



## **Technical Specification Document**

# **Sales Data Analysis and Data Modelling to support Business Requirements**

**Author - Shantanu Zodey**

**Date – 30/05/2025**

## Objective

The purpose of this project is to transform raw orders, customer, and shipping data from different formats (Excel, CSV, JSON) into a unified data model that supports advanced business reporting and analytics. The goal is to enable the business to gain valuable insights into customer behavior, product performance, shipping efficiency, and geographic trends, which will drive strategic decision-making.

By leveraging SQL-based data modeling, this project aims to ensure data consistency, integrity, and traceability, while preparing the dataset for scalable reporting across key metrics like total sales, customer transactions, product popularity, and delivery statuses.

# Project Scope

This project includes the following activities:

## 1. Source Data Integration

- Ingesting data from three sources:
  - Customer (Excel)
  - Orders (CSV)
  - Shipping (JSON)

## 2. Data Quality Checks

- Performing completeness, accuracy, and referential integrity checks using SQL.

## 3. Data Modeling

- Designing a star schema using dimension and fact tables.
- Resolving inconsistencies in item prices and categorize customers by age.

## 4. ETL Logic Development

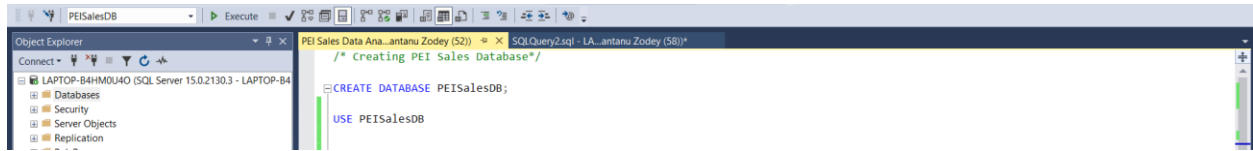
- Defining and implement transformations to populate the target model.

## 5. Enable Reporting Requirements

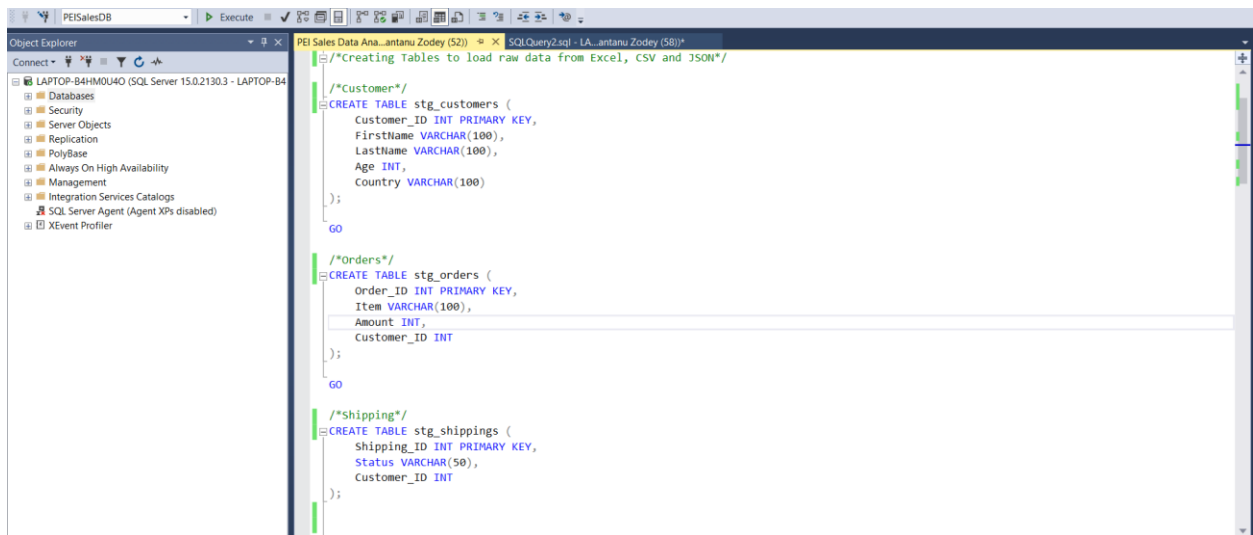
- Structure data to support five business reporting use cases including:
  - Total spent per country with pending deliveries
  - Product performance by age category
  - Transaction volume and sales per customer
  - Country with lowest sales and transaction count

# Source Data Integration

## a) Creating Database PEISalesDB



## b) Creating Staging Tables for Customers, Orders and Shipping.



- The above staging tables will be used to load raw data and perform exploratory data analysis, data quality, data consistency and data accuracy checks.
- Once the staging tables are created, we loaded the data for Customer data from Excel and Orders data from CSV using Import Data functionality in Microsoft SQL Server. As Shipping data is in JSON format, Microsoft SQL Server does not support Import Data functionality for JSON file format. So, to convert JSON data into CSV data, we leveraged python and converted shipping.json data to shipping.csv file and then loaded the shipping data using Import Data functionality. Please find below snapshot.

jupyter Untitled18 Last Checkpoint: 7 minutes ago (unsaved changes) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

### Converting Shipping.json data to CSV data

```
In [1]: import pandas as pd
import json

In [8]: pwd

Out[8]: 'C:\\Users\\Shantanu Zodey\\Desktop\\Data Task'

In [9]: # Loading JSON data from Shipping.json file
with open('Shipping.json', 'r') as file:
    data = json.load(file)

In [10]: # Converting to DataFrame
df = pd.DataFrame(data)

In [12]: # Saving to CSV
df.to_csv('Shipping.csv', index=False)

In [13]: ls

Volume in drive C is Windows
Volume Serial Number is F46E-2E44

Directory of C:\\Users\\Shantanu Zodey\\Desktop\\Data Task

05/30/2025  02:13 AM  <DIR>          .
05/30/2025  01:20 AM  <DIR>          ..
05/29/2025  12:34 AM              31,232 Customer.xls
05/29/2025  12:34 AM              5,221 Order.csv
05/30/2025  02:13 AM              4,260 Shipping.csv
05/29/2025  12:34 AM             20,230 Shipping.json
                4 File(s)          60,943 bytes
                2 Dir(s)  4,913,369,088 bytes free
```

c) Validating all the records and counts to ensure all records are loaded into staging table.

## Stg\_customer

Object Explorer: LAPTOP-B4HM0U40 (SQL Server 15.0.2130.3 - LAPTOP-B4)

