**Amazon Advance SQL Data Analysis Project**

**Task 1:**

**In the Orders table, the order\_id column was mistakenly defined as a primary key with the**

**MONEY data type instead of INT. Update the order\_id column to use the INT data type.**

**Step 1: Drop the primary key constraint**

ALTER TABLE Orders DROP CONSTRAINT PK\_orders;

**Step 2: Change the data type of the column**

ALTER TABLE Orders ALTER COLUMN order\_id INT NOT NULL;

**Step 3: Re-add the primary key constraint**

ALTER TABLE orders ADD CONSTRAINT pk\_orders PRIMARY KEY (order\_id);

**Adding Foreign Key constraint to Tables**

ALTER TABLE Products

ADD CONSTRAINT fk\_category\_id

FOREIGN KEY (category\_id)

REFERENCES Category (category\_id);

ALTER TABLE Orders

ADD CONSTRAINT fk\_customer\_id

FOREIGN KEY (customer\_id)

REFERENCES Customers (Customer\_id);

ALTER TABLE Orders

ADD CONSTRAINT fk\_seller\_id

FOREIGN KEY (seller\_id)

REFERENCES Sellers (seller\_id);

ALTER TABLE Order\_items

ADD CONSTRAINT fk\_order\_id

FOREIGN KEY (order\_id)

REFERENCES Orders (order\_id);

ALTER TABLE Order\_items

ADD CONSTRAINT fk\_Product\_id

FOREIGN KEY (product\_id)

REFERENCES Products (product\_id);

ALTER TABLE Payments

ADD CONSTRAINT fk\_order\_id\_payment

FOREIGN KEY (order\_id)

REFERENCES Orders (order\_id);

ALTER TABLE Shipping

ADD CONSTRAINT fk\_order\_id\_shipping

FOREIGN KEY (order\_id)

REFERENCES Orders (order\_id);

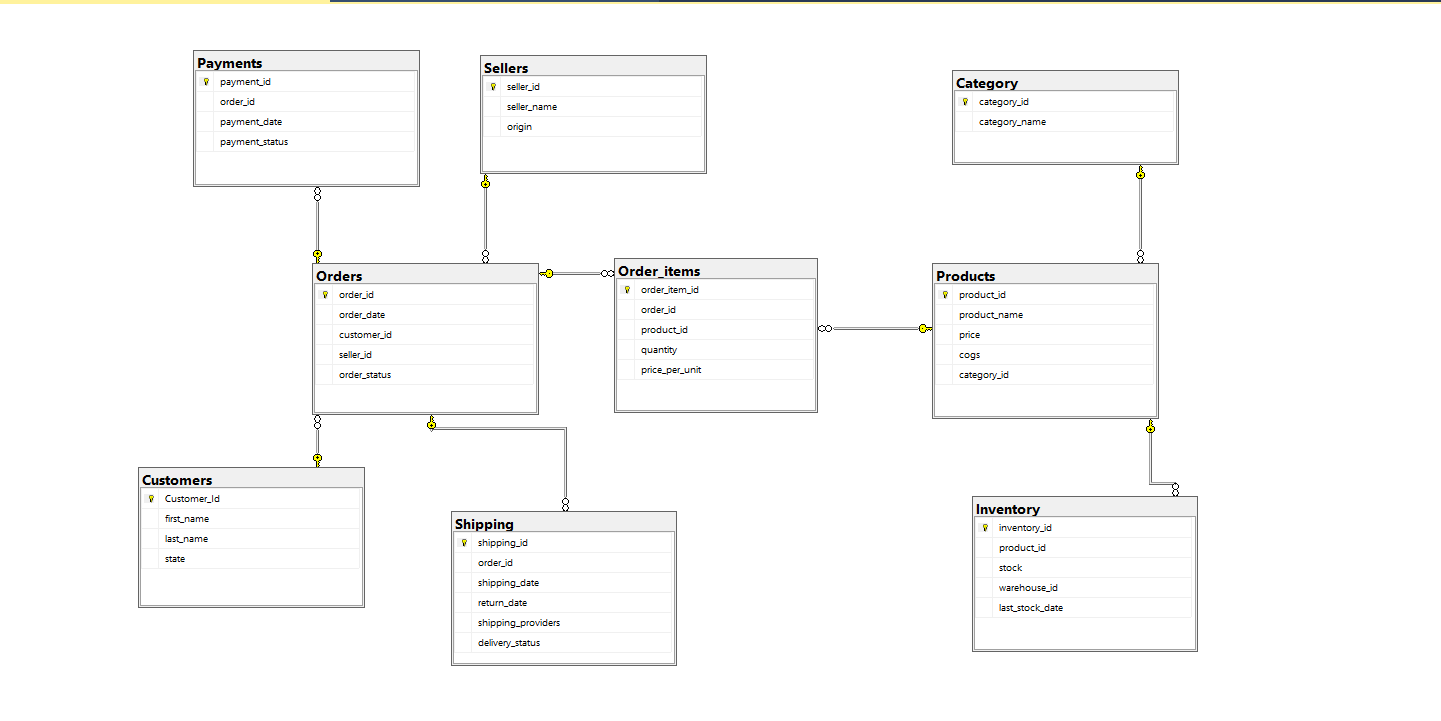
ALTER TABLE Inventory

ADD CONSTRAINT fk\_product\_Inventory

FOREIGN KEY (product\_id)

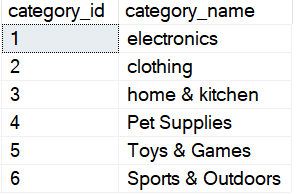
REFERENCES Products (product\_id);

\*/

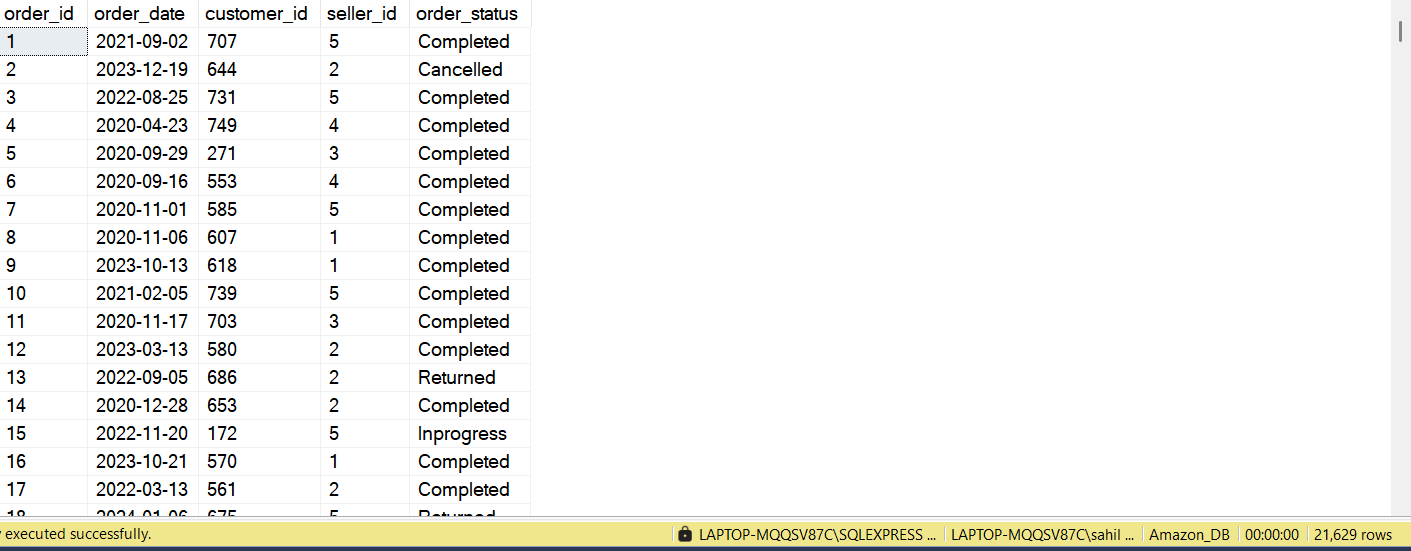


**EXPLORATORY DATA ANALYSIS**

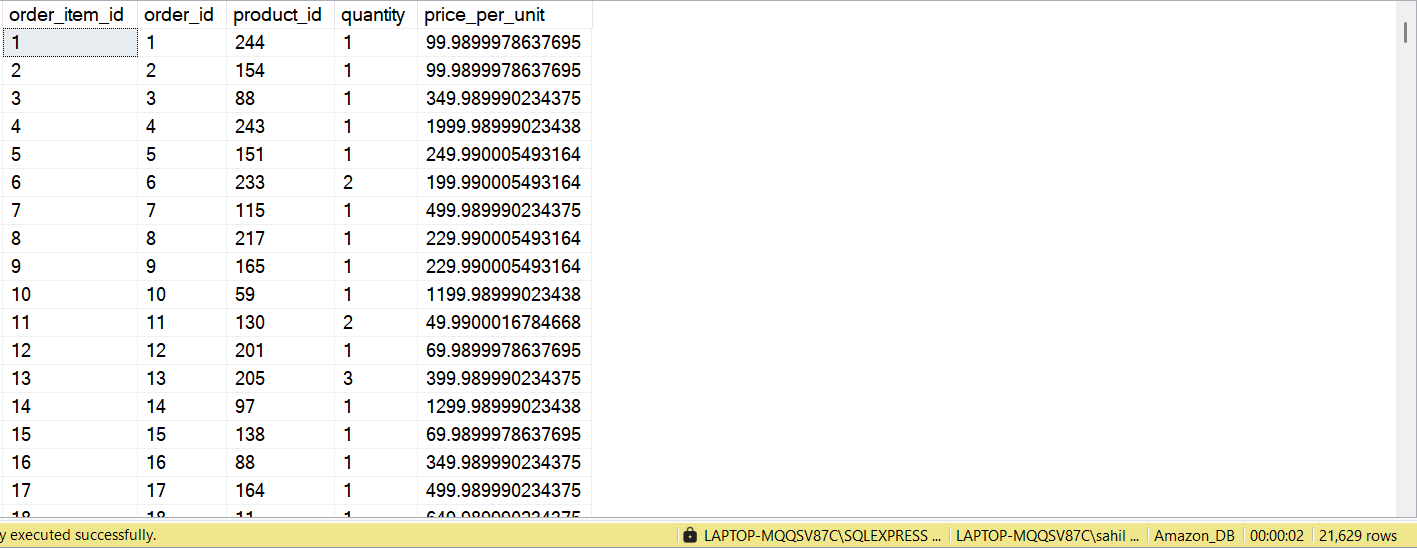
SELECT \* FROM category;



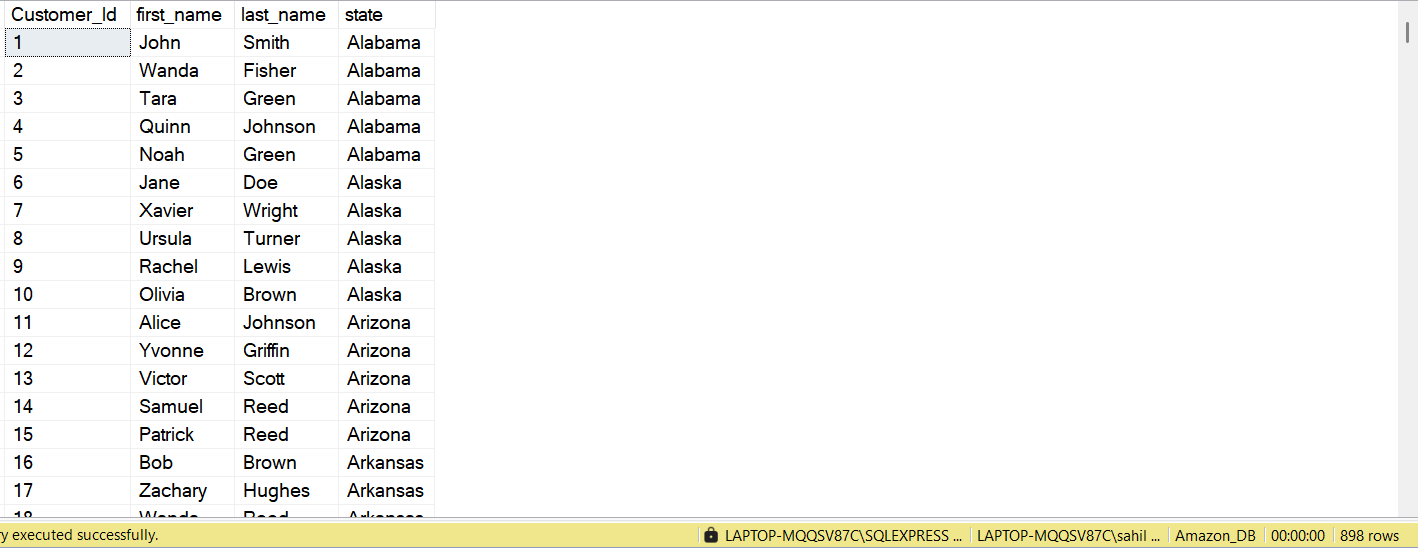
SELECT \* FROM Orders;



