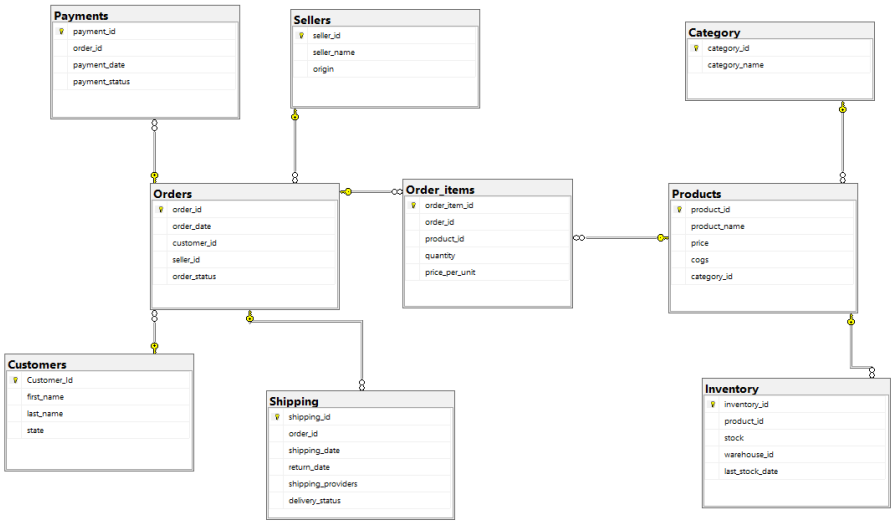
Amazon Advance SQL Data Analysis Project

```
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;
```

category_id	category_name
1	electronics
2	clothing
3	home & kitchen
4	Pet Supplies
5	Toys & Games
6	Sports & Outdoors

```
SELECT * FROM Orders;
```

order_id	order_date	customer_id	seller_id	order_status
1	2021-09-02	707	5	Completed
2	2023-12-19	644	2	Cancelled
3	2022-08-25	731	5	Completed
4	2020-04-23	749	4	Completed
5	2020-09-29	271	3	Completed
6	2020-09-16	553	4	Completed
7	2020-11-01	585	5	Completed
8	2020-11-06	607	1	Completed
9	2023-10-13	618	1	Completed
10	2021-02-05	739	5	Completed
11	2020-11-17	703	3	Completed
12	2023-03-13	580	2	Completed
13	2022-09-05	686	2	Returned
14	2020-12-28	653	2	Completed
15	2022-11-20	172	5	Inprogress
16	2023-10-21	570	1	Completed
17	2022-03-13	561	2	Completed

executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 21,629 rows

```
SELECT * FROM Order_items;
```

order_item_id	order_id	product_id	quantity	price_per_unit
1	1	244	1	99.9899978637695
2	2	154	1	99.9899978637695
3	3	88	1	349.989990234375
4	4	243	1	1999.98999023438
5	5	151	1	249.990005493164
6	6	233	2	199.990005493164
7	7	115	1	499.989990234375
8	8	217	1	229.990005493164
9	9	165	1	229.990005493164
10	10	59	1	1199.98999023438
11	11	130	2	49.9900016784668
12	12	201	1	69.9899978637695
13	13	205	3	399.989990234375
14	14	97	1	1299.98999023438
15	15	138	1	69.9899978637695
16	16	88	1	349.989990234375
17	17	164	1	499.989990234375

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:02 21,629 rows

```sql
SELECT * FROM Customers;
```

Customer_Id	first_name	last_name	state
1	John	Smith	Alabama
2	Wanda	Fisher	Alabama
3	Tara	Green	Alabama
4	Quinn	Johnson	Alabama
5	Noah	Green	Alabama
6	Jane	Doe	Alaska
7	Xavier	Wright	Alaska
8	Ursula	Turner	Alaska
9	Rachel	Lewis	Alaska
10	Olivia	Brown	Alaska
11	Alice	Johnson	Arizona
12	Yvonne	Griffin	Arizona
13	Victor	Scott	Arizona
14	Samuel	Reed	Arizona
15	Patrick	Reed	Arizona
16	Bob	Brown	Arkansas
17	Zachary	Hughes	Arkansas

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 898 rows

```sql
SELECT * FROM Payments;
```

payment_id	order_id	payment_date	payment_status
1	7136	2024-01-05	Payment Successed
2	1107	2023-06-19	Payment Successed
3	2787	2022-12-30	Payment Successed
4	1033	2022-11-24	Payment Successed
5	19241	2022-09-12	Payment Successed
6	6035	2021-07-06	Payment Successed
7	1069	2023-04-28	Payment Successed
8	4499	2023-03-05	Payment Successed
9	11560	2022-11-08	Payment Successed
10	17500	2022-01-20	Payment Successed
11	14043	2021-10-14	Payment Successed
12	1761	2020-09-09	Payment Successed
13	11416	2020-03-06	Payment Successed
14	4461	2024-01-24	Payment Successed
15	3291	2023-09-27	Payment Successed
16	8262	2023-04-05	Payment Successed
17	11768	2021-07-25	Payment Successed

executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 21,629 rows

```sql
SELECT DISTINCT payment_status FROM  Payments;
```

payment_status
Payment Failed
Refunded
Payment Successed

```sql
SELECT DISTINCT order_status FROM  Orders;
```

order_status
Inprogress
Returned
Completed
Cancelled

```sql
SELECT * FROM Inventory;
```

inventory_id	product_id	stock	warehouse_id	last_stock_date
1	1	55	1	2024-04-08
2	2	39	1	2024-02-02
3	3	92	1	2024-05-10
4	4	57	1	2024-04-18
5	5	87	1	2024-01-09
6	6	31	1	2024-03-07
7	7	87	1	2022-09-02
8	8	89	1	2022-08-08
9	9	30	1	2022-07-04
10	10	53	1	2023-06-03
11	11	60	1	2022-01-03
12	12	59	1	2023-06-08
13	13	39	1	2022-02-09
14	14	89	1	2023-08-22
15	15	40	1	2022-12-30
16	16	48	1	2023-10-04
17	17	57	1	2022-03-18