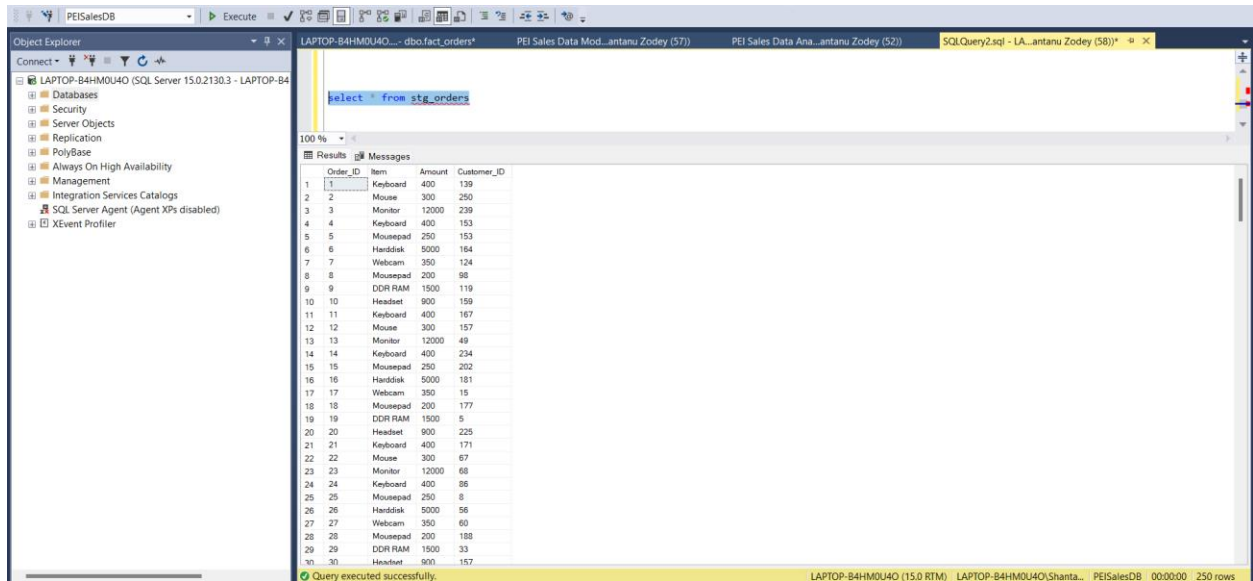
SQLQuery2.sql - LA...antanu Zodey (58)\*

```
select * from stg_customers
```

Customer_ID	FirstName	LastName	Age	Country
1	Joseph	Rice	43	USA
2	Gary	Moore	71	USA
3	John	Walker	44	UK
4	Eric	Carter	38	UK
5	William	Jackson	58	USA
6	Nicole	Jones	33	USA
7	David	Davis	59	USA
8	Jason	Montgomery	56	UK
9	Kent	Weaver	61	UK
10	Darrell	Dillon	50	USA
11	Jacqueline	Wang	22	USA
12	Jodi	Gonzalez	69	USA
13	Omar	Martin	28	UK
14	Nicole	Lara	77	UK
15	Jason	Brown	63	USA
16	David	Benson	61	USA
17	Kimberly	Moss	34	USA
18	Erin	Macias	25	UK
19	James	Johnson	67	UK
20	Derek	Peterson	36	USA
21	Diane	Henson	74	USA
22	Cody	Loyon	47	USA
23	Margaret	Hardy	25	UK
24	Arthur	Hayes	75	UK
25	Raymond	Taylor	36	USA
26	Tina	Moore	34	USA
27	Kyle	White	63	USA
28	Jonathan	Manning	30	UK
29	Angela	Bryant	23	UK
30	Michael	Mann	56	USA

Query executed successfully. LAPTOP-B4HM0U40 (15.0 RTM) LAPTOP-B4HM0U40/Shanta... PEISalesDB 00:00:00 250 rows

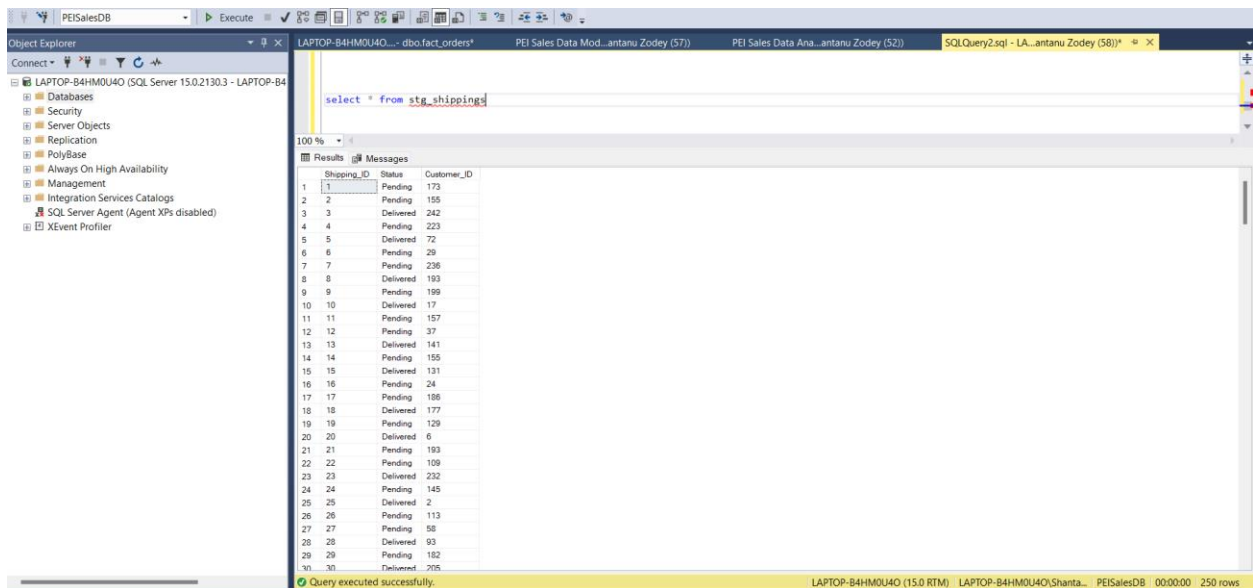
## stg\_orders



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the Object Explorer with the server 'LAPTOP-B4HM0U4O' expanded. The right pane shows the query editor with the query 'select \* from stg\_orders'. The bottom pane displays the results of the query, which is a table with 30 rows and 4 columns: Order\_ID, Item, Amount, and Customer\_ID. The status bar at the bottom indicates that the query was executed successfully and returned 250 rows.

Order_ID	Item	Amount	Customer_ID
1	Keyboard	400	139
2	Mouse	300	250
3	Monitor	12000	239
4	Keyboard	400	153
5	Mousepad	250	153
6	Harddisk	5000	164
7	Webcam	350	124
8	Mousepad	200	98
9	DDR RAM	1500	119
10	Headset	900	159
11	Keyboard	400	167
12	Mouse	300	157
13	Monitor	12000	49
14	Keyboard	400	234
15	Mousepad	250	202
16	Harddisk	5000	181
17	Webcam	350	15
18	Mousepad	200	177
19	DDR RAM	1500	5
20	Headset	900	225
21	Keyboard	400	171
22	Mouse	300	67
23	Monitor	12000	68
24	Keyboard	400	98
25	Mousepad	250	8
26	Harddisk	5000	56
27	Webcam	350	60
28	Mousepad	200	188
29	DDR RAM	1500	33
30	Headset	900	157

## stg\_shippings

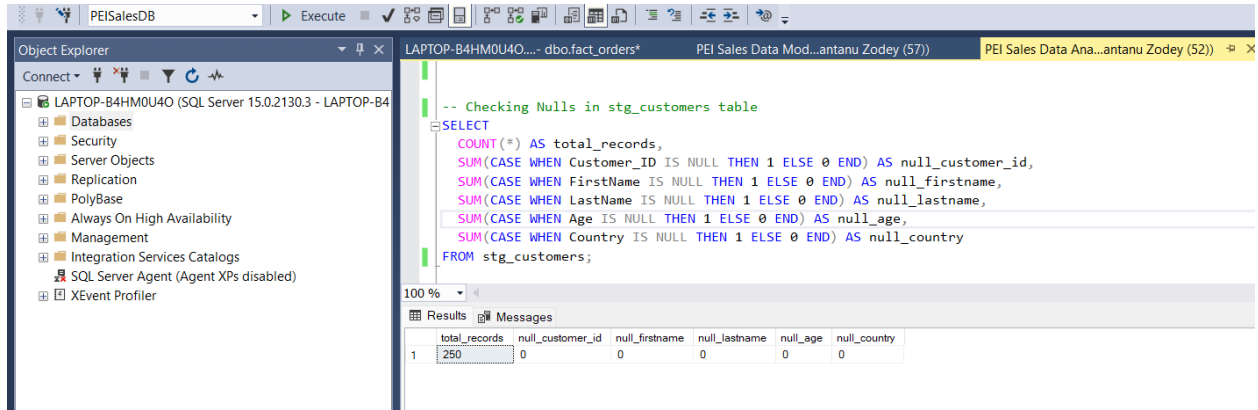


The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the Object Explorer with the server 'LAPTOP-B4HM0U4O' expanded. The right pane shows the query editor with the query 'select \* from stg\_shippings'. The bottom pane displays the results of the query, which is a table with 30 rows and 3 columns: Shipping\_ID, Status, and Customer\_ID. The status bar at the bottom indicates that the query was executed successfully and returned 250 rows.