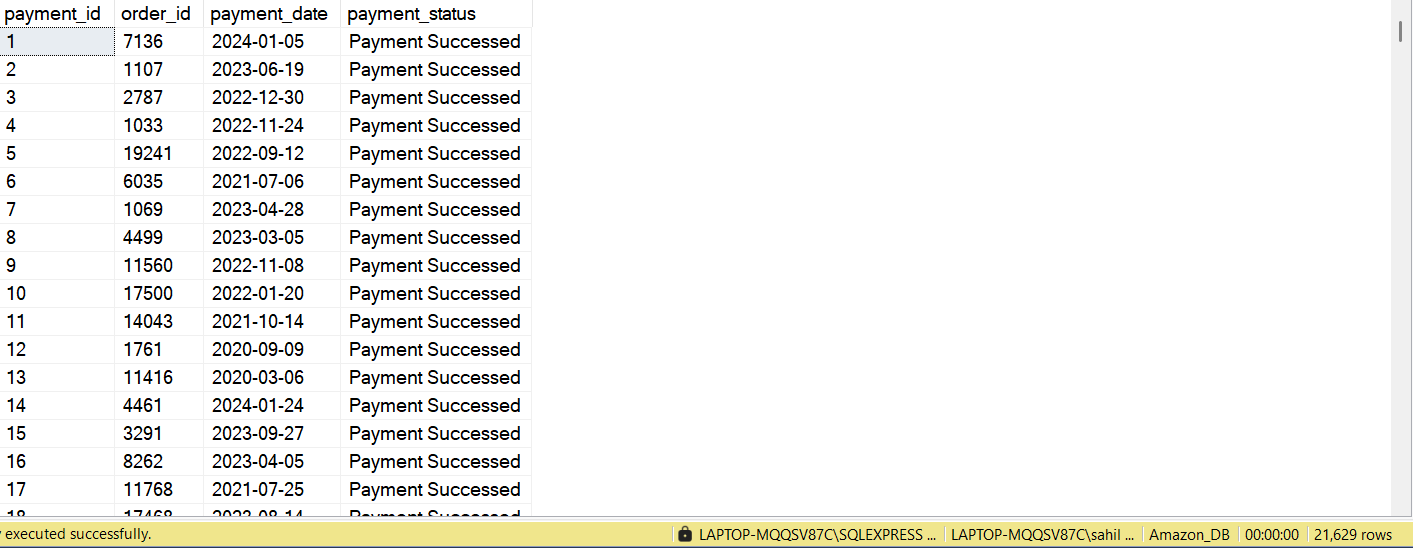
SELECT \* FROM Order\_items;



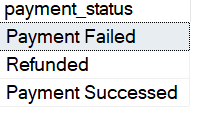
SELECT \* FROM Customers;



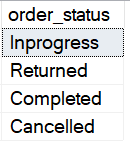
SELECT \* FROM Payments;



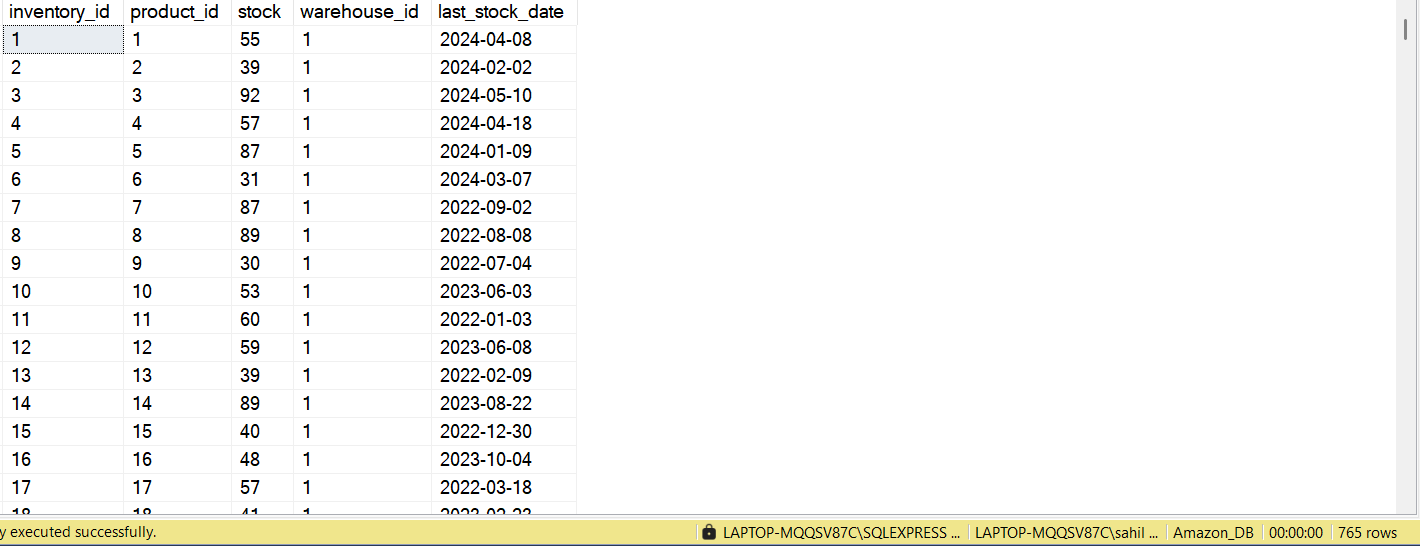
SELECT DISTINCT payment\_status FROM Payments;



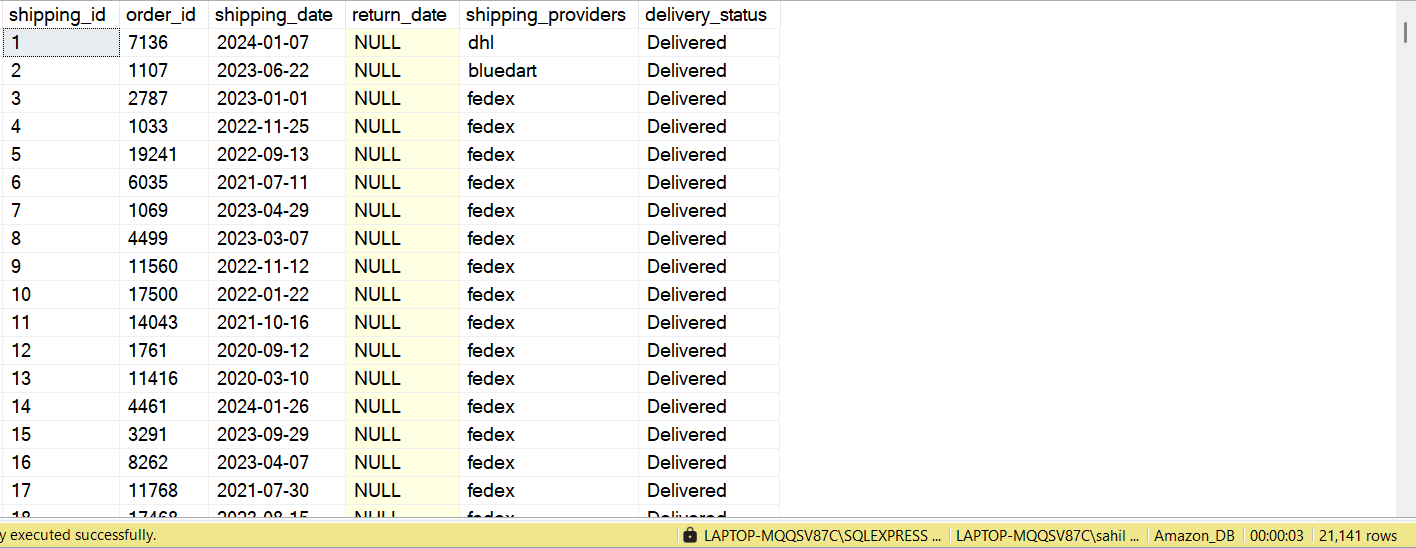
SELECT DISTINCT order\_status FROM Orders;



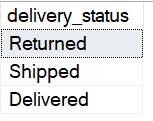
SELECT \* FROM Inventory;



SELECT \* FROM Shipping;

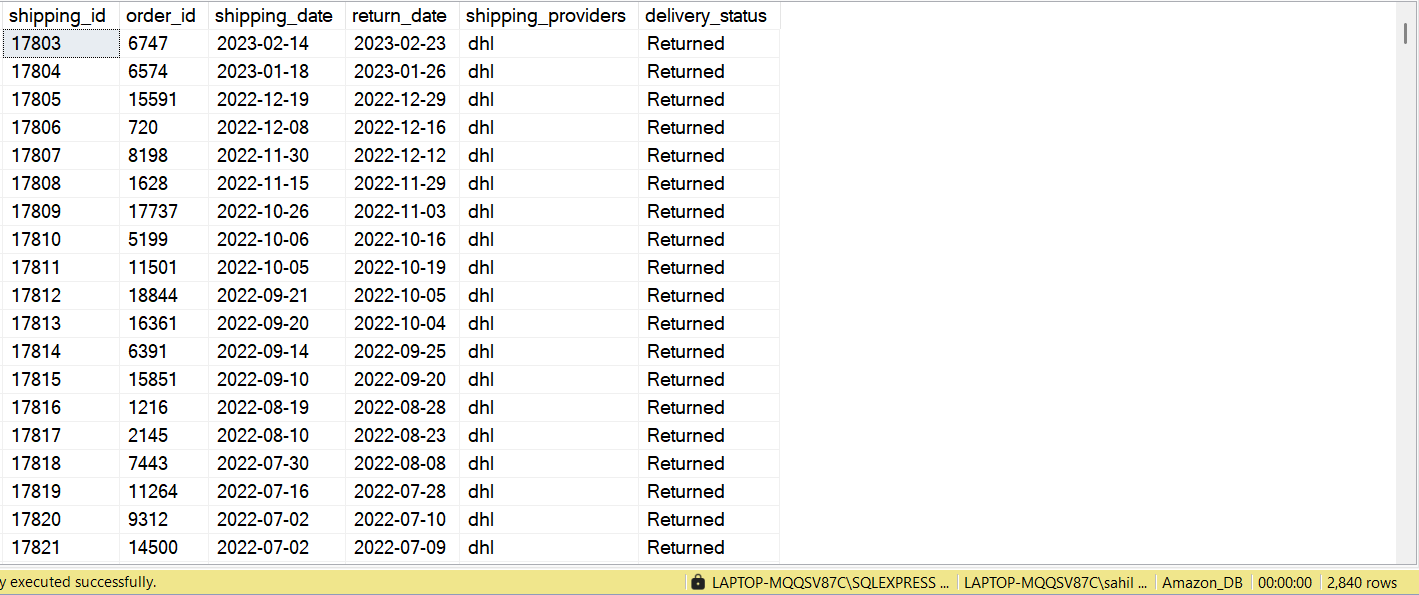


SELECT DISTINCT delivery\_status FROM Shipping;



SELECT \* FROM Shipping

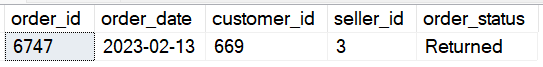
WHERE return\_date IS NOT NULL



**This means 2,840 products were returned**

SELECT \* FROM Orders

WHERE order\_id= 6747



**Here you can verify that Order\_id with 6747 which has return date in shipping table also has Returned status in Orders table**

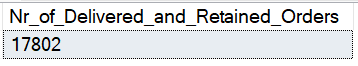
**This is how we have verified that data follows the Integrity**

SELECT COUNT(\*) Nr\_of\_Delivered\_and\_Retained\_Orders

FROM Shipping

WHERE return\_date IS NULL

AND delivery\_status = 'delivered';



**Task 2 : Top Selling Products**

**Query the Top 10 Products by Total sales value.**

**Challenge: Include Product name, total quantity sold, and total sales value**

To solve this Task first I added Sales column by doing following steps and then solve the Question

ALTER TABLE Order\_items

ADD Sales FLOAT;

UPDATE Order\_items

SET Sales = quantity \* price\_per\_unit;

WITH Top\_Product\_by\_Sales

AS

(

SELECT

oi.product\_id,

p.product\_name,

SUM(oi.quantity) Total\_Quantity\_Sold,

CAST(SUM(Sales) as decimal(10,2)) Total\_Sales

FROM Orders o

LEFT JOIN Order\_items oi

ON o.order\_id= oi.order\_id

LEFT JOIN Products p

ON oi.product\_id = p.product\_id

GROUP BY

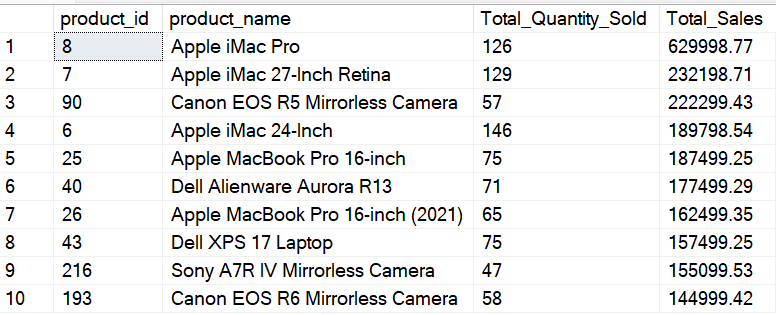
oi.product\_id,

p.product\_name

)

SELECT TOP 10 \* FROM Top\_Product\_by\_Sales

ORDER BY Total\_Sales DESC;



**Task 3 Revenue by Category**

**Calculate total revenue generated by each product category.**

**Challenge: Include the percentage contribution of each category to total revenue**

SELECT

p.category\_id,

c.category\_name,

CAST(SUM(oi.Sales) as decimal(10,2)) as Total\_Sales\_by\_Category,

CAST(SUM(oi.Sales)/(SELECT SUM(Sales) FROM Order\_items) \* 100 as decimal(10,2)) Percent\_Contribution

