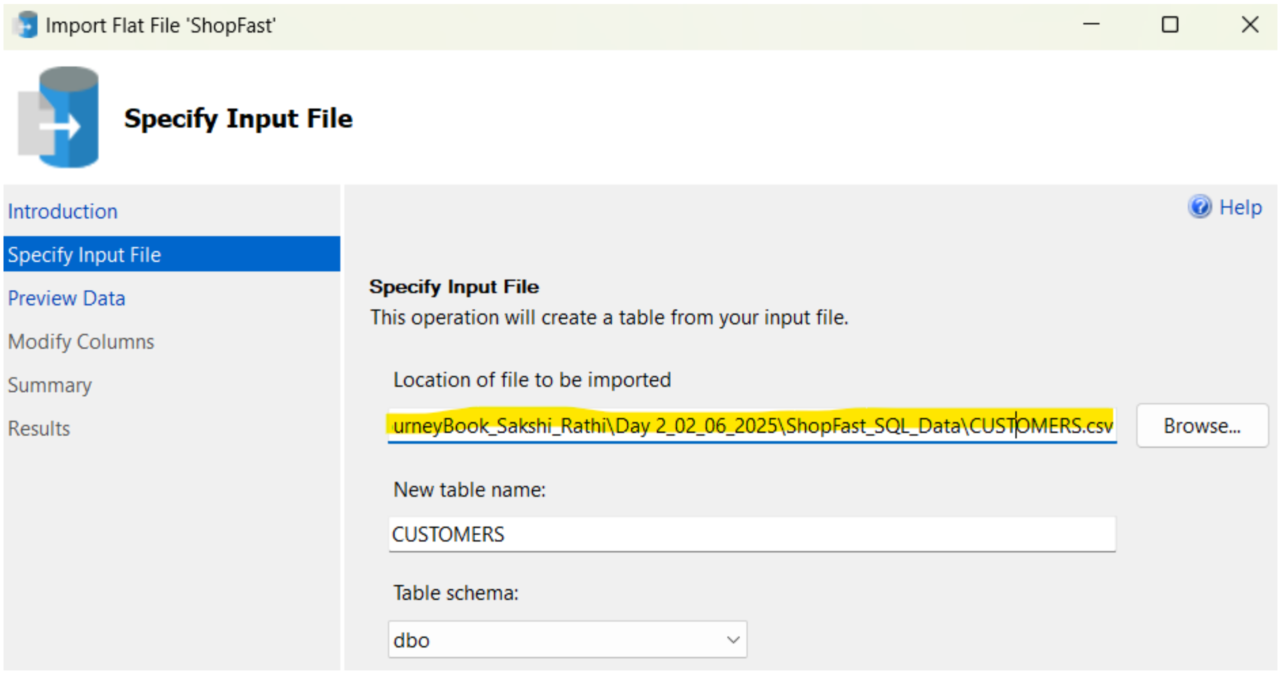
PROJECT – SHOPFAST ECOMMERCE SQL ANALYSIS SCENARIO

– 02nd June 2025

DB CREATION

```
CREATE DATABASE ShopFast;
GO
USE ShopFast;
```

CSV File Import


Import Flat File 'ShopFast'

Specify Input File

Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results

Help

Specify Input File
This operation will create a table from your input file.

Location of file to be imported
urneyBook_Sakshi_Rathi\Day 2_02_06_2025\ShopFast_SQL_Data\CUSTOMERS.csv

