use job\_history\_data;

-- create the sales table

CREATE TABLE Sales (

sale\_id INT PRIMARY KEY NOT NULL,

product\_id INT NOT NULL,

quantity\_sold INT NOT NULL,

sale\_date DATE NOT NULL,

total\_price DECIMAL(10 , 2 )

);

-- Insert sample data into Sales table

INSERT