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

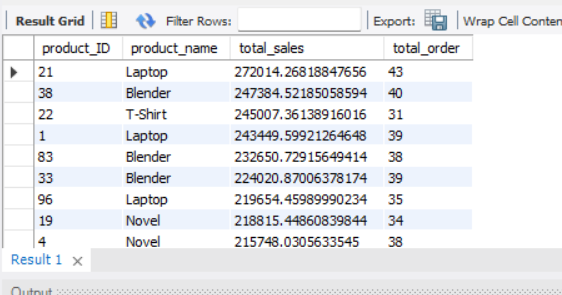
JOIN

product p ON p.product\_ID = oi.product\_ID

GROUP BY oi.product\_ID , p.product\_name

ORDER BY total\_sales DESC

LIMIT 10;



----- 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

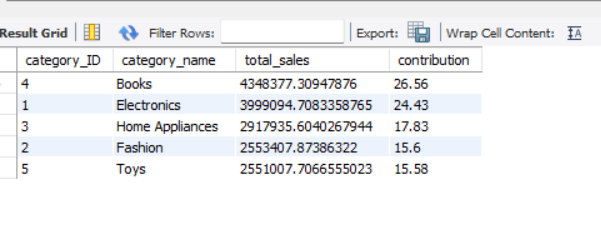
product p ON oi.product\_ID = p.product\_ID

LEFT JOIN

category c ON c.category\_ID = p.category\_ID

GROUP BY p.category\_ID , c.category\_name

ORDER BY total\_sales DESC;



----- 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

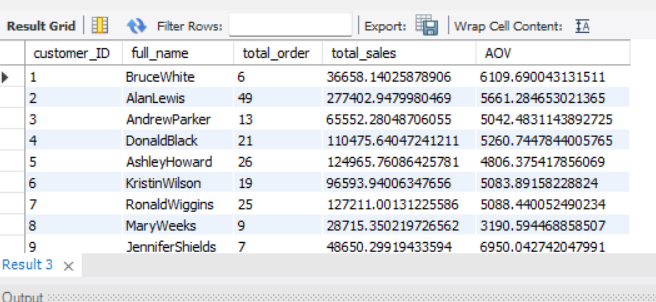
customer cu ON o.customer\_ID = cu.customer\_ID

JOIN

order\_item oi ON oi.order\_ID = o.order\_ID

GROUP BY cu.customer\_ID , full\_name

HAVING COUNT(o.order\_ID) >= 5;



------ 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

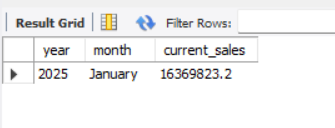
JOIN order\_item oi ON o.order\_ID = oi.order\_ID

WHERE

o.order\_date >= CURRENT\_DATE - INTERVAL 1 YEAR

GROUP BY MONTHNAME(o.order\_date) , YEAR(o.order\_date)

ORDER BY year , month) ti;



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

SELECT

\*

FROM

customer

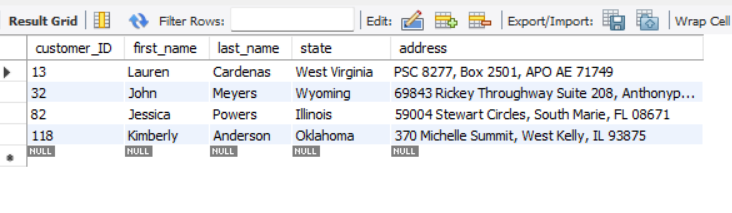
WHERE

customer\_ID NOT IN (SELECT DISTINCT

customer\_ID

FROM

orders);



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

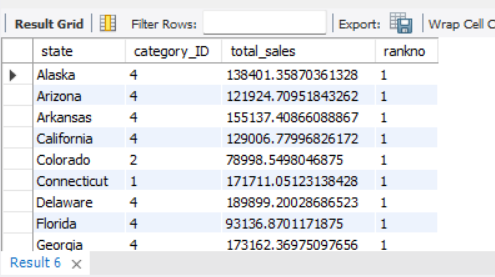
with ranking as( select c.state,cat.category\_ID,sum(oi.total\_sales) as total\_sales,

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;

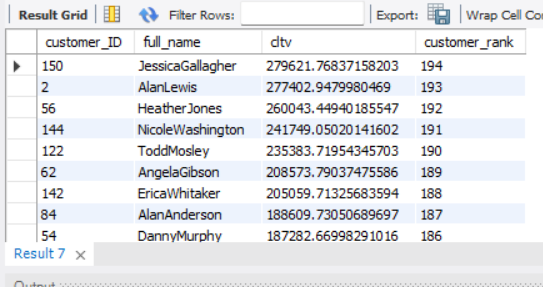


----- Customer lifetime value (cltv) >> calculate the total value of orders placed by each customer over their lifetime

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;



------ 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

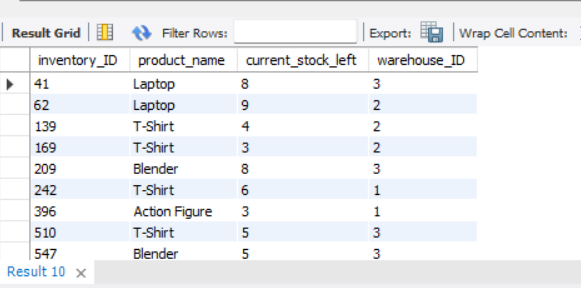
inventory i

JOIN

product p ON p.product\_ID = i.product\_ID

WHERE

stock < 10;