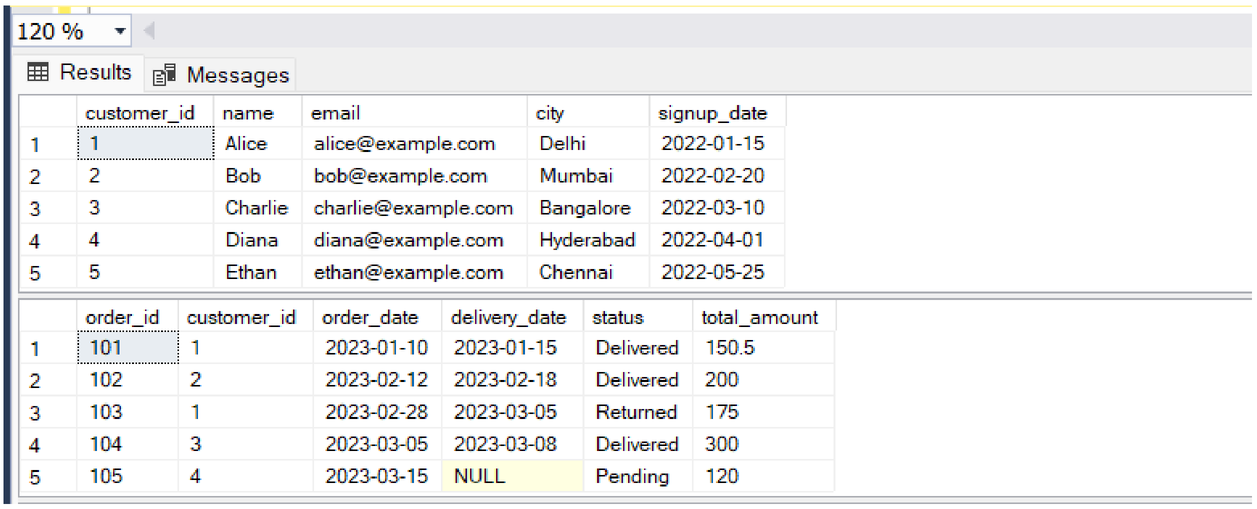
Browse...

New table name:
CUSTOMERS

Table schema:
dbo

Imported tables in MSSQL Server

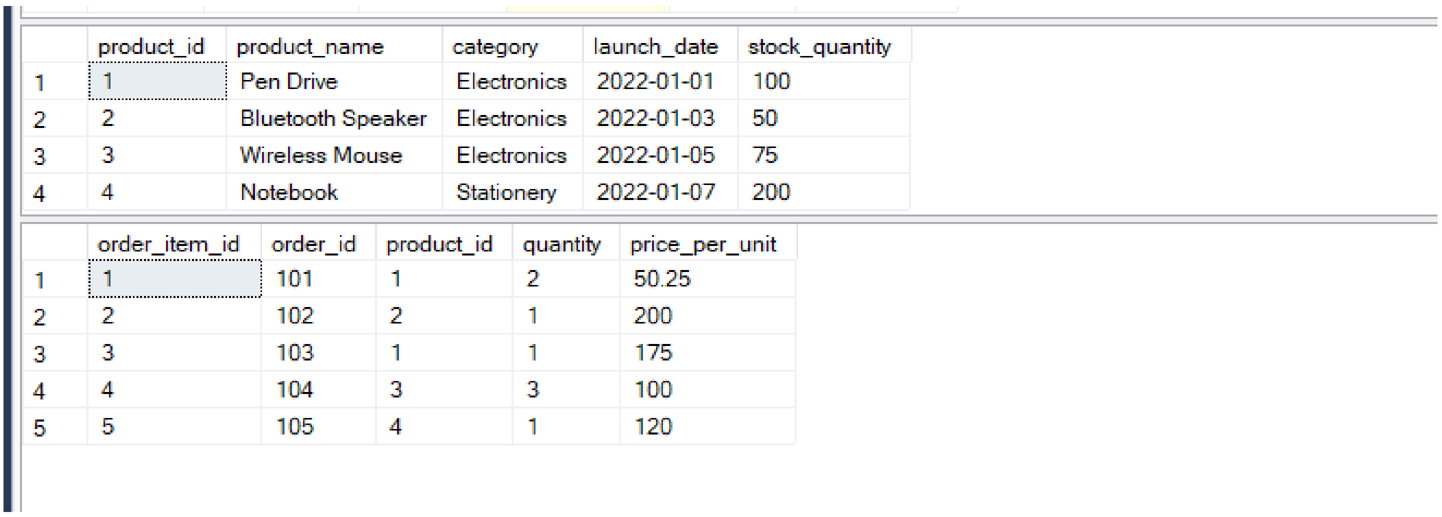
```
SELECT * FROM CUSTOMERS;
SELECT * FROM ORDERS;
SELECT * FROM PRODUCTS;
SELECT * FROM ORDER_ITEMS;
```


