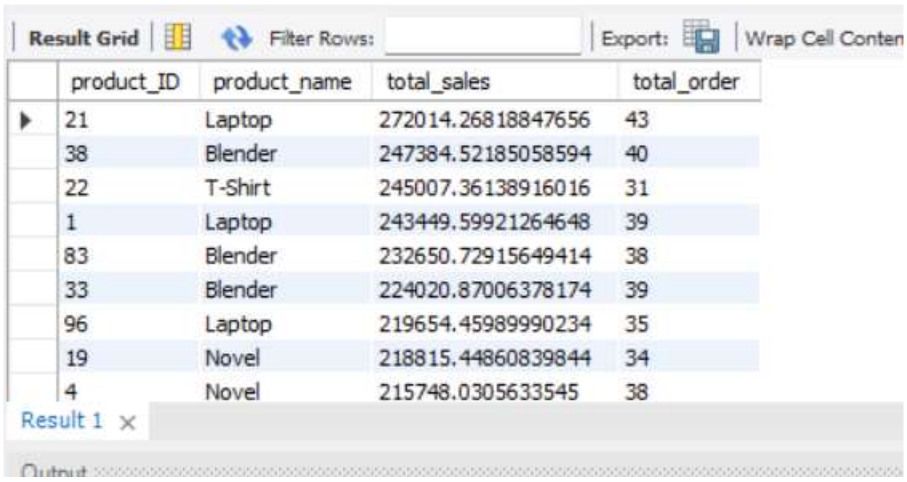
```sql
create database amazon_selles;

-- Parent Tables

CREATE TABLE Category (
    category_ID INT PRIMARY KEY,
    category_name VARCHAR(50) NOT NULL
);

CREATE TABLE Customer (
    customer_ID INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    state VARCHAR(50) NOT NULL,
    address VARCHAR(255) NOT NULL
);

CREATE TABLE Seller (
    seller_ID INT PRIMARY KEY,
    seller_name VARCHAR(50) NOT NULL,
    origin VARCHAR(50) NOT NULL
);

-- Child Tables

CREATE TABLE Product (
    product_ID INT PRIMARY KEY,
    product_name VARCHAR(50) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    cogs DECIMAL(10, 2) NOT NULL,
    category_ID INT,
    FOREIGN KEY (category_ID) REFERENCES Category(category_ID)
```

```
);

CREATE TABLE Orders (
   order_ID INT PRIMARY KEY,
   order_date DATE NOT NULL,
   customer_ID INT,
   seller_ID INT,
   order_status VARCHAR(20) NOT NULL,
   FOREIGN KEY (customer_ID) REFERENCES Customer(customer_ID),
   FOREIGN KEY (seller_ID) REFERENCES Seller(seller_ID)
);

CREATE TABLE Order_Item (
   order_item_ID INT PRIMARY KEY,
   order_ID INT,
   product_ID INT,
   quantity INT NOT NULL,
   price_per_unit DECIMAL(10, 2) NOT NULL,
   FOREIGN KEY (order_ID) REFERENCES Orders(order_ID),
   FOREIGN KEY (product_ID) REFERENCES Product(product_ID)
);

CREATE TABLE Payment (
   payment_ID INT PRIMARY KEY,
   order_ID INT,
   payment_date DATE NOT NULL,
   payment_status VARCHAR(20) NOT NULL,
   FOREIGN KEY (order_ID) REFERENCES Orders(order_ID)
);
CREATE TABLE shipping (
   shipping_ID INT PRIMARY KEY,
   order_ID INT,
   shipping_date DATE NOT NULL,
   FOREIGN KEY (order_ID) REFERENCES Orders(order_ID)
);
CREATE TABLE inventory (
   inventory_ID INT PRIMARY KEY,
   product_ID INT,
   stock INT NOT NULL,
   warehouse_ID INT NOT NULL,
   last_stock_date DATE NOT NULL,
   FOREIGN KEY (product_ID) REFERENCES Product(product_ID)
);


select * from seller;
select * from category;
select * from orders;
select * from order_item;
select * from product;
select * from seller;
select * from payment;
select * from customer;
select * from shipping;
select * from inventory;
SHOW WARNINGS;
SELECT COUNT(*) FROM payment;
SELECT COUNT(*) FROM seller;
desc inventory;
use amazon_selles;
 alter table order_item add column total_sales float;
update order_item set total_sales=quantity*price_per_unit;
```

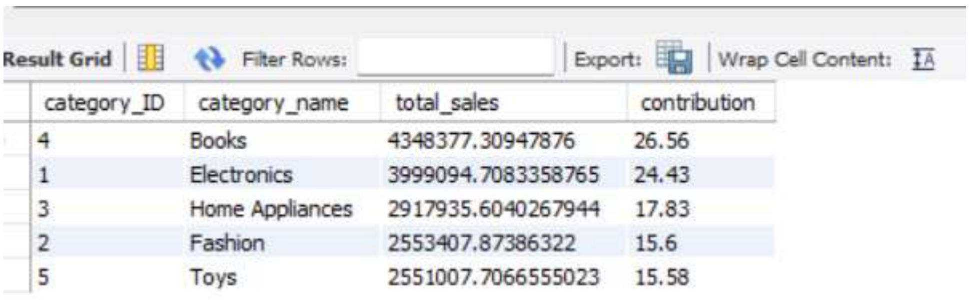
```
SET SQL_SAFE_UPDATES = 0;
UPDATE order_item
SET total_sales = quantity * price_per_unit;
SET SQL_SAFE_UPDATES = 1;
select * from order_item;
------ find Top 10 product by total sales>> include- product_name,total_quantity,total_sales value


SELECT
   oi.product_ID,
   p.product_name,
   SUM(oi.total_sales) AS total_sales,
   COUNT(o.order_ID) AS total_order
FROM
   orders o
      JOIN
   order_item oi ON o.order_ID = oi.order_ID
      JOIN
   product p ON p.product_ID = oi.product_ID
GROUP BY oi.product_ID , p.product_name
ORDER BY total_sales DESC
LIMIT 10;
```