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 765 rows

```sql
SELECT * FROM Shipping;
```

shipping_id	order_id	shipping_date	return_date	shipping_providers	delivery_status
1	7136	2024-01-07	NULL	dhl	Delivered
2	1107	2023-06-22	NULL	bluedart	Delivered
3	2787	2023-01-01	NULL	fedex	Delivered
4	1033	2022-11-25	NULL	fedex	Delivered
5	19241	2022-09-13	NULL	fedex	Delivered
6	6035	2021-07-11	NULL	fedex	Delivered
7	1069	2023-04-29	NULL	fedex	Delivered
8	4499	2023-03-07	NULL	fedex	Delivered
9	11560	2022-11-12	NULL	fedex	Delivered
10	17500	2022-01-22	NULL	fedex	Delivered
11	14043	2021-10-16	NULL	fedex	Delivered
12	1761	2020-09-12	NULL	fedex	Delivered
13	11416	2020-03-10	NULL	fedex	Delivered
14	4461	2024-01-26	NULL	fedex	Delivered
15	3291	2023-09-29	NULL	fedex	Delivered
16	8262	2023-04-07	NULL	fedex	Delivered
17	11768	2021-07-30	NULL	fedex	Delivered

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:03 21,141 rows

```sql
SELECT  DISTINCT delivery_status FROM Shipping;
```

delivery_status
Returned
Shipped
Delivered

```sql
SELECT * FROM Shipping
WHERE return_date IS NOT NULL
```

shipping_id	order_id	shipping_date	return_date	shipping_providers	delivery_status
17803	6747	2023-02-14	2023-02-23	dhl	Returned
17804	6574	2023-01-18	2023-01-26	dhl	Returned
17805	15591	2022-12-19	2022-12-29	dhl	Returned
17806	720	2022-12-08	2022-12-16	dhl	Returned
17807	8198	2022-11-30	2022-12-12	dhl	Returned
17808	1628	2022-11-15	2022-11-29	dhl	Returned
17809	17737	2022-10-26	2022-11-03	dhl	Returned
17810	5199	2022-10-06	2022-10-16	dhl	Returned
17811	11501	2022-10-05	2022-10-19	dhl	Returned
17812	18844	2022-09-21	2022-10-05	dhl	Returned
17813	16361	2022-09-20	2022-10-04	dhl	Returned
17814	6391	2022-09-14	2022-09-25	dhl	Returned
17815	15851	2022-09-10	2022-09-20	dhl	Returned
17816	1216	2022-08-19	2022-08-28	dhl	Returned
17817	2145	2022-08-10	2022-08-23	dhl	Returned
17818	7443	2022-07-30	2022-08-08	dhl	Returned
17819	11264	2022-07-16	2022-07-28	dhl	Returned
17820	9312	2022-07-02	2022-07-10	dhl	Returned
17821	14500	2022-07-02	2022-07-09	dhl	Returned

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 2,840 rows

This means 2,840 products were returned

```
SELECT * FROM Orders
WHERE order_id= 6747
```

order_id	order_date	customer_id	seller_id	order_status
6747	2023-02-13	669	3	Returned

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';
```

Nr_of_Delivered_and_Retained_Orders
17802

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;
```

	product_id	product_name	Total_Quantity_Sold	Total_Sales
1	8	Apple iMac Pro	126	629998.77
2	7	Apple iMac 27-Inch Retina	129	232198.71
3	90	Canon EOS R5 Mirrorless Camera	57	222299.43
4	6	Apple iMac 24-Inch	146	189798.54
5	25	Apple MacBook Pro 16-inch	75	187499.25
6	40	Dell Alienware Aurora R13	71	177499.29
7	26	Apple MacBook Pro 16-inch (2021)	65	162499.35
8	43	Dell XPS 17 Laptop	75	157499.25
9	216	Sony A7R IV Mirrorless Camera	47	155099.53
10	193	Canon EOS R6 Mirrorless Camera	58	144999.42

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
```

category_id	category_name	Total_Sales_by_Category	Percent_Contribution
1	electronics	11343909.65	89.73
6	Sports & Outdoors	457462.79	3.62
5	Toys & Games	354165.59	2.80
4	Pet Supplies	262478.77	2.08
2	clothing	133775.88	1.06
3	home & kitchen	90277.84	0.71

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
```

Customer_Id	Customer_Name	Avg_Order_Value	Nr_of_Orders
189	Yvonne Turner	1792.83	7
103	Samuel Reed	1524.27	7
314	Quinn Green	1366.81	6
203	Xavier Green	1299.98	7
145	Emma Scott	1276.42	7
330	Hugo Smith	1238.31	6
530	Gina Coleman	1196.65	6
275	Ulysses Parker	1077.26	11
371	Felix Lee	1048.31	6
465	Rachel Turner	1031.86	8
218	Amelia Green	962.48	8
331	Liam Brown	914.31	6
711	Fred Davis	856.03	96
494	Daniel Young	853.87	9
591	Quinn Davis	851.67	93
513	Abigail Davis	846.65	6
197	Zachary Murphy	846.64	6
80	Emma Brown	842.86	8
554	Yvonne Reed	839.9	106
680	Yara Davis	820.79	91
265	Steve Baker	812.64	6