120 %

Results | Messages

	customer_id	name	email	city	signup_date
1	1	Alice	alice@example.com	Delhi	2022-01-15
2	2	Bob	bob@example.com	Mumbai	2022-02-20
3	3	Charlie	charlie@example.com	Bangalore	2022-03-10
4	4	Diana	diana@example.com	Hyderabad	2022-04-01
5	5	Ethan	ethan@example.com	Chennai	2022-05-25

	order_id	customer_id	order_date	delivery_date	status	total_amount
1	101	1	2023-01-10	2023-01-15	Delivered	150.5
2	102	2	2023-02-12	2023-02-18	Delivered	200
3	103	1	2023-02-28	2023-03-05	Returned	175
4	104	3	2023-03-05	2023-03-08	Delivered	300
5	105	4	2023-03-15	NULL	Pending	120

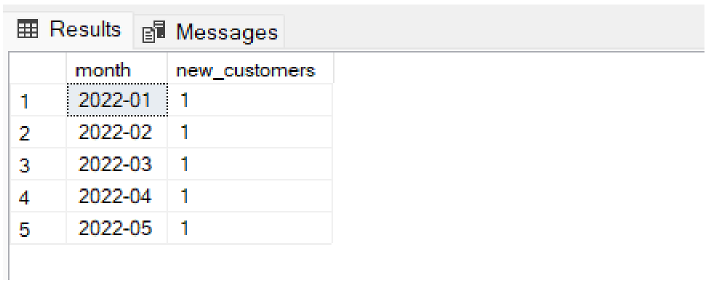


	product_id	product_name	category	launch_date	stock_quantity
1	1	Pen Drive	Electronics	2022-01-01	100
2	2	Bluetooth Speaker	Electronics	2022-01-03	50
3	3	Wireless Mouse	Electronics	2022-01-05	75
4	4	Notebook	Stationery	2022-01-07	200

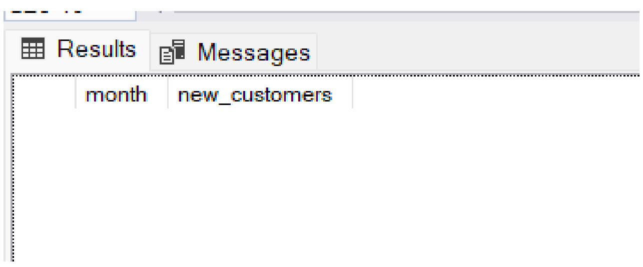
	order_item_id	order_id	product_id	quantity	price_per_unit
1	1	101	1	2	50.25
2	2	102	2	1	200
3	3	103	1	1	175
4	4	104	3	3	100
5	5	105	4	1	120

SQL QUERIES FOR PROBLEM SET

```
--1. Customer Sign-up Trend: New customers per month (last 12
months)
SELECT FORMAT(signup_date, 'yyyy-MM') AS month, COUNT(*) AS
new_customers
FROM CUSTOMERS
WHERE signup_date >= DATEADD(MONTH, -12, year(2023)) --getdate()
GROUP BY FORMAT(signup_date, 'yyyy-MM')
ORDER BY month;
```



	month	new_customers
1	2022-01	1
2	2022-02	1
3	2022-03	1
4	2022-04	1
5	2022-05	1



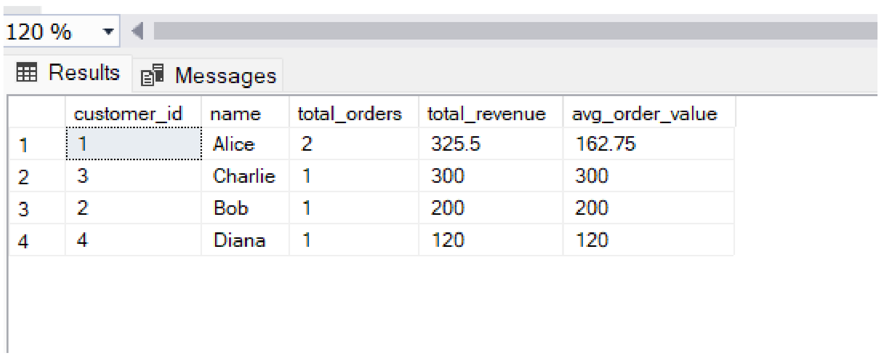
month	new_customers

Output 1 – With 2023 Year

Output 2 – With GETDATE() Curretnt year 2025

```
--2. Top 5 Customers by Revenue
SELECT TOP 5 c.customer_id, c.name, COUNT(o.order_id) AS
total_orders,
       SUM(o.total_amount) AS total_revenue,
       AVG(o.total_amount) AS avg_order_value
FROM CUSTOMERS c
JOIN ORDERS o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_revenue DESC;
```