FROM Order\_items oi

LEFT JOIN Products p

ON oi.product\_id = p.product\_id

LEFT JOIN Category c

ON c.category\_id = p.category\_id

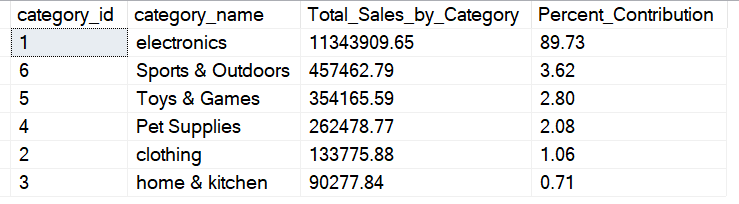
Group by

p.category\_id,

c.category\_name

ORDER BY

CAST(SUM(oi.Sales) as decimal(10,2)) DESC



**Task 4 : Average Order Value (AOV)**

**Compute the average Order value for each customer**

**Challenge: Include Only the Customers with more than 5 Orders**

SELECT

c.Customer\_Id,

CONCAT(c.first\_name,' ', c.last\_name) Customer\_Name,

ROUND(SUM(oi.Sales)/COUNT(o.order\_id),2) Avg\_Order\_Value,

COUNT(o.order\_id) Nr\_of\_Orders

FROM Orders o

LEFT JOIN Order\_items oi

ON oi.order\_id = o.order\_id

LEFT JOIN Customers c

ON o.customer\_id = c.Customer\_Id

GROUP BY

c.Customer\_Id,

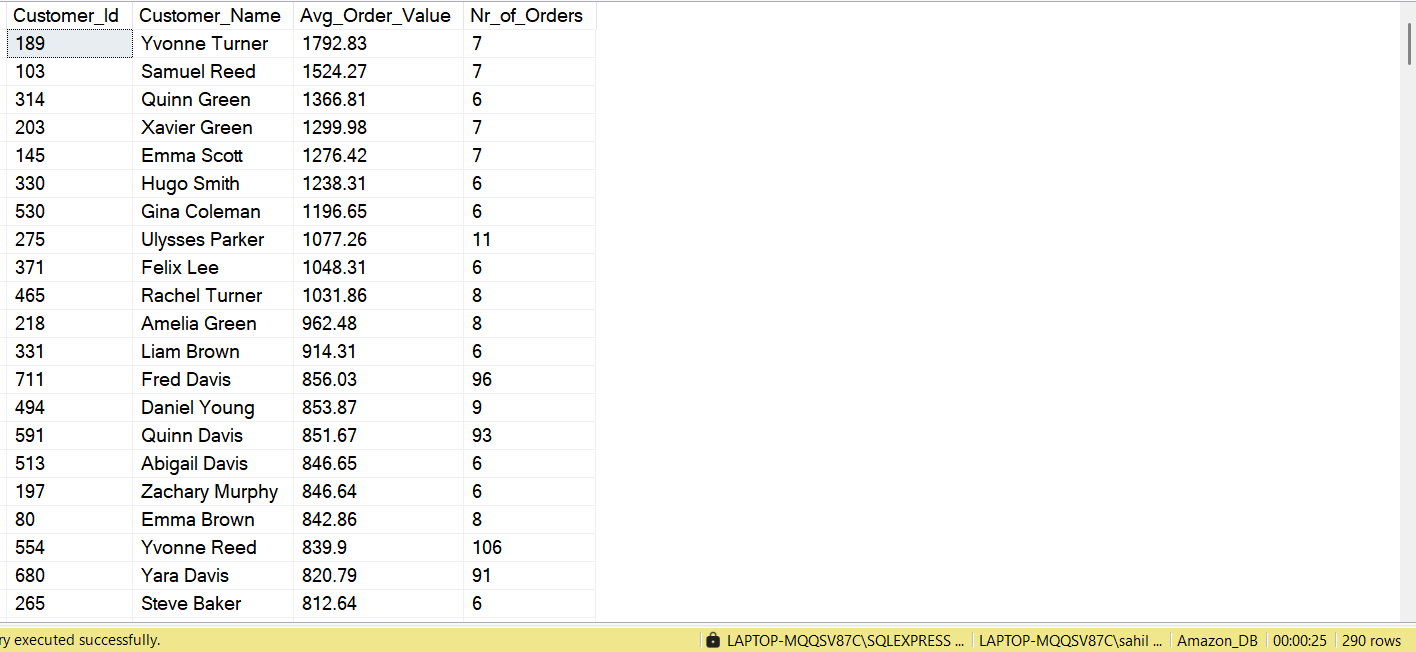
c.first\_name,

c.last\_name

HAVING COUNT(o.order\_id) > 5

ORDER BY

SUM(oi.Sales)/COUNT(o.order\_id) DESC



**Task 5 Monthly Sales Trend**

**Query Monthly Total Sales Over the Past year.**

**Challenge: Display the sales trend, grouping by month, return current\_month\_sales, Last Month Sales**

WITH Monthly\_Trend\_2023

AS

(

SELECT

YEAR(order\_date) Year,

MONTH(order\_date) Month\_Num,

DATENAME(MONTH, o.order\_date) Month\_Name,

ROUND(SUM(Sales),2) Current\_Month\_Sales,

ROUND(LAG(SUM(Sales)) OVER(ORDER BY MONTH(order\_date)),2) Prev\_Month\_Sales

FROM Orders o

JOIN Order\_items oi

ON o.order\_id = oi.order\_id

WHERE YEAR(order\_date) = 2023

GROUP BY

YEAR(order\_date),

MONTH(order\_date) ,

DATENAME(MONTH, o.order\_date)

)

SELECT

Year,

Month\_Name,

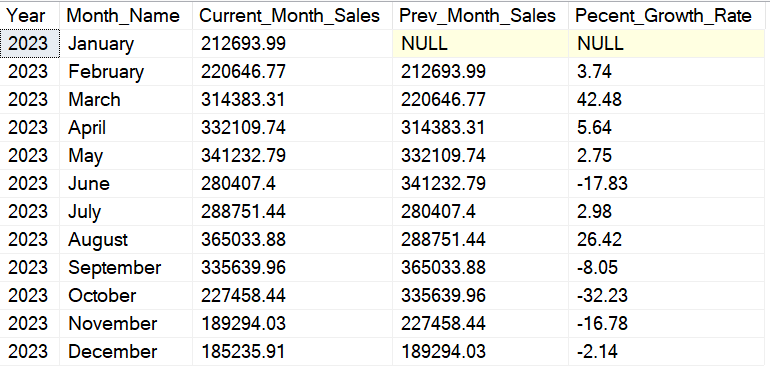
Current\_Month\_Sales,

Prev\_Month\_Sales,

ROUND((Current\_Month\_Sales - Prev\_Month\_Sales)/Prev\_Month\_Sales \* 100,2) Pecent\_Growth\_Rate

FROM Monthly\_Trend\_2023

ORDER BY Year, Month\_Num



-- Here I have gone One step further and applied Monthly trend for all four years

WITH Monthly\_Trend AS

(

SELECT

YEAR(order\_date) AS Year,

MONTH(order\_date) AS Month\_Num,

DATENAME(MONTH, order\_date) AS Month\_Name,

ROUND(SUM(Sales), 2) AS Current\_Month\_Sales,

LAG(SUM(Sales)) OVER (ORDER BY YEAR(order\_date), MONTH(order\_date)) AS Prev\_Month\_Sales

FROM Orders o

JOIN Order\_items oi

ON o.order\_id = oi.order\_id

GROUP BY

YEAR(order\_date),

MONTH(order\_date),

DATENAME(MONTH, order\_date)

)

SELECT

Year,

Month\_Name,

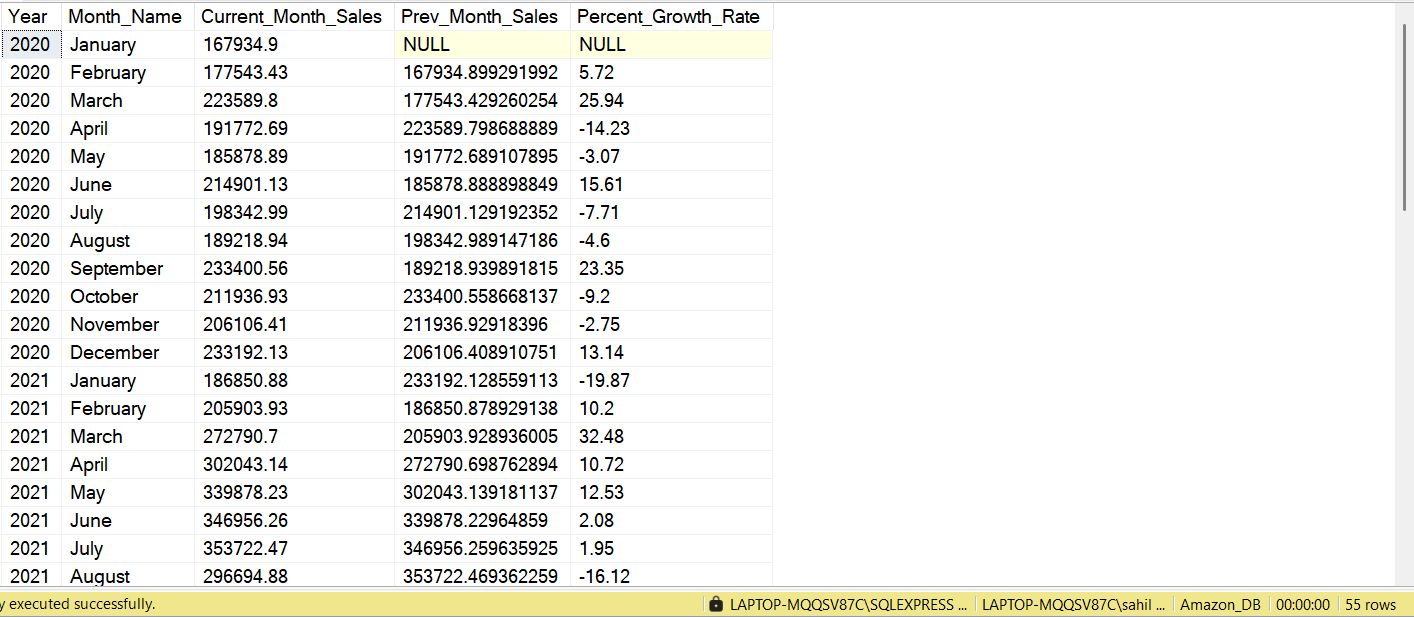
Current\_Month\_Sales,

Prev\_Month\_Sales,

ROUND((Current\_Month\_Sales - Prev\_Month\_Sales) / NULLIF(Prev\_Month\_Sales, 0) \* 100, 2) AS Percent\_Growth\_Rate

FROM Monthly\_Trend

ORDER BY Year, Month\_Num;



**Year Over Year Analysis**

WITH Yearly\_Trend\_Analysis

AS

(

SELECT

DISTINCT YEAR(order\_date) Years,

ROUND(SUM(Sales),2) Current\_Year\_Sales,

ROUND(LAG(SUM(Sales)) OVER (ORDER BY YEAR(order\_date)),2) AS Prev\_Year\_Sales

FROM Orders o

JOIN Order\_items oi

ON o.order\_id = oi.order\_id

GROUP BY

YEAR(order\_date)

)

SELECT

Years,

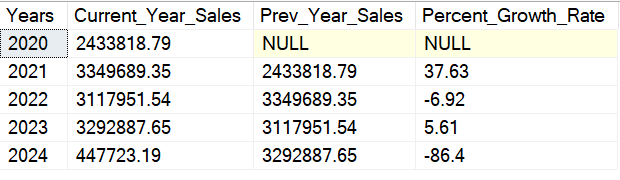
Current\_Year\_Sales,

Prev\_Year\_Sales,

ROUND((Current\_Year\_Sales - Prev\_Year\_Sales)/Prev\_Year\_Sales \* 100,2) Percent\_Growth\_Rate

FROM Yearly\_Trend\_Analysis

ORDER BY Years



**Task 6: Customers with No purchases**

**Find the Customer registered but never placed an order**

**Challenge: List customer details and the time since their registration.**

**Solution:**

**To demonstrate my understanding of SQL queries and showcase multiple ways to solve the**

**problem effectively, I have utilized three different approaches. Each approach achieves the same goal but employs distinct techniques to illustrate versatility in SQL**

SELECT \* FROM Customers

WHERE customer\_id IN

(

SELECT

DISTINCT c.Customer\_Id

FROM Customers c

LEFT JOIN Orders o

ON o.customer\_id = c.Customer\_Id

WHERE order\_id IS NULL

GROUP BY c.Customer\_Id

)

ORDER BY Customer\_Id

APPROACH 2

SELECT \* FROM Customers

WHERE customer\_id IN

(

SELECT

DISTINCT c.Customer\_Id

FROM Customers c

LEFT JOIN Orders o

ON o.customer\_id = c.Customer\_Id

GROUP BY c.Customer\_Id

HAVING COUNT(o.order\_id) = 0

)

