



ATLIQ
MARTS



OPTIMIZING SUPPLY CHAIN PERFORMANCE FOR **ATLIQ MART**

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



1- خالد شهاب

2- بسنت ياسر

3- لبنى عيد

4- أحمد علاء

5- محمد صبري

,

Prepared for

**ATLIQ MART
STAKEHOLDERS**

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.**



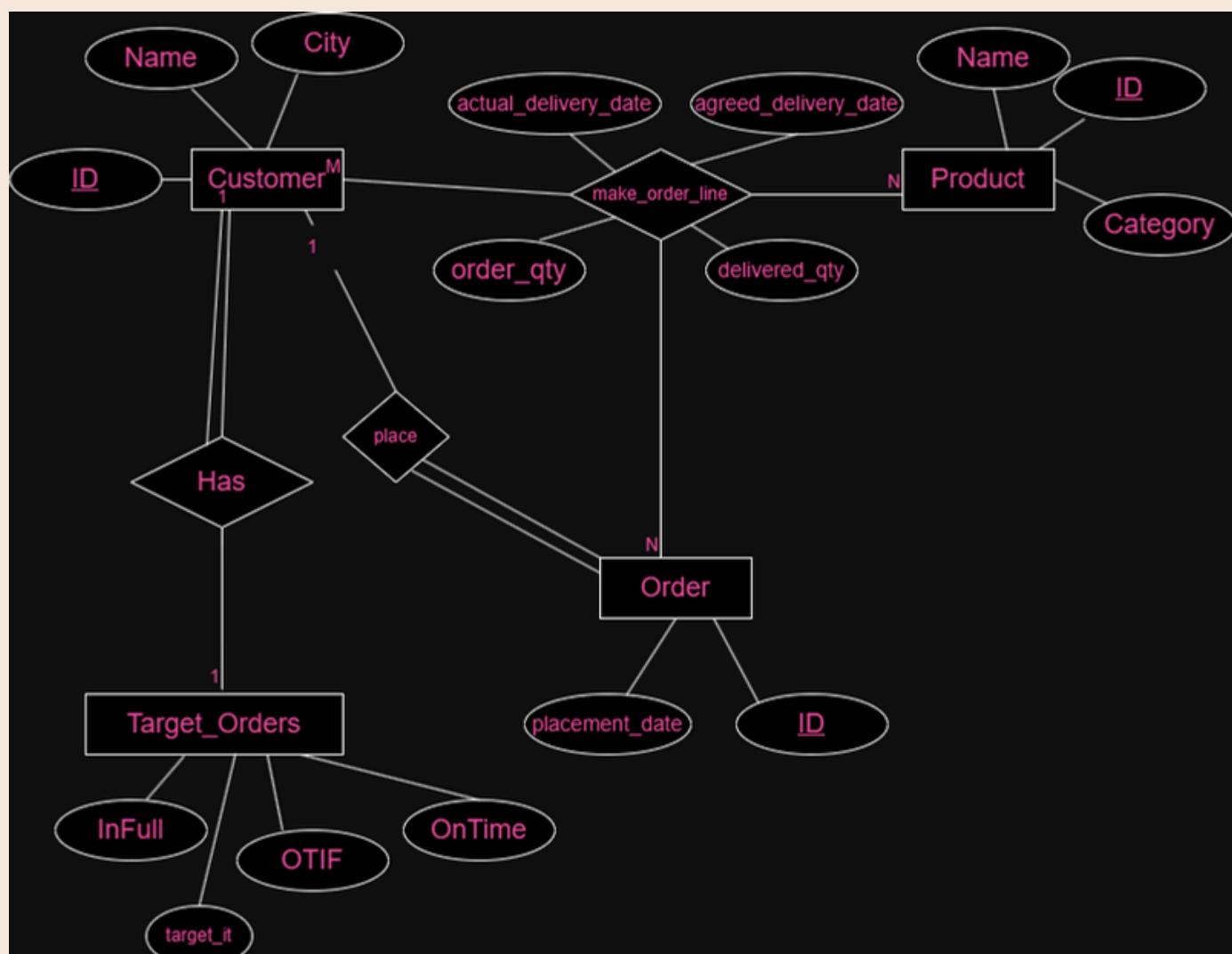
01 - Entity Relationship Diagram (ERD) and Data Mapping

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



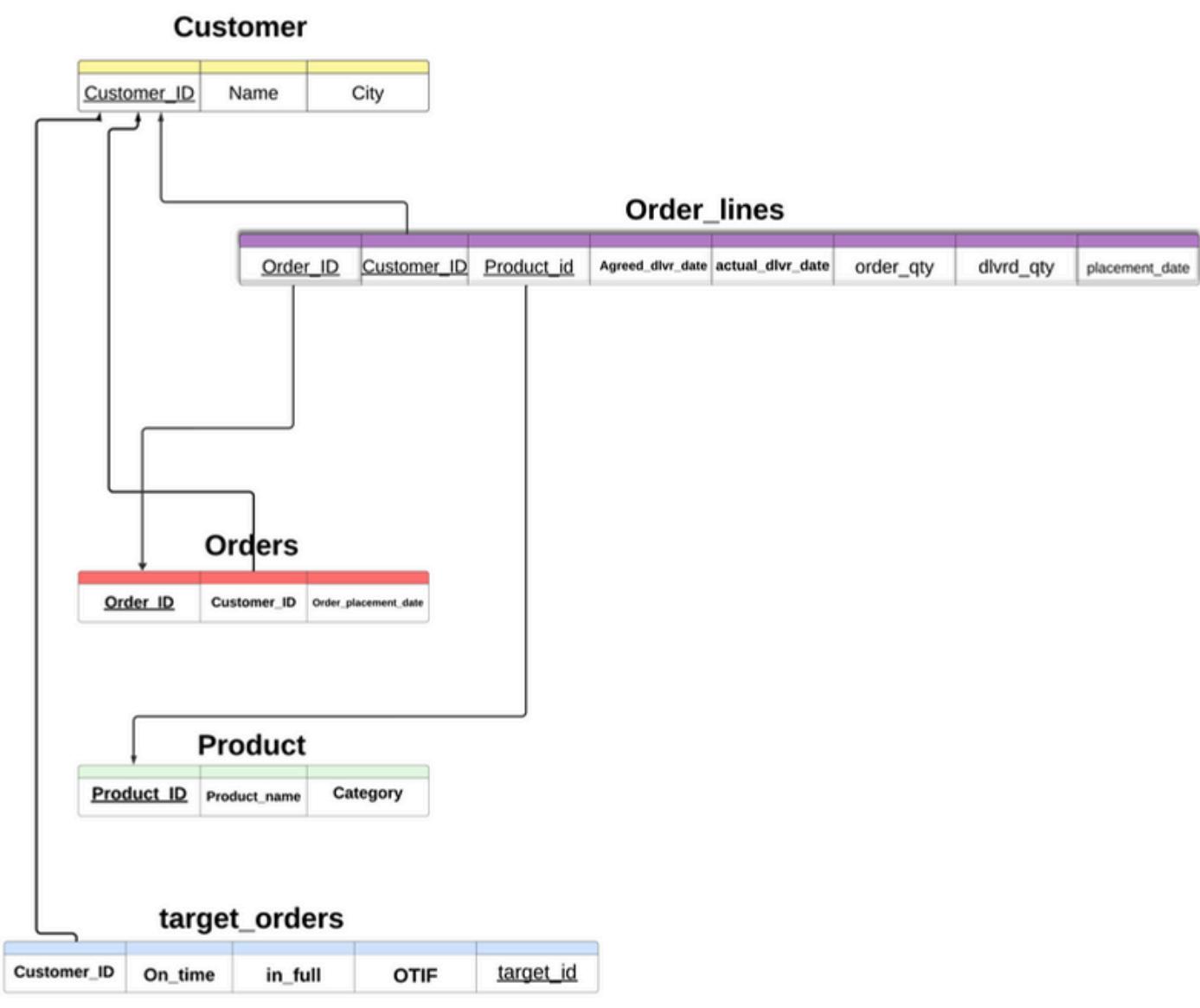
01 - Entity Relationship Diagram (ERD) and Data Mapping

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))  ↗ X
- 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:

```
-- 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:

```
-- 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',
    FIRSTROW = 2
);
```

10% < Connected. (1/1) | D

02 - Data Management and SQL Database Setup

Data extraction from CSV files.