Shipping_ID	Status	Customer_ID
1	Pending	173
2	Pending	155
3	Delivered	242
4	Pending	223
5	Delivered	72
6	Pending	29
7	Pending	236
8	Delivered	193
9	Pending	199
10	Delivered	17
11	Pending	157
12	Pending	37
13	Delivered	141
14	Pending	155
15	Delivered	131
16	Pending	24
17	Pending	186
18	Delivered	177
19	Pending	129
20	Delivered	6
21	Pending	193
22	Pending	109
23	Delivered	232
24	Pending	145
25	Delivered	2
26	Pending	113
27	Pending	58
28	Delivered	93
29	Pending	182
30	Delivered	205

# Data Quality Checks

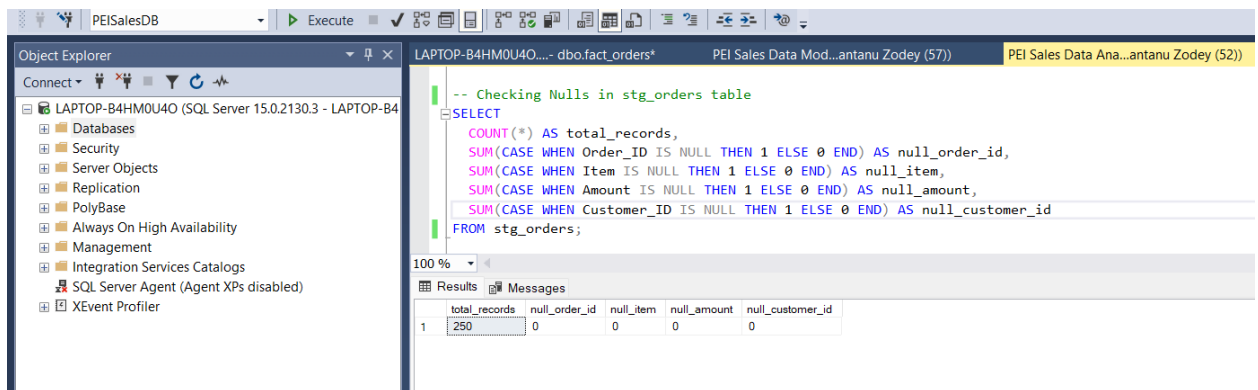
## a) Checking Nulls in stg\_customer table



The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for LAPTOP-B4HM0U40. The main window shows a query titled "Checking Nulls in stg\_customers table". The query is a SELECT statement that counts the total records and sums the number of nulls for Customer\_ID, FirstName, LastName, Age, and Country. The results pane shows a single row with the following values:

	total_records	null_customer_id	null_firstname	null_lastname	null_age	null_country
1	250	0	0	0	0	0

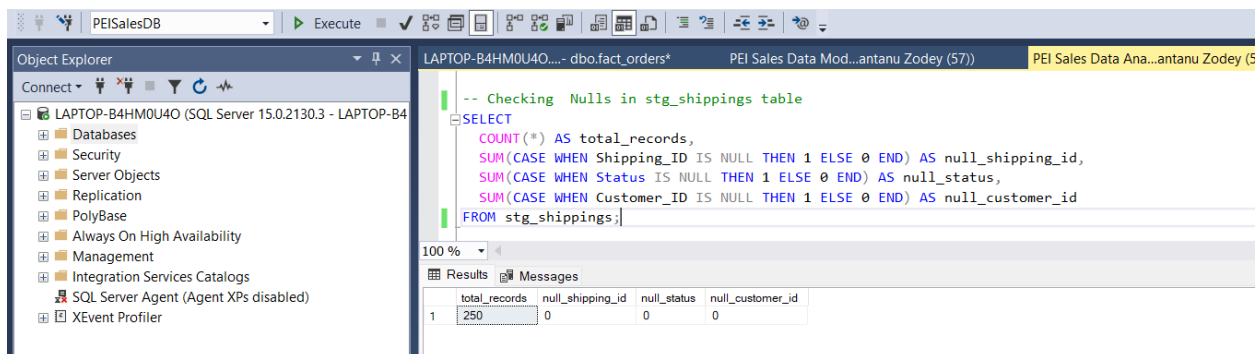
## b) Checking Nulls in stg\_orders table



The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for LAPTOP-B4HM0U40. The main window shows a query titled "Checking Nulls in stg\_orders table". The query is a SELECT statement that counts the total records and sums the number of nulls for Order\_ID, Item, Amount, and Customer\_ID. The results pane shows a single row with the following values:

	total_records	null_order_id	null_item	null_amount	null_customer_id
1	250	0	0	0	0

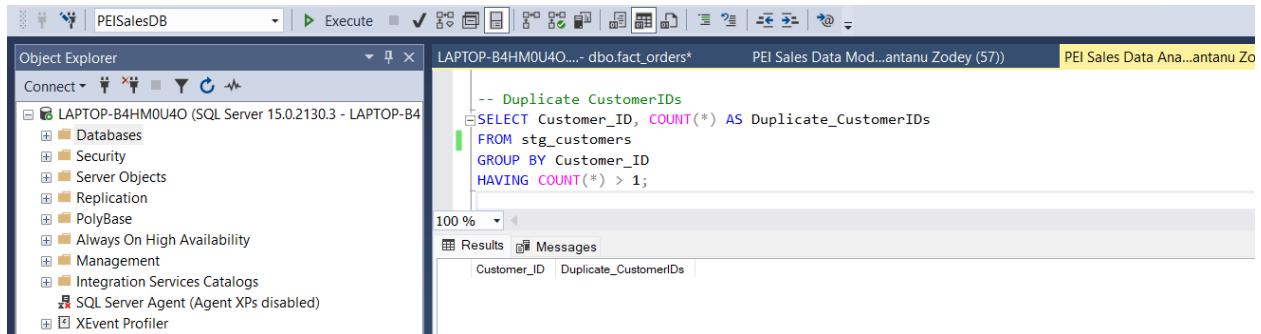
## c) Checking Nulls in stg\_shipping table