ORDER BY Customer\_Id

SMART Approach

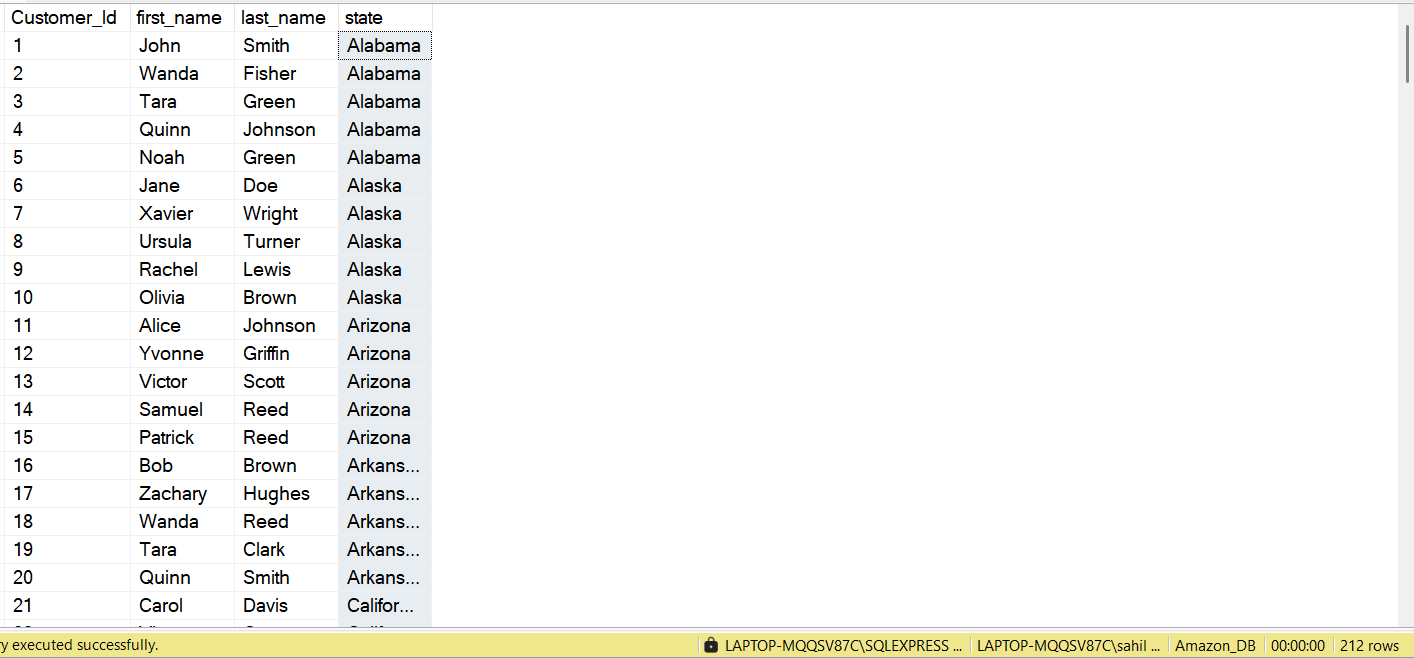
SELECT \* FROM Customers

WHERE Customer\_Id NOT IN

(SELECT DISTINCT Customer\_Id

FROM Orders)

ORDER BY Customer\_Id



**Task 7 : Best Selling Category by State**

**Identify the best-selling product for each state**

**Challenge: Include the Total Sales for that category within each State.**

**I have identified the Top 3 categories for each state to provide a focused analysis of the most significant contributors. Notably, the Electronics category accounts for approximately 89% of total sales, which highlights its dominant impact. This analysis allows stakeholders to make informed decisions based on the Top 3 categories, or expand the scope to the Top 5 or Top 10 categories if needed, depending on their strategic preferences**

WITH Best\_Selling\_Category\_by\_State

AS

(

SELECT

c.state as State,

ct.category\_name Category,

ROUND(SUM(oi.Sales),2) Total\_Sales\_by\_Category,

RANK() OVER(PARTITION BY c.state ORDER BY SUM(Sales) DESC) Rank

FROM Orders o

LEFT JOIN Customers c

ON c.Customer\_Id = o.customer\_id

LEFT JOIN Order\_items oi

ON oi.order\_id = o.order\_id

LEFT JOIN Products p

ON p.product\_id = oi.product\_id

LEFT JOIN Category ct

ON ct.category\_id = p.category\_id

GROUP BY

c.state,

ct.category\_name

)

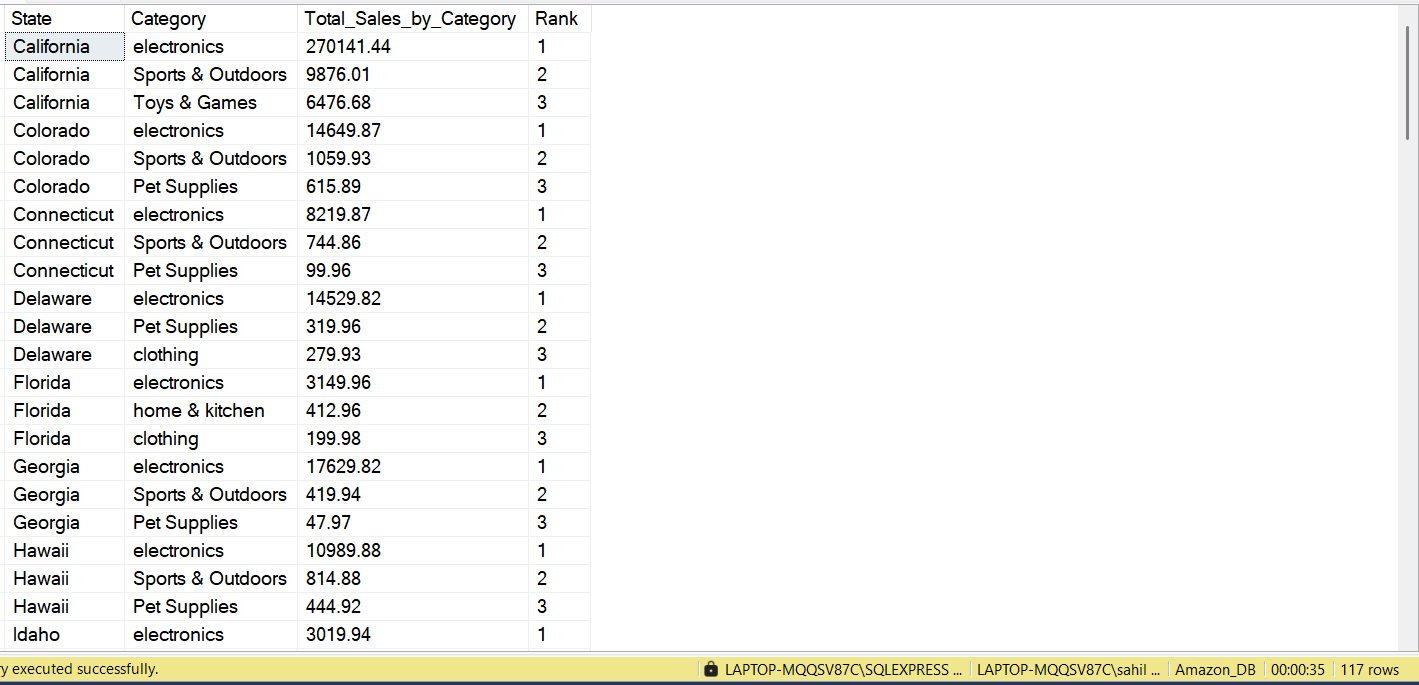
SELECT State,Category,Total\_Sales\_by\_Category, Rank FROM Best\_Selling\_Category\_by\_State

WHERE Rank <= 3

ORDER BY

State,

Total\_Sales\_by\_Category DESC



**Task 8 : Least Selling Category by State**

**Identify the least-selling product for each state**

**Challenge: Include the Total Sales for that category within each State.**

WITH Sales\_Category\_by\_State

AS

(

SELECT

c.state as State,

ct.category\_name Category,

ROUND(SUM(oi.Sales),2) Total\_Sales\_by\_Category,

RANK() OVER(PARTITION BY c.state ORDER BY SUM(Sales) ASC) Rank

FROM Orders o

LEFT JOIN Customers c

ON c.Customer\_Id = o.customer\_id

LEFT JOIN Order\_items oi

ON oi.order\_id = o.order\_id

LEFT JOIN Products p

ON p.product\_id = oi.product\_id

LEFT JOIN Category ct

ON ct.category\_id = p.category\_id

GROUP BY

c.state,

ct.category\_name

)