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:25 290 rows

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
```

Year	Month_Name	Current_Month_Sales	Prev_Month_Sales	Pecent_Growth_Rate
2023	January	212693.99	NULL	NULL
2023	February	220646.77	212693.99	3.74
2023	March	314383.31	220646.77	42.48
2023	April	332109.74	314383.31	5.64
2023	May	341232.79	332109.74	2.75
2023	June	280407.4	341232.79	-17.83
2023	July	288751.44	280407.4	2.98
2023	August	365033.88	288751.44	26.42
2023	September	335639.96	365033.88	-8.05
2023	October	227458.44	335639.96	-32.23
2023	November	189294.03	227458.44	-16.78
2023	December	185235.91	189294.03	-2.14

```
-- 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	Month_Name	Current_Month_Sales	Prev_Month_Sales	Percent_Growth_Rate
2020	January	167934.9	NULL	NULL
2020	February	177543.43	167934.899291992	5.72
2020	March	223589.8	177543.429260254	25.94
2020	April	191772.69	223589.798688889	-14.23
2020	May	185878.89	191772.689107895	-3.07
2020	June	214901.13	185878.888898849	15.61
2020	July	198342.99	214901.129192352	-7.71
2020	August	189218.94	198342.989147186	-4.6
2020	September	233400.56	189218.939891815	23.35
2020	October	211936.93	233400.558668137	-9.2
2020	November	206106.41	211936.92918396	-2.75
2020	December	233192.13	206106.408910751	13.14
2021	January	186850.88	233192.128559113	-19.87
2021	February	205903.93	186850.878929138	10.2
2021	March	272790.7	205903.928936005	32.48
2021	April	302043.14	272790.698762894	10.72
2021	May	339878.23	302043.139181137	12.53
2021	June	346956.26	339878.22964859	2.08
2021	July	353722.47	346956.259635925	1.95
2021	August	296694.88	353722.469362259	-16.12

executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 55 rows

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
```

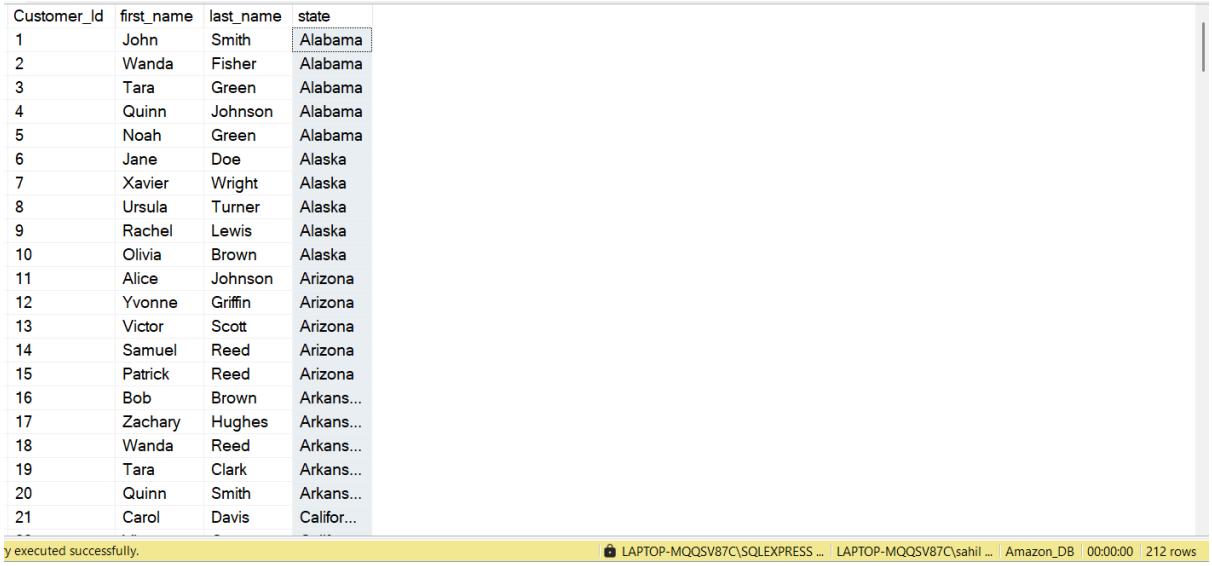
Years	Current_Year_Sales	Prev_Year_Sales	Percent_Growth_Rate
2020	2433818.79	NULL	NULL
2021	3349689.35	2433818.79	37.63
2022	3117951.54	3349689.35	-6.92
2023	3292887.65	3117951.54	5.61
2024	447723.19	3292887.65	-86.4

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
```