```
-- Bulk insert data into the fact_orders_aggregate table
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

```
-- 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

```
-- 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

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

```
-- 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

```
-- 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);
```

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

-- Convert order_placement_date to DATE format and store it in the new column
UPDATE fact_orders_aggregate
SET order_placement_dateConverted = 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_dateConverted 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_dateConverted', '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 %

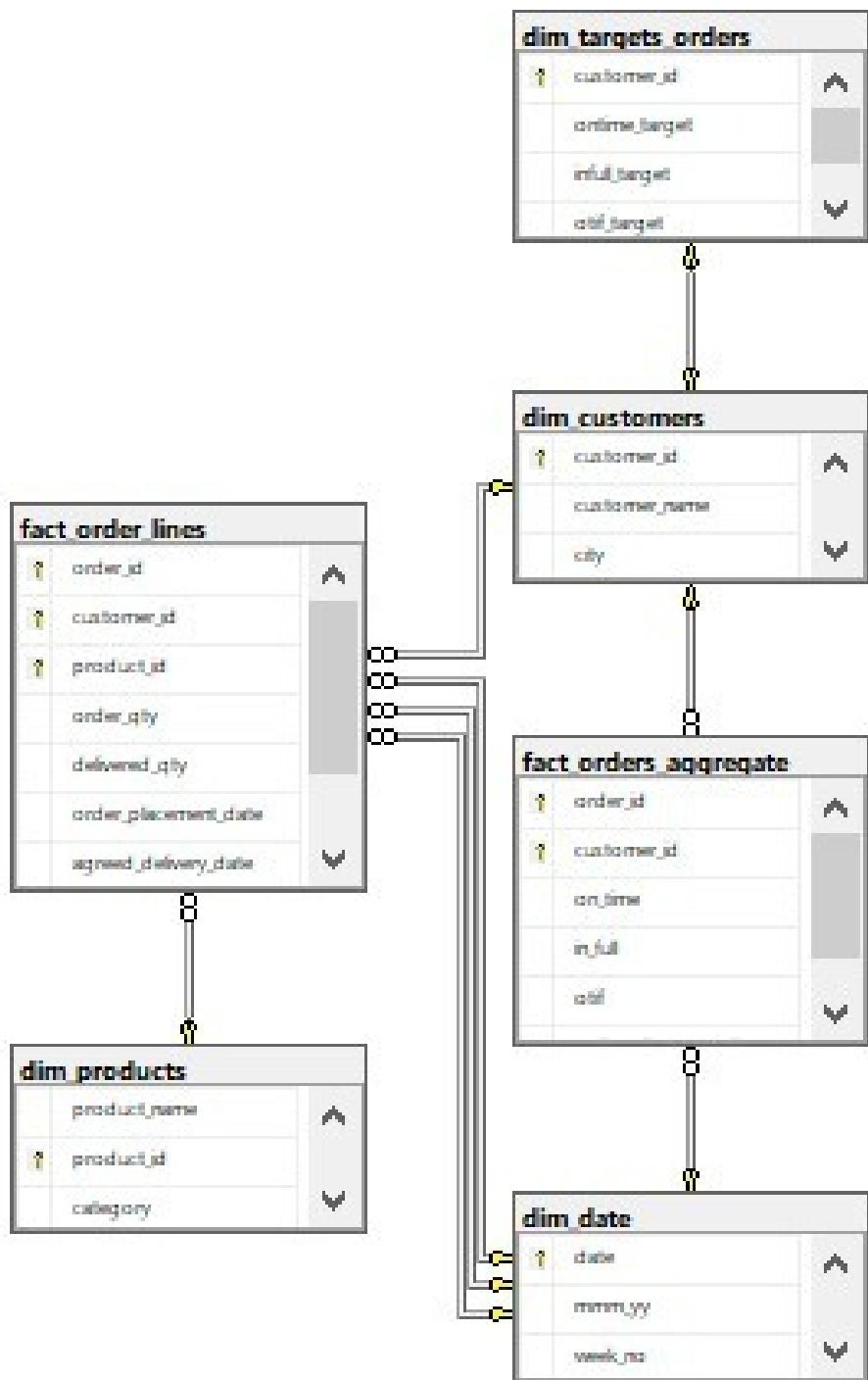
Results Messages

	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
17	FAP410102602	789102	25891202	20	20	2022-04-09	2022-04-10	2022-04-10

Query executed successfully.

DESKTOP-3B6G2

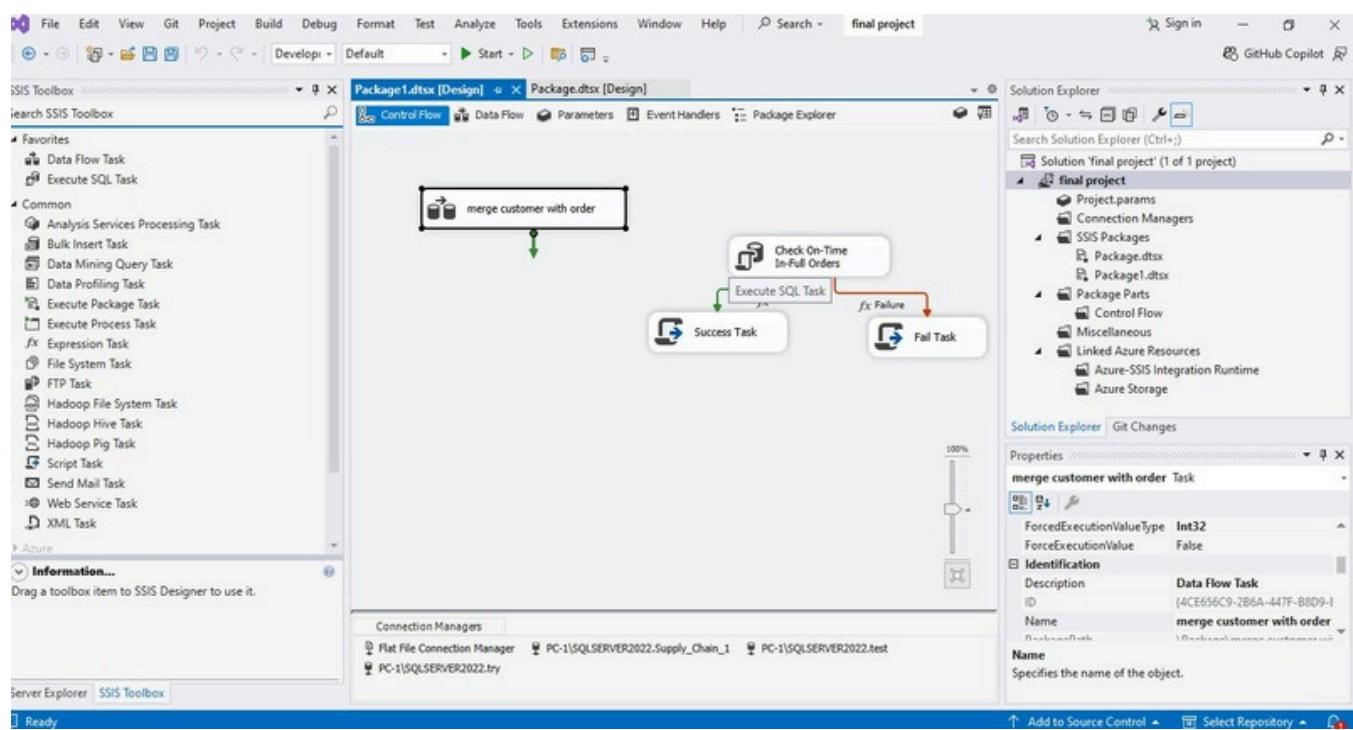
02 - Data Management and SQL Database Setup



Data Warehousing and Python Programming

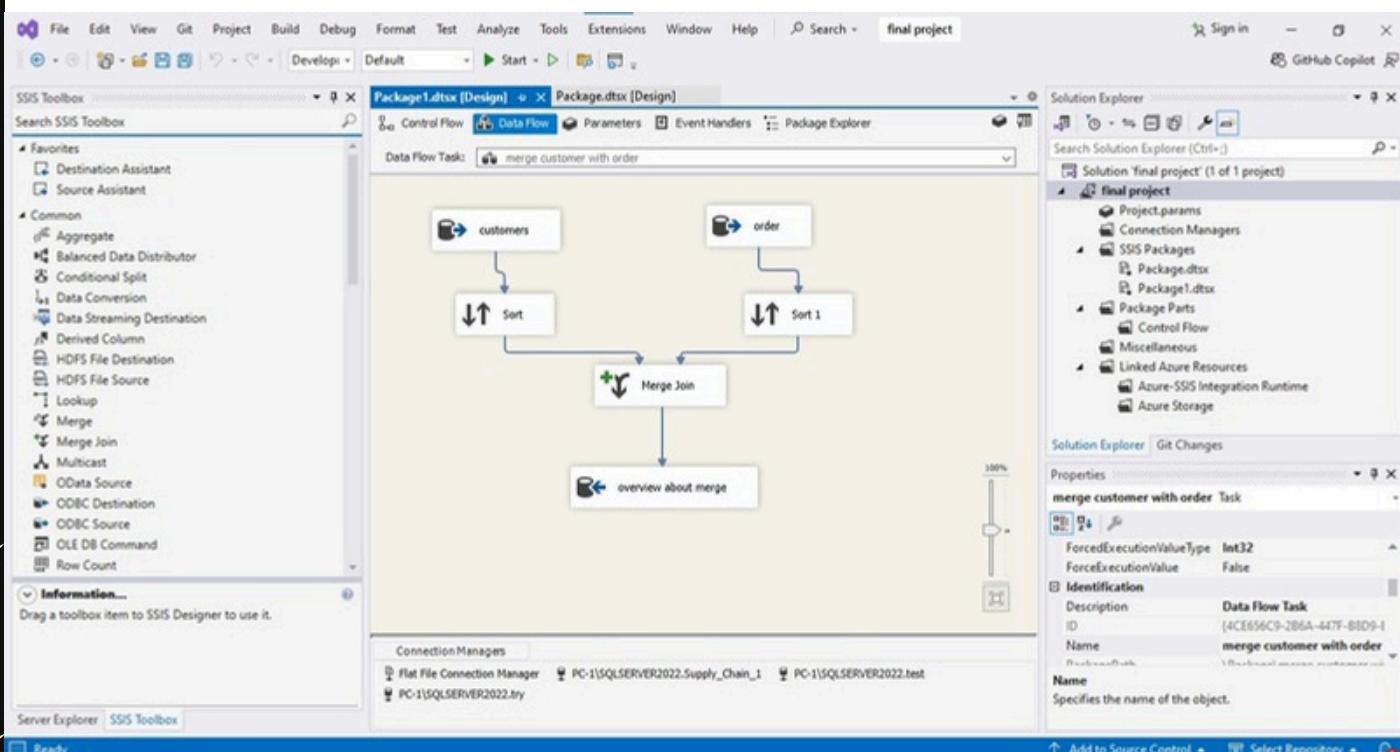
Data Warehouse Implementation

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



Data Warehousing and Python Programming

Data Warehouse Implementation



Data Warehousing and Python Programming

Data Warehouse Implementation



SQlQuery4.sql - PC-1\SQLSERVER2022 (PC-1\EI-Wattaneya (55)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Execute

try

Object Explorer

Connect

PC-1\SQLSERVER2022 (SQL Server 1)

- Databases
 - System Databases
 - Database Snapshots
 - AdventureWorks2019
 - bassant
 - Company_SD
 - DQS_MAIN
 - DQS_PROJECTS
 - DQS_STAGING_DATA
 - Supply_Chain_1
 - test
 - try
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dboAggregate Order
 - dbo.check
 - dbo.merge pro with i
 - dbo.OLE DB Destinati
 - Dropped Ledger Tabl
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store

SELECT TOP (1000) [customer_id]
, [customer_name]