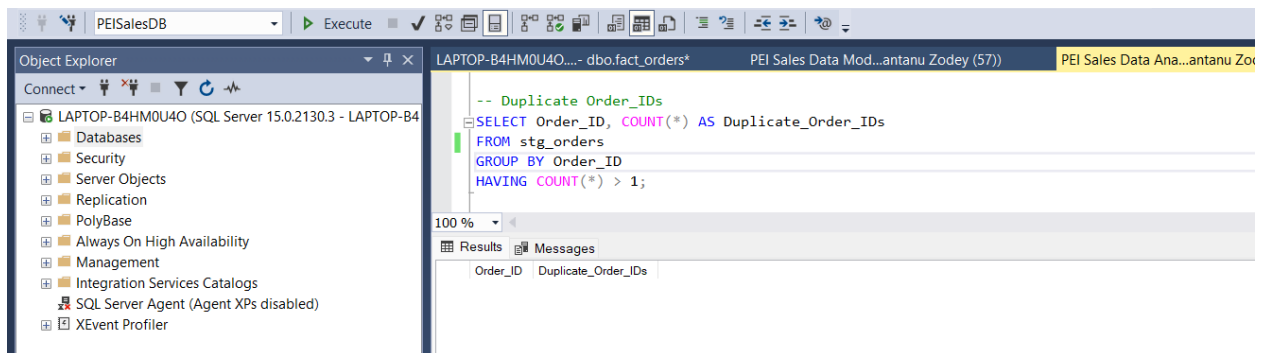
The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for LAPTOP-B4HM0U40. The main window shows a query titled "Checking Nulls in stg\_shippings table". The query is a SELECT statement that counts the total records and sums the number of nulls for Shipping\_ID, Status, and Customer\_ID. The results pane shows a single row with the following values:

	total_records	null_shipping_id	null_status	null_customer_id
1	250	0	0	0

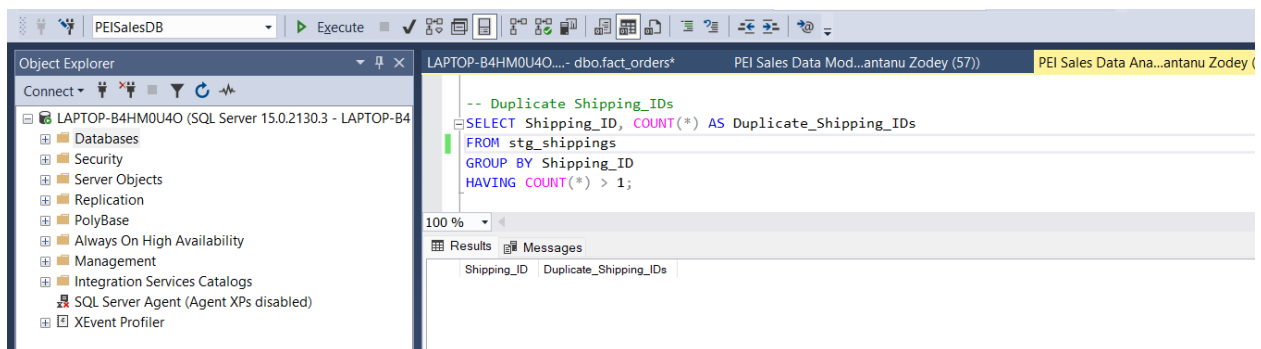
#### d) Checking Duplicate Customer IDs in stg\_customer table



#### e) Checking Duplicate Order IDs in stg\_orders table



#### f) Checking Duplicate Shipping IDs in stg\_shipping table



#### g) Checking Alpha-Numeric and Special characters in First Name



Object Explorer: LAPTOP-B4HM0U4O (SQL Server 15.0.2130.3 - LAPTOP-B4)

Query: LAPTOP-B4HM0U4O...- dbo.fact\_orders\* PEI Sales Data Mod...antanu Zodey (57) PEI Sales Data Ana...antanu Zodey

```

/*Checking Data Type Validity*/
-- Alpha-Numeric and Special Characters in First Name
SELECT * FROM stg_customers
WHERE FirstName LIKE '%[^a-zA-Z0-9 _]%'

```

Results:

	Customer_ID	FirstName	LastName	Age	Country
1	6	Nicole	Jones	33	USA
2	14	Nicole	Lara	77	UK
3	162	Nicole	Bennett	51	USA
4	171	L@rry	Cole	50	USA
5	214	Nicole	Mcintyre	18	UK

In the above screenshot, there are few records in First Name that contains special characters. This column needs transformation in order to get the clean and accurate data.

#### h) Checking Alpha-Numeric and Special characters in Last Name, Country and Item

Object Explorer: LAPTOP-B4HM0U4O (SQL Server 15.0.2130.3 - LAPTOP-B4)

Query: LAPTOP-B4HM0U4O...- dbo.fact\_orders\* PEI Sales Data Mod...antanu Zodey (57) PEI Sales Data Ana...antanu Zodey

```

-- Alpha-Numeric and Special Characters in Last Name
SELECT * FROM stg_customers
WHERE LastName LIKE '%[^a-zA-Z0-9 _]%'

-- Alpha-Numeric and Special Characters in Country
SELECT * FROM stg_customers
WHERE Country LIKE '%[^a-zA-Z0-9 _]%'

-- Alpha-Numeric and Special Characters in Item
SELECT * FROM stg_orders
WHERE Item LIKE '%[^a-zA-Z0-9 _]%'

```

Results:

Customer_ID	FirstName	LastName	Age	Country
-------------	-----------	----------	-----	---------

Customer_ID	FirstName	LastName	Age	Country
-------------	-----------	----------	-----	---------

Order_ID	Item	Amount	Customer_ID
----------	------	--------	-------------

#### i) Checking Negative or Unrealistic Ages.

PEISalesDB

Execute

Object Explorer

Connect

LAPTOP-B4HM0U40 (SQL Server 15.0.2130.3 - LAPTOP-B4)

Databases

Security

Server Objects

Replication

PolyBase

Always On High Availability

Management

Integration Services Catalogs

LAPTOP-B4HM0U40...- dbo.fact\_orders\* PEI Sales Data Mod...antanu Zodey (57)) PEI Sales Data

```
-- Negative or unrealistic ages
SELECT * FROM stg_customers WHERE Age <= 0 OR Age >= 120;
```

100 %

Results Messages

Customer_ID	FirstName	LastName	Age	Country
-------------	-----------	----------	-----	---------

## j) Checking Invalid Amounts

PEISalesDB

Execute

Object Explorer

Connect

LAPTOP-B4HM0U40 (SQL Server 15.0.2130.3 - LAPTOP-B4)

Databases

Security

Server Objects

Replication

PolyBase

Always On High Availability

Management

Integration Services Catalogs

LAPTOP-B4HM0U40...- dbo.fact\_orders\* PEI Sales Data Mod...antanu Zodey (57)) PEI Sales Data An

```
-- Invalid Amounts
SELECT * FROM stg_orders WHERE Amount <= 0;
```

100 %

Results Messages

Order_ID	Item	Amount	Customer_ID
----------	------	--------	-------------

## k) Referential Integrity Checks for valid customers present in stg\_orders

PEISalesDB

Execute

Object Explorer

Connect

LAPTOP-B4HM0U40 (SQL Server 15.0.2130.3 - LAPTOP-B4)

Databases

Security

Server Objects

Replication

PolyBase

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

LAPTOP-B4HM0U40...- dbo.fact\_orders\* PEI Sales Data Mod...antanu Zodey (57)) PEI Sales Data Ana...antanu Zo