product_ID	product_name	total_sales	total_order
21	Laptop	272014.26818847656	43
38	Blender	247384.52185058594	40
22	T-Shirt	245007.36138916016	31
1	Laptop	243449.59921264648	39
83	Blender	232650.72915649414	38
33	Blender	224020.87006378174	39
96	Laptop	219654.45989990234	35
19	Novel	218815.44860839844	34
4	Novel	215748.0305633545	38

```
----- Revenue by category >> calculate total revenue genrated by each product category
------ Include percentage contribution of each category to tatal_revenue
```

```
SELECT
    *
FROM
    category;
SELECT
    *
FROM
    product;
SELECT
    p.category_ID,
    c.category_name,
    SUM(oi.total_sales) AS total_sales,
    ROUND(SUM(oi.total_sales) / (SELECT
                    SUM(total_sales)
                FROM
                    order_item) * 100,
            2) AS contribution
FROM
    order_item oi
        JOIN
    product 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 total_sales DESC;
```

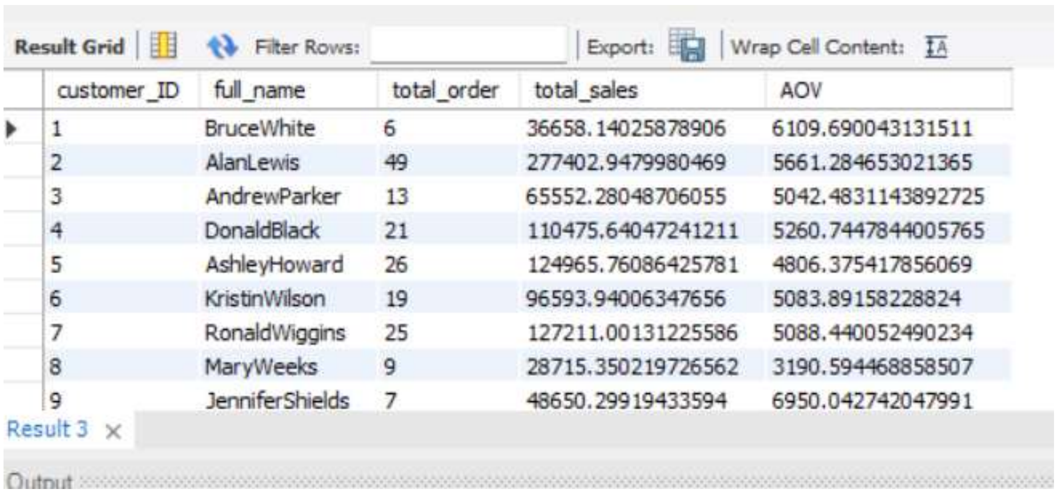

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_ID	category_name	total_sales	contribution
4	Books	4348377.30947876	26.56
1	Electronics	3999094.7083358765	24.43
3	Home Appliances	2917935.6040267944	17.83
2	Fashion	2553407.87386322	15.6
5	Toys	2551007.7066555023	15.58

----- Average order value (AOV)>>coumpute average order value of each category>>> include only customer with more than 4 order

----- Aov=sum(total_sales)/count(o.order_ID)

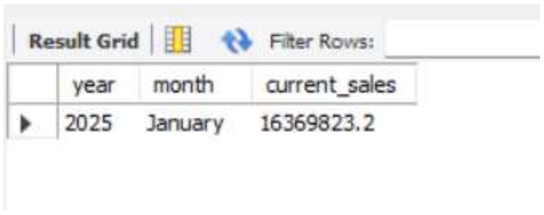
```
SELECT
   cu.customer_ID,
   CONCAT(cu.first_name, '', cu.last_name) AS full_name,
   COUNT(o.order_ID) AS total_order,
   SUM(oi.total_sales) AS total_sales,
   SUM(oi.total_sales) / COUNT(o.order_ID) AS AOV
FROM
   orders o
      JOIN
   customer cu ON o.customer_ID = cu.customer_ID
      JOIN
   order_item oi ON oi.order_ID = o.order_ID
GROUP BY cu.customer_ID , full_name
HAVING COUNT(o.order_ID) >= 5;
```



customer_ID	full_name	total_order	total_sales	AOV
1	BruceWhite	6	36658.14025878906	6109.690043131511
2	AlanLewis	49	277402.9479980469	5661.284653021365
3	AndrewParker	13	65552.28048706055	5042.4831143892725
4	DonaldBlack	21	110475.64047241211	5260.7447844005765
5	AshleyHoward	26	124965.76086425781	4806.375417856069
6	KristinWilson	19	96593.94006347656	5083.89158228824
7	RonaldWiggins	25	127211.00131225586	5088.440052490234
8	MaryWeeks	9	28715.350219726562	3190.594468858507
9	JenniferShields	7	48650.29919433594	6950.042742047991