----- 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

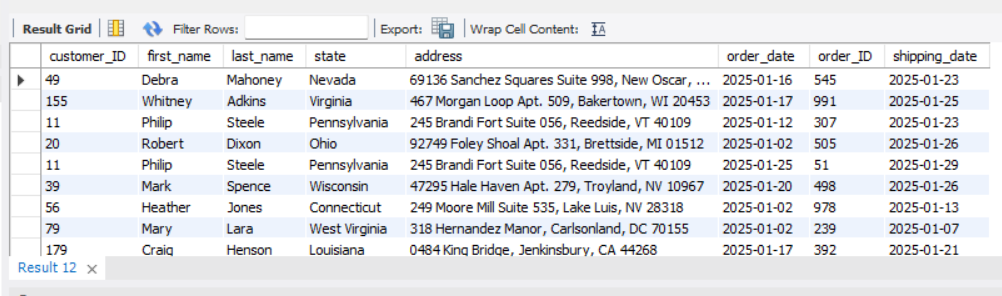
customer c ON o.customer\_ID = c.customer\_ID

JOIN

shipping s ON o.order\_ID = s.order\_ID

WHERE

s.shipping\_date - o.order\_date > 3;



------- 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

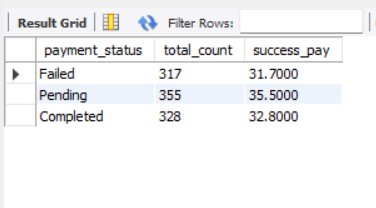
FROM

orders o

JOIN

payment pay ON o.order\_ID = pay.order\_ID

GROUP BY pay.payment\_status;



----- 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,

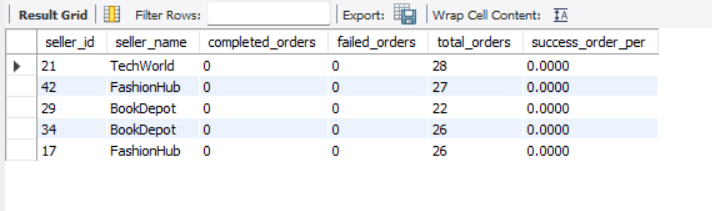
SUM(CASE WHEN order\_status = 'failed' THEN total\_orders ELSE 0 END) AS failed\_orders,

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;



----- Product profit margin >>> calculate the profit margin for each product (difference between price and cost of goods sold)>>>

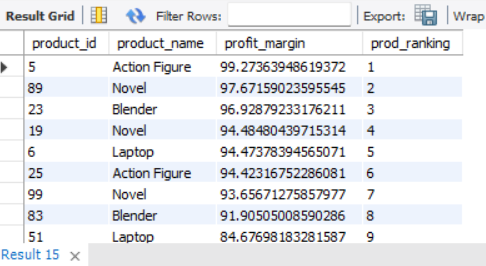
------ challenge: rank product by their profit margin showing highest of lowest

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;



------- 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

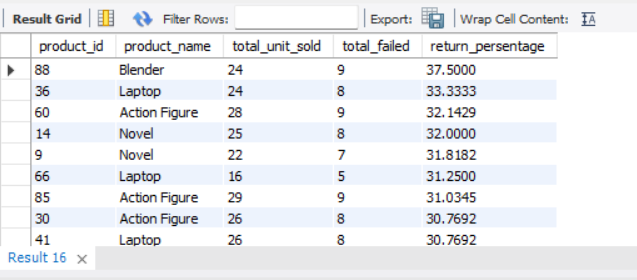
GROUP BY p.product\_id , p.product\_name

ORDER BY SUM(CASE

WHEN o.order\_status = 'Cancelled' THEN 1

ELSE 0

END) / COUNT(\*) \* 100 DESC;



------- 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

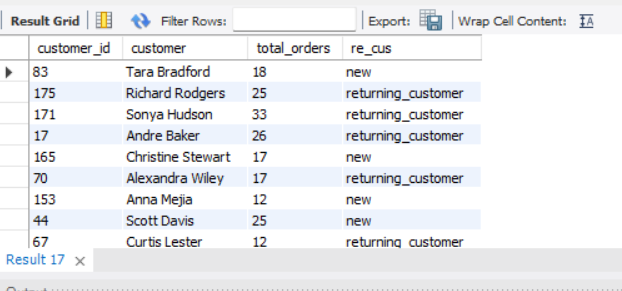
END) AS total\_return

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;



------ 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

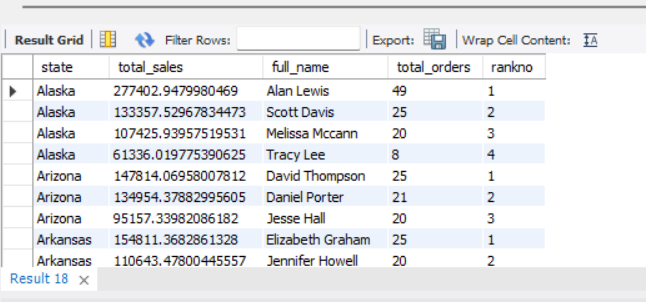
select \* from (select c.state,sum(oi.total\_sales) as total\_sales,

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;



------- 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

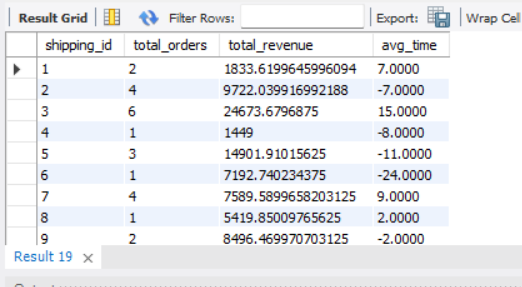
JOIN

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;



------ 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;