, [order_id]
[order_qty]

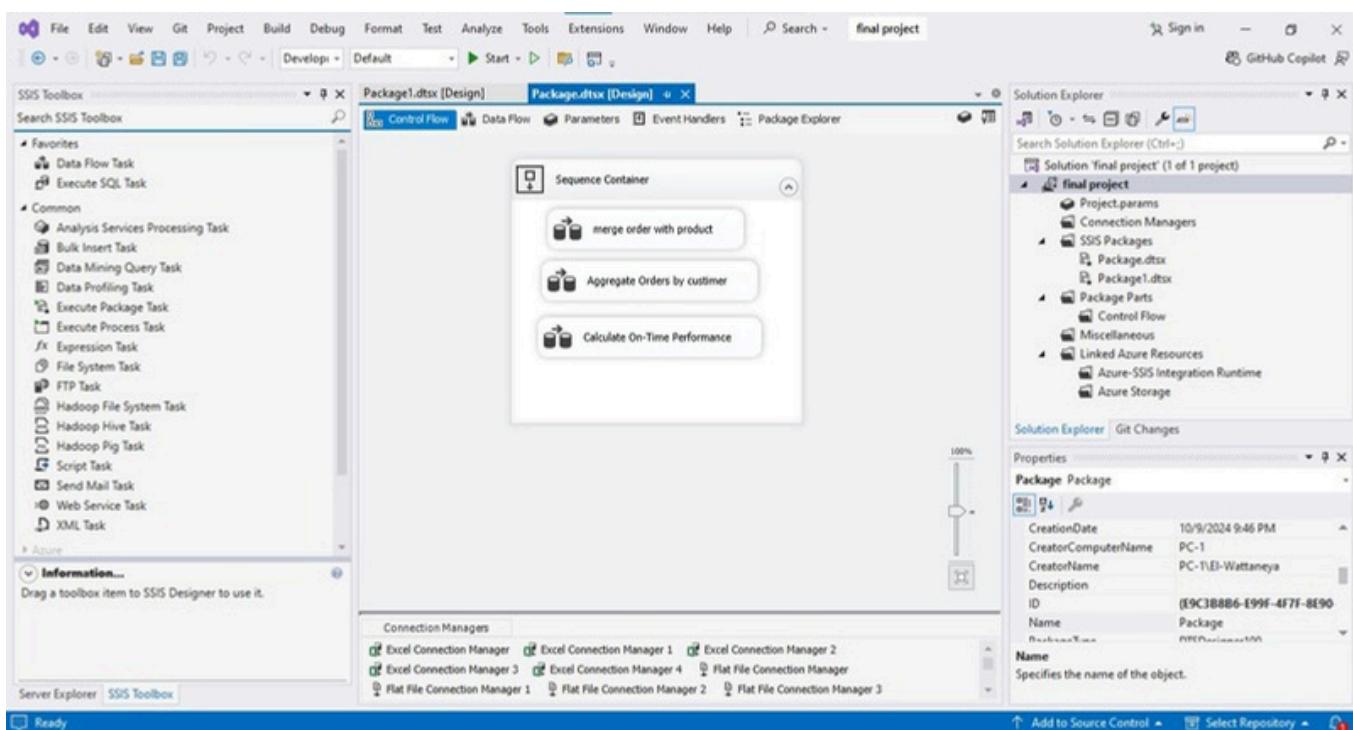
	customer_id	customer_name	order_id	order_qty	delivered_qty
1	789101	Vjay Stores	FAP430101302	493	493
2	789101	Vjay Stores	FAUG820101402	350	350
3	789101	Vjay Stores	FJUL717101103	404	404
4	789101	Vjay Stores	FMR312101503	147	147
5	789101	Vjay Stores	FAP430101503	198	198
6	789101	Vjay Stores	FAUG815101403	468	468
7	789101	Vjay Stores	FJUL722101602	23	23
8	789101	Vjay Stores	FMR311101501	219	219
9	789101	Vjay Stores	FAP430101503	169	169
10	789101	Vjay Stores	FAUG815101401	404	404
11	789101	Vjay Stores	FJUL722101602	423	423
12	789101	Vjay Stores	FMR311101501	303	288
13	789101	Vjay Stores	FAP430101502	207	207
14	789101	Vjay Stores	FAUG815101303	45	45
15	789101	Vjay Stores	FJUL722101602	302	302
16	789101	Vjay Stores	FJUL628101403	454	454
17	789101	Vjay Stores	FAP430101501	186	186
18	789101	Vjay Stores	FAUG815101303	386	347
19	789101	Vjay Stores	FJUL722101503	191	191

Query executed successfully.

PC-1\SQLSERVER2022 (16.0 RTM) PC-1\EI-Wattaneya (55) try 00:00:00 1,000 rows

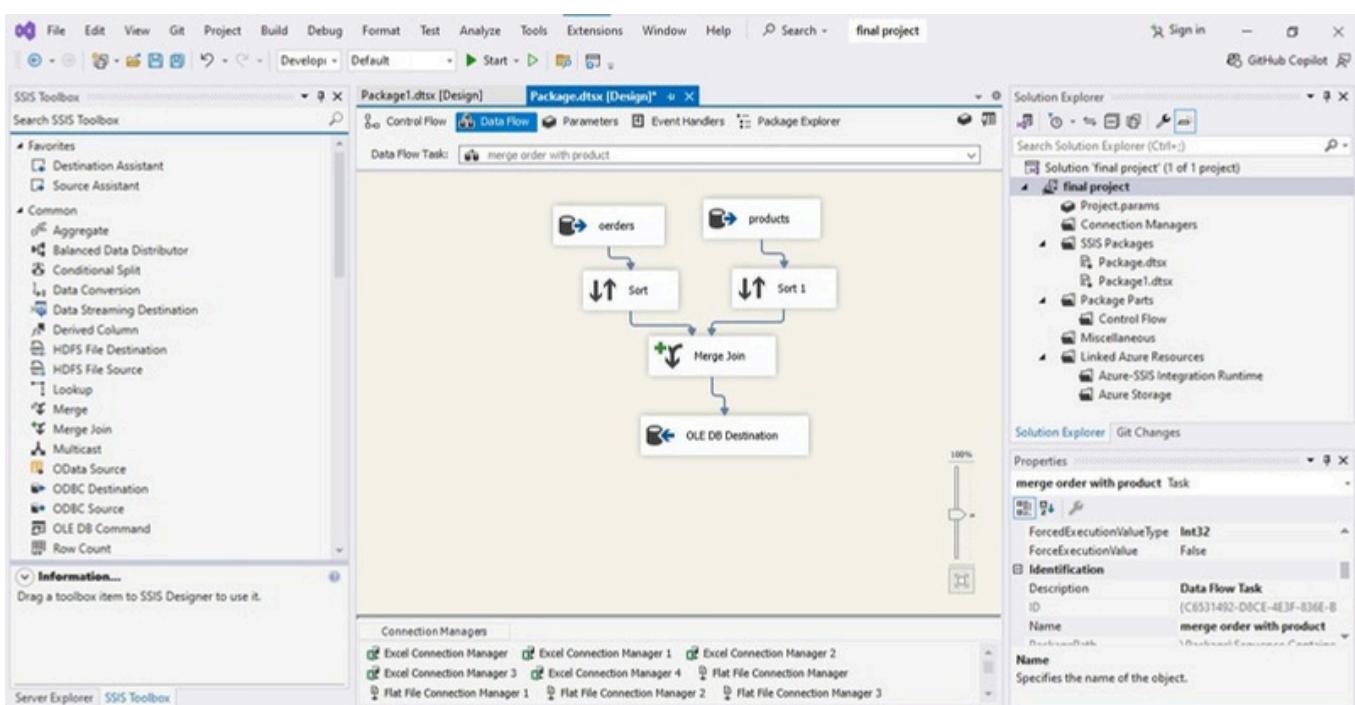
Data Warehousing and Python Programming

Data Warehouse Implementation



Data Warehousing and Python Programming

Data Warehouse Implementation



Data Warehousing and Python Programming

Data Warehouse Implementation



SQLQuery3.sql - PC-1\SQLSERVER2022.trv (PC-1\EI-Wattaneya (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Execute

Object Explorer

try

Object Explorer

File Edit View Query Project Tools Window Help

New Query Execute

Object Explorer

try

Databases

- System Databases
- AdventureWorks2019
- bassant
- Company_SD
- DQS_MAIN
- DQS_PROJECTS
- DQS_STAGING_DATA
- Supply_Chain_1
- test
- try

Tables

- System Tables
- FileTables
- External Tables
- Graph Tables
- dbo.Aggregate Order
- dbo.check
- dbo.merge pro with c
- dbo.OLE DB Destinati
- Dropped Ledger Tabl

Views

External Resources

Synonyms

Programmability

Query Store

SQLQuery3.sql - PC-1\EI-Wattaneya (53))

SQLQuery2.sql - PC-1\EI-Wattaneya (71))

SQLQuery1.sql - PC-1\EI-Wattaneya (67))