------ Monthly sales trend >> monthly total sales over the past year>>display the sales trend,grouping by month,return current month sales ,last month sales

```
SELECT
   year, month, total_sales AS current_sales
FROM
   (SELECT
      MONTHNAME(o.order_date) AS month,
         YEAR(o.order_date) AS year,
         ROUND(SUM(oi.total_sales), 2) AS total_sales
   FROM
      orders o
   JOIN order_item oi ON o.order_ID = oi.order_ID
   WHERE
      o.order_date >= CURRENT_DATE - INTERVAL 1 YEAR
   GROUP BY MONTHNAME(o.order_date) , YEAR(o.order_date)
   ORDER BY year , month) ti;
```

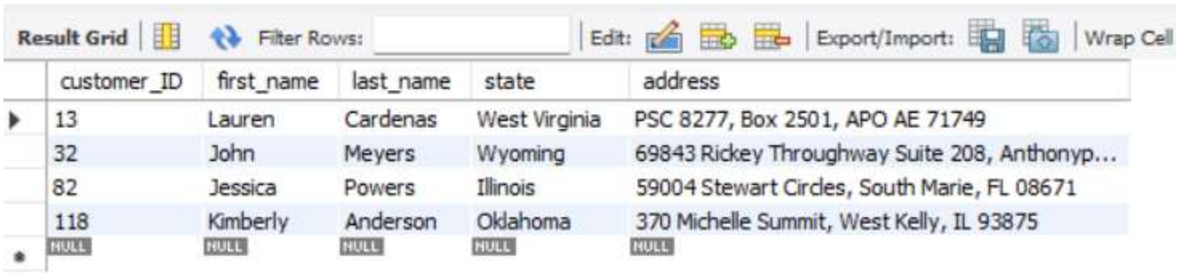


year	month	current_sales
2025	January	16369823.2

------ customer with no purches find customer who have registered but never place order

```
SELECT
   *
```

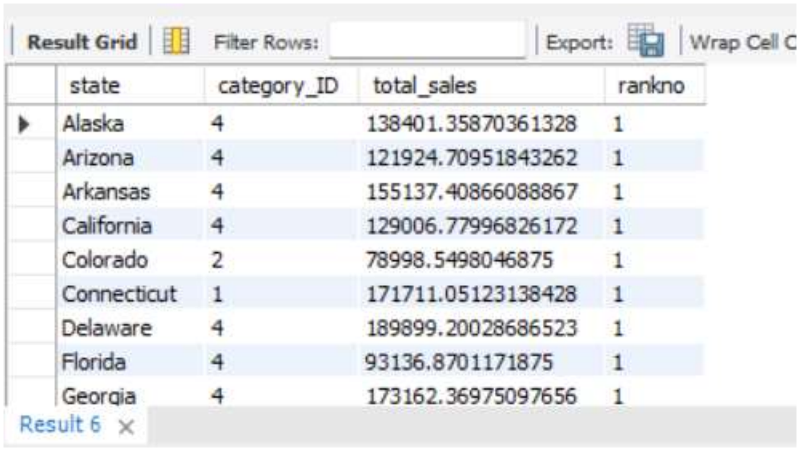
```
FROM
  customer
WHERE
  customer_ID NOT IN (SELECT DISTINCT
      customer_ID
    FROM
      orders);
```



customer_ID	first_name	last_name	state	address
13	Lauren	Cardenas	West Virginia	PSC 8277, Box 2501, APO AE 71749
32	John	Meyers	Wyoming	69843 Rickey Throughway Suite 208, Anthonyp...
82	Jessica	Powers	Illinois	59004 Stewart Circles, South Marie, FL 08671
118	Kimberly	Anderson	Oklahoma	370 Michelle Summit, West Kelly, IL 93875
NULL	NULL	NULL	NULL	NULL

----- best selling categories by state >> identify the best selling product category for each state>> include the tatal sales for that category within each state

```
with ranking as( select c.state,cat.category_ID,sum(oi.total_sales) as total_sales,
rank() over(partition by state order by sum(oi.total_sales) desc) as rankno from orders o join customer c on o.customer_ID=c.customer_ID
join order_item oi on o.order_ID=oi.order_ID join product p on oi.product_ID=p.product_ID join category cat on cat.category_ID=p.category_ID
group by c.state,cat.category_ID)
select * from ranking where rankno=1;
```

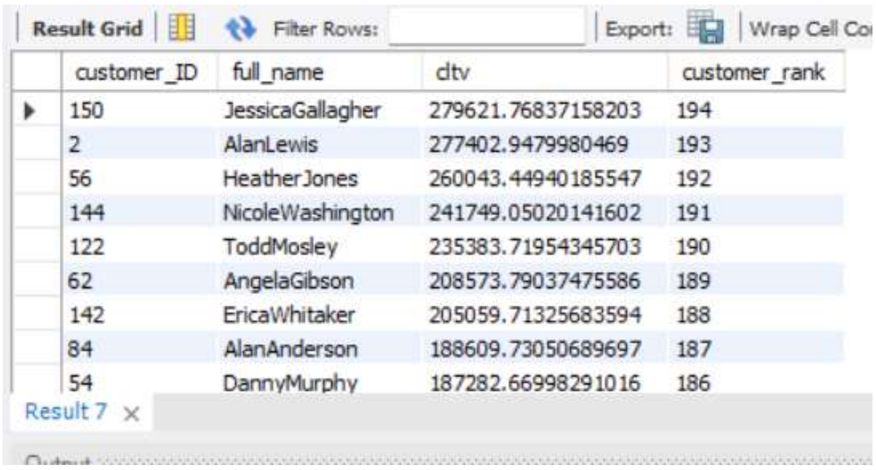


state	category_ID	total_sales	rankno
Alaska	4	138401.35870361328	1
Arizona	4	121924.70951843262	1
Arkansas	4	155137.40866088867	1
California	4	129006.77996826172	1
Colorado	2	78998.5498046875	1
Connecticut	1	171711.05123138428	1
Delaware	4	189899.20028686523	1
Florida	4	93136.8701171875	1
Georgia	4	173162.36975097656	1