SELECT State,Category,Total\_Sales\_by\_Category, Rank

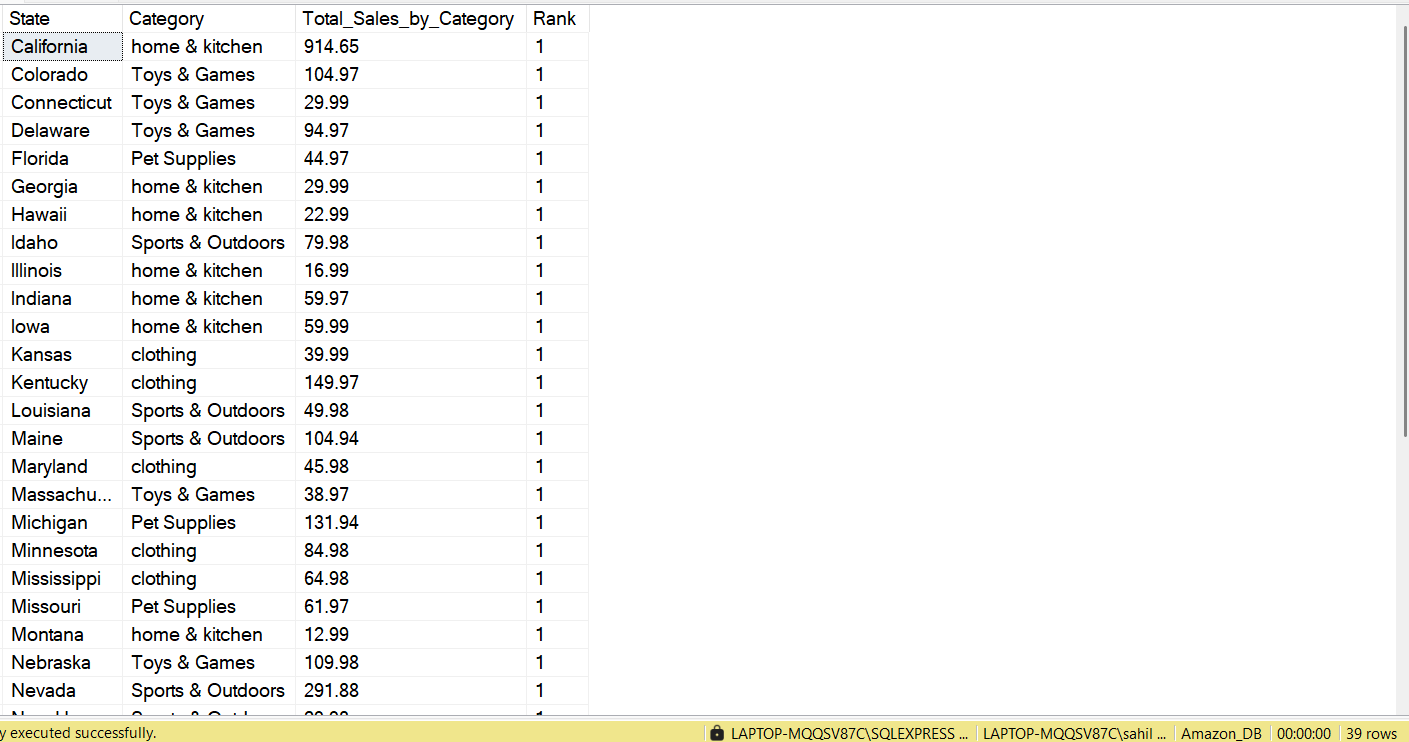
FROM Sales\_Category\_by\_State

WHERE Rank = 1

ORDER BY

State,

Total\_Sales\_by\_Category



**Task 9 Customer Lifetime Value**

**Calculate the total value of orders placed by each customer over their lifetime.**

**Challenge: Rank customers based on their Customer Lifetime Sales**

SELECT

c.customer\_id,

Concat(c.first\_name,' ' ,c.last\_name) Customer\_Name,

FORMAT(ROUND(SUM(oi.Sales),2),'N') Customer\_Lifetime\_Sales,

DENSE\_RANK() OVER(ORDER BY ROUND(SUM(oi.Sales),2) DESC) Rank

FROM Orders o

LEFT JOIN Customers c

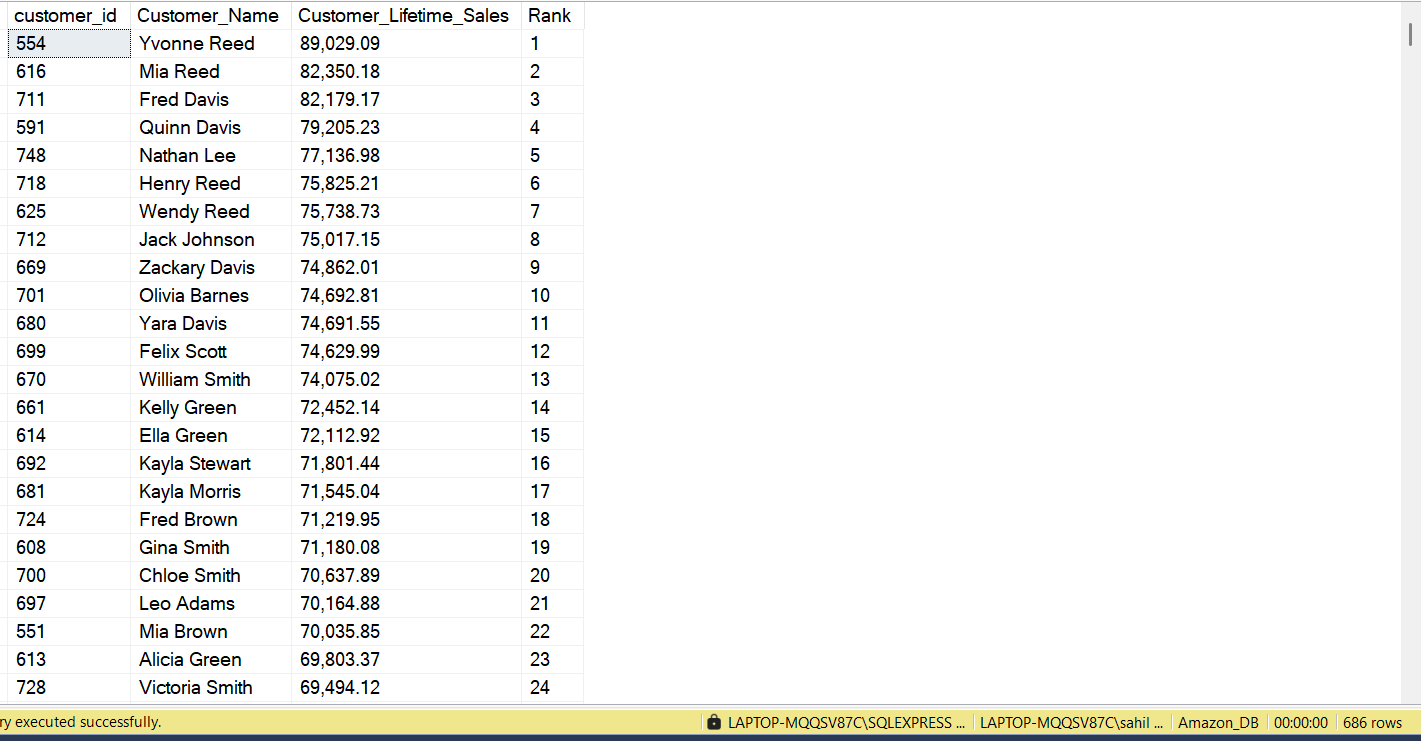
ON c.Customer\_Id = o.customer\_id

LEFT JOIN Order\_items oi

ON oi.order\_id = o.order\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

ORDER BY SUM(oi.Sales) DESC



**Task 10 Inventory Stock Alerts**

**Query Products with stock levels below a certain threshold(e., less than 10 units)**

**Challenge: Include last restock date and warehouse information**

SELECT

i.product\_id,

p.product\_name,

stock,

i.warehouse\_id ,

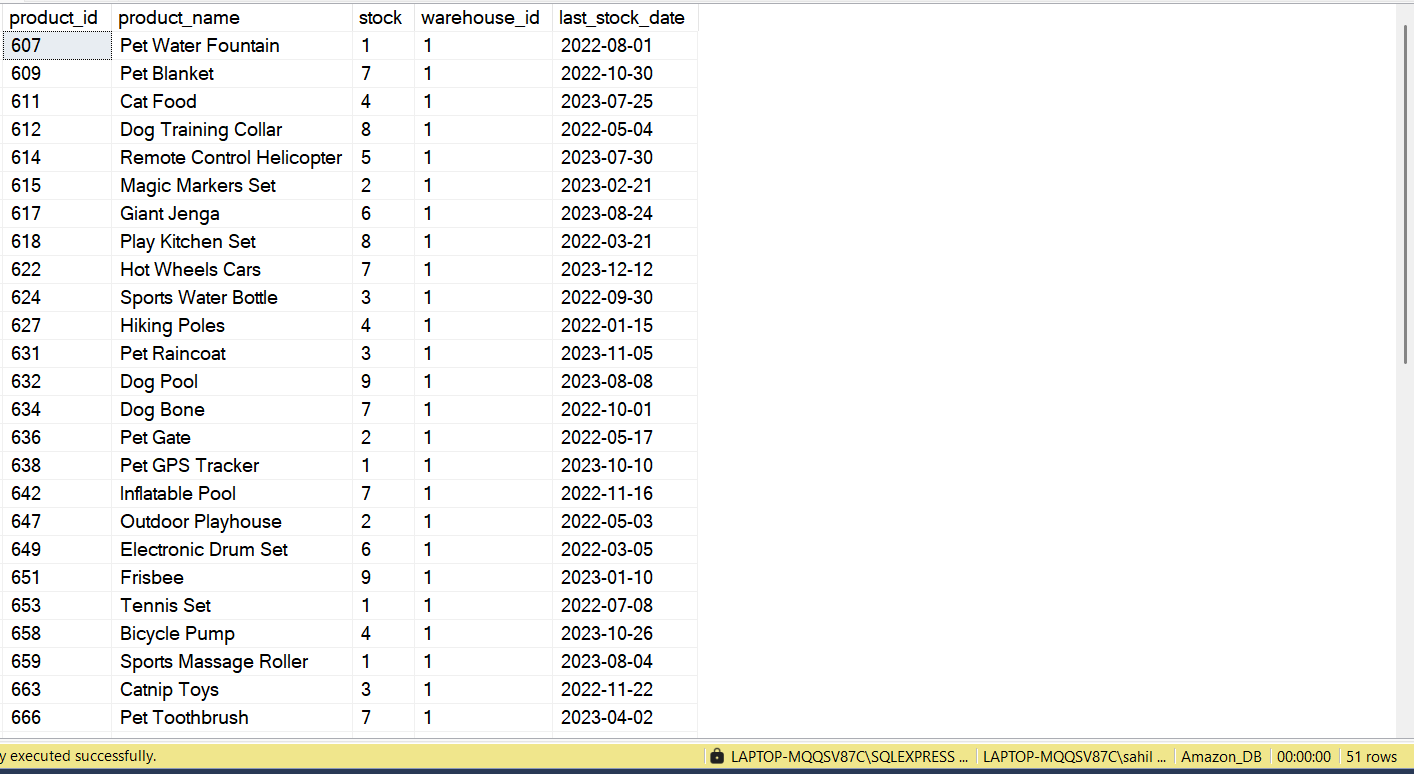
last\_stock\_date

FROM Inventory i

JOIN Products p

ON i.product\_id = p.product\_id

WHERE stock < 10



**Task 11 Shipping Delays**

**Identify orders where the shipping date is later than 4 days after the order date.**

**Challenge: Include customer, Order details, and delivery provider.**

SELECT

c.Customer\_Id,

CONCAT(c.first\_name,' ' ,c.last\_name) Customer\_Name,

o.\*,

shipping\_providers,

DATEDIFF(DAY,o.order\_date, sh.shipping\_date) Days\_took\_to\_ship

FROM Orders o

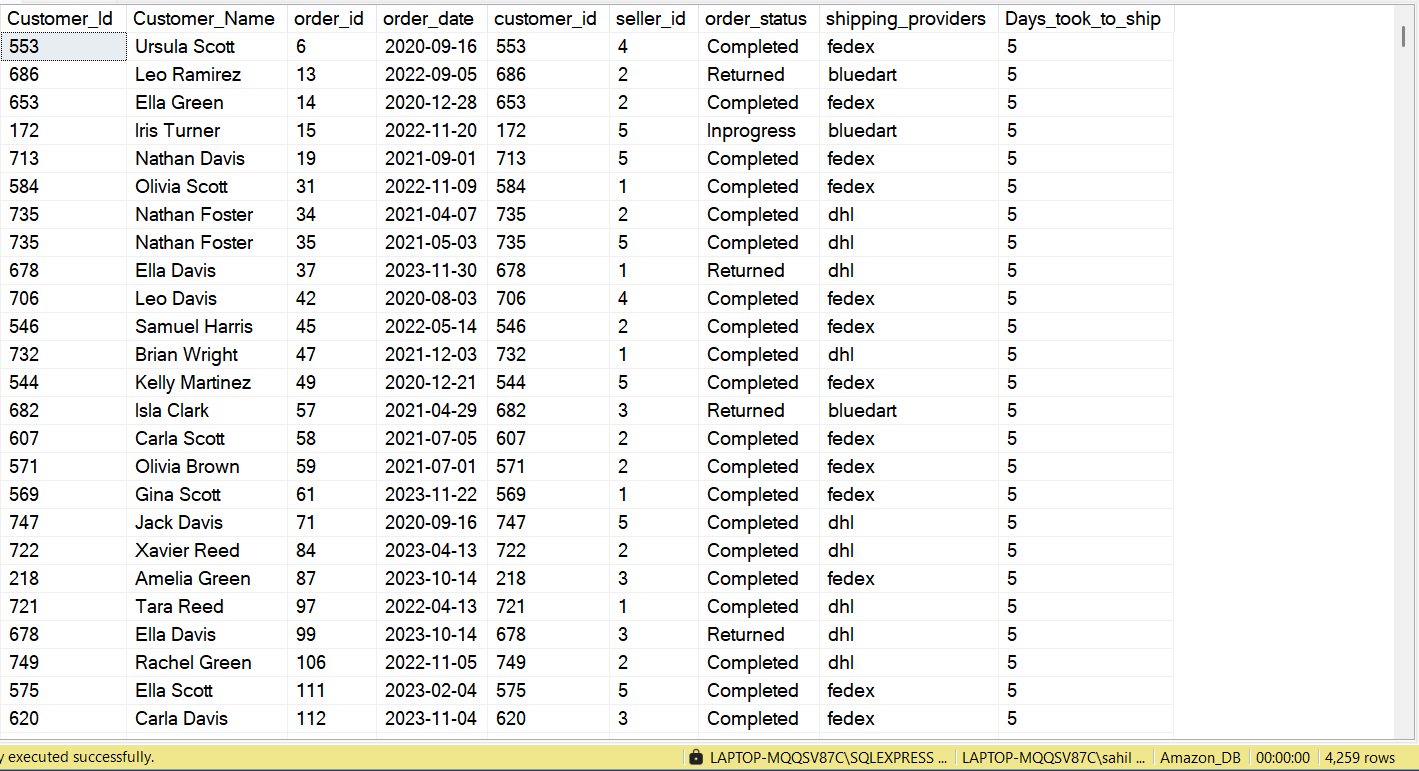
LEFT JOIN Shipping sh

ON sh.order\_id = o.order\_id

LEFT JOIN Customers c

ON c.Customer\_Id = o.customer\_id

WHERE DATEDIFF(DAY,o.order\_date, sh.shipping\_date) > 4



**Task 12 Payment Success Rate**

**Calculate the Percentage of successful payments, access all orders.**

**Challenge: Include breakdown by Payment status (eg., Failed,pending)**

SELECT

p.payment\_status Payment\_Status,

COUNT(p.payment\_id) Nr\_of\_payments,

ROUND(CAST(COUNT(p.payment\_id) AS FLOAT)/

(SELECT COUNT(payment\_id) FROM Payments) \* 100, 2) AS Percent\_Breakdown

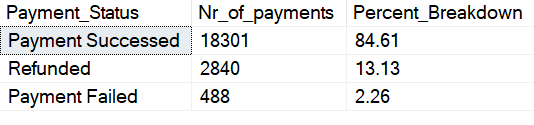
FROM Orders o

LEFT JOIN Payments p

ON p.order\_id = o.order\_id

GROUP BY p.payment\_status

ORDER BY Percent\_Breakdown DESC



**Task 13 Top Performing Sellers**

**Find the Top 5 sellers based on total Sales value.**

**Challenge: Include both successful and failed Orders, and Display their percentage of**

**successful Orders**

**Here, I have solved Solved this question in three phases, and In each phase of Query I have given insights at different level, so that Stakeholders can Get Insight at different Granularity**

SELECT TOP 5

s.seller\_id,

s.seller\_name,

FORMAT(ROUND(SUM(oi.Sales),2),'N') Total\_Sales\_by\_Sellers

FROM Orders o

JOIN Order\_items oi

ON oi.order\_id = o.order\_id

JOIN Sellers s

ON s.seller\_id = o.seller\_id

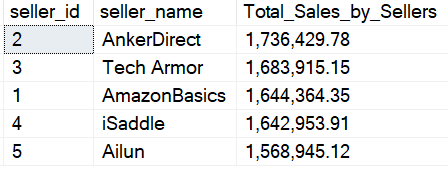
GROUP BY

s.seller\_id,

s.seller\_name

ORDER BY

ROUND(SUM(oi.Sales),2) DESC



----------------------------------------------------------------------

**Top 5 Best Sellers by Total Sales and Percentage Order Distribution based On Order\_Status**

WITH Top\_Sellers

AS

(

SELECT TOP 5

s.seller\_id,

s.seller\_name,

ROUND(SUM(oi.Sales),2) Total\_Sales

FROM Orders o

JOIN Order\_items oi

ON oi.order\_id = o.order\_id

JOIN Sellers s

ON s.seller\_id = o.seller\_id

GROUP BY

s.seller\_id,

s.seller\_name

ORDER BY

Total\_Sales DESC

),

Seller\_Orders\_Statuses

as

(

SELECT

seller\_id,

order\_status,

CAST(COUNT(\*) as float) Nr\_Orders\_By\_Seller

FROM Orders

GROUP BY seller\_id, order\_status

),

Total\_Orders

AS

(SELECT Seller\_id, COUNT(order\_status) Total\_Orders\_by\_Each\_Seller

FROM Orders

GROUP BY Seller\_id

)

SELECT

ts.seller\_id,

ts.seller\_name,

ts.Total\_Sales,

sos.order\_status,

Nr\_Orders\_By\_Seller,

-- Total\_Orders\_by\_Each\_Seller,

ROUND((Nr\_Orders\_By\_Seller/Total\_Orders\_by\_Each\_Seller) \* 100,2) Percent\_of\_Orders