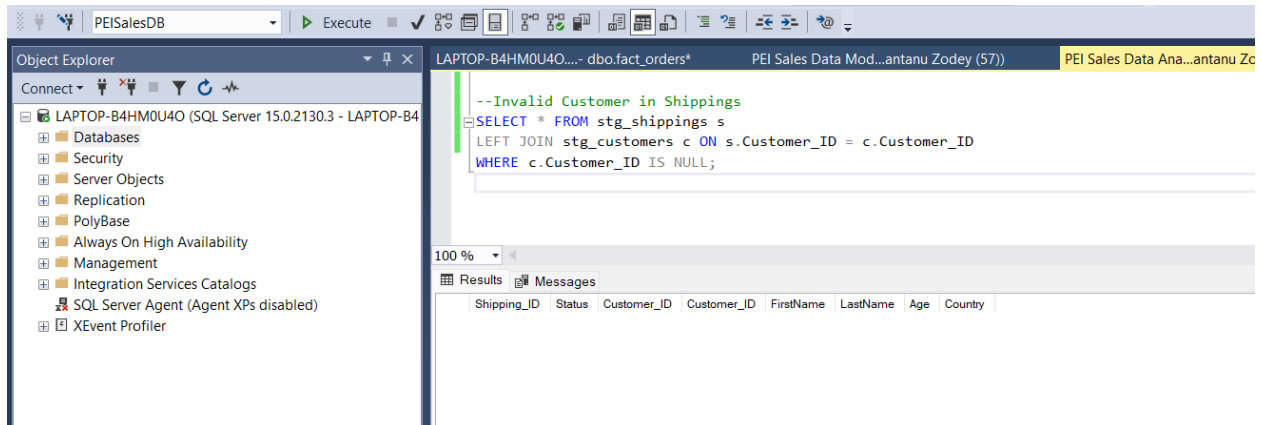
```
/*Checking Referential Integrity*/
/*Ensuring all CustomerIDs in orders Data and shippings Data exist in customers Data*/
SELECT * FROM stg_orders o
LEFT JOIN stg_customers c ON o.Customer_ID = c.Customer_ID
WHERE c.Customer_ID IS NULL;
```

100 %

Results Messages

Order_ID	Item	Amount	Customer_ID	Customer_ID	FirstName	LastName	Age	Country
----------	------	--------	-------------	-------------	-----------	----------	-----	---------

## l) Referential Integrity Checks for valid customers present in stg\_shippings



### Findings for Data Quality checks:

- First Name column in stg\_customer table contains special characters
- Duplicate Items with different amounts.
- All datasets are joinable via CustomerID.
- No nulls found in critical fields.
- No duplicates found in critical fields.

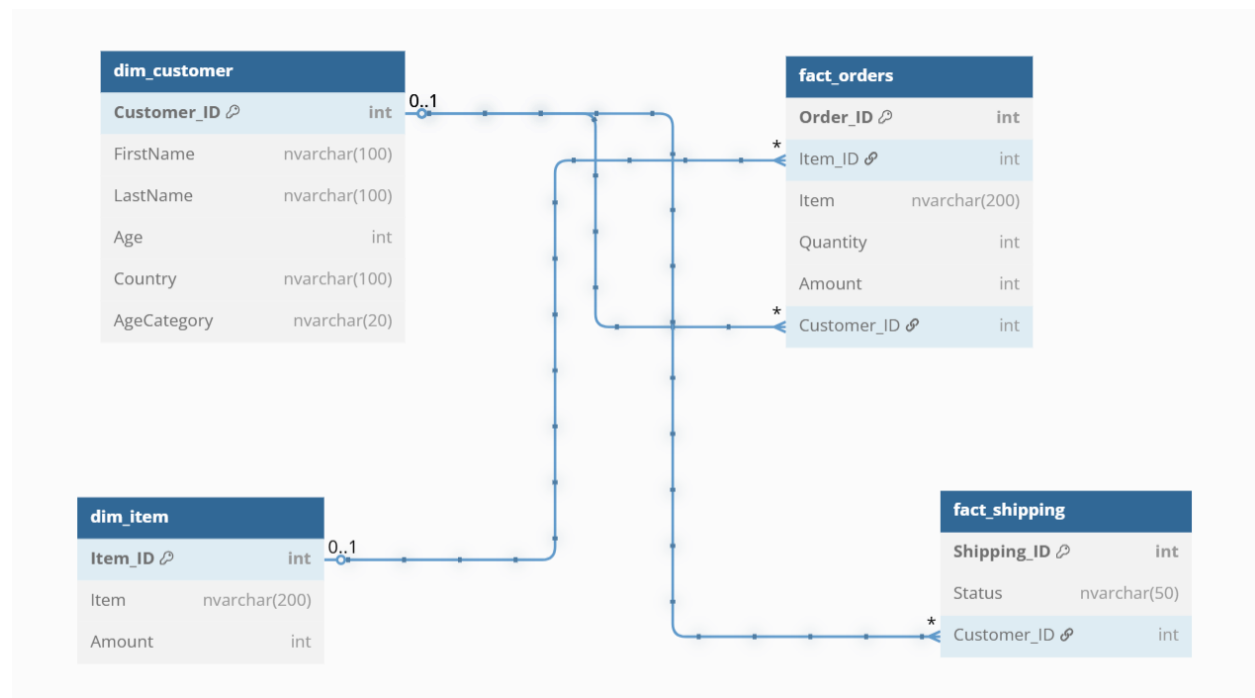
# Data Modelling

To fulfill the Business reporting requirement, we designed a **Star Schema** with:

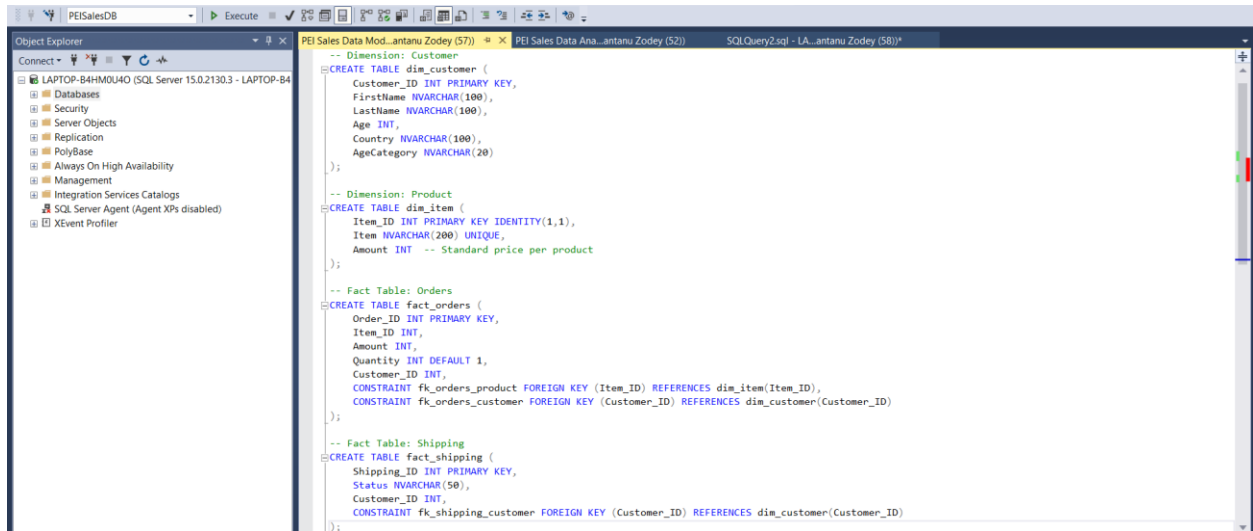
- **2 Dimension Tables:** dim\_customer, dim\_product
- **2 Fact Tables:** fact\_orders, fact\_shipping

Please find below snapshot.

## Domain Model (ERD)



Below are the table schema's that has been created in Database.