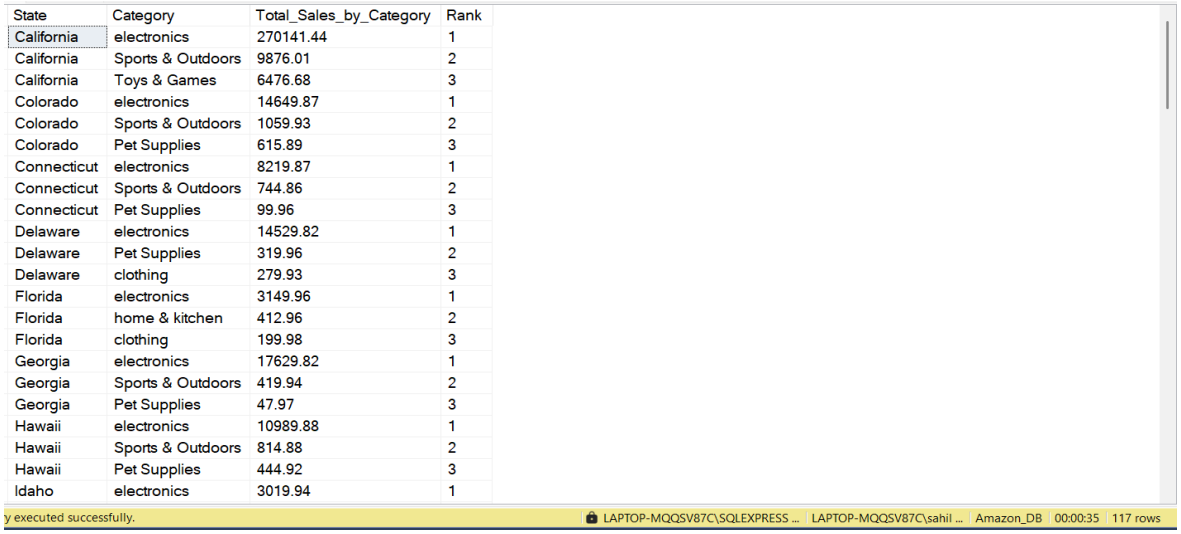


Customer_Id	first_name	last_name	state
1	John	Smith	Alabama
2	Wanda	Fisher	Alabama
3	Tara	Green	Alabama
4	Quinn	Johnson	Alabama
5	Noah	Green	Alabama
6	Jane	Doe	Alaska
7	Xavier	Wright	Alaska
8	Ursula	Turner	Alaska
9	Rachel	Lewis	Alaska
10	Olivia	Brown	Alaska
11	Alice	Johnson	Arizona
12	Yvonne	Griffin	Arizona
13	Victor	Scott	Arizona
14	Samuel	Reed	Arizona
15	Patrick	Reed	Arizona
16	Bob	Brown	Arkans...
17	Zachary	Hughes	Arkans...
18	Wanda	Reed	Arkans...
19	Tara	Clark	Arkans...
20	Quinn	Smith	Arkans...
21	Carol	Davis	Califor...

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
```



State	Category	Total_Sales_by_Category	Rank
California	electronics	270141.44	1
California	Sports & Outdoors	9876.01	2
California	Toys & Games	6476.68	3
Colorado	electronics	14649.87	1
Colorado	Sports & Outdoors	1059.93	2
Colorado	Pet Supplies	615.89	3
Connecticut	electronics	8219.87	1
Connecticut	Sports & Outdoors	744.86	2
Connecticut	Pet Supplies	99.96	3
Delaware	electronics	14529.82	1
Delaware	Pet Supplies	319.96	2
Delaware	clothing	279.93	3
Florida	electronics	3149.96	1
Florida	home & kitchen	412.96	2
Florida	clothing	199.98	3
Georgia	electronics	17629.82	1
Georgia	Sports & Outdoors	419.94	2
Georgia	Pet Supplies	47.97	3
Hawaii	electronics	10989.88	1
Hawaii	Sports & Outdoors	814.88	2
Hawaii	Pet Supplies	444.92	3
Idaho	electronics	3019.94	1

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.

```sql
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
```

State	Category	Total_Sales_by_Category	Rank
California	home & kitchen	914.65	1
Colorado	Toys & Games	104.97	1
Connecticut	Toys & Games	29.99	1
Delaware	Toys & Games	94.97	1
Florida	Pet Supplies	44.97	1
Georgia	home & kitchen	29.99	1
Hawaii	home & kitchen	22.99	1
Idaho	Sports & Outdoors	79.98	1
Illinois	home & kitchen	16.99	1
Indiana	home & kitchen	59.97	1
Iowa	home & kitchen	59.99	1
Kansas	clothing	39.99	1
Kentucky	clothing	149.97	1
Louisiana	Sports & Outdoors	49.98	1
Maine	Sports & Outdoors	104.94	1
Maryland	clothing	45.98	1
Massachu...	Toys & Games	38.97	1
Michigan	Pet Supplies	131.94	1
Minnesota	clothing	84.98	1
Mississippi	clothing	64.98	1
Missouri	Pet Supplies	61.97	1
Montana	home & kitchen	12.99	1
Nebraska	Toys & Games	109.98	1
Nevada	Sports & Outdoors	291.88	1

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 39 rows

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
```

customer_id	Customer_Name	Customer_Lifetime_Sales	Rank
554	Yvonne Reed	89,029.09	1
616	Mia Reed	82,350.18	2
711	Fred Davis	82,179.17	3
591	Quinn Davis	79,205.23	4
748	Nathan Lee	77,136.98	5
718	Henry Reed	75,825.21	6
625	Wendy Reed	75,738.73	7
712	Jack Johnson	75,017.15	8
669	Zackary Davis	74,862.01	9
701	Olivia Barnes	74,692.81	10
680	Yara Davis	74,691.55	11
699	Felix Scott	74,629.99	12
670	William Smith	74,075.02	13
661	Kelly Green	72,452.14	14
614	Ella Green	72,112.92	15
692	Kayla Stewart	71,801.44	16
681	Kayla Morris	71,545.04	17
724	Fred Brown	71,219.95	18
608	Gina Smith	71,180.08	19
700	Chloe Smith	70,637.89	20
697	Leo Adams	70,164.88	21
551	Mia Brown	70,035.85	22
613	Alicia Green	69,803.37	23
728	Victoria Smith	69,494.12	24

ry executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Amazon_DB 00:00:00 686 rows