FROM Top\_Sellers ts

LEFT JOIN Seller\_Orders\_Statuses sos

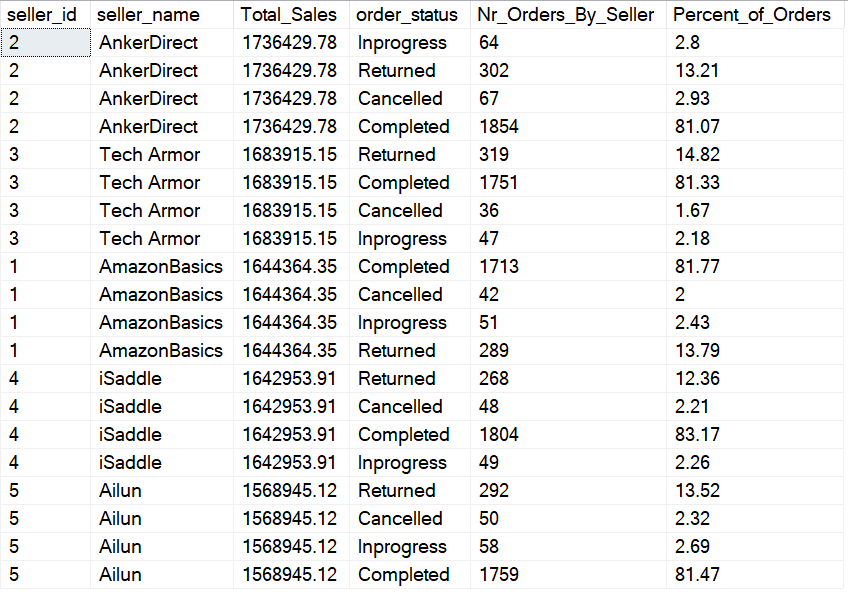
ON sos.seller\_id = ts.seller\_id

LEFT JOIN Total\_Orders ot

ON ot.seller\_id = ts.seller\_id

ORDER BY

ts.Total\_Sales DESC;



**Final Solution of Task 13**

**Top 5 Sellers by Total Revenue and Percentage distributio of Cancelled Orders and Completed**

**orders**

WITH Top\_Sellers

AS

(

SELECT TOP 5

s.seller\_id,

s.seller\_name,

ROUND(SUM(oi.Sales),2) Total\_Sales

FROM Orders o

JOIN Order\_items oi

ON oi.order\_id = o.order\_id

JOIN Sellers s

ON s.seller\_id = o.seller\_id

GROUP BY

s.seller\_id,

s.seller\_name

ORDER BY

Total\_Sales DESC

),

Seller\_Orders\_Statuses

as

(

SELECT

seller\_id,

order\_status,

CAST(COUNT(\*) as float) Nr\_Orders\_By\_Seller

FROM Orders

GROUP BY seller\_id, order\_status

),

Total\_Orders

AS

(SELECT Seller\_id, COUNT(order\_status) Total\_Orders\_by\_Each\_Seller

FROM Orders

WHERE

order\_status != 'Inprogress'

AND

order\_status != 'returned'

GROUP BY Seller\_id

)

SELECT

ts.seller\_id,

ts.seller\_name,

ts.Total\_Sales,

sos.order\_status,

Nr\_Orders\_By\_Seller,

Total\_Orders\_by\_Each\_Seller,

ROUND((Nr\_Orders\_By\_Seller/Total\_Orders\_by\_Each\_Seller) \* 100,2) Percent\_of\_Orders

FROM Top\_Sellers ts

LEFT JOIN Seller\_Orders\_Statuses sos

ON sos.seller\_id = ts.seller\_id

LEFT JOIN Total\_Orders ot

ON ot.seller\_id = ts.seller\_id

WHERE

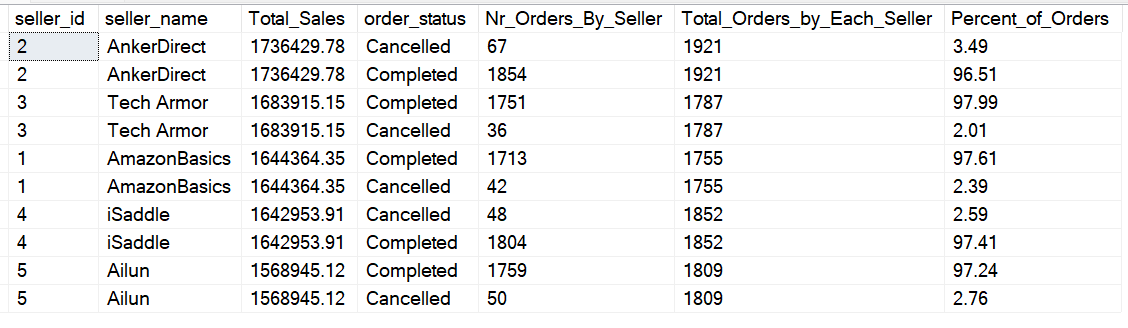
order\_status != 'Inprogress'

AND

order\_status != 'returned'

ORDER BY

ts.Total\_Sales DESC



**Task 14 Product Profit Margin**

**Calculate the profit margin for much product (difference between price and cost of goods sold)**

**Challenge: Rank products by their profit margin, showing highest to lowest**

**Here I have gone one step ahead and calculated Profit Margin Percentage**

WITH Profit\_Margin

AS

(

SELECT

p.product\_id,

p.product\_name,

ROUND(SUM(oi.Sales - (p.cogs \* oi.quantity)),2) Profit,

ROUND(SUM(oi.Sales),2) Total\_Sales

FROM Orders o

LEFT JOIN Order\_items oi

ON oi.order\_id = o.order\_id

LEFT JOIN Products p

ON oi.product\_id = p.product\_id

GROUP BY

p.product\_id,

p.product\_name

)

SELECT

product\_id,

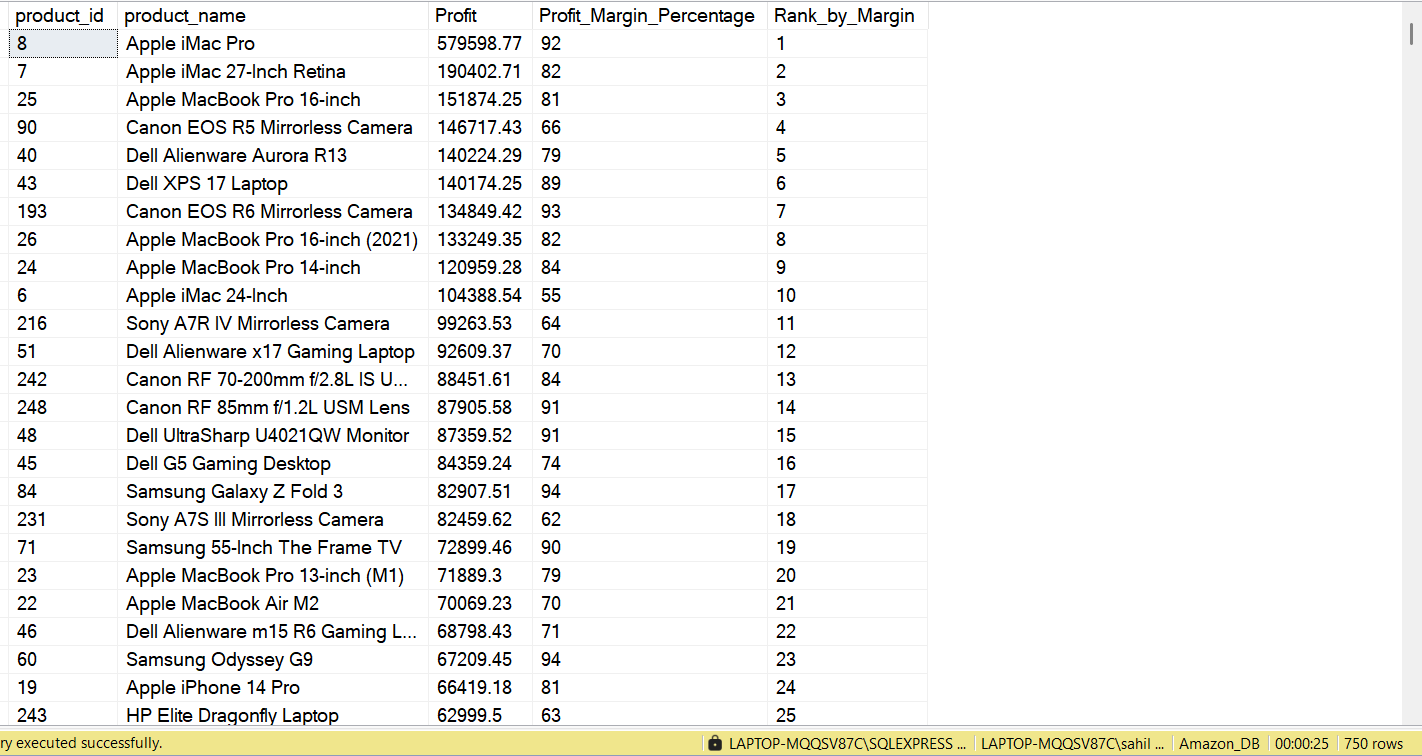
product\_name,

Profit,

ROUND((Profit/Total\_Sales),4) \* 100 Profit\_Margin\_Percentage ,

DENSE\_RANK() OVER(ORDER BY Profit DESC) Rank\_by\_Margin

FROM Profit\_Margin



**Task 15 Most Returned Products**

**Query the top 10 products by the number of return.**

**Challenge: Display the return rate as a percentage of total unitssold for each product**

WITH Top\_Returned\_Products AS (

SELECT

p.product\_id,

p.product\_name,

SUM(CASE WHEN o.order\_status = 'returned' THEN oi.quantity ELSE 0 END) AS Nr\_of\_return\_Units,

SUM(oi.quantity) AS Total\_Unit\_Sold

FROM Orders o

INNER JOIN Order\_items oi ON oi.order\_id = o.order\_id

INNER JOIN Products p ON p.product\_id = oi.product\_id

GROUP BY p.product\_id, p.product\_name

)

SELECT TOP 10

product\_id,

product\_name,

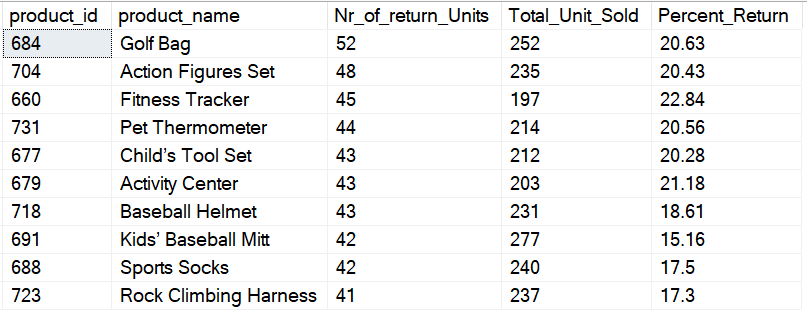
Nr\_of\_return\_Units,

Total\_Unit\_Sold,

ROUND((CAST(Nr\_of\_return\_Units AS FLOAT) / NULLIF(Total\_Unit\_Sold, 0)) \* 100, 2) AS Percent\_Return

FROM Top\_Returned\_Products

ORDER BY Nr\_of\_return\_Units DESC;



**Product Order Status and Sales Performance Summary**

WITH Total\_Orders\_of\_Products\_by\_Order\_Status AS (

SELECT

p.product\_id,

p.product\_name,

o.order\_status,

SUM(oi.quantity) AS Nr\_of\_Units

FROM Orders o

LEFT JOIN Order\_items oi

ON oi.order\_id = o.order\_id

LEFT JOIN Products p

ON p.product\_id = oi.product\_id

--WHERE p.category\_id = 1 -- You can filter Products by Category Id

--AND

--YEAR(order\_date) = 2024 -- You can filter by Year

GROUP BY p.product\_id, p.product\_name, o.order\_status

),

Total\_Orders\_of\_Each\_Products AS (

SELECT

p.product\_id,

SUM(oi.quantity) AS Total\_Units\_by\_Product

FROM Orders o

LEFT JOIN Order\_items oi

ON oi.order\_id = o.order\_id

LEFT JOIN Products p

ON p.product\_id = oi.product\_id

-- Where YEAR(order\_date) = 2024 -- You can filer by Year

-- AND

-- p.category\_id = 1 -- You can filter Products by Category Id

GROUP BY p.product\_id

),

Product\_Order\_Summary AS (

SELECT

tobos.product\_id,

tobos.product\_name,

tobos.order\_status,

tobos.Nr\_of\_Units,

tooep.Total\_Units\_by\_Product

FROM Total\_Orders\_of\_Products\_by\_Order\_Status tobos

JOIN Total\_Orders\_of\_Each\_Products tooep

ON tobos.product\_id = tooep.product\_id

)

SELECT TOP 10

product\_id,

product\_name,

order\_status,

Nr\_of\_Units,

Total\_Units\_by\_Product,

ROUND((CAST(Nr\_of\_Units AS FLOAT) / NULLIF(Total\_Units\_by\_Product, 0)) \* 100, 2) AS Percentage -- FIX: Prevent division by zero

--WHERE Nr\_of\_Units > 100 -- You can filter by Nr of units

--WHERE order\_status = 'Completed' -- You can filer by Order\_statuses like Completed, Inprogress, Complete, Returned

FROM Product\_Order\_Summary

ORDER BY Total\_Units\_by\_Product DESC, product\_id;