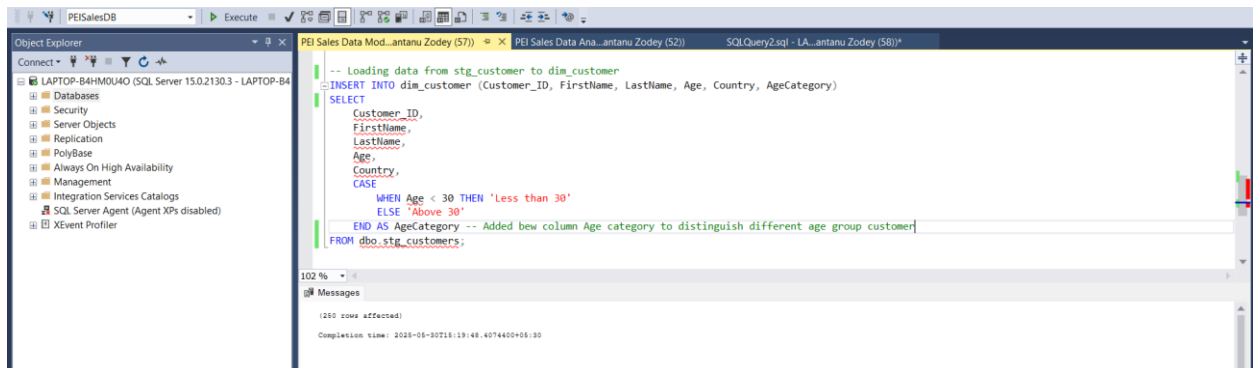
```
-- Dimension: Customer
CREATE TABLE dim_customer (
    Customer_ID INT PRIMARY KEY,
    FirstName NVARCHAR(100),
    LastName NVARCHAR(100),
    Age INT,
    Country NVARCHAR(100),
    AgeCategory NVARCHAR(20)
);

-- Dimension: Product
CREATE TABLE dim_item (
    Item_ID INT PRIMARY KEY IDENTITY(1,1),
    Item NVARCHAR(200) UNIQUE,
    Amount INT -- Standard price per product
);

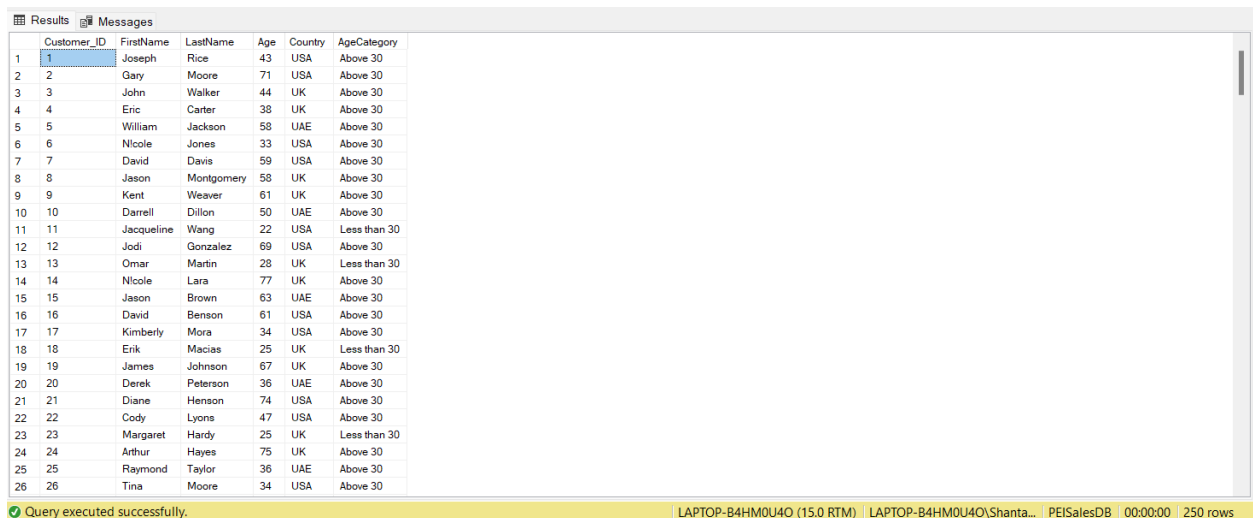
-- Fact Table: Orders
CREATE TABLE fact_orders (
    Order_ID INT PRIMARY KEY,
    Item_ID INT,
    Amount INT,
    Quantity INT DEFAULT 1,
    Customer_ID INT,
    CONSTRAINT fk_orders_product FOREIGN KEY (Item_ID) REFERENCES dim_item(Item_ID),
    CONSTRAINT fk_orders_customer FOREIGN KEY (Customer_ID) REFERENCES dim_customer(Customer_ID)
);

-- Fact Table: Shipping
CREATE TABLE fact_shipping (
    Shipping_ID INT PRIMARY KEY,
    Status NVARCHAR(50),
    Customer_ID INT,
    CONSTRAINT fk_shipping_customer FOREIGN KEY (Customer_ID) REFERENCES dim_customer(Customer_ID)
);
```

Loading data from stg\_customer to dim\_customer.



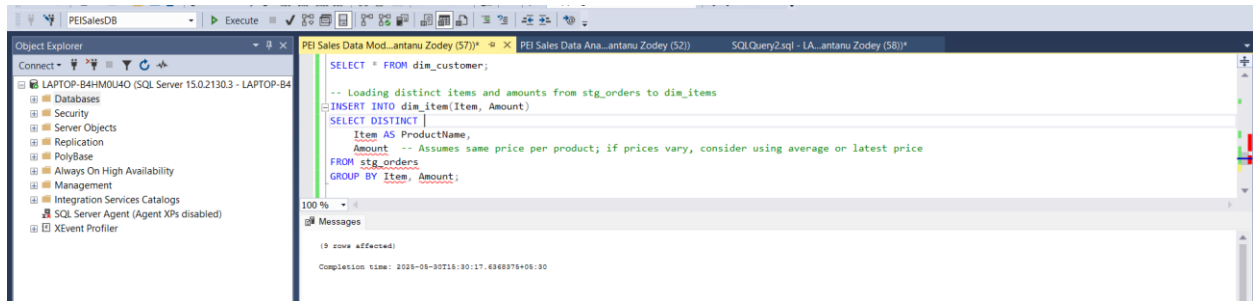
```
-- Loading data from stg_customer to dim_customer
INSERT INTO dim_customer (Customer_ID, FirstName, LastName, Age, Country, AgeCategory)
SELECT
    Customer_ID,
    FirstName,
    LastName,
    Age,
    Country,
    CASE
        WHEN Age < 30 THEN 'Less than 30'
        ELSE 'Above 30'
    END AS AgeCategory -- Added new column Age category to distinguish different age group customer
FROM dbo.stg_customers;
```



	Customer_ID	FirstName	LastName	Age	Country	AgeCategory
1	1	Joseph	Rice	43	USA	Above 30
2	2	Gary	Moore	71	USA	Above 30
3	3	John	Walker	44	UK	Above 30
4	4	Eric	Carter	38	UK	Above 30
5	5	William	Jackson	58	UAE	Above 30
6	6	Nicole	Jones	33	USA	Above 30
7	7	David	Davis	59	USA	Above 30
8	8	Jason	Montgomery	58	UK	Above 30
9	9	Kent	Weaver	61	UK	Above 30
10	10	Darrell	Dillon	50	UAE	Above 30
11	11	Jacqueline	Wang	22	USA	Less than 30
12	12	Jodi	Gonzalez	69	USA	Above 30
13	13	Omar	Martin	28	UK	Less than 30
14	14	Nicole	Lara	77	UK	Above 30
15	15	Jason	Brown	63	UAE	Above 30
16	16	David	Benson	61	USA	Above 30
17	17	Kimberly	Mora	34	USA	Above 30
18	18	Erik	Macias	25	UK	Less than 30
19	19	James	Johnson	67	UK	Above 30
20	20	Derek	Peterson	36	UAE	Above 30
21	21	Diane	Henson	74	USA	Above 30
22	22	Cody	Lyons	47	USA	Above 30
23	23	Margaret	Hardy	25	UK	Less than 30
24	24	Arthur	Hayes	75	UK	Above 30
25	25	Raymond	Taylor	36	UAE	Above 30
26	26	Tina	Moore	34	USA	Above 30

Query executed successfully. LAPTOP-B4HMOU4O (15.0 RTM) LAPTOP-B4HMOU4O\Shanta... PEISalesDB 00:00:00 250 rows

Loading Distinct Items and Amounts from stg\_orders into dim\_items.