```
SELECT TOP (1000) [product_id]
      ,[order_id]
      ,[customer_id]
```

Results Messages

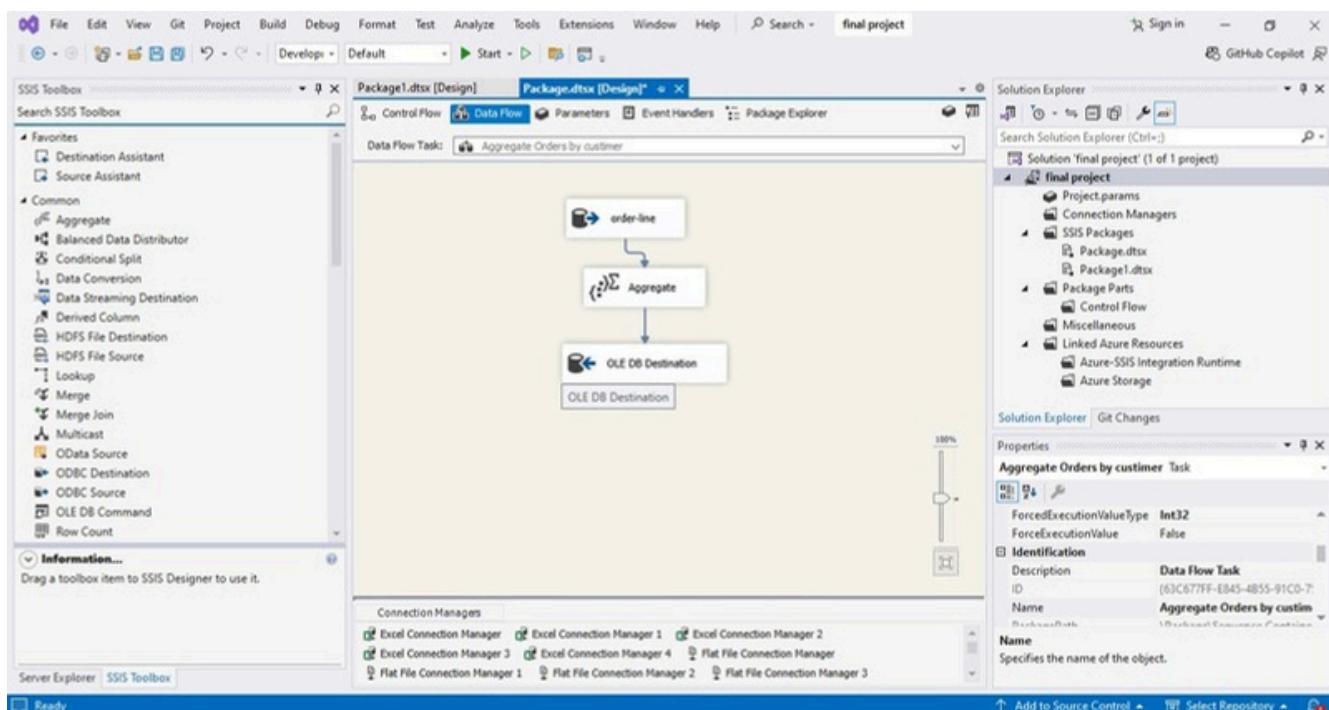
product_id	order_id	customer_id	order_qty	delivered_qty	order_placement_date	product_name	category
1	FAP429903601	789903	371	297	2022-04-28	AM Milk 500	Dairy
2	FAPG83042101	789421	379	341	2022-08-28	AM Milk 500	Dairy
3	FJUN611721403	789721	335	318	2022-06-10	AM Milk 500	Dairy
4	FJUN617122101	789122	363	327	2022-06-15	AM Milk 500	Dairy
5	FAP43702401	789702	329	329	2022-03-31	AM Milk 500	Dairy
6	FAUG824703101	789703	408	408	2022-08-23	AM Milk 500	Dairy
7	FJUL722603502	789603	362	344	2022-07-21	AM Milk 500	Dairy
8	FJUN69621102	789621	491	442	2022-06-08	AM Milk 500	Dairy
9	FAP43720101	789720	449	427	2022-04-02	AM Milk 500	Dairy
10	FAUG89303601	789303	479	479	2022-08-06	AM Milk 500	Dairy
11	FJUL722621502	789621	318	318	2022-07-19	AM Milk 500	Dairy
12	FJUN62720602	789720	472	472	2022-05-30	AM Milk 500	Dairy
13	FAP414520101	789520	456	456	2022-04-13	AM Milk 500	Dairy
14	FAUG814422101	789422	300	240	2022-08-12	AM Milk 500	Dairy
15	FJUL722622302	789622	352	352	2022-07-19	AM Milk 500	Dairy
16	FJUN623420601	789420	324	324	2022-06-20	AM Milk 500	Dairy
17	FAP43902403	789902	422	422	2022-03-31	AM Milk 500	Dairy
18	FAUG831202503	789202	327	327	2022-08-29	AM Milk 500	Dairy
19	FJUL73621101	789621	301	301	2022-07-02	AM Milk 500	Dairy
20	FJUN617220601	789220	449	449	2022-06-16	AM Milk 500	Dairy

Query executed successfully.

PC-1\SQLSERVER2022 (16.0 RTM) | PC-1\EI-Wattaneya (53) | trv | 00:00:00 | 1.000 rows

Data Warehousing and Python Programming

Data Warehouse Implementation



Data Warehousing and Python Programming

Data Warehouse Implementation



A screenshot of the SQL Server Management Studio (SSMS) interface. The left sidebar shows a tree view of the database structure under 'try'. The main query window contains a T-SQL SELECT statement:

```
SELECT TOP (1000) [customer_id]
      ,[order_qty]
  FROM [try].[dbo].[Aggregate Orders by Customer]
```

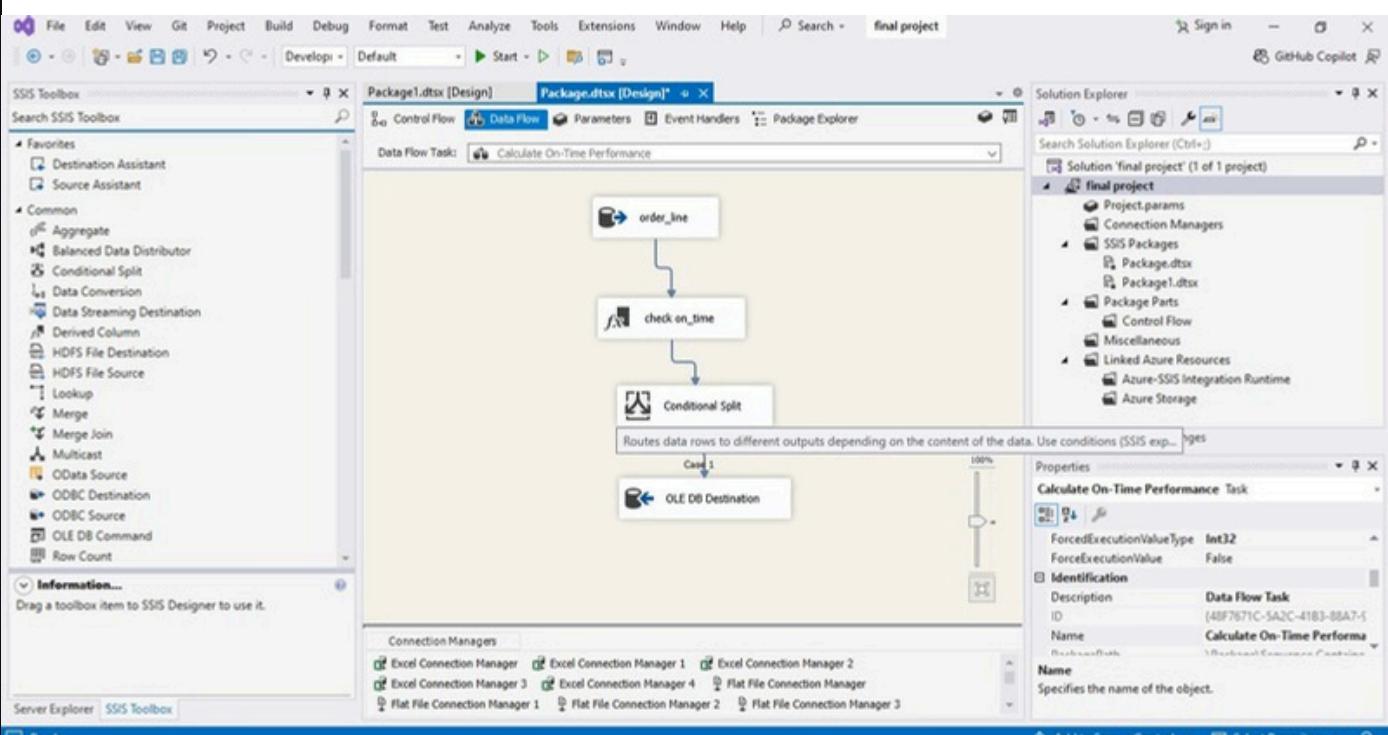
The results pane displays a table with two columns: 'customer_id' and 'order_qty'. The data is as follows:

	customer_id	order_qty
1	709101	380409
2	709102	398405
3	709103	397479
4	709121	383162
5	709122	393462
6	709201	377103
7	709202	398489
8	709203	380006
9	709220	383675
10	709221	377036
11	709301	389880
12	709303	378866
13	709320	383153
14	709321	373499
15	709401	386520
16	709402	384657
17	709403	372586

At the bottom of the results pane, a message states: "Query executed successfully."

Data Warehousing and Python Programming

Data Warehouse Implementation



The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows the database structure, including databases like "AdventureWorks2019", "bassant", "Company_SD", "DQS_MAIN", "DQS_PROJECTS", "DQS_STAGING_DATA", "Supply_Chain_1", and a "try" database. The "Tables" node under "try" is expanded, showing tables such as "dbo.Aggregate Order", "dbo.check", "dbo.merge pro with c", "dbo.OLE DB Destinati", "Dropped Ledger Tabl", "Views", "External Resources", and "System Catalogs".

The central pane displays the results of a SQL query:

```
try
SELECT TOP (1000) [order_id]
,[customer_id]
,[product_id]
,[order_qty]
,[delivered_qty]
```

The results grid shows 15 rows of data:

order_id	customer_id	product_id	order_qty	delivered_qty	order_placement_date	agreed_delivery_date	actual_delivery_date	on_time
FJA715622602	799622	25891201	351	351	2022-07-12	2022-07-15	2022-07-15	1
FJA715622602	799622	25891203	295	295	2022-07-12	2022-07-15	2022-07-15	1
FJA715622602	799622	25891402	281	281	2022-07-12	2022-07-15	2022-07-15	1
FJA715622602	799622	25891503	234	234	2022-07-12	2022-07-15	2022-07-15	1
FJA715622602	799622	25891602	119	95	2022-07-12	2022-07-15	2022-07-15	1
FJA715622603	799622	25891603	192	192	2022-07-13	2022-07-15	2022-07-15	1
FJA715702203	799702	25891203	312	312	2022-07-14	2022-07-15	2022-07-15	1
FJA715702302	799702	25891202	300	270	2022-07-12	2022-07-15	2022-07-15	1
FJA715702502	799702	25891501	229	229	2022-07-12	2022-07-15	2022-07-15	1
FJA715702502	799702	25891502	184	184	2022-07-12	2022-07-15	2022-07-15	1
FJA715702603	799702	25891101	418	397	2022-07-13	2022-07-15	2022-07-15	1
FJA715702603	799702	25891403	421	337	2022-07-13	2022-07-15	2022-07-15	1
FJA715702603	799702	25891503	105	105	2022-07-13	2022-07-15	2022-07-15	1
FJA715702603	799702	25891603	128	115	2022-07-13	2022-07-15	2022-07-15	1
FJA715703002	799703	25891302	50	48	2022-07-12	2022-07-15	2022-07-15	1

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\NSQLSERVER01;'
                       '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 (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
    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_delivery_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    57096.00000
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

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

```
1 SELECT
2     order_id,
3         customer_id,
4         order_placement_date,
5         on_time
6 FROM
7     OPENROWSET(
8         BULK 'https://supplyproject.dfs.core.windows.net/input/csvdata/fact_orders_aggregate_1.csv',
9         FORMAT = 'CSV',
10        HEADER_ROW=TRUE,
11        PARSER_VERSION = '2.0'
12
13    ) as [result]
14 WHERE
15     on_time = 0;
```

Azure Integration

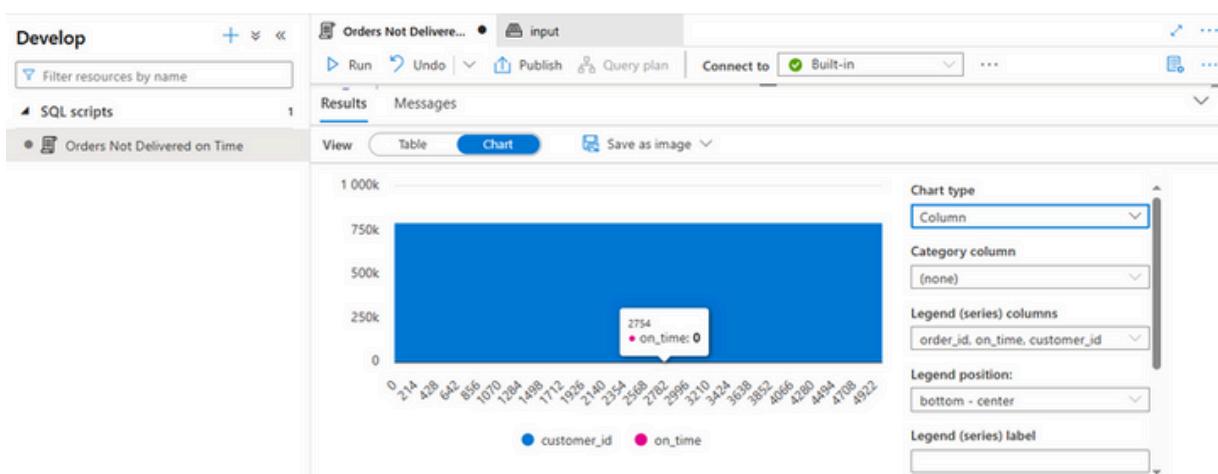
Result in table and chart



```
Develop Orders Not Deliver... input Count Orders Deliver... Total Delivered Quan...
+ Filter resources by name
SQL scripts 3
Count Orders Delivered Late
Orders Not Delivered on Time
Total Delivered Quantity per Product

Run Undo Publish Query plan Connect to Built-in ...
Results Messages
View Table Chart Export results ...
Search

