate_rows}")


print("\nDuplicate rows:")
print(df[df.duplicated()])
```

```
Number of duplicate rows in the dataset: 0

Duplicate rows:
Empty DataFrame
Columns: [order_id, customer_id, product_id, order_qty, delivered_qty, order_placement_date, agreed_delivery_date, actual_deliv
ery_date]
Index: []
```

# Data Warehousing and Python Programming

## DATA CLEANING

### Add a new column for the number of delayed days

In [15]:
```
df['order_placement_date'] = pd.to_datetime(df['order_placement_date'])
df['agreed_delivery_date'] = pd.to_datetime(df['agreed_delivery_date'])
df['actual_delivery_date'] = pd.to_datetime(df['actual_delivery_date'])


df['delay_days'] = (df['actual_delivery_date'] - df['agreed_delivery_date']).dt.days
```

# Data Warehousing and Python Programming

## DATA CLEANING

In [16]:
```
print("\nSample of the data with delay days:")
print(df.tail())
```

```
Sample of the data with delay days:
          order_id  customer_id  product_id  order_qty  delivered_qty  \
57091  FMY59903502       789903    25891502        105            100
57092  FMY59903503       789903    25891102        327            294
57093  FMY59903503       789903    25891503        127            121
57094  FMY59903601       789903    25891601         91             86
57095  FMY59903603       789903    25891603         96             91

      order_placement_date agreed_delivery_date actual_delivery_date  \
57091           2022-05-07           2022-05-09           2022-05-09
57092           2022-05-08           2022-05-09           2022-05-09
57093           2022-05-08           2022-05-09           2022-05-09
57094           2022-05-08           2022-05-09           2022-05-12
57095           2022-05-06           2022-05-09           2022-05-10

       delay_days
57091           0
57092           0
57093           0
57094           3
57095           1
```

# Data Warehousing and Python Programming

## DATA CLEANING

**Calculate not delivered quantity by subtracting delivered_qty from order_qty**

```
In [22]: df['not_delivered'] = (df['order_qty'] - df['delivered_qty']).astype(int)


         # Show specific columns (order_id, customer_id, product_id, delay_days, not_delivered)
         print("\nSample of selected columns with delay days and not delivered quantity:")
         print(df[['order_id','product_id', 'order_qty', 'delivered_qty', 'not_delivered']].head())
```

```
Sample of selected columns with delay days and not delivered quantity:
      order_id  product_id  order_qty  delivered_qty  not_delivered
0  FAP410101302    25891103        493            493              0
1  FAP410101302    25891203        374            374              0
2  FAP410101302    25891302         46             44              2
3  FAP410101402    25891101        311            311              0
4  FAP410101402    25891201        442            442              0
```

# Data Warehousing and Python Programming

## DATA CLEANING

In [23]:
```
# Describe the 'not_delivered' column to get statistical summary
not_delivered_summary = df['not_delivered'].describe()


# Print the summary
print("\nStatistical summary of not delivered quantities:")
print(not_delivered_summary)
```

```
Statistical summary of not delivered quantities:
count    57096.000000
mean         8.017707
std         16.484549
min          0.000000
25%          0.000000
50%          0.000000
75%          9.000000
max        100.000000
Name: not_delivered, dtype: float64
```

# Data Warehousing and Python Programming

## MODEL

jupyter supp_xgboost_final_demo Last Checkpoint: 16/10/2024 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [14]:
```
# Predict and evaluate delay model
best_model_delay = grid_search_delay.best_estimator_
y_pred_delay = best_model_delay.predict(X_test)

print("\nDelay Prediction Report:")
print(classification_report(y_test_delay, y_pred_delay))

# Predict and evaluate full delivery model
best_model_full = grid_search_full.best_estimator_
y_pred_full = best_model_full.predict(X_test_full)

print("\nFull Delivery Prediction Report:")
print(classification_report(y_test_full, y_pred_full))