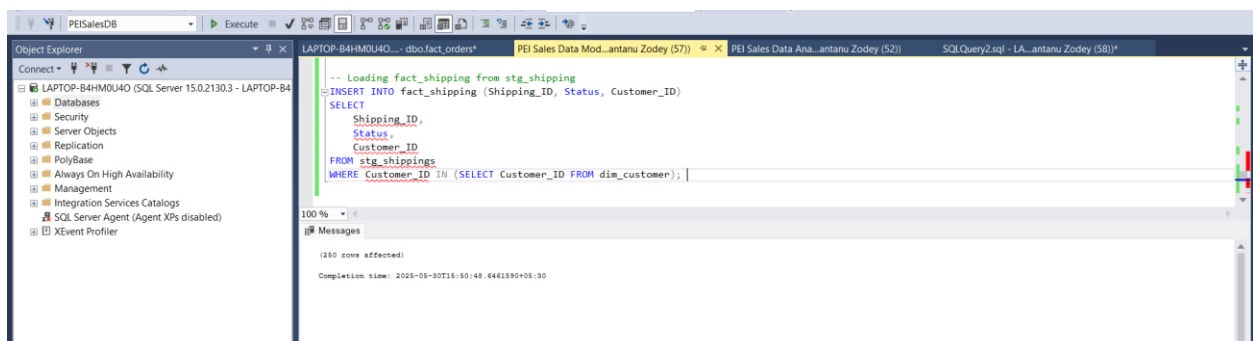
```
SELECT * FROM dim_item;
```

100 %

Results Messages

	Item_ID	Item	Amount
1	1	DDR RAM	1500
2	2	Harddisk	5000
3	3	Headset	900
4	4	Keyboard	400
5	5	Monitor	12000
6	6	Mouse	300
7	7	Mousepad	200
8	8	Mousepad	250
9	9	Webcam	350

Loading data from stg\_shipping to fact\_shipping.



Object Explorer: Connect to LAPTOP-B4HMOU40 (SQL Server 15.0.2130.3 - LAPTOP-B4HMOU40) | PEISalesDB | Execute | SQL Query 2.sql - LA...antanz Zodey (581)\*

Query: SELECT \* FROM fact\_shipping

Shipping_ID	Status	Customer_ID
1	Pending	173
2	Pending	155
3	Delivered	242
4	Pending	223
5	Delivered	72
6	Pending	29
7	Pending	236
8	Delivered	193
9	Pending	199
10	Delivered	17
11	Pending	157
12	Pending	27
13	Delivered	141
14	Pending	155
15	Delivered	131
16	Pending	24
17	Pending	186
18	Delivered	177
19	Pending	129
20	Delivered	6
21	Pending	193
22	Pending	109
23	Delivered	232
24	Pending	145
25	Delivered	2
26	Pending	113
27	Pending	58
28	Delivered	93
29	Pending	182
...	...	...

Query executed successfully. LAPTOP-B4HMOU40 (15.0 RTM) LAPTOP-B4HMOU40\Shanta... PEISalesDB 00:00:00 250 rows

## Loading fact\_order table.

To populate the fact\_orders table for transactional reporting. A Data engineer should create a table in such a way that it captures individual product purchase events linked to customer and item dimensions.

In a way Business stakeholders can analyze total sales, customer purchasing behavior, and product performance across various dimensions such as country, age category, and product type.

## Business Context

The fact\_orders table will store transactional-level data based on raw order records from the sales team. Each record in the raw orders CSV contains an Order\_ID, Item, Amount, and CustomerID. The goal is to link this to:

- A **product dimension (dim\_product)** for consistent product representation.
- A **customer dimension (dim\_customer)** to associate orders with customer demographics.

This table supports reporting on:

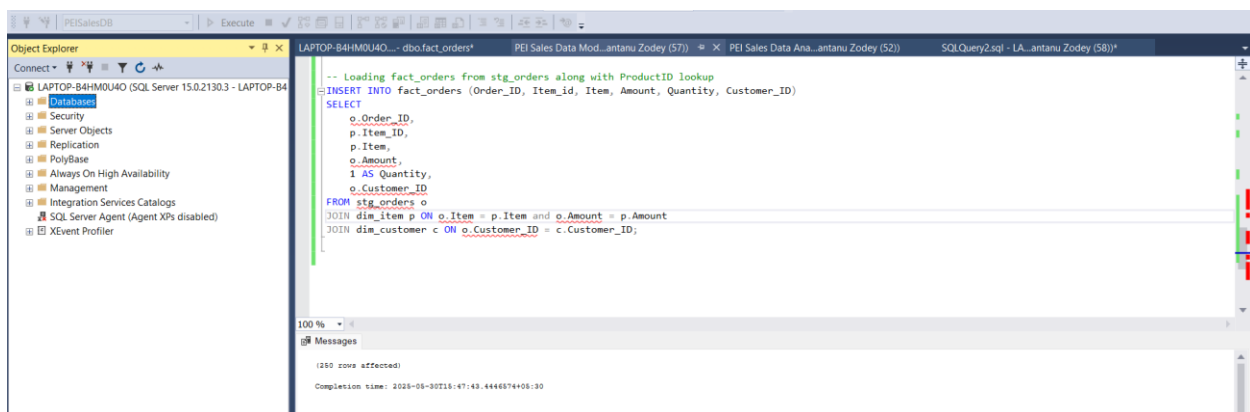
- Total sales by customer
- Product purchases by age category
- Quantity sold per item

## Transformations

```
SELECT
    o.Order_ID,
    p.Item_ID,
    p.Item,
    o.Amount,
    1 AS Quantity,
    o.Customer_ID
FROM stg_orders o
JOIN dim_item p ON o.Item = p.Item and o.Amount = p.Amount
JOIN dim_customer c ON o.Customer_ID = c.Customer_ID;
```

- Join with dim\_item on Item to obtain Item\_ID.
- Validate Customer\_ID exists in dim\_customer.
- Derive Quantity: Hardcoded to 1 per transaction.
- Exclude null or malformed records in Order\_ID, Item, Amount, or CustomerID.