----- Customer lifetime value (cltv) >> calculate the total value of orders placed by each customer over their lifetime select c.customer_ID ,concat(first_name,'',last_name) as full_name,sum(oi.total_sales) as cltv,DENSE_RANK() over(order by sum(oi.total_sales))

as customer_rank from orders o join customer c on o.customer_ID=c.customer_ID join order_item oi on oi.order_ID=o.order_ID group by

c.customer_ID ,full_name order by sum(oi.total_sales) desc;

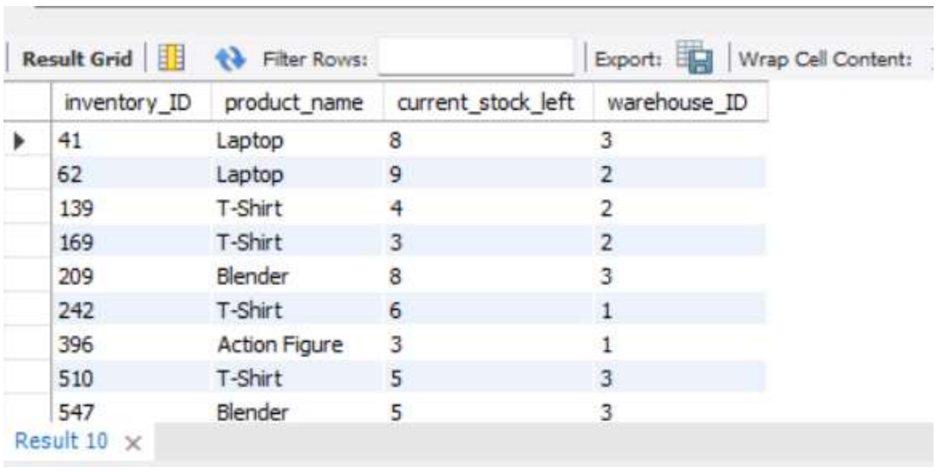


customer_ID	full_name	cltv	customer_rank
150	JessicaGallagher	279621.76837158203	194
2	AlanLewis	277402.9479980469	193
56	HeatherJones	260043.44940185547	192
144	NicoleWashington	241749.05020141602	191
122	ToddMosley	235383.71954345703	190
62	AngelaGibson	208573.79037475586	189
142	EricaWhitaker	205059.71325683594	188
84	AlanAnderson	188609.73050689697	187
54	DannyMurphy	187282.66998291016	186

------ Invenntory stock alter >> query products with stock leavel below a certauin thresholds(ex. less tham 10 unit

-- Include last restock date and warehouse Information.

```
SELECT
   i.inventory_ID,
   p.product_name,
   i.stock AS current_stock_left,
   i.warehouse_ID
FROM
   inventory i
      JOIN
   product p ON p.product_ID = i.product_ID
WHERE
   stock < 10;
```

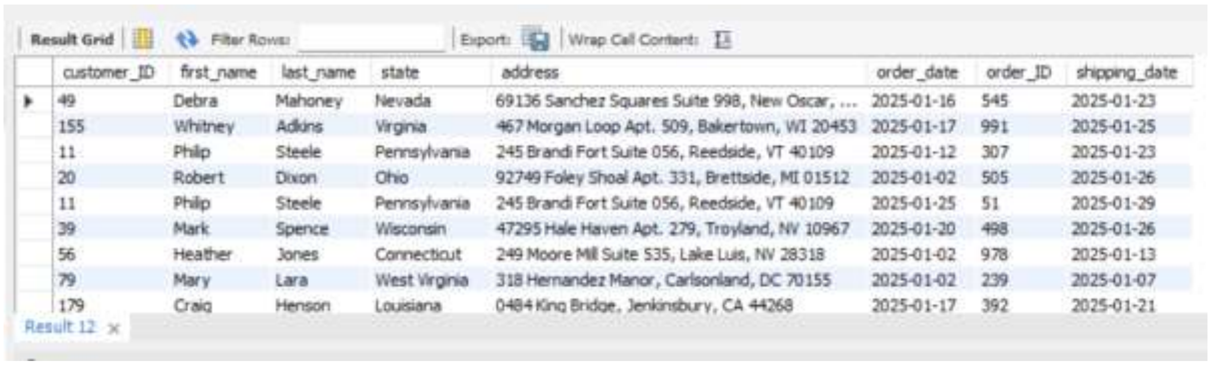


inventory_ID	product_name	current_stock_left	warehouse_ID
41	Laptop	8	3
62	Laptop	9	2
139	T-Shirt	4	2
169	T-Shirt	3	2
209	Blender	8	3
242	T-Shirt	6	1
396	Action Figure	3	1
510	T-Shirt	5	3
547	Blender	5	3