# Predict and evaluate on-time in full model
best_model_on_time = grid_search_on_time.best_estimator_
y_pred_on_time = best_model_on_time.predict(X_test_on_time)

print("\nOn-Time In Full Prediction Report:")
print(classification_report(y_test_on_time, y_pred_on_time))
```

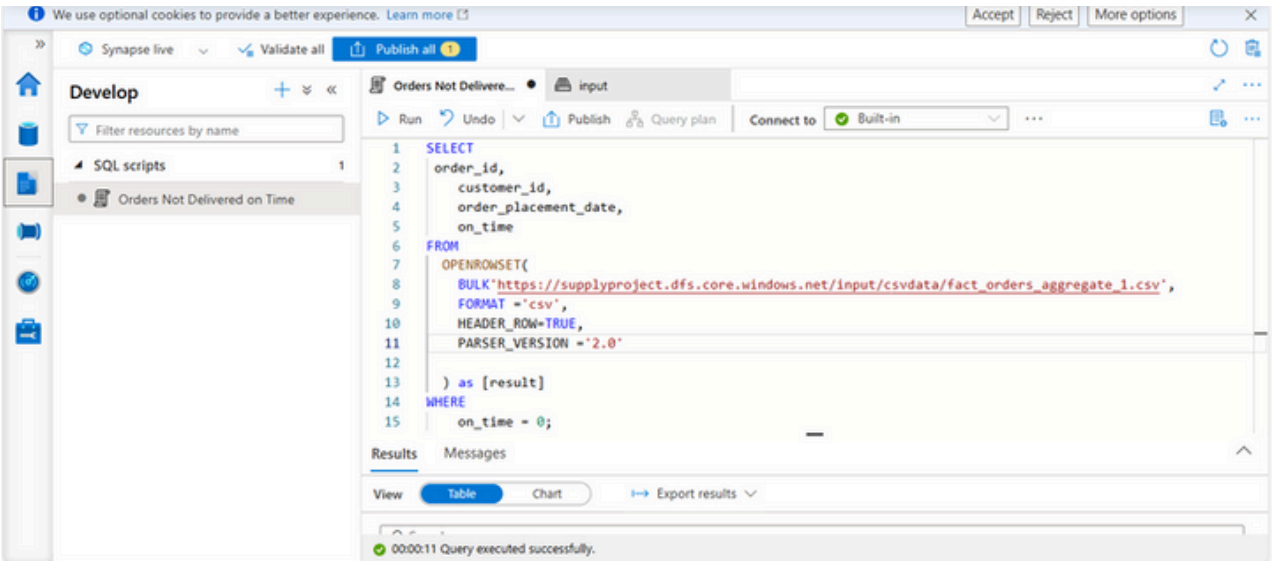
```
Delay Prediction Report:
              precision    recall  f1-score   support

           0       0.84      0.92      0.88      8098
           1       0.74      0.58      0.65      3322