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

```sql
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
```

product_id	product_name	stock	warehouse_id	last_stock_date
607	Pet Water Fountain	1	1	2022-08-01
609	Pet Blanket	7	1	2022-10-30
611	Cat Food	4	1	2023-07-25
612	Dog Training Collar	8	1	2022-05-04
614	Remote Control Helicopter	5	1	2023-07-30
615	Magic Markers Set	2	1	2023-02-21
617	Giant Jenga	6	1	2023-08-24
618	Play Kitchen Set	8	1	2022-03-21
622	Hot Wheels Cars	7	1	2023-12-12
624	Sports Water Bottle	3	1	2022-09-30
627	Hiking Poles	4	1	2022-01-15
631	Pet Raincoat	3	1	2023-11-05
632	Dog Pool	9	1	2023-08-08
634	Dog Bone	7	1	2022-10-01
636	Pet Gate	2	1	2022-05-17
638	Pet GPS Tracker	1	1	2023-10-10
642	Inflatable Pool	7	1	2022-11-16
647	Outdoor Playhouse	2	1	2022-05-03
649	Electronic Drum Set	6	1	2022-03-05
651	Frisbee	9	1	2023-01-10
653	Tennis Set	1	1	2022-07-08
658	Bicycle Pump	4	1	2023-10-26
659	Sports Massage Roller	1	1	2023-08-04
663	Catnip Toys	3	1	2022-11-22
666	Pet Toothbrush	7	1	2023-04-02

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.

```sql
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
```

Customer_Id	Customer_Name	order_id	order_date	customer_id	seller_id	order_status	shipping_providers	Days_took_to_ship
553	Ursula Scott	6	2020-09-16	553	4	Completed	fedex	5
686	Leo Ramirez	13	2022-09-05	686	2	Returned	bluedart	5
653	Ella Green	14	2020-12-28	653	2	Completed	fedex	5
172	Iris Turner	15	2022-11-20	172	5	Inprogress	bluedart	5
713	Nathan Davis	19	2021-09-01	713	5	Completed	fedex	5
584	Olivia Scott	31	2022-11-09	584	1	Completed	fedex	5
735	Nathan Foster	34	2021-04-07	735	2	Completed	dhl	5
735	Nathan Foster	35	2021-05-03	735	5	Completed	dhl	5
678	Ella Davis	37	2023-11-30	678	1	Returned	dhl	5
706	Leo Davis	42	2020-08-03	706	4	Completed	fedex	5
546	Samuel Harris	45	2022-05-14	546	2	Completed	fedex	5
732	Brian Wright	47	2021-12-03	732	1	Completed	dhl	5
544	Kelly Martinez	49	2020-12-21	544	5	Completed	fedex	5
682	Isla Clark	57	2021-04-29	682	3	Returned	bluedart	5
607	Carla Scott	58	2021-07-05	607	2	Completed	fedex	5
571	Olivia Brown	59	2021-07-01	571	2	Completed	fedex	5
569	Gina Scott	61	2023-11-22	569	1	Completed	fedex	5
747	Jack Davis	71	2020-09-16	747	5	Completed	dhl	5
722	Xavier Reed	84	2023-04-13	722	2	Completed	dhl	5
218	Amelia Green	87	2023-10-14	218	3	Completed	fedex	5
721	Tara Reed	97	2022-04-13	721	1	Completed	dhl	5
678	Ella Davis	99	2023-10-14	678	3	Returned	dhl	5
749	Rachel Green	106	2022-11-05	749	2	Completed	dhl	5
575	Ella Scott	111	2023-02-04	575	5	Completed	fedex	5
620	Carla Davis	112	2023-11-04	620	3	Completed	fedex	5

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
```

Payment_Status	Nr_of_payments	Percent_Breakdown
Payment Successed	18301	84.61
Refunded	2840	13.13
Payment Failed	488	2.26

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
```

seller_id	seller_name	Total_Sales_by_Sellers
2	AnkerDirect	1,736,429.78
3	Tech Armor	1,683,915.15
1	AmazonBasics	1,644,364.35
4	iSaddle	1,642,953.91
5	Ailun	1,568,945.12

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;
```

seller_id	seller_name	Total_Sales	order_status	Nr_Orders_By_Seller	Percent_of_Orders
2	AnkerDirect	1736429.78	Inprogress	64	2.8
2	AnkerDirect	1736429.78	Returned	302	13.21
2	AnkerDirect	1736429.78	Cancelled	67	2.93
2	AnkerDirect	1736429.78	Completed	1854	81.07
3	Tech Armor	1683915.15	Returned	319	14.82
3	Tech Armor	1683915.15	Completed	1751	81.33
3	Tech Armor	1683915.15	Cancelled	36	1.67
3	Tech Armor	1683915.15	Inprogress	47	2.18
1	AmazonBasics	1644364.35	Completed	1713	81.77
1	AmazonBasics	1644364.35	Cancelled	42	2
1	AmazonBasics	1644364.35	Inprogress	51	2.43
1	AmazonBasics	1644364.35	Returned	289	13.79
4	iSaddle	1642953.91	Returned	268	12.36
4	iSaddle	1642953.91	Cancelled	48	2.21
4	iSaddle	1642953.91	Completed	1804	83.17
4	iSaddle	1642953.91	Inprogress	49	2.26
5	Ailun	1568945.12	Returned	292	13.52
5	Ailun	1568945.12	Cancelled	50	2.32
5	Ailun	1568945.12	Inprogress	58	2.69
5	Ailun	1568945.12	Completed	1759	81.47

Final Solution of Task 13
Top 5 Sellers by Total Revenue and Percentage distributio of Cancelled Orders and Completed
orders