----- Shipping delays << identify orders where the shipping date is later(more) then 3 days after the order date >>

-------- Include customer, order details, and delivery provider .

```
SELECT
    c.*, o.order_date, o.order_ID, s.shipping_date
FROM
    orders o
        JOIN
    customer c ON o.customer_ID = c.customer_ID
        JOIN
    shipping s ON o.order_ID = s.order_ID
WHERE
    s.shipping_date - o.order_date > 3;
```

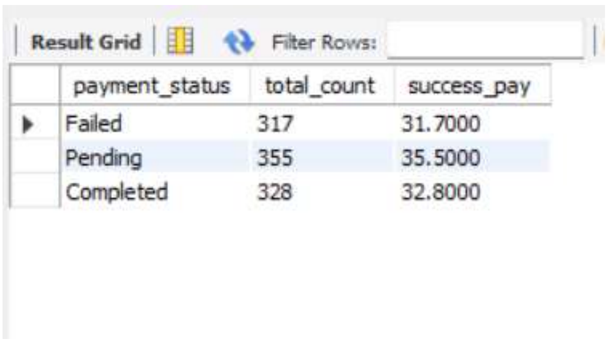


customer_ID	first_name	last_name	state	address	order_date	order_ID	shipping_date
49	Debra	Mahoney	Nevada	69136 Sanchez Squares Suite 998, New Oscar, ...	2025-01-16	545	2025-01-23
155	Whitney	Adkins	Virginia	467 Morgan Loop Apt. 509, Bakertown, WI 20453	2025-01-17	991	2025-01-25
11	Philip	Steele	Pennsylvania	245 Brandi Fort Suite 056, Reedside, VT 40109	2025-01-12	307	2025-01-23
20	Robert	Dixon	Ohio	92749 Foley Shoal Apt. 331, Brettside, MI 01512	2025-01-02	505	2025-01-26
11	Philip	Steele	Pennsylvania	245 Brandi Fort Suite 056, Reedside, VT 40109	2025-01-25	51	2025-01-29
39	Mark	Spence	Wisconsin	47295 Hale Haven Apt. 279, Troyland, NV 10967	2025-01-20	498	2025-01-26
56	Heather	Jones	Connecticut	249 Moore Mill Suite 535, Lake Luis, NV 28318	2025-01-02	978	2025-01-13
79	Mary	Lara	West Virginia	318 Hernandez Manor, Carlsonland, DC 70155	2025-01-02	239	2025-01-07
179	Craig	Henson	Louisiana	0484 King Bridge, Jenkinsbury, CA 44268	2025-01-17	392	2025-01-21

------- Payment success rate <<calculate The percentage of successful payment across all orders >>

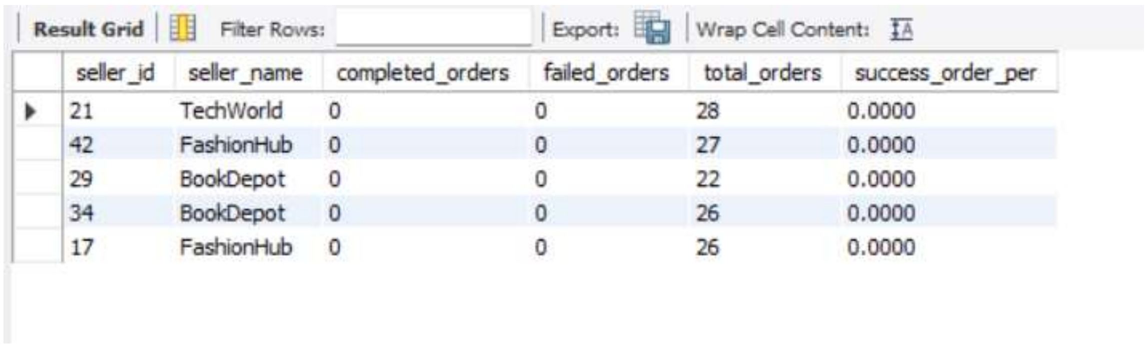
------- include breakdown by payment status (eg.failer,pending)

```
SELECT
    pay.payment_status,
    COUNT(*) AS total_count,
    (COUNT(*) / (SELECT
            COUNT(*)
        FROM
            orders)) * 100 AS success_pay
FROM
    orders o
        JOIN
    payment pay ON o.order_ID = pay.order_ID
GROUP BY pay.payment_status;
```