    accuracy                           0.82     11420
   macro avg       0.79      0.75      0.76     11420
weighted avg       0.81      0.82      0.81     11420
```

# Azure Integration

## We use SQL Scripts in Synapse Analytics for Data Analysis:

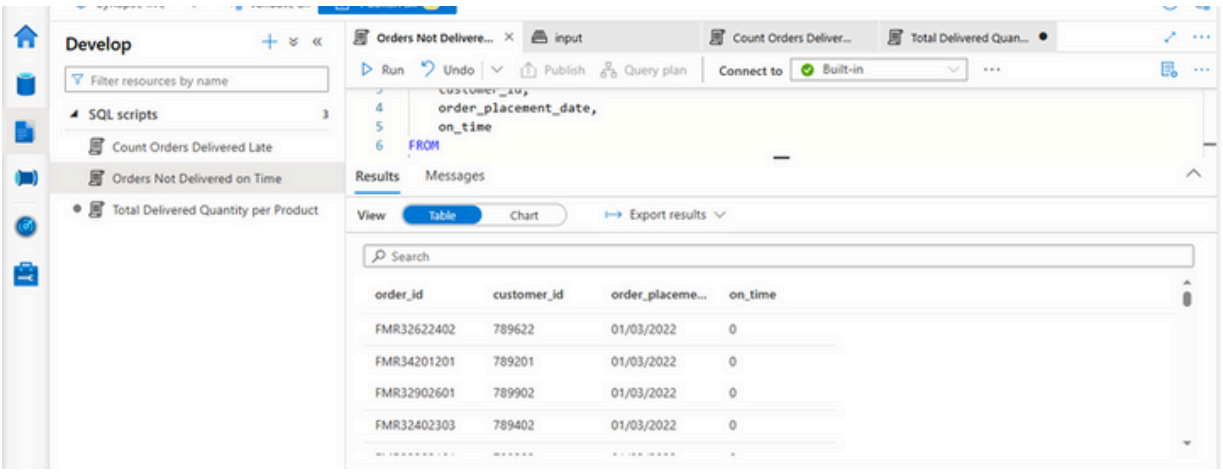
**retrieve information from data by writing queries to fetch, filter, and analyze data in csv files.**

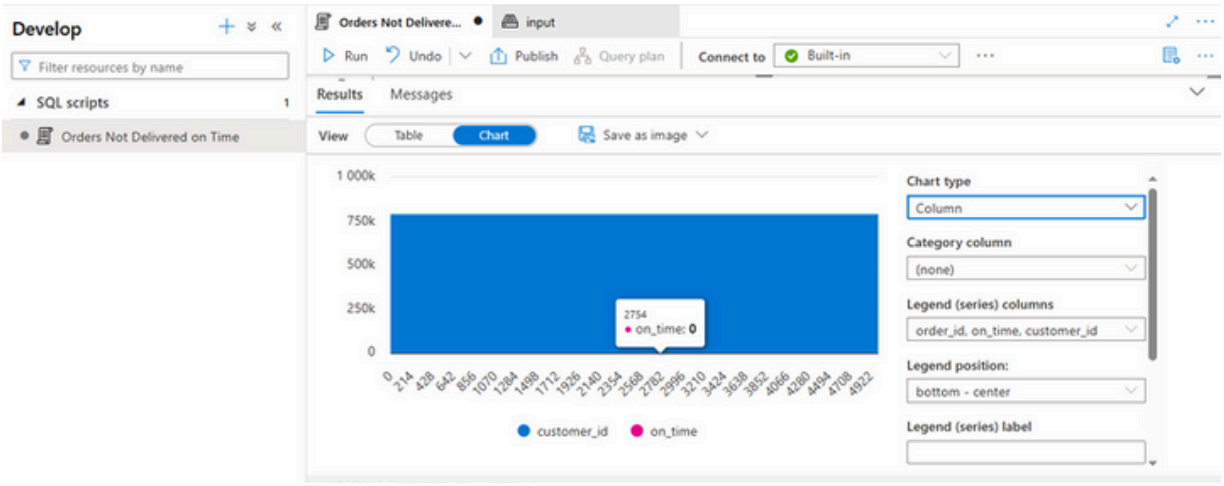