Output -

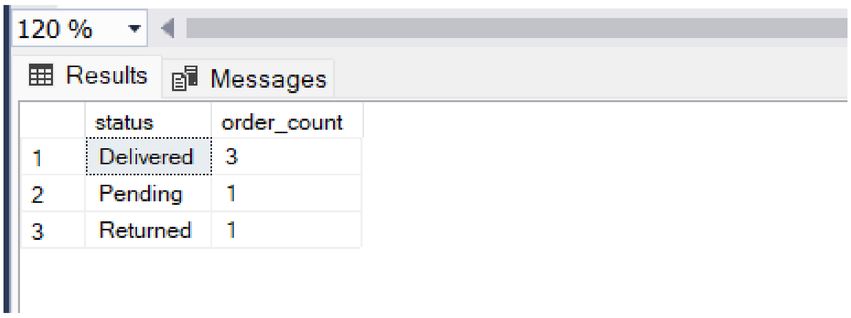

120 %

Results | Messages

	customer_id	name	total_orders	total_revenue	avg_order_value
1	1	Alice	2	325.5	162.75
2	3	Charlie	1	300	300
3	2	Bob	1	200	200
4	4	Diana	1	120	120

```
--3. Order Status Distribution
SELECT status, COUNT(*) AS order_count
FROM ORDERS
GROUP BY status;
```

Ouput –

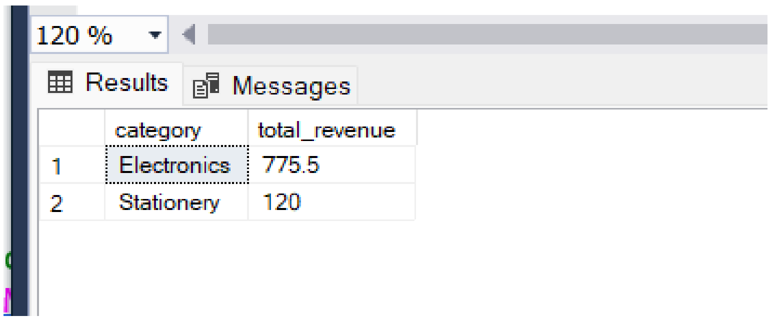

120 %

Results | Messages

	status	order_count
1	Delivered	3
2	Pending	1
3	Returned	1

```
--4. Revenue by Category
SELECT p.category, SUM(oi.quantity * oi.price_per_unit) AS
total_revenue
FROM ORDER_ITEMS oi
JOIN PRODUCTS p ON oi.product_id = p.product_id
GROUP BY p.category
ORDER BY total_revenue DESC;
```

Output –

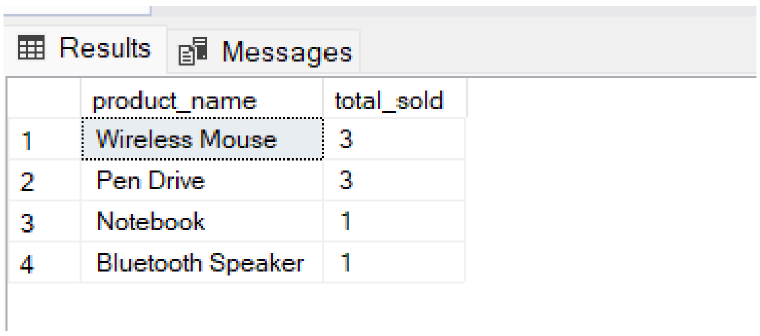

120 %

Results | Messages

	category	total_revenue
1	Electronics	775.5
2	Stationery	120

```sql
--5. Best-Selling Products (Top 5 by quantity sold)
SELECT TOP 5 p.product_name, SUM(oi.quantity) AS total_sold
FROM ORDER_ITEMS oi
JOIN PRODUCTS p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sold DESC;
```