order_id customer_id order_placeme... on_time
FMR32622402 789622 01/03/2022 0
FMR34201201 789201 01/03/2022 0
FMR32902601 789902 01/03/2022 0
FMR32402303 789402 01/03/2022 0
```



Azure Integration

2-Count orders delivered late



Synapse live Validate all Publish all

Develop + <>

Filter resources by name

SQL scripts 2

Orders Not Delivered on Time

Count Orders Delivered Late

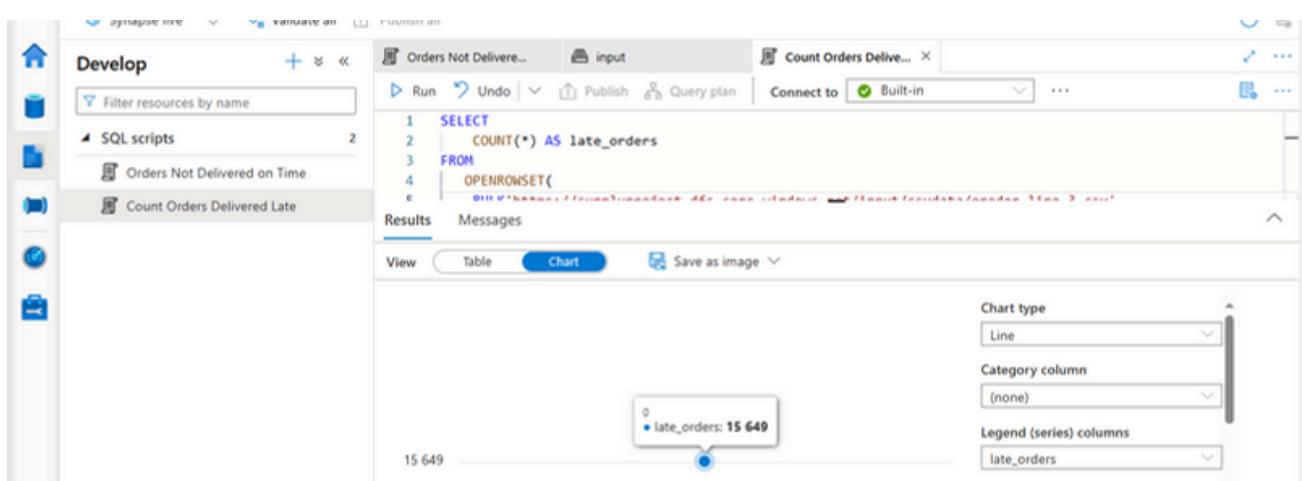
```
1 SELECT
2     COUNT(*) AS late_orders
3 FROM
4     OPENROWSET(
5         BULK'https://supplyproject.dfs.core.windows.net/input/csvdata/oreder_line_3.csv',
6         FORMAT = 'csv',
7         HEADER_ROW=TRUE,
8         PARSER_VERSION = '2.0'
9
10    ) as [result]
11 WHERE
12     actual_delivery_date > agreed_delivery_date;
```

Results Messages

View Table Chart Save as image

Publishing completed
Successfully published

Connect to Built-in



Azure Integration

3-Total delivered quantity per product



Synapse live Validate all Publish all

Develop + <>

Filter resources by name

SQL scripts 3

- Count Orders Delivered Late
- Orders Not Delivered on Time
- Total Delivered Quantity per Product**

Orders Not Deliver... input Count Orders Deliver... Total Delivered Qua... Run Undo Publish Query plan Connect to Built-in ...

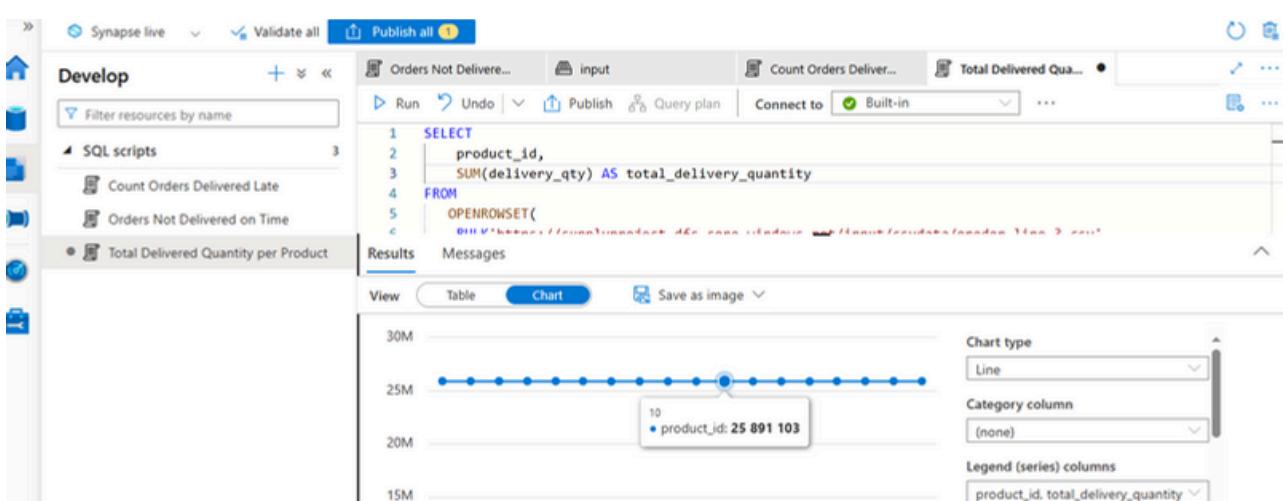
```
1 SELECT
2     product_id,
3     SUM(delivery_qty) AS total_delivery_quantity
4 FROM
5     OPENROWSET(
6         BULK 'https://supplyproject.dfs.core.windows.net/input/csvdata/oreder_line_3.csv',
7         FORMAT = 'csv',
8         HEADER ROW=TRUE,
```

Results Messages

View Table Export results

Search

product_id	total_delivery_quantity
25891601	382766
25891203	948395



Azure Integration

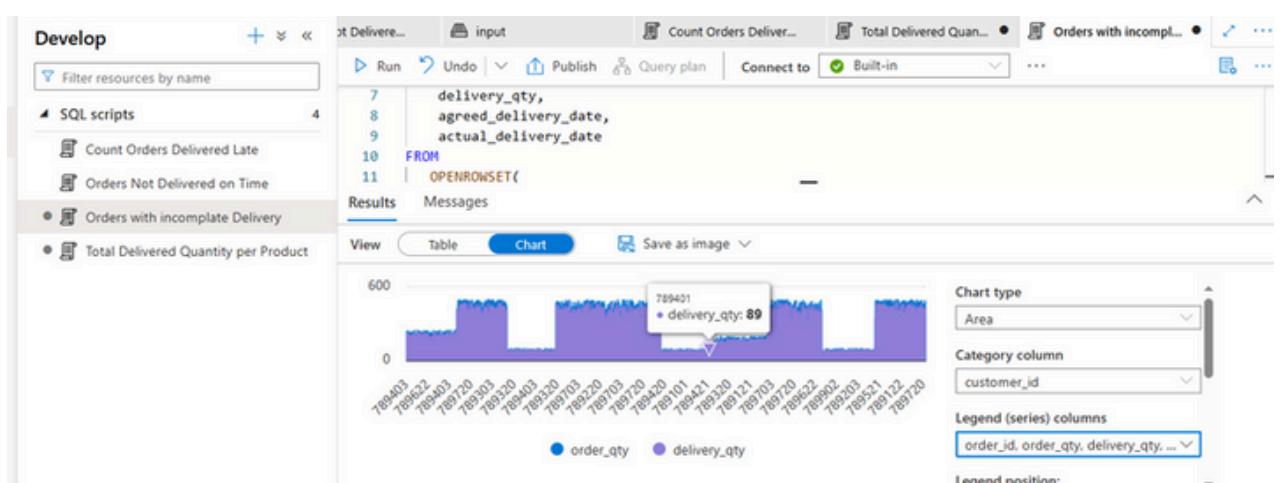
4-Orders with incomplete delivery

The screenshot shows the Azure Synapse Studio interface. On the left, there's a sidebar titled 'Develop' under 'SQL scripts' which lists several queries: 'Count Orders Delivered Late', 'Orders Not Delivered on Time', 'Orders with incomplete Delivery' (which is selected), and 'Total Delivered Quantity per Product'. The main area has tabs for 'Run', 'Undo', 'Publish', 'Query plan', and 'Connect to'. The 'Connect to' dropdown is set to 'Built-in'. Below these tabs, the query text is displayed:

```
7   delivery_qty,  
8   agreed_delivery_date,  
9   actual_delivery_date  
10  FROM  
11  OPENROWSET(
```

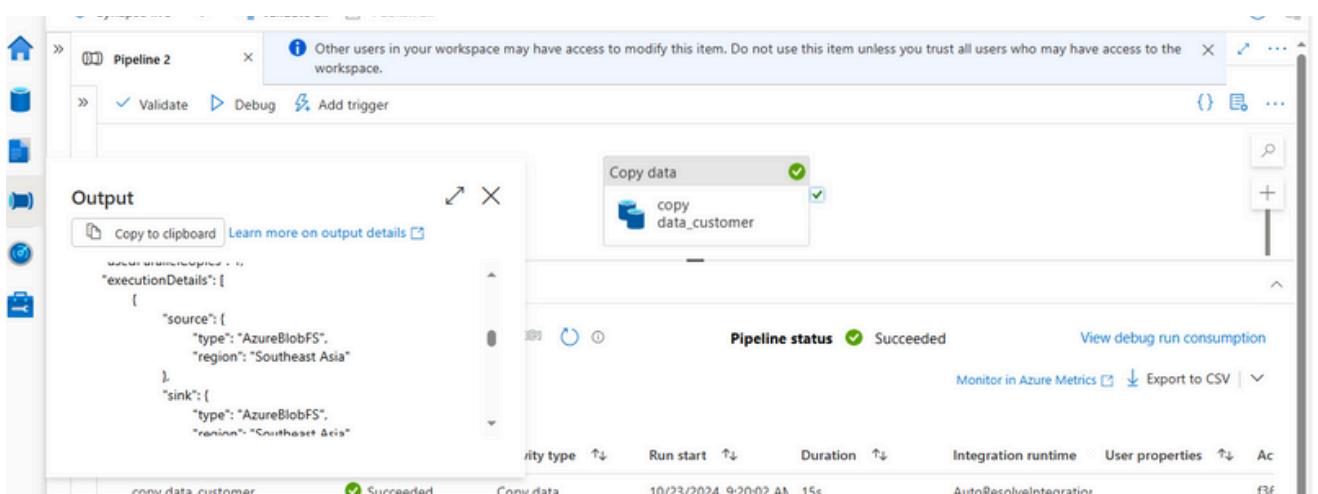
Under the 'Results' tab, there's a table with the following data:

order_id	order_placeme...	customer_id	product_id	order_qty	delivery_qty	agreed_deliver...
FMR36503501	04/03/2022	789503	25891501	196	176	06/03/2022
FMR37321501	04/03/2022	789321	25891501	225	214	07/03/2022
FMR37401602	04/03/2022	789401	25891501	243	219	07/03/2022
FMR37202501	04/03/2022	789202	25891501	227	216	07/03/2022



Azure Integration

Using pipeline activities: 1-Copy Data

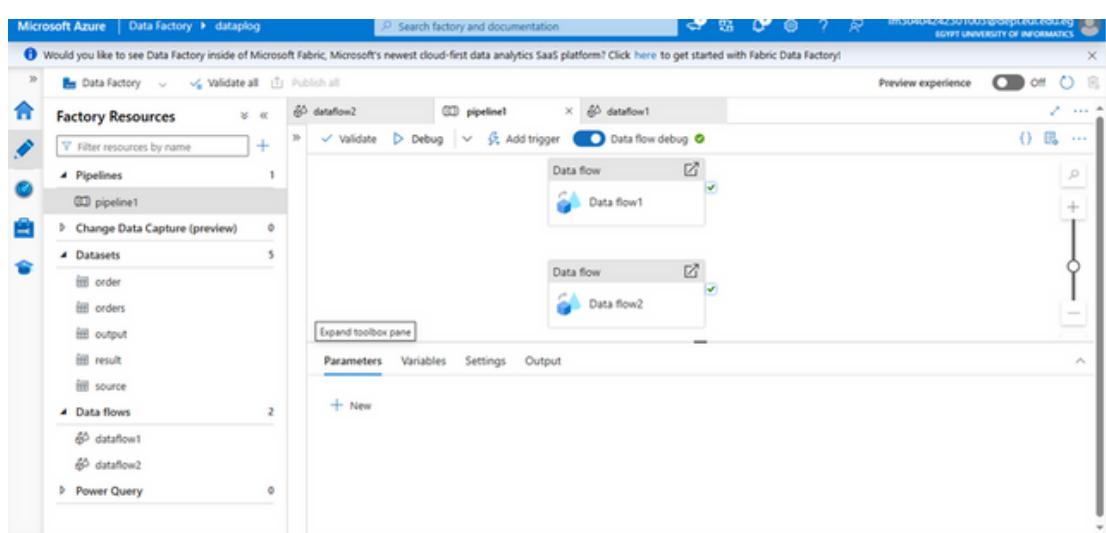
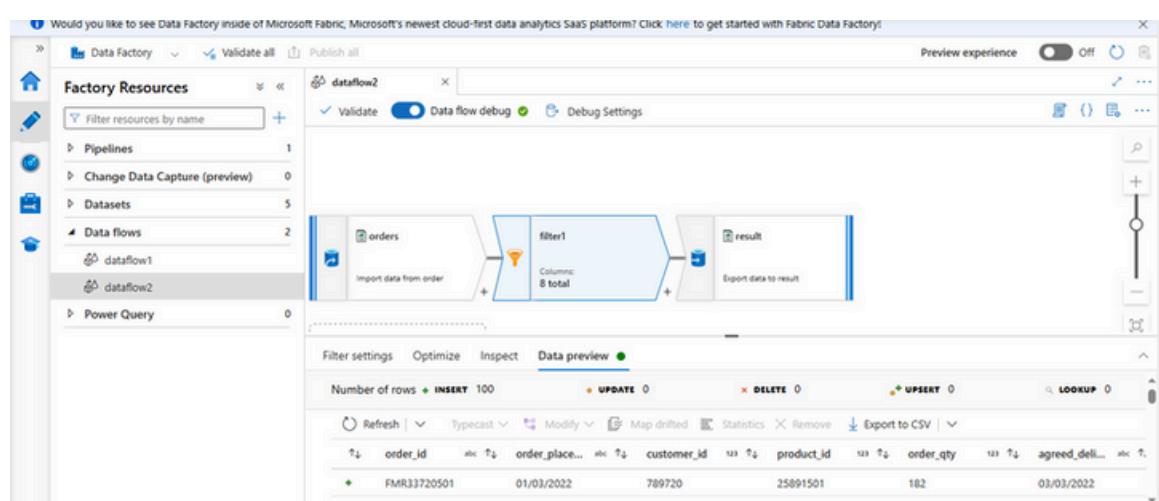
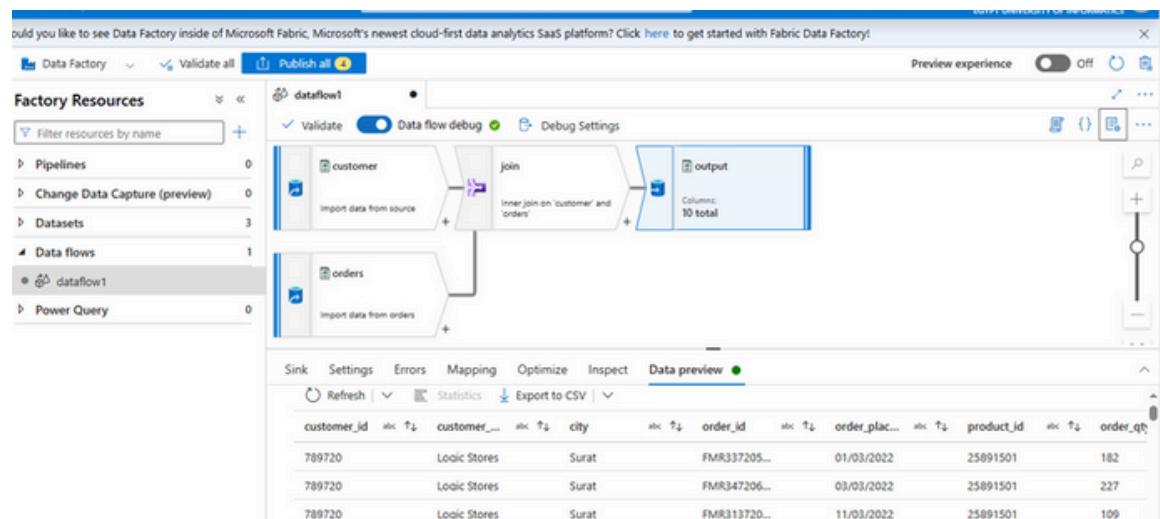


This screenshot shows the Azure Storage Blob container interface. On the left, the 'output' container is selected, showing its 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. A blob named 'result.csv' is listed under 'Name'. On the right, the 'result.csv' blob is previewed. The CSV file contains 16 rows of data, each with three columns: 'customer_id', 'customer_name', and 'city'. The data includes various store locations such as Rel Fresh, Ahmedabad, Vadodara, Surat, and Chipte Stores in cities like Rel Fresh, Ahmedabad, Vadodara, Surat, and Chipte.

customer_id	customer_name	city
789201	Rel Fresh	Surat
789202	Rel Fresh	Ahmedabad
789203	Rel Fresh	Vadodara
789301	Expression Stores	Surat
789303	Expression Stores	Vadodara
789101	Vijay Stores	Surat
789102	Vijay Stores	Ahmedabad
789103	Vijay Stores	Vadodara
789121	Coolblue	Ahmedabad
789122	Coolblue	Vadodara
789220	Atlas Stores	Surat
789221	Atlas Stores	Ahmedabad
789320	Chipte Stores	Ahmedabad
789321	Chipte Stores	Ahmedabad
789401	Propel Mart	Surat

Azure Integration

Using Data Factory: 1-Data Flows



04

THANK YOU!!

instructor

Hanaa gharib



ATLIQ
GRANDS





ATLIQ
GRANDS