### 1-Orders not delivered on time



# Azure Integration

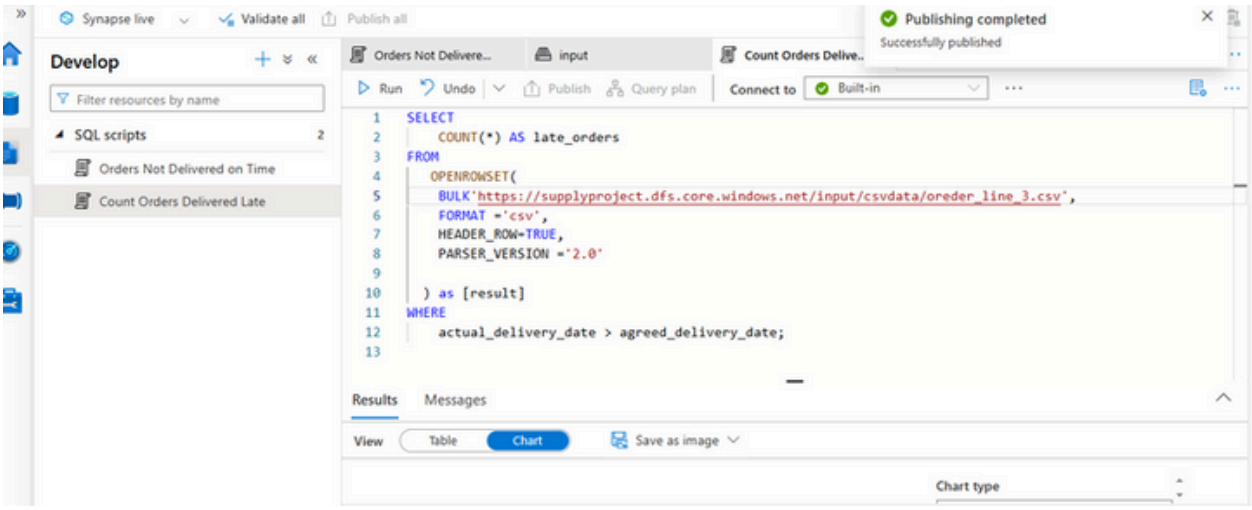
## Result in table and chart

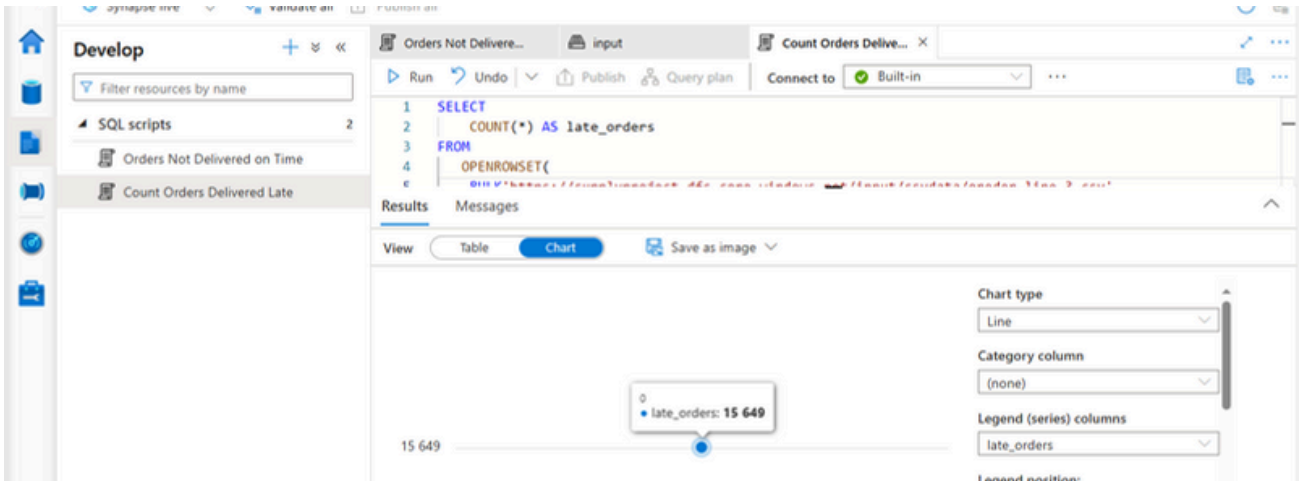


# Azure Integration

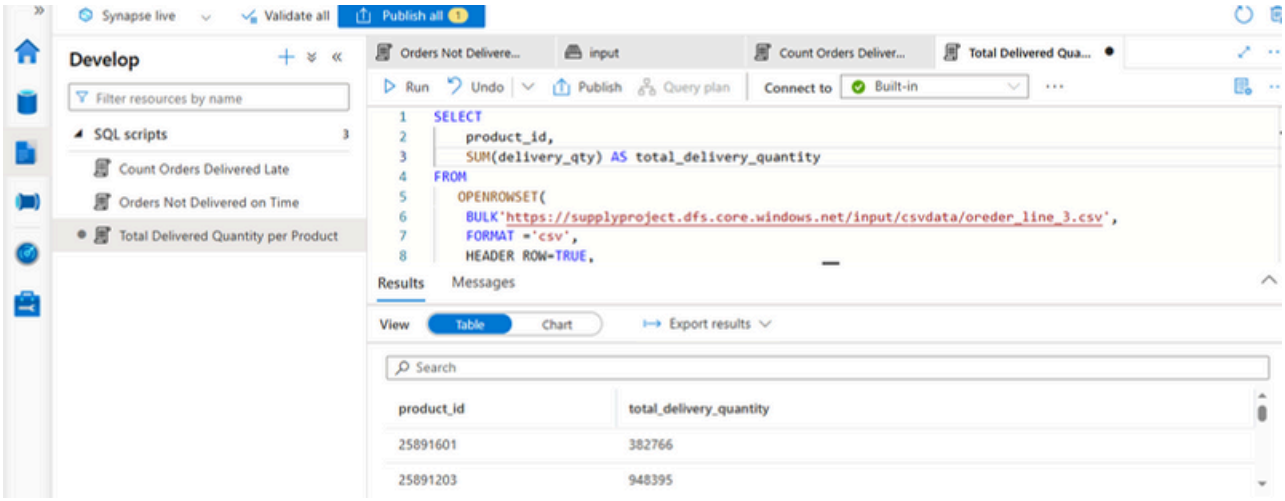