```
WITH Top_Sellers
```

```sql
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
```

seller_id	seller_name	Total_Sales	order_status	Nr_Orders_By_Seller	Total_Orders_by_Each_Seller	Percent_of_Orders
2	AnkerDirect	1736429.78	Cancelled	67	1921	3.49
2	AnkerDirect	1736429.78	Completed	1854	1921	96.51
3	Tech Armor	1683915.15	Completed	1751	1787	97.99
3	Tech Armor	1683915.15	Cancelled	36	1787	2.01
1	AmazonBasics	1644364.35	Completed	1713	1755	97.61
1	AmazonBasics	1644364.35	Cancelled	42	1755	2.39
4	iSaddle	1642953.91	Cancelled	48	1852	2.59
4	iSaddle	1642953.91	Completed	1804	1852	97.41
5	Ailun	1568945.12	Completed	1759	1809	97.24
5	Ailun	1568945.12	Cancelled	50	1809	2.76

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
```

product_id	product_name	Profit	Profit_Margin_Percentage	Rank_by_Margin
8	Apple iMac Pro	579598.77	92	1
7	Apple iMac 27-Inch Retina	190402.71	82	2
25	Apple MacBook Pro 16-inch	151874.25	81	3
90	Canon EOS R5 Mirrorless Camera	146717.43	66	4
40	Dell Alienware Aurora R13	140224.29	79	5
43	Dell XPS 17 Laptop	140174.25	89	6
193	Canon EOS R6 Mirrorless Camera	134849.42	93	7
26	Apple MacBook Pro 16-inch (2021)	133249.35	82	8
24	Apple MacBook Pro 14-inch	120959.28	84	9
6	Apple iMac 24-Inch	104388.54	55	10
216	Sony A7R IV Mirrorless Camera	99263.53	64	11
51	Dell Alienware x17 Gaming Laptop	92609.37	70	12
242	Canon RF 70-200mm f/2.8L IS U...	88451.61	84	13
248	Canon RF 85mm f/1.2L USM Lens	87905.58	91	14
48	Dell UltraSharp U4021QW Monitor	87359.52	91	15
45	Dell G5 Gaming Desktop	84359.24	74	16
84	Samsung Galaxy Z Fold 3	82907.51	94	17
231	Sony A7S III Mirrorless Camera	82459.62	62	18
71	Samsung 55-Inch The Frame TV	72899.46	90	19
23	Apple MacBook Pro 13-inch (M1)	71889.3	79	20
22	Apple MacBook Air M2	70069.23	70	21
46	Dell Alienware m15 R6 Gaming L...	68798.43	71	22
60	Samsung Odyssey G9	67209.45	94	23
19	Apple iPhone 14 Pro	66419.18	81	24
243	HP Elite Dragonfly Laptop	62999.5	63	25

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

```sql
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_id	product_name	Nr_of_return_Units	Total_Unit_Sold	Percent_Return
684	Golf Bag	52	252	20.63
704	Action Figures Set	48	235	20.43
660	Fitness Tracker	45	197	22.84
731	Pet Thermometer	44	214	20.56
677	Child's Tool Set	43	212	20.28
679	Activity Center	43	203	21.18
718	Baseball Helmet	43	231	18.61
691	Kids' Baseball Mitt	42	277	15.16
688	Sports Socks	42	240	17.5
723	Rock Climbing Harness	41	237	17.3

Product Order Status and Sales Performance Summary

```sql
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.

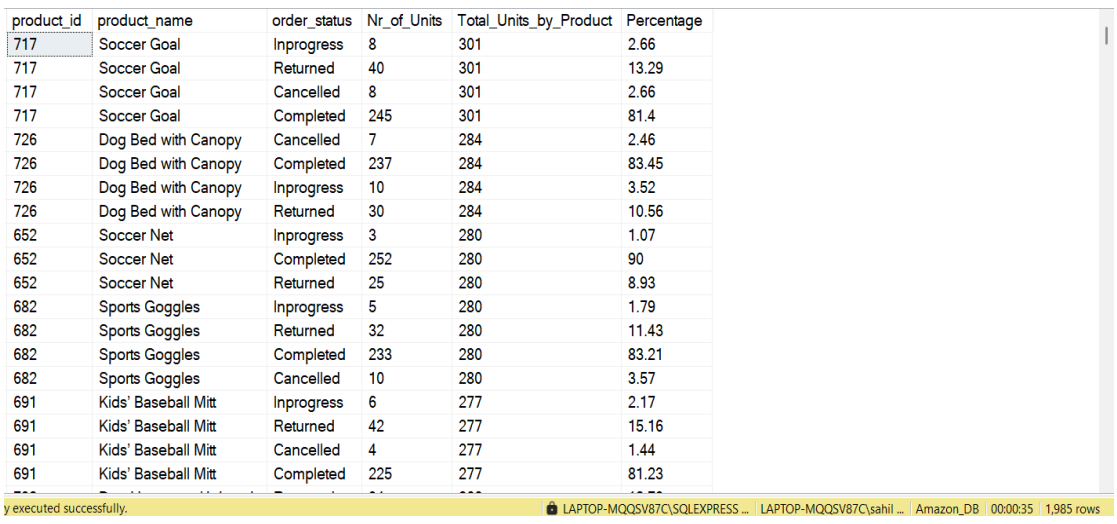
Year: YEAR(order_date) = 2024 to focus on a particular year's data.