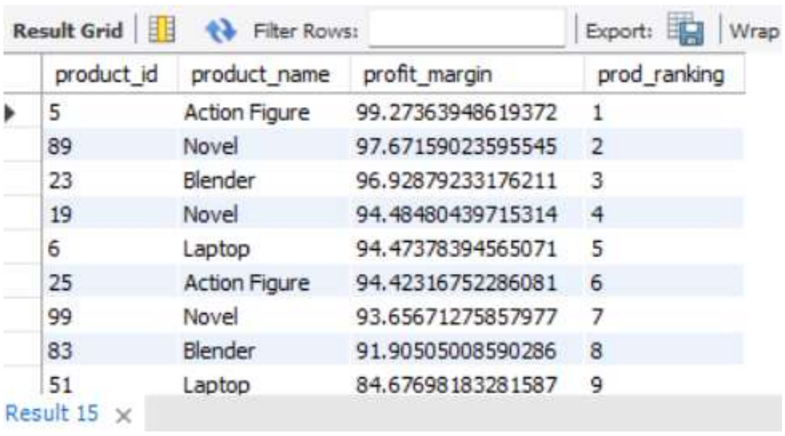
payment_status	total_count	success_pay
Failed	317	31.7000
Pending	355	35.5000
Completed	328	32.8000

```
----- Top performing seller>> find top 5 saller based on total sales value>>
---------- include both successfull and falier orders and display their percentage of successful order
WITH top_seller AS (
    SELECT s.seller_id, s.seller_name, SUM(oi.total_sales) AS total_sales
    FROM orders o
    JOIN seller s ON o.seller_ID = s.seller_ID
    JOIN order_item oi ON o.order_ID = oi.order_ID
    GROUP BY s.seller_id, s.seller_name
    ORDER BY SUM(oi.total_sales) DESC
    LIMIT 5
),
seller_report AS (
    SELECT o.seller_ID, ts.seller_name, o.order_status, COUNT(*) AS total_orders
    FROM orders o
    JOIN top_seller ts ON ts.seller_id = o.seller_id
    GROUP BY o.seller_ID, ts.seller_name, o.order_status
)
SELECT seller_id,
       seller_name,
       SUM(CASE WHEN order_status = 'completed' THEN total_orders ELSE 0 END) AS
completed_orders,
       SUM(CASE WHEN order_status = 'failed' THEN total_orders ELSE 0 END) AS failed_orders,
       SUM(total_orders) AS total_orders,
       SUM(CASE WHEN order_status = 'completed' THEN total_orders ELSE 0 END) /
SUM(total_orders) * 100 AS success_order_per
FROM seller_report
GROUP BY seller_id, seller_name;
```



seller_id	seller_name	completed_orders	failed_orders	total_orders	success_order_per
21	TechWorld	0	0	28	0.0000
42	FashionHub	0	0	27	0.0000
29	BookDepot	0	0	22	0.0000
34	BookDepot	0	0	26	0.0000
17	FashionHub	0	0	26	0.0000

```
----- Product profit margin >>> calculate the profit margin for each product (difference between price
and cost of goods sold)>>>
------  challenge: rank product by their profit margin showing highest of lowest
select product_id,product_name,profit_margin,dense_rank() over(order by profit_margin desc) as
prod_ranking from
(select p.product_id,p.product_name,sum(oi.total_sales-(p.cogs*oi.quantity)) as profit,
sum(oi.total_sales-(p.cogs*oi.quantity))/sum(oi.total_sales)*100 as profit_margin from order_item oi
join product p on oi.product_id=p.product_id group by p.product_id,p.product_name) as t1;
```

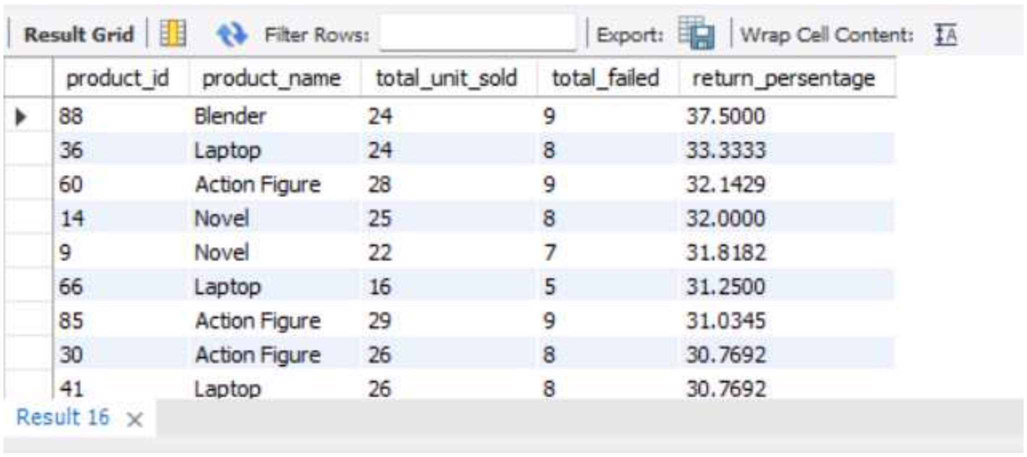

Result Grid | Filter Rows: | Export: | Wrap

product_id	product_name	profit_margin	prod_ranking
5	Action Figure	99.27363948619372	1
89	Novel	97.67159023595545	2
23	Blender	96.92879233176211	3
19	Novel	94.48480439715314	4
6	Laptop	94.47378394565071	5
25	Action Figure	94.42316752286081	6
99	Novel	93.65671275857977	7
83	Blender	91.90505008590286	8
51	Laptop	84.67698183281587	9