## 2-Count orders delivered late

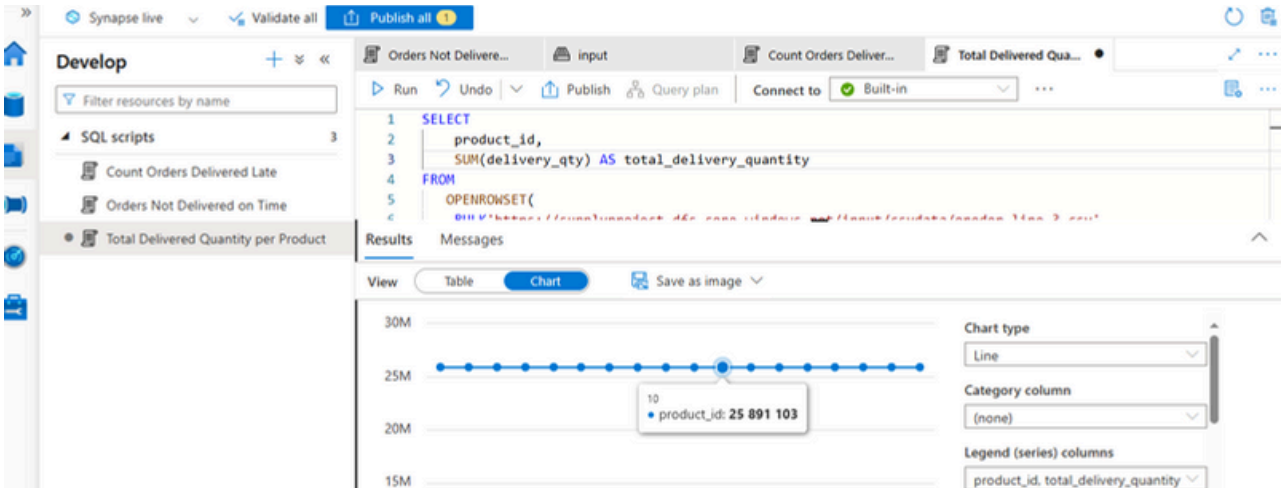


# Azure Integration

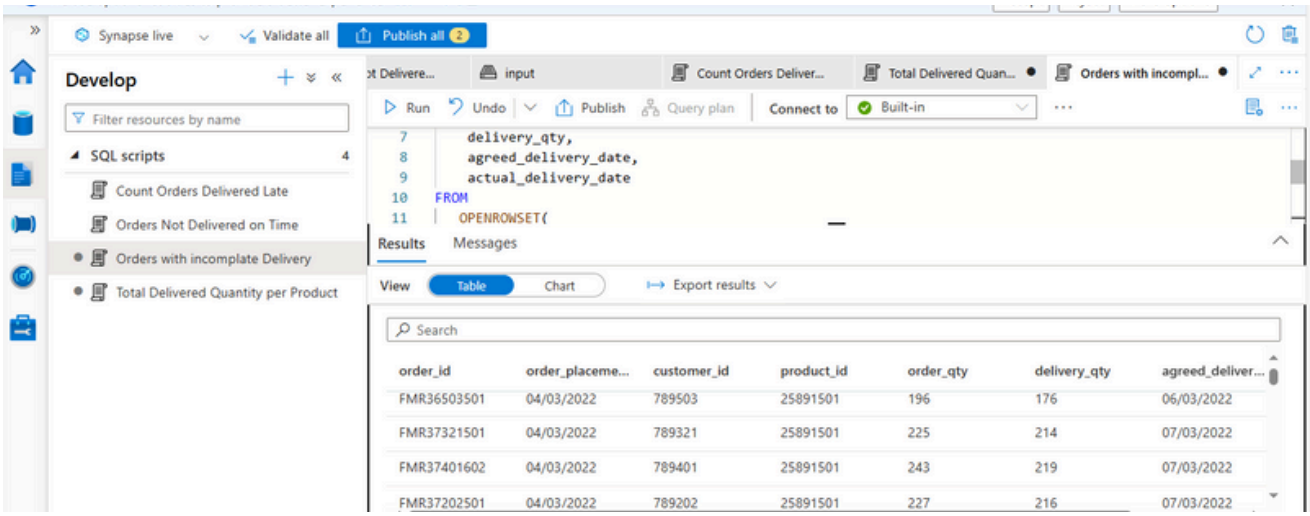
## 3-Total delivered quantity per product

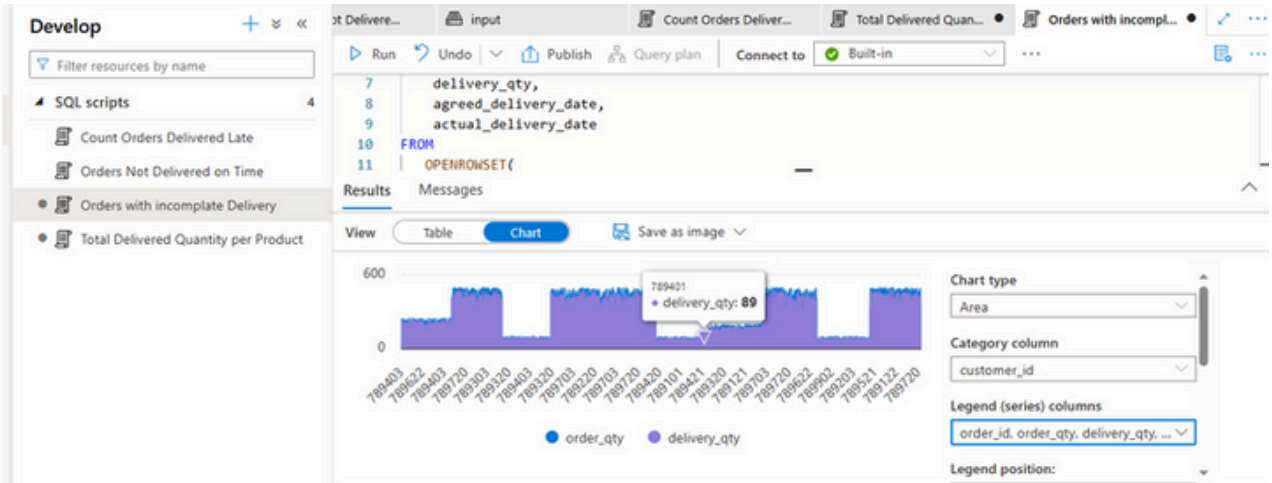