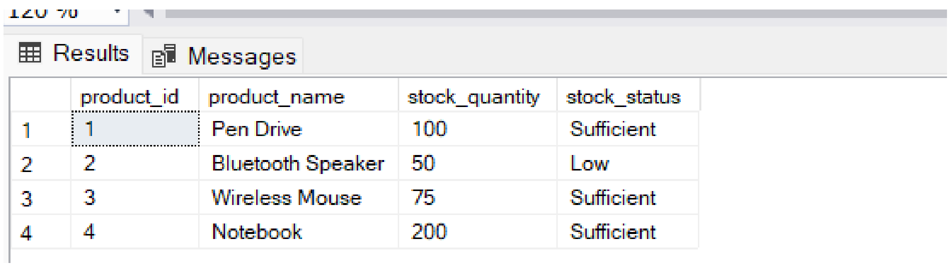
Output –



	product_name	total_sold
1	Wireless Mouse	3
2	Pen Drive	3
3	Notebook	1
4	Bluetooth Speaker	1

```sql
--6. Low-Stock Products (<10% stock using CASE)
SELECT product_id, product_name, stock_quantity,
       CASE
           WHEN stock_quantity < 10 THEN 'Critical Low'
           WHEN stock_quantity BETWEEN 10 AND 50 THEN 'Low'
           ELSE 'Sufficient'
       END AS stock_status
FROM PRODUCTS;
```

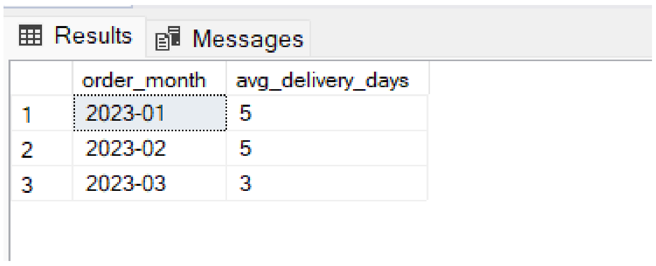
Output –



	product_id	product_name	stock_quantity	stock_status
1	1	Pen Drive	100	Sufficient
2	2	Bluetooth Speaker	50	Low
3	3	Wireless Mouse	75	Sufficient
4	4	Notebook	200	Sufficient

```sql
--7. Avg Delivery Time per Month
SELECT FORMAT(order_date, 'yyyy-MM') AS order_month,
       AVG(DATEDIFF(DAY, order_date, delivery_date)) AS
avg_delivery_days
FROM ORDERS
WHERE delivery_date IS NOT NULL
GROUP BY FORMAT(order_date, 'yyyy-MM')
ORDER BY order_month;
```

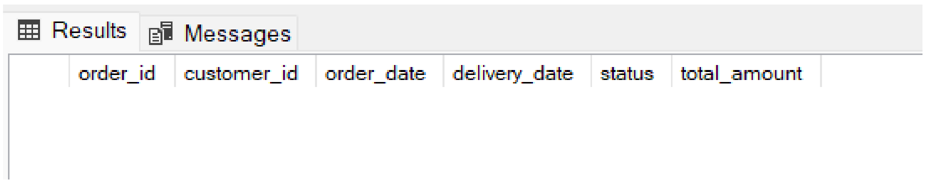
Output –



	order_month	avg_delivery_days
1	2023-01	5
2	2023-02	5
3	2023-03	3

```sql
--8. Orders with Delivery >7 days
SELECT *
FROM ORDERS
WHERE DATEDIFF(DAY, order_date, delivery_date) > 7;
```

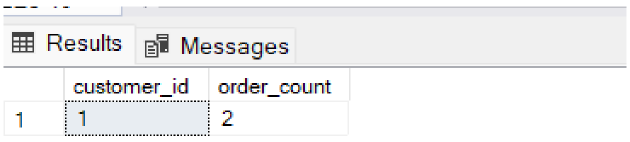
Output (No orders with 7 days date difference)-



order_id	customer_id	order_date	delivery_date	status	total_amount

```sql
--9. Repeat Customers
SELECT customer_id, COUNT(*) AS order_count
FROM ORDERS
GROUP BY customer_id
HAVING COUNT(*) > 1;
```