Result 15 ×

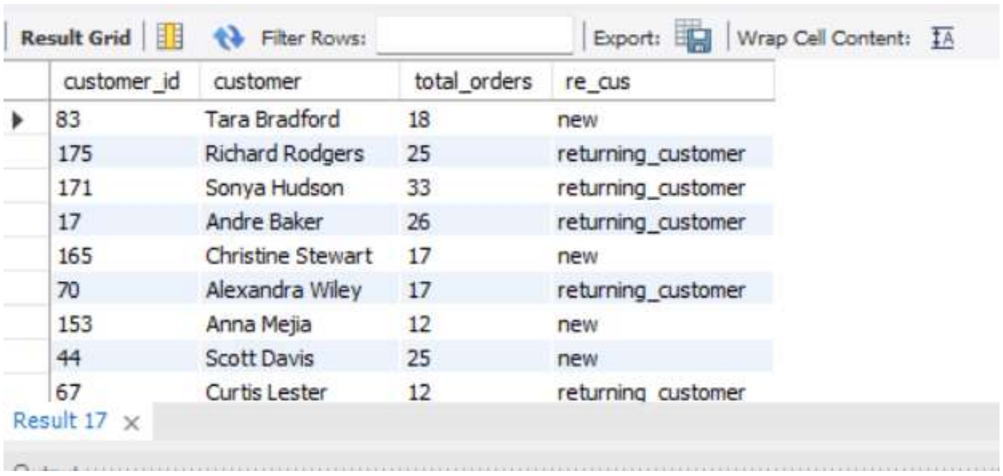
```
------- Most returned  products >>>> the top 10 products by the number of failed

--   challenge: display the returned rate as a percentage of total units solved for each product

SELECT

   p.product_id,

   p.product_name,

   COUNT(*) AS total_unit_sold,

   SUM(CASE

      WHEN o.order_status = 'Cancelled' THEN 1

      ELSE 0

   END) AS total_failed,

   SUM(CASE

      WHEN o.order_status = 'Cancelled' THEN 1

      ELSE 0

   END) / COUNT(*) * 100 AS return_persentage

FROM

   order_item oi

      JOIN

   product p ON oi.product_ID = p.product_ID

      JOIN

   orders o ON oi.order_ID = o.order_ID

GROUP BY p.product_id , p.product_name

ORDER BY SUM(CASE

   WHEN o.order_status = 'Cancelled' THEN 1
```

```
    ELSE 0
END) / COUNT(*) * 100 DESC;
```



product_id	product_name	total_unit_sold	total_failed	return_persentage
88	Blender	24	9	37.5000
36	Laptop	24	8	33.3333
60	Action Figure	28	9	32.1429
14	Novel	25	8	32.0000
9	Novel	22	7	31.8182
66	Laptop	16	5	31.2500
85	Action Figure	29	9	31.0345
30	Action Figure	26	8	30.7692
41	Laptop	26	8	30.7692

```
------- Inactive seller >>> identify seller who have not made any sale in last six month
------ >>challenge :show last sale date and total sale from those sellers
select * from seller;


  SELECT
  t1.customer_id,
  t1.full_name AS customer,
  t1.total_orders,
  CASE
    WHEN t1.total_return > 5 THEN 'returning_customer'
    ELSE 'new'
  END AS re_cus
FROM
  (SELECT
    c.customer_ID,
    CONCAT(c.first_name, ' ', c.last_name) AS full_name,
    COUNT(o.order_id) AS total_orders,
    SUM(CASE
      WHEN o.order_status = 'Cancelled' THEN 1
      ELSE 0
    END) AS total_return
  FROM orders o
  JOIN customer c ON o.customer_ID = c.customer_ID
  JOIN order_item AS oi ON o.order_ID = oi.order_ID
  GROUP BY c.customer_id, c.first_name, c.last_name) AS t1;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

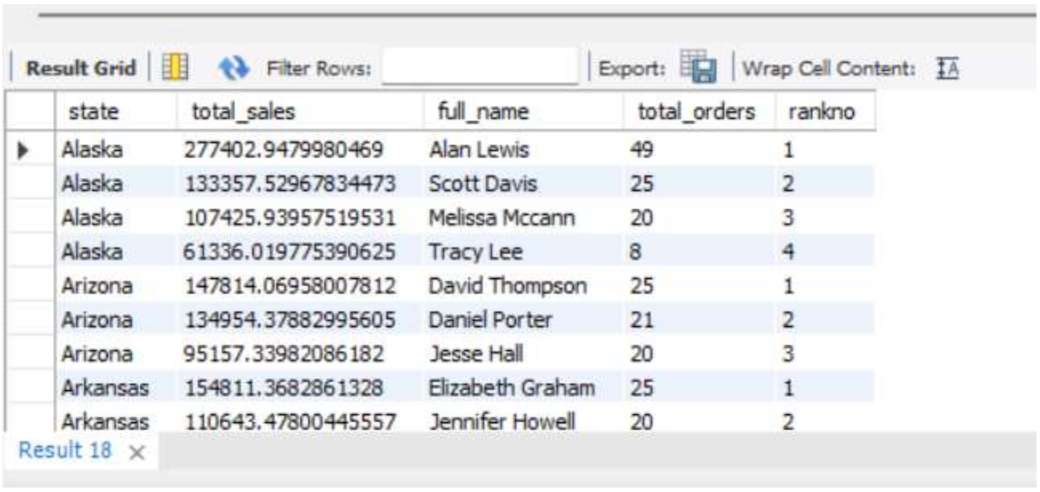
customer_id	customer	total_orders	re_cus
83	Tara Bradford	18	new
175	Richard Rodgers	25	returning_customer
171	Sonya Hudson	33	returning_customer
17	Andre Baker	26	returning_customer
165	Christine Stewart	17	new
70	Alexandra Wiley	17	returning_customer
153	Anna Mejia	12	new
44	Scott Davis	25	new
67	Curtis Lester	12	returning_customer

Result 17

------ top 5 customers by orders in each state identify the top 5 customer with the highest number of orders for each state

--- >>>challenge include the number of orders in total sales for each customer