## Implementation





Object Explorer: LAPTOP-B4HM0U4O (SQL Server 15.0.2130.3 - LAPTOP-B4HM0U4O)

Query: SELECT \* FROM fact\_orders

Order_ID	Item_ID	Item	Amount	Quantity	Customer_ID
1	4	Keyboard	400	1	139
2	2	Mouse	300	1	250
3	3	Monitor	12000	1	239
4	4	Keyboard	400	1	153
5	5	Mousepad	250	1	153
6	6	Harddisk	5000	1	164
7	7	Webcam	350	1	124
8	8	Mousepad	200	1	98
9	9	DDR RAM	1500	1	119
10	10	Headset	900	1	159
11	11	Keyboard	400	1	167
12	12	Mouse	300	1	157
13	13	Monitor	12000	1	49
14	14	Keyboard	400	1	234
15	15	Mousepad	250	1	202
16	16	Harddisk	5000	1	181
17	17	Webcam	350	1	15
18	18	Mousepad	200	1	177
19	19	DDR RAM	1500	1	5
20	20	Headset	900	1	225
21	21	Keyboard	400	1	171
22	22	Mouse	300	1	67
23	23	Monitor	12000	1	68
24	24	Keyboard	400	1	86
25	25	Mousepad	250	1	8
26	26	Harddisk	5000	1	56
27	27	Webcam	350	1	80
28	28	Mousepad	250	1	188
29	29	DDR RAM	1500	1	33
30	30	Headset	900	1	157
31	31	Keyboard	400	1	87

Query executed successfully. LAPTOP-B4HM0U4O (15.0 RTM) LAPTOP-B4HM0U4O\Shanta... PEISalesDB 00:00:00 250 rows

## QA Test Criteria

Test Case #	Description	Expected Outcome
TC1	Validate Order_ID uniqueness	No duplicates in fact_orders.Order_ID
TC2	Verify Item_ID and Customer_ID exist in dimension	All values must match dimension records
TC3	Validate Amount is not null and > 0	All records must pass
TC4	Ensure Quantity is always 1	No deviation
TC5	Join test with dim_Item and dim_customer	Ensure joins produce expected results

# Enable Reporting Requirements

1. Total amount spent and the country for the "Pending" delivery status for each country

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the server structure for 'LAPTOP-B4HM0U40'. The central pane shows a SQL query in the 'SQLQuery2.sql' file. The query is designed to calculate the total amount spent for each country where the delivery status is 'Pending'. The results pane at the bottom shows the output of the query.

```
-- Total amount spent and the country for the "Pending" delivery status for each country
SELECT
    c.Country,
    SUM(o.Amount) AS TotalAmountSpent
FROM fact_shipping s
JOIN dim_customer c ON s.Customer_ID = c.Customer_ID
JOIN fact_orders o ON s.Customer_ID = o.Customer_ID
WHERE s.Status = 'Pending'
GROUP BY c.Country;
```

Country	TotalAmountSpent
1 UAE	53800
2 UK	136300
3 USA	65500

2. Total number of transactions, total quantity sold, and total amount spent for each customer, along with the product details

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the server structure for 'LAPTOP-B4HM0U40'. The central pane shows a SQL query in the 'SQLQuery2.sql' file. The query is designed to provide a detailed view of customer transactions, including the number of transactions, total quantity sold, and total amount spent, along with the customer's name and the product details. The results pane at the bottom shows the output of the query.

```
-- Total number of transactions, total quantity sold, and total amount spent for each customer, along with the product details
SELECT
    c.Customer_ID,
    c.FirstName,
    c.LastName,
    p.Item,
    COUNT(o.Order_ID) AS TotalTransactions,
    SUM(o.Quantity) AS TotalQuantitySold,
    SUM(o.Amount) AS TotalAmountSpent
FROM fact_orders o
JOIN dim_customer c ON o.Customer_ID = c.Customer_ID
JOIN dim_item p ON o.Item_ID = p.Item_ID
GROUP BY c.Customer_ID, c.FirstName, c.LastName, p.Item
ORDER BY c.Customer_ID;
```

Customer_ID	FirstName	LastName	Item	TotalTransactions	TotalQuantitySold	TotalAmountSpent
1	Eric	Carler	Mousepad	1	1	200
2	William	Jackson	DDR RAM	1	1	1500
3	Jason	Montgomery	DDR RAM	1	1	1500
4	Jason	Montgomery	Mousepad	2	2	450
5	Jason	Montgomery	Webcam	1	1	350
6	Danell	Dillon	Keyboard	1	1	400
7	Jodi	Gonzalez	Headlink	1	1	5000
8	Omar	Martin	Headset	1	1	900
9	Omar	Martin	Keyboard	1	1	400
10	Omar	Martin	Monitor	1	1	12000
11	Jason	Brown	Webcam	1	1	350
12	Kimberly	Mora	Webcam	1	1	350
13	Derek	Peterson	DDR RAM	1	1	1500
14	Margaret	Hardy	Keyboard	2	2	800
15	Margaret	Hardy	Monitor	1	1	12000
16	Arthur	Hayes	Keyboard	1	1	400
17	Arthur	Hayes	Mousepad	1	1	250

### 3. The maximum product purchased for each country

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'PEISalesDB'. The central pane shows a SQL query titled 'The maximum product purchased for each country'. The query uses a Common Table Expression (CTE) named 'product\_country\_sales' to calculate the total quantity for each country and item, and then a window function 'ROW\_NUMBER()' to rank them by total quantity in descending order. The final result set shows the top product for each country.

```
-- The maximum product purchased for each country

WITH product_country_sales AS (
    SELECT
        c.Country,
        p.Item,
        SUM(o.Quantity) AS TotalQuantity
    FROM fact_orders o
    JOIN dim_customer c ON o.Customer_ID = c.Customer_ID
    JOIN dim_item p ON o.Item_ID = p.Item_ID
    GROUP BY c.Country, p.Item
),
ranked_products AS (
    SELECT *,
        ROW_NUMBER() OVER (PARTITION BY Country ORDER BY TotalQuantity DESC) AS rn
    FROM product_country_sales
)
SELECT Country, Item, TotalQuantity
FROM ranked_products
WHERE rn = 1;
```

Country	Item	TotalQuantity
1 UAE	Keyboard	12
2 UK	Mousepad	24
3 USA	Mousepad	18

### 4. The most purchased product based on the age category: less than 30 and 30 or above

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'PEISalesDB'. The central pane shows a SQL query titled 'The most purchased product based on the age category: less than 30 and 30 or above'. The query uses a Common Table Expression (CTE) named 'product\_age\_sales' to calculate the total quantity for each age category and item, and then a window function 'ROW\_NUMBER()' to rank them by total quantity in descending order. The final result set shows the top product for each age category.

```
--The most purchased product based on the age category: less than 30 and 30 or above

WITH product_age_sales AS (
    SELECT
        c.AgeCategory,
        p.Item,
        SUM(o.Quantity) AS TotalQuantity
    FROM fact_orders o
    JOIN dim_customer c ON o.Customer_ID = c.Customer_ID
    JOIN dim_item p ON o.Item_ID = p.Item_ID
    GROUP BY c.AgeCategory, p.Item
),
ranked_age_products AS (
    SELECT *,
        ROW_NUMBER() OVER (PARTITION BY AgeCategory ORDER BY TotalQuantity DESC) AS rn
    FROM product_age_sales
)
SELECT AgeCategory, Item, TotalQuantity
FROM ranked_age_products
WHERE rn = 1;
```

AgeCategory	Item	TotalQuantity
1 Above 30	Keyboard	35
2 Less than 30	Mousepad	17

## 5. The country that had the minimum transactions and sales amount

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' for a server named 'LAPTOP-B4HM0U4O'. The right pane shows a query window with the following SQL code:

```
--The country that had the minimum transactions and sales amount

SELECT TOP 1
    c.Country,
    COUNT(o.Order_ID) AS TotalTransactions,
    SUM(o.Amount) AS TotalSalesAmount
FROM fact_orders o
JOIN dim_customer c ON o.Customer_ID = c.Customer_ID
GROUP BY c.Country
ORDER BY
    COUNT(o.Order_ID) ASC,
    SUM(o.Amount) ASC;
```

Below the query window, the 'Results' pane shows the output of the query:

Country	TotalTransactions	TotalSalesAmount
1 UAE	40	49950