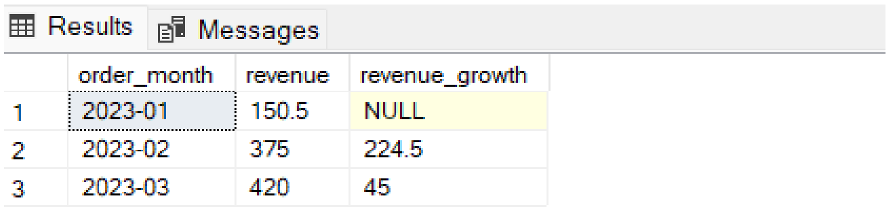
Output –



	customer_id	order_count
1	1	2

```sql
--10. Monthly Revenue Growth with LAG()
WITH MonthlyRevenue AS (
    SELECT FORMAT(order_date, 'yyyy-MM') AS order_month,
           SUM(total_amount) AS revenue
    FROM ORDERS
    GROUP BY FORMAT(order_date, 'yyyy-MM')
)
SELECT order_month, revenue,
       revenue - LAG(revenue) OVER (ORDER BY order_month) AS
revenue_growth
FROM MonthlyRevenue;
```

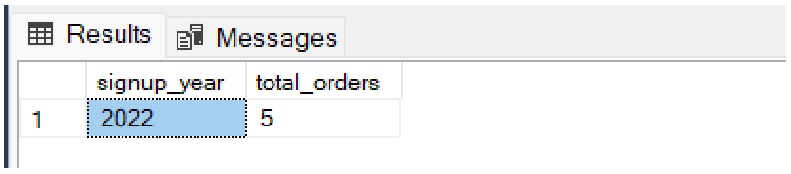
Output –



	order_month	revenue	revenue_growth
1	2023-01	150.5	NULL
2	2023-02	375	224.5
3	2023-03	420	45

```
--11. Cohort Analysis using CTE (signup year)
WITH Cohorts AS (
    SELECT customer_id, YEAR(signup_date) AS signup_year
    FROM CUSTOMERS
)
SELECT c.signup_year, COUNT(o.order_id) AS total_orders
FROM Cohorts c
JOIN ORDERS o ON c.customer_id = o.customer_id
GROUP BY c.signup_year;
```

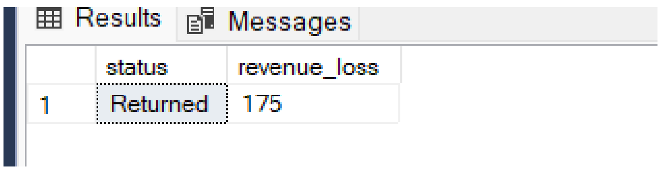
Output –



	signup_year	total_orders
1	2022	5

```
--12. Cancelled/Returned Product Revenue Loss
SELECT status, SUM(total_amount) AS revenue_loss
FROM ORDERS
WHERE status IN ('Cancelled', 'Returned')
GROUP BY status;
```

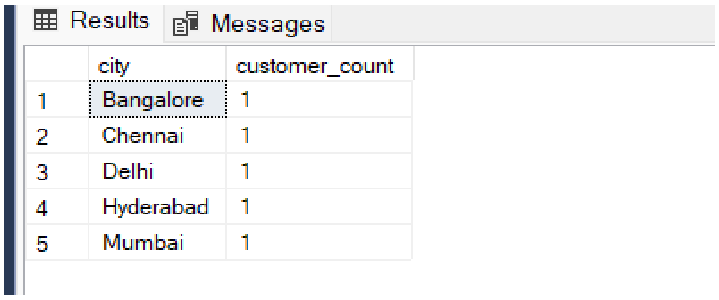
Output –



	status	revenue_loss
1	Returned	175

```
--13. Customer City Heatmap
SELECT city, COUNT(*) AS customer_count
FROM CUSTOMERS
GROUP BY city
ORDER BY customer_count DESC;
```