# Azure Integration

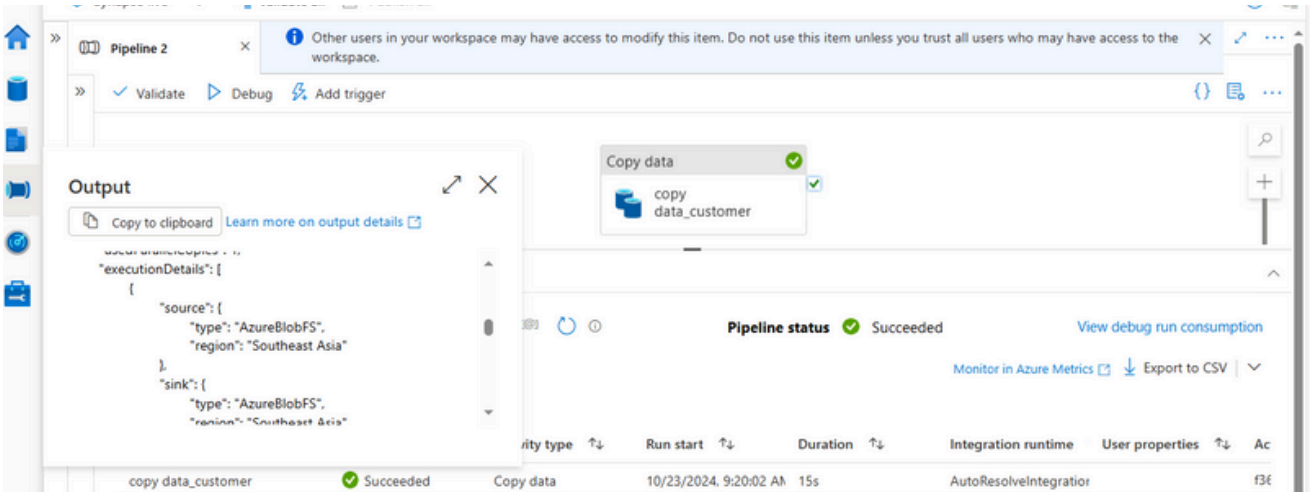
## 4-Orders with incomplate delivery

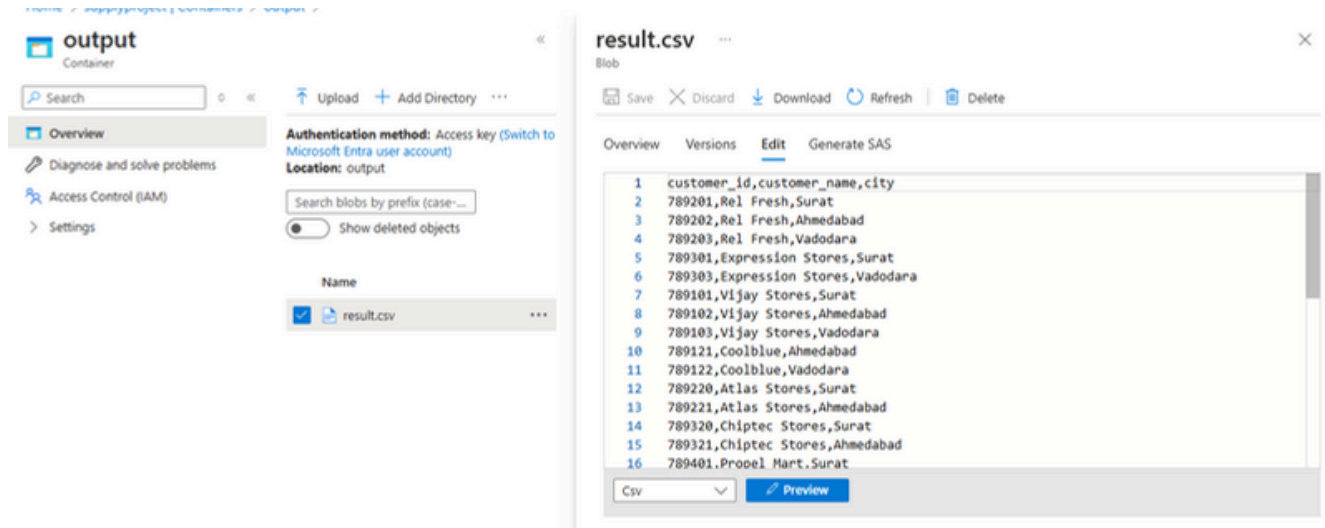


# Azure Integration

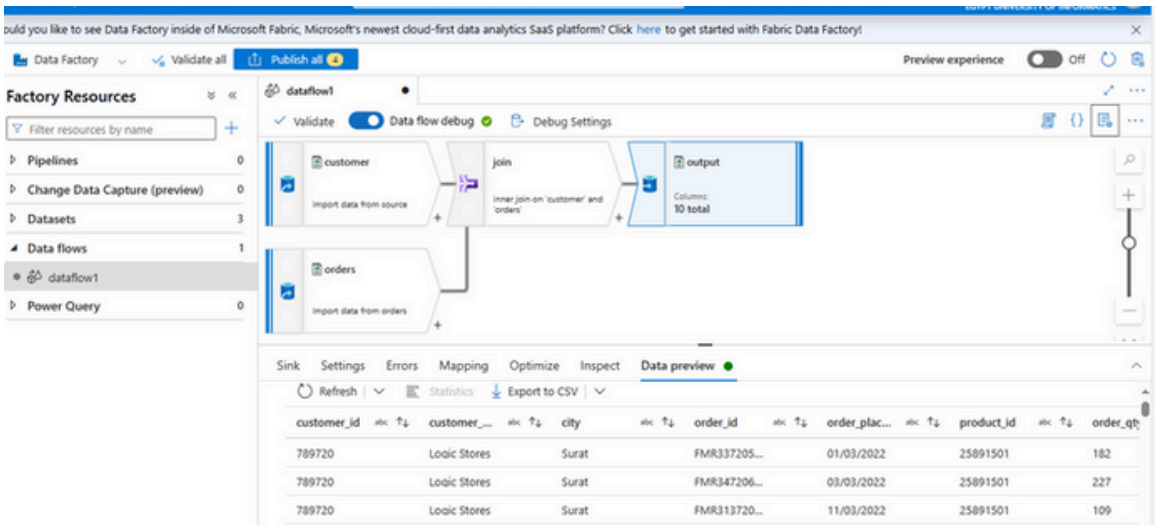
## Using pipeline activities:
## 1-Copy