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.

product_id	product_name	order_status	Nr_of_Units	Total_Units_by_Product	Percentage
717	Soccer Goal	Inprogress	8	301	2.66
717	Soccer Goal	Returned	40	301	13.29
717	Soccer Goal	Cancelled	8	301	2.66
717	Soccer Goal	Completed	245	301	81.4
726	Dog Bed with Canopy	Cancelled	7	284	2.46
726	Dog Bed with Canopy	Completed	237	284	83.45
726	Dog Bed with Canopy	Inprogress	10	284	3.52
726	Dog Bed with Canopy	Returned	30	284	10.56
652	Soccer Net	Inprogress	3	280	1.07
652	Soccer Net	Completed	252	280	90
652	Soccer Net	Returned	25	280	8.93
682	Sports Goggles	Inprogress	5	280	1.79
682	Sports Goggles	Returned	32	280	11.43
682	Sports Goggles	Completed	233	280	83.21
682	Sports Goggles	Cancelled	10	280	3.57
691	Kids' Baseball Mitt	Inprogress	6	277	2.17
691	Kids' Baseball Mitt	Returned	42	277	15.16
691	Kids' Baseball Mitt	Cancelled	4	277	1.44
691	Kids' Baseball Mitt	Completed	225	277	81.23



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
```

Last_Order_Date
2024-07-30

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;
```

Date_before_6_Months
2024-02-01 00:00:00.000

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
```

seller_id	seller_name	origin	Last_date_Sale	Total_Sales
53	Clorox	USA	NULL	0
54	Lysol	USA	NULL	0

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

```sql
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
```

Customer_Id	Customer_Full_Name	Total_Orders	Nr_of_Returns	New_Category
625	Wendy Reed	127	0	New_Customer
694	Ella Reed	117	117	Returning_Customer
647	Gina Reed	115	0	New_Customer
697	Leo Adams	114	114	Returning_Customer
701	Olivia Barnes	114	0	New_Customer
731	Henry Davis	114	0	New_Customer
587	Alicia Green	113	0	New_Customer
693	Henry Harris	112	112	Returning_Customer
689	Daniel Green	112	112	Returning_Customer
700	Chloe Smith	110	0	New_Customer
699	Felix Scott	109	0	New_Customer
716	Zackary Smith	109	0	New_Customer
742	Patrick Rogers	109	0	New_Customer
670	William Smith	108	108	Returning_Customer
610	Olivia Scott	108	0	New_Customer
636	Olivia Green	108	0	New_Customer
541	Yvonne Clark	108	0	New_Customer
552	Quinn Harris	107	0	New_Customer
641	Isabella Davis	107	0	New_Customer
614	Ella Green	107	0	New_Customer
615	Isabella Davis	107	0	New_Customer
727	Rachel Johnson	107	0	New_Customer
723	Brian Smith	106	0	New_Customer
656	Quinn Scott	106	0	New_Customer

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.

```sql
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
```

state	Customer_Id	Customer_Full_Name	Total_Orders	Total_Sales	Rank_by_Orders
California	98	Yvonne Smith	10	6219.79	1
California	75	Kayla Reed	8	3229.86	2
California	80	Emma Brown	8	6742.87	2
California	103	Samuel Reed	7	10669.89	3
California	189	Yvonne Turner	7	12549.84	3
California	55	Iris Richardson	7	2224.91	3
California	156	Wanda Green	7	1359.84	3
California	150	Yvonne Green	7	2439.83	3
California	50	Yara Smith	7	3909.93	3
California	170	Alicia Green	7	1204.91	3
California	153	Kayla Reed	7	3019.93	3
California	145	Emma Scott	7	8934.91	3
California	83	Quinn Wilson	7	4239.88	3
California	162	Ulysses Turner	6	3759.88	4
California	76	Olivia Turner	6	2709.87	4
California	59	Yvonne Turner	6	2921.85	4
California	84	Ulysses Clark	6	2334.86	4
California	104	Wanda Harris	6	282.89	4
California	188	Ulysses Green	6	4579.91	4
California	164	Chloe Green	6	3089.93	4
California	67	Emma Johnson	6	679.84	4
California	181	Samuel Davis	6	3719.87	4
California	158	Emma Turner	6	2088.84	4
California	68	Iris Taylor	6	3199.91	4

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
```

shipping_providers	Total_Sales	Nr_of_Orders	Time_taken_to_Deliver
fedex	8352946.26	14346	3.010000
dhl	2601748.22	4403	2.940000
bluedart	1379151.63	2392	3.040000

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
```

Product_Id	Product_Name	Category_Name	Total_Sales_2022	Total_Sales_2023	Percent_Change
351	Unisex Sports Cap	clothing	271.84	16.99	-93.75
252	HP Spectre x360 13 Laptop	electronics	14299.89	1299.99	-90.91
593	Magic Set	Toys & Games	329.89	29.99	-90.91
168	DJI Osmo Pocket 2	electronics	5399.82	599.98	-88.89
570	Yoga Mat	Sports & Outdoors	314.91	34.99	-88.89
370	Women's Long Sleeve Shirt	clothing	269.91	29.99	-88.89
571	Dumbbell Set	Sports & Outdoors	719.91	79.99	-88.89
580	Pet Shampoo	Pet Supplies	254.83	29.98	-88.24
743	Dog Toothpaste	Pet Supplies	69.93	9.99	-85.71
579	Dog Bed	Pet Supplies	349.93	49.99	-85.71

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

```sql
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;
```

```
(1 row affected)

(1 row affected)

(1 row affected)
Thank you. Product Sale: Apple AirPods 3rd Gen has been added. Inventory stock updated.