Output –



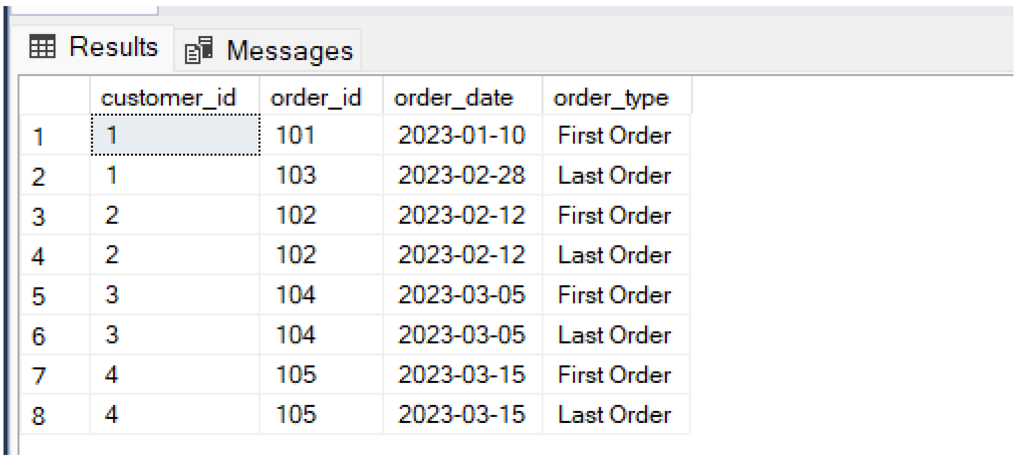
	city	customer_count
1	Bangalore	1
2	Chennai	1
3	Delhi	1
4	Hyderabad	1
5	Mumbai	1

```
--14. First & Last Order per Customer using ROW_NUMBER()
WITH OrderedData AS (
    SELECT customer_id, order_id, order_date,
           ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY 
order_date ASC) AS rn_asc,
           ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY 
order_date DESC) AS rn_desc
    FROM ORDERS
)
SELECT customer_id, order_id, order_date, 'First Order' AS 
order_type
FROM OrderedData WHERE rn_asc = 1

UNION ALL

SELECT customer_id, order_id, order_date, 'Last Order' AS order_type
FROM OrderedData WHERE rn_desc = 1
ORDER BY customer_id, order_type;
```

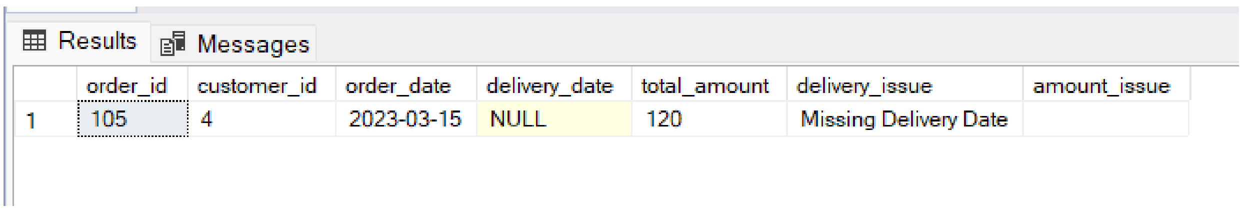
Output –



	customer_id	order_id	order_date	order_type
1	1	101	2023-01-10	First Order
2	1	103	2023-02-28	Last Order
3	2	102	2023-02-12	First Order
4	2	102	2023-02-12	Last Order
5	3	104	2023-03-05	First Order
6	3	104	2023-03-05	Last Order
7	4	105	2023-03-15	First Order
8	4	105	2023-03-15	Last Order

```
-- 15. NULL Handling: Orders with missing delivery date or total
amount
SELECT order_id, customer_id, order_date, delivery_date,
total_amount,
       CASE WHEN delivery_date IS NULL THEN 'Missing Delivery Date'
ELSE '' END AS delivery_issue,
       CASE WHEN total_amount IS NULL THEN 'Missing Amount' ELSE ''
END AS amount_issue
FROM ORDERS
WHERE delivery_date IS NULL OR total_amount IS NULL;
```

Output –



	order_id	customer_id	order_date	delivery_date	total_amount	delivery_issue	amount_issue
1	105	4	2023-03-15	NULL	120	Missing Delivery Date	