Data

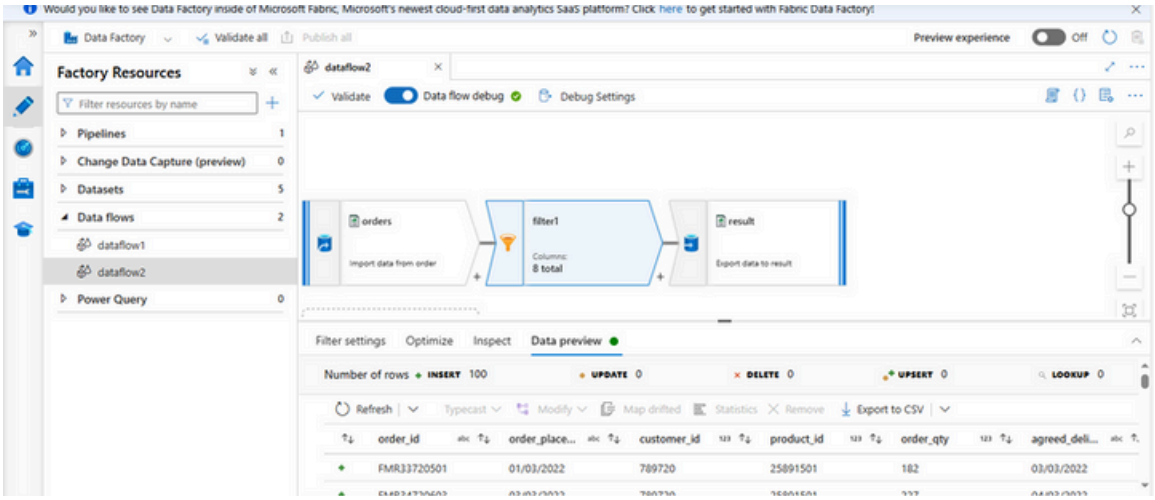


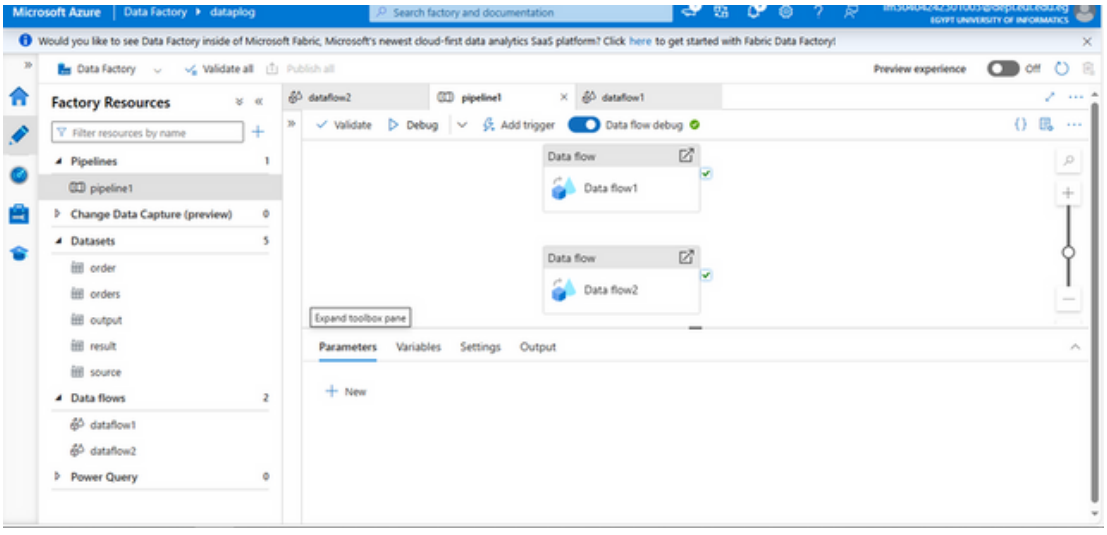



# Azure Integration

## Using Data Factory:
## 1-Data Flows



# THANK YOU!!

instructor **Hanaa gharib**

ATLIQ
GRANDS

# ATLIQ
## GRANDS