**This query provides key insights into product sales and order status:**

**Order Distribution: Track product performance across different order statuses**

**(Completed, In Progress, Returned).**

**Product Performance: Identify top-selling and underperforming products by comparing units sold to**

**total units.**

**Percentage Breakdown: View the percentage of units sold in each status to spot trends**

**(e.g., returns or unfulfilled orders).**

**Filters You Can Apply:**

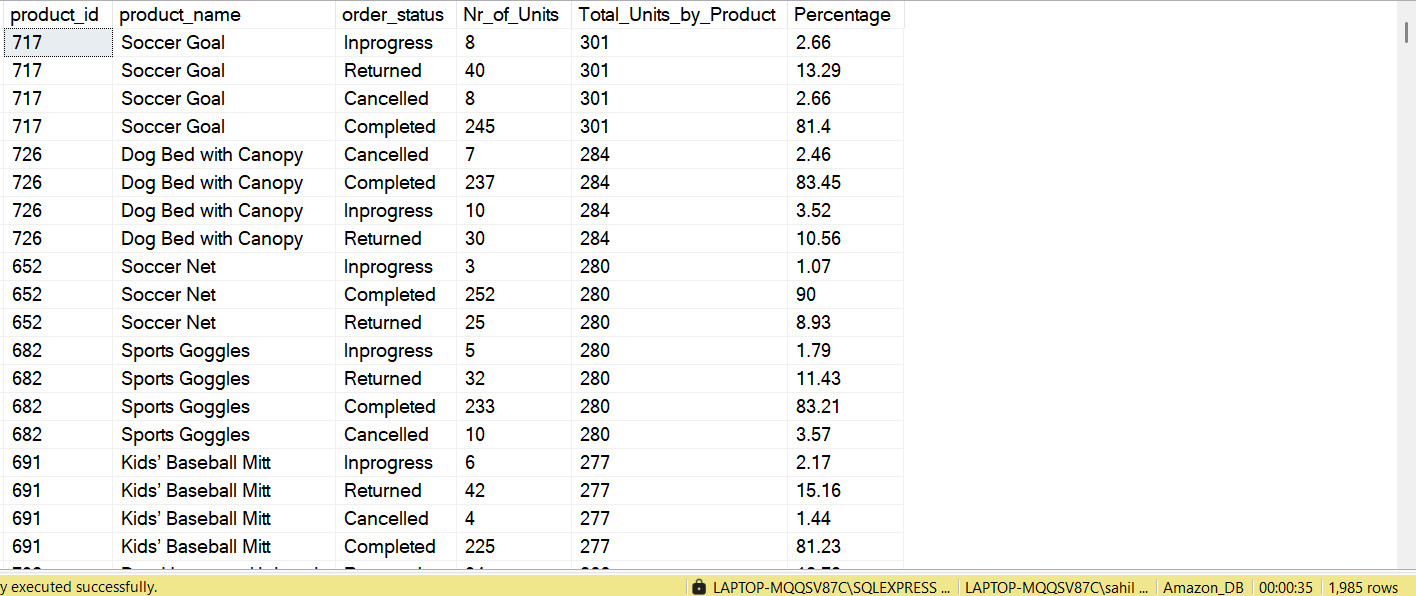
**Product Category: p.category\_id = 1 to analyze specific product categories.**

**Year: YEAR(order\_date) = 2024 to focus on a particular year's data.**

**Order Status: order\_status = 'Completed' (or other statuses) to focus on specific order types.**

**Units Sold: Nr\_of\_Units > 100 to filter products with significant sales.**

**These filters will help you get focused insights on sales, product performance, and order trends.**



**Task 16 Inactive Sellers**

**Identify Seller who haven't made any sales in the Last 6 months**

**Challenge : Show the last sale date and total sales from those sellers**

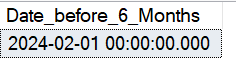
SELECT MAX(order\_date) Last\_Order\_Date FROM Orders



**Since this dataset is 1 year old as I am doing this project at Jan 2025**

**So I will consider '2024-08-01' as My Today's date**

SELECT DATEADD(MONTH,-6, '2024-08-01') Date\_before\_6\_Months;



**order\_date before 6 months that is '2024-02-01'**

WITH Seller\_cte

AS

(

SELECT

seller\_id,

seller\_name,

origin

FROM Sellers

WHERE Seller\_id NOT IN

(

SELECT DISTINCT seller\_id FROM Orders

WHERE order\_date > DATEADD(MONTH,-6, '2024-08-01'))

),

Seller\_cte2

AS

(

SELECT

s.seller\_id,

MAX(order\_date) Last\_date\_Sale,

SUM(oi.Sales) Total\_Sales

FROM Orders o

JOIN Sellers s

ON o.seller\_id = s.seller\_id

JOIN Order\_items oi

ON oi.order\_id = o.order\_id

WHERE

o.seller\_id IN (SELECT seller\_id FROM Seller\_cte)

GROUP BY s.seller\_id

)

SELECT

sc1.seller\_id,

sc1.seller\_name,

sc1.origin,

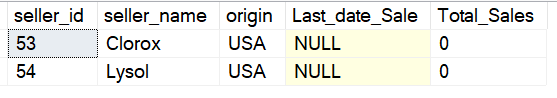
sc2.Last\_date\_Sale,

COALESCE(sc2.Total\_Sales,0) Total\_Sales

FROM Seller\_cte sc1

LEFT JOIN Seller\_cte2 sc2

ON sc1.seller\_id =sc2.seller\_id



**Task 17 Identify customers into returning or new**

**if the customer has done more than 5 return categorize them as returning otherwise new**

**challenge: List customers id, name, total orders, total returns**

WITH Customers\_Orders\_Summary

AS

(

SELECT

c.Customer\_Id,

CONCAT(c.first\_name,' ' ,c.last\_name) Customer\_Full\_Name,

COUNT( DISTINCT o.order\_id) Total\_Orders,

COALESCE(SUM(CASE WHEN

O.order\_status = 'Returned'

THEN 1 ELSE 0 END ),0) Nr\_of\_Returns

FROM Customers c

JOIN Orders o

ON c.Customer\_Id = o.customer\_id

GROUP BY

c.Customer\_Id,

c.first\_name,

c.last\_name

)

SELECT \* ,

CASE

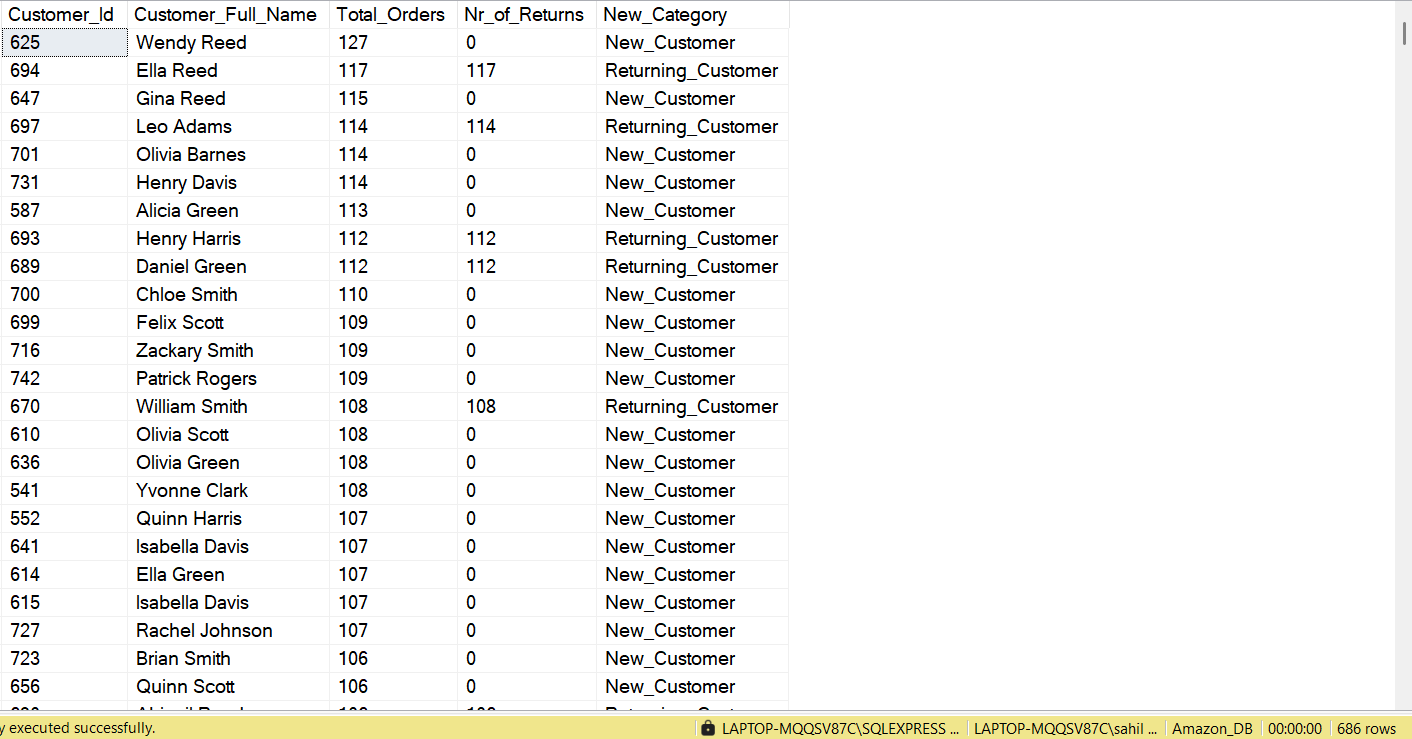
WHEN Nr\_of\_Returns > 5 THEN 'Returning\_Customer'

ELSE 'New\_Customer'

END New\_Category

FROM Customers\_Orders\_Summary

ORDER BY Total\_Orders DESC



**Task 18: Top 5 Customers by Orders in Each State**

**Identify the Top 5 Customers with the Highest number of Orders for Each State.**

**Challenge: Include the Number of Orders and total Sales for each Customer.**

SELECT \* FROM

(

SELECT

c.state,

c.Customer\_Id,

CONCAT(c.first\_name,' ' ,c.last\_name) Customer\_Full\_Name,

COUNT(o.order\_id) Total\_Orders,

ROUND(SUM(oi.Sales),2) Total\_Sales,

DENSE\_RANK() OVER(PARTITION BY c.state ORDER BY COUNT(o.order\_id) DESC) Rank\_by\_Orders

FROM Orders o

JOIN Customers c

ON o.customer\_id = c.Customer\_Id

JOIN Order\_items oi

ON oi.order\_id = o.order\_id

GROUP BY

c.state,

c.Customer\_Id,

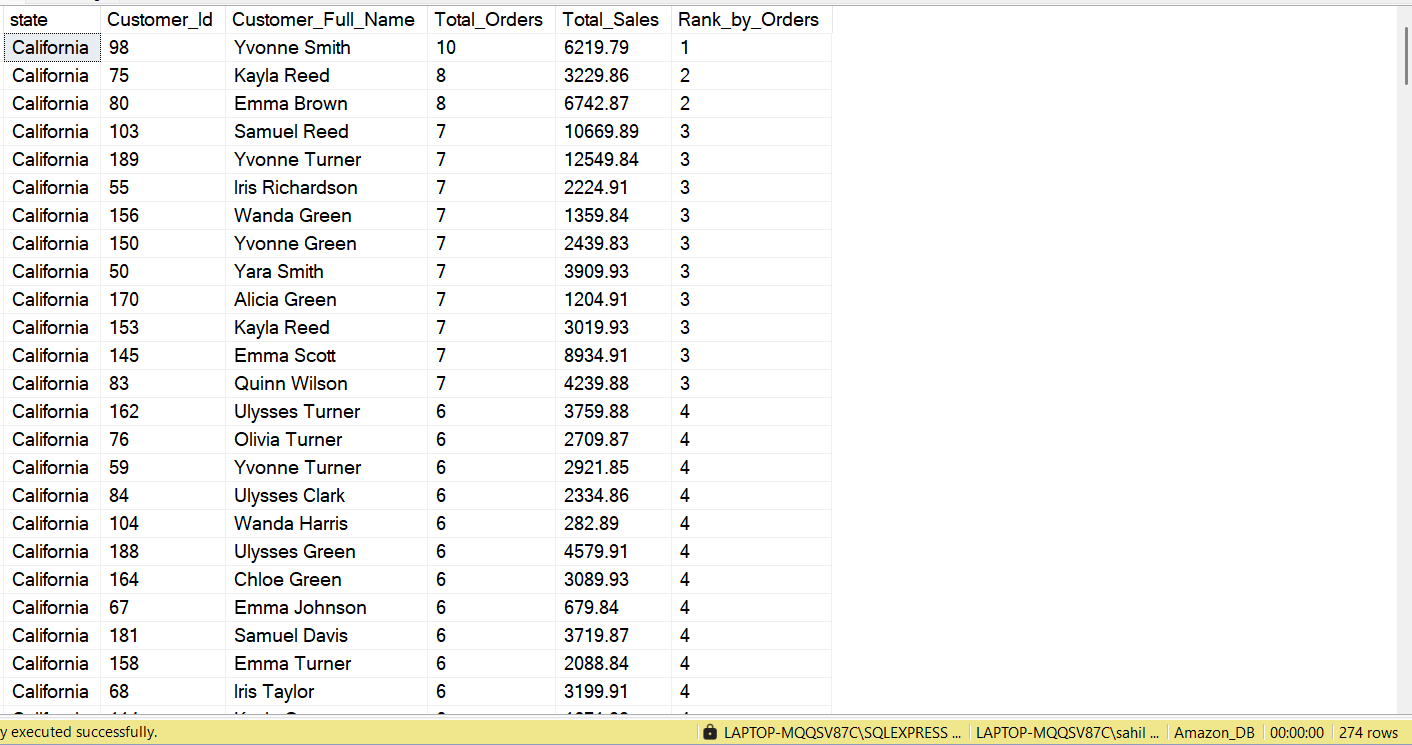
c.first\_name,

c.last\_name

) t1

WHERE Rank\_by\_Orders <= 5

ORDER BY state



**Task 20: Revenue by Shipping Provider**

**Calculate the Total Revenue handled by each shipping provider.**

**Challenge: Include the Total Number of Orders handled and the Average delivery time for each provider**

SELECT

sh.shipping\_providers,

ROUND(SUM(oi.Sales),2) Total\_Sales,

COUNT(o.order\_id) Nr\_of\_Orders,

ROUND(AVG(DATEDIFF(DAY,o.order\_date,sh.shipping\_date)\* 1.00) ,2) Time\_taken\_to\_Deliver

