```
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,s.seller_name,o.order_status,count(*) as total_orders from orders
o join top_seller ts on ts.seller_id=o.seller_id
where o.order_status not in ('pending') group by o.seller_ID,s.seller_name,o.order_status)

select seller_id,s.seller_name,sum(case when order_status='completed' then total_orders else 0
end) as completed_order,
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;