Completion time: 2025-01-30T04:33:01.5762463+05:30
```

Now after Executing Update_Sales Procedure

```sql
SELECT * FROM Orders
-- Nr_of_Records were 21629
```

	order_id	order_date	customer_id	seller_id	order_status
21620	21620	2024-02-29	677	27	Returned
21621	21621	2024-01-27	562	30	Completed
21622	21622	2024-06-29	614	42	Completed
21623	21623	2024-07-15	749	27	Completed
21624	21624	2024-03-27	558	18	Completed
21625	21625	2024-02-18	543	21	Completed
21626	21626	2024-03-19	587	32	Completed
21627	21627	2024-07-26	659	15	Completed
21628	21628	2024-01-20	710	51	Completed
21629	21629	2024-07-26	645	26	Cancelled
21630	21630	2025-01-30	2	8	Inprogress

```sql
SELECT * FROM Order_items;
-- Nr_of_Records were 21629
```

	order_item_id	order_id	product_id	quantity	price_per_unit	Sales
21620	21620	21620	639	3	89.9899978637695	269.969993591309
21621	21621	21621	455	1	79.9899978637695	79.9899978637695
21622	21622	21622	405	3	49.9900016784668	149.9700050354
21623	21623	21623	415	1	59.9900016784668	59.9900016784668
21624	21624	21624	513	1	34.9900016784668	34.9900016784668
21625	21625	21625	385	2	59.9900016784668	119.980003356934
21626	21626	21626	373	3	64.9899978637695	194.969993591309
21627	21627	21627	734	3	34.9900016784668	104.9700050354
21628	21628	21628	607	2	49.9900016784668	99.9800033569336
21629	21629	21629	367	2	39.9900016784668	79.9800033569336
21630	21630	21630	1	10	199.990005493164	1999.90005493164

```sql
SELECT * FROM Inventory
```

inventory_id	product_id	stock	warehouse_id	last_stock_date
1	1	35	1	2024-04-08
2	2	39	1	2024-02-02
3	3	92	1	2024-05-10
4	4	57	1	2024-04-18
5	5	87	1	2024-01-09
6	6	31	1	2024-03-07
7	7	87	1	2022-09-02
8	8	89	1	2022-08-08
9	9	30	1	2022-07-04
10	10	53	1	2023-06-03
11	11	60	1	2022-01-03
12	12	59	1	2023-06-08

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;
```

```
(1 row affected)

(1 row affected)

(1 row affected)
Thank you. Product Sale: Apple AirPods Max has been added. Inventory stock updated.

Completion time: 2025-01-30T04:53:22.1250289+05:30
```

Apple Airpods Max Product_id 2---> Stock---> 39

```
SELECT * FROM Orders
```

order_id	order_date	customer_id	seller_id	order_status
21621	2024-01-27	562	30	Completed
21622	2024-06-29	614	42	Completed
21623	2024-07-15	749	27	Completed
21624	2024-03-27	558	18	Completed
21625	2024-02-18	543	21	Completed
21626	2024-03-19	587	32	Completed
21627	2024-07-26	659	15	Completed
21628	2024-01-20	710	51	Completed
21629	2024-07-26	645	26	Cancelled
21630	2025-01-30	2	8	Inprogress
21631	2025-01-30	10	4	Inprogress

```
SELECT * FROM Order_items
```

order_item_id	order_id	product_id	quantity	price_per_unit	Sales
21621	21621	455	1	79.9899978637695	79.9899978637695
21622	21622	405	3	49.9900016784668	149.9700050354
21623	21623	415	1	59.9900016784668	59.9900016784668
21624	21624	513	1	34.9900016784668	34.9900016784668
21625	21625	385	2	59.9900016784668	119.980003356934
21626	21626	373	3	64.9899978637695	194.969993591309
21627	21627	734	3	34.9900016784668	104.9700050354
21628	21628	607	2	49.9900016784668	99.9800033569336
21629	21629	367	2	39.9900016784668	79.9800033569336
21630	21630	1	10	199.990005493164	1999.90005493164
21631	21631	2	9	549.989990234375	4949.90991210938

```
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.**

inventory_id	product_id	stock	warehouse_id	last_stock_date
1	1	35	1	2024-04-08
2	2	30	1	2024-02-02
3	3	92	1	2024-05-10
4	4	57	1	2024-04-18
5	5	87	1	2024-01-09
6	6	31	1	2024-03-07
7	7	87	1	2022-09-02
8	8	89	1	2022-08-08
9	9	30	1	2022-07-04
10	10	53	1	2023-06-03
11	11	60	1	2022-01-03

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

```
SELECT * FROM Inventory
```

inventory_id	product_id	stock	warehouse_id	last_stock_date
273	273	67	1	2022-06-09
274	274	73	1	2023-12-25
275	275	81	1	2022-09-27
276	276	48	1	2022-09-03
277	277	55	1	2023-10-10
278	278	89	1	2022-04-01
279	279	64	1	2022-05-11
280	280	26	1	2022-02-09
281	281	38	1	2022-01-26
282	282	57	1	2023-01-16
283	283	78	1	2022-04-21

**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;
```

Thank you for the info. The product Sony WH-1000XM3 Wireless Headphones is not available.

Completion time: 2025-01-30T12:46:53.9810603+05:30