```
 select * from (select c.state,sum(oi.total_sales) as total_sales,
      CONCAT(c.first_name, ' ', c.last_name) AS full_name,
      COUNT(o.order_id) AS total_orders ,dense_rank() over(partition by state order by COUNT(o.order_id) desc) as rankno from
      orders o join order_item oi on o.order_ID=oi.order_ID join customer c on c.customer_ID= o.customer_ID group by c.state,
       CONCAT(c.first_name, ' ', c.last_name)) t1 where  rankno<=5;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

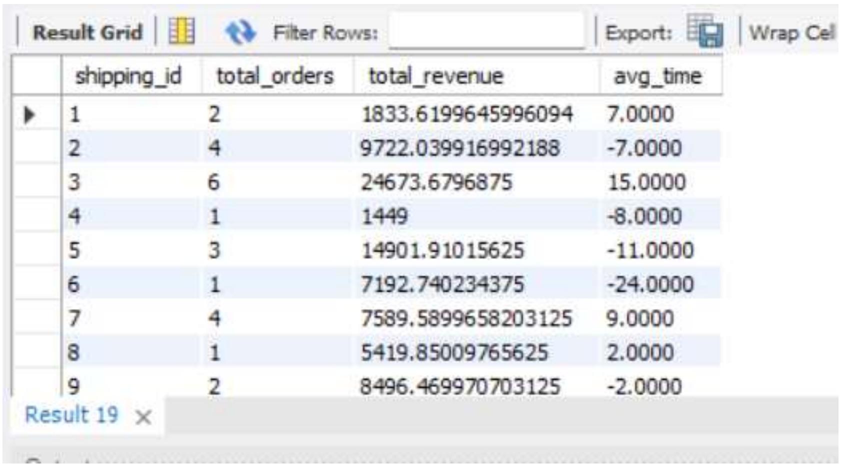
state	total_sales	full_name	total_orders	rankno
Alaska	277402.9479980469	Alan Lewis	49	1
Alaska	133357.52967834473	Scott Davis	25	2
Alaska	107425.93957519531	Melissa Mccann	20	3
Alaska	61336.019775390625	Tracy Lee	8	4
Arizona	147814.06958007812	David Thompson	25	1
Arizona	134954.37882995605	Daniel Porter	21	2
Arizona	95157.33982086182	Jesse Hall	20	3
Arkansas	154811.3682861328	Elizabeth Graham	25	1
Arkansas	110643.47800445557	Jennifer Howell	20	2

Result 18

------- Revenue by shipping_id calculate the total revenue handled by shipping_id>>

----- challenge: include the total number of orders handled and the average delivery time for each provider

```
SELECT
   s.shipping_id,
   COUNT(o.order_id) AS total_orders,
   SUM(oi.total_sales) AS total_revenue,
   COALESCE(AVG((o.order_date) - (s.shipping_date)),
         0) AS avg_time
FROM
   orders o
      JOIN
   order_item oi ON o.order_ID = oi.order_ID
      JOIN
   shipping s ON o.order_ID = s.order_ID
GROUP BY s.shipping_id;
```



shipping_id	total_orders	total_revenue	avg_time
1	2	1833.6199645996094	7.0000
2	4	9722.039916992188	-7.0000
3	6	24673.6796875	15.0000
4	1	1449	-8.0000
5	3	14901.91015625	-11.0000
6	1	7192.740234375	-24.0000
7	4	7589.5899658203125	9.0000
8	1	5419.85009765625	2.0000
9	2	8496.469970703125	-2.0000

------ Top 10 product with highest decreasing revenue ratio compared to last 2022 and current year 2023 challenging>>>

-- return product ID, product name, 2022 revenue, and 2023 revenue, decrese ratio at end round the result

```
with last_year_sale as (select p.product_id,p.product_name,sum(oi.total_sales) as revenue
```

from orders o join order_item oi on o.order_ID=oi.order_ID join product p on oi.product_ID=p.product_ID where year(o.order_date)=2022 group by p.product_id,p.product_name),

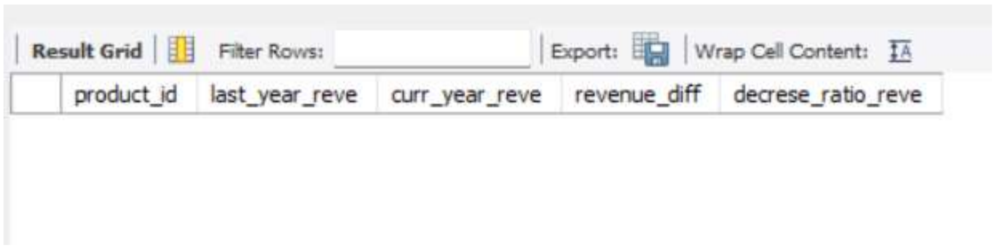
curr_year_sale as(select p.product_id,p.product_name,sum(oi.total_sales) as revenue

from orders o join order_item oi on o.order_ID=oi.order_ID join product p on oi.product_ID=p.product_ID where year(o.order_date)=2023 group by p.product_id,p.product_name)

select ls.product_id,ls.revenue as last_year_reve,cs.revenue as curr_year_reve,(ls.revenue-cs.revenue) as revenue_diff,

round((cs.revenue-ls.revenue)/ls.revenue*100,2) as decrese_ratio_reve from last_year_sale ls join curr_year_sale cs on

ls.product_id=cs.product_id where ls.revenue >cs.revenue;



	product_id	last_year_reve	curr_year_reve	revenue_diff	decrese_ratio_reve