FROM Orders o

JOIN Shipping sh

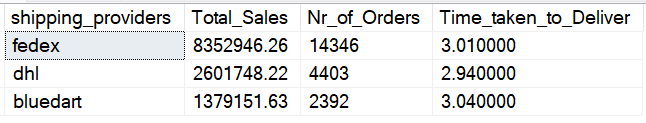
ON sh.order\_id = o.order\_id

JOIN Order\_items oi

ON oi.order\_id = o.order\_id

GROUP BY

sh.shipping\_providers



**Task 21 Top 10 Product with Highest decreasing revenue ratio compare to last year(2022) and current year(2023)**

**Challenge: Return product\_id, Product\_name, category\_name,2022 revenue and 2023 Revenue decrease ratio at end Round the result**

WITH Product\_2023

AS

(

SELECT

p.product\_id,

p.product\_name,

c.category\_name,

ROUND(SUM(oi.Sales),2) Total\_Sales\_2023

FROM Orders o

JOIN Order\_items oi

ON o.order\_id = oi.order\_id

JOIN Products p

ON p.product\_id = oi.product\_id

JOIN Category c

ON c.category\_id = p.category\_id

WHERE YEAR(o.order\_date) = 2023

GROUP BY

p.product\_id,

p.product\_name,

c.category\_name

), Product\_2022

AS

(

SELECT

p.product\_id,

p.product\_name,

c.category\_name,

ROUND(SUM(oi.Sales),2) Total\_Sales\_2022

FROM Orders o

JOIN Order\_items oi

ON o.order\_id = oi.order\_id

JOIN Products p

ON p.product\_id = oi.product\_id

JOIN Category c

ON c.category\_id = p.category\_id

WHERE YEAR(o.order\_date) = 2022

GROUP BY

p.product\_id,

p.product\_name,

c.category\_name)

, Product\_Sales\_Comparison\_2022\_2023

AS

(

SELECT

pr23.product\_id as Product\_Id,

pr23.product\_name as Product\_Name,

pr23.category\_name as Category\_Name,

pr22.Total\_Sales\_2022,

pr23.Total\_Sales\_2023

FROM Product\_2023 pr23

LEFT JOIN Product\_2022 pr22

ON pr23.product\_id = pr22.product\_id

)

SELECT TOP 10

Product\_Id,

Product\_Name,

Category\_Name,

Total\_Sales\_2022,

Total\_Sales\_2023,

ROUND(CASE WHEN Total\_Sales\_2022 > 0 THEN (Total\_Sales\_2023 - Total\_Sales\_2022)

/ Total\_Sales\_2022 \* 100 ELSE NULL END, 2) AS Percent\_Change

FROM Product\_Sales\_Comparison\_2022\_2023

WHERE Total\_Sales\_2022 > Total\_Sales\_2023

ORDER BY Percent\_Change



**Task 22: Store Procedure**

**Create a function as soon as the product is sold the same quantity should reduced from**

**Inventory table**

**Apple Air pode 3rd gen Product\_id 1--> Stock -->45**

**Apple Airpods Max Product\_id 2---> Stock---> 39**

CREATE PROCEDURE Update\_Sales

@order\_id INT,

@customer\_id INT,

@seller\_id INT,

@order\_item\_id INT,

@product\_id INT,

@quantity INT

AS

BEGIN

-- Declaring Variables

DECLARE

@v\_count INT,

@v\_price FLOAT,

@v\_product VARCHAR(100)

-- Checking Stock and Product Availability in Inventory

SELECT

@v\_price = price,

@v\_product = product\_name

FROM Products

WHERE product\_id = @product\_id

SELECT @v\_count = COUNT(\*)

FROM Inventory

WHERE product\_id = @product\_id AND stock >= @quantity

IF @v\_count > 0

BEGIN

-- Adding into Orders Table

INSERT INTO Orders(order\_id, order\_date, customer\_id, seller\_id, order\_status)

VALUES (@order\_id, CAST(GETDATE() AS DATE), @customer\_id, @seller\_id, 'Inprogress')

-- Adding into Order\_items Table

INSERT INTO Order\_items(order\_item\_id, order\_id, product\_id, quantity, price\_per\_unit, Sales)

VALUES (@order\_item\_id, @order\_id, @product\_id, @quantity, @v\_price, @v\_price \* @quantity)

-- Updating Inventory table

UPDATE Inventory

SET stock = stock - @quantity

WHERE product\_id = @product\_id

PRINT 'Thank you. Product Sale: ' + @v\_product + ' has been added. Inventory stock updated.'

END

ELSE

BEGIN

PRINT 'Thank you for the info. The product ' + @v\_product + ' is not available.'

END

END

-------------------------------------------------------------------------------

**Scenario before Execution was this**

**Apple Air pode 3rd gen Product\_id 1--> Stock -->45**

SELECT \* FROM Orders**-- Nr\_of\_Records--> 21629**

SELECT \* FROM Order\_items**;-- Nr\_of\_Records-->21629**

**TEST 1**

EXEC Update\_Sales

@order\_id = 21630,

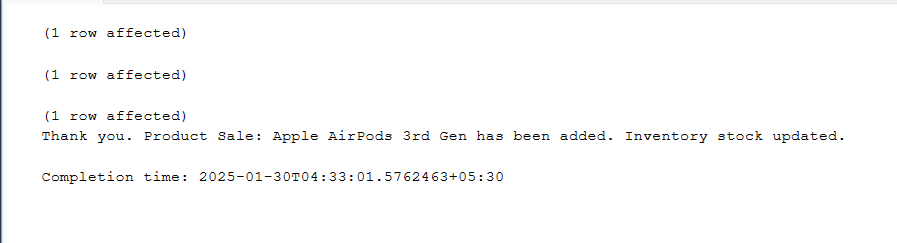
@customer\_id = 2,

@seller\_id = 8,

@order\_item\_id = 21630,

@product\_id = 1,

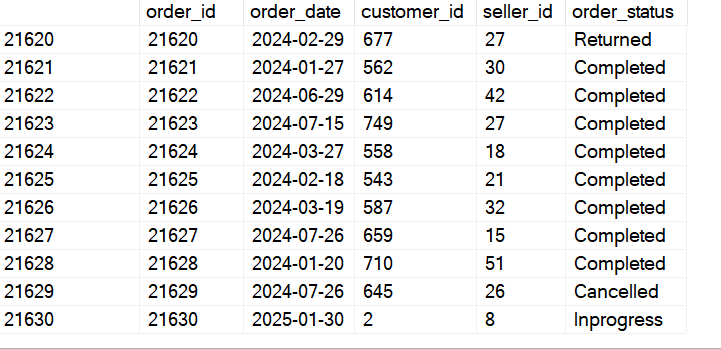
@quantity = 10;



**Now after Executing Update\_Sales Procedure**

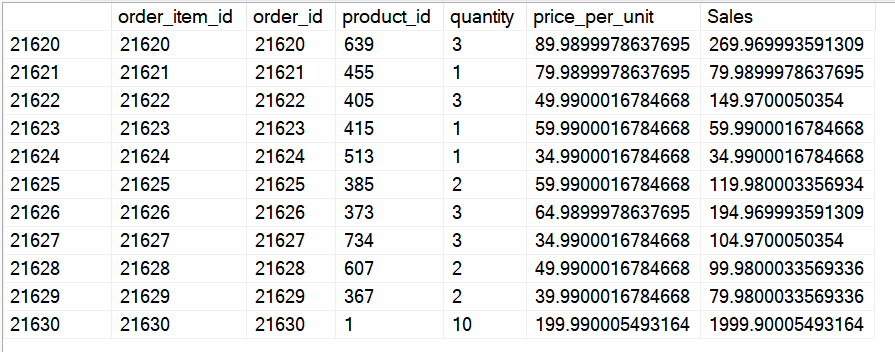
SELECT \* FROM Orders

-- Nr\_of\_Records were 21629

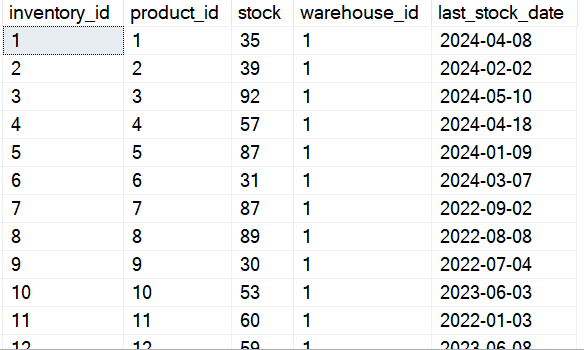


SELECT \* FROM Order\_items;

**-- Nr\_of\_Records were 21629**



SELECT \* FROM Inventory



**Apple Air pode 3rd gen Product\_id 1--> Stocks updated from 45 to 35**

**TEST 2**

**Before execution**

**Apple Airpods Max Product\_id 2---> Stock---> 39**

**Nr\_of\_Records were 21630**

**Nr\_of\_Records were 21630**

EXEC Update\_Sales

@order\_id = 21631,

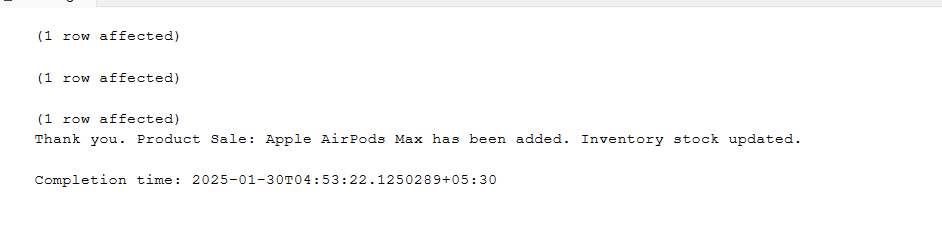
@customer\_id = 10,

@seller\_id = 4,

@order\_item\_id = 21631,

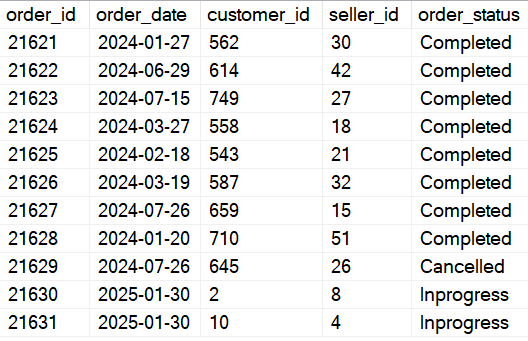
@product\_id = 2,

@quantity = 9;

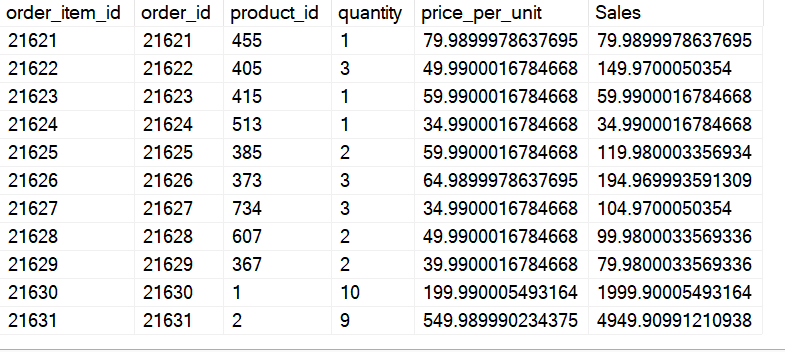


**Apple Airpods Max Product\_id 2---> Stock---> 39**

SELECT \* FROM Orders



SELECT \* FROM Order\_items



SELECT \* FROM Inventory

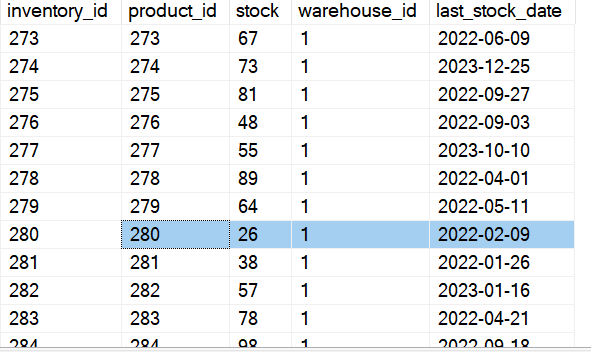
**AS you can observe Apple Airpods Max Product\_id 2**

**Stocks changed from 39 to 30 as it was sold to customer\_id = 10, Hence my second test was also successful and Store Procedure is working perfectly.**



**Test 3: what if user asked for Product and its quantity is not available**

SELECT \* FROM Inventory



**For product\_id = 280, stocks before test 3 are 26,Now if user want to retrieve 30 quantity**

**then what will happen**

EXEC Update\_Sales

@order\_id = 21632,

@customer\_id = 11,

@seller\_id = 7,

@order\_item\_id = 21632,

@product\_id = 280,

@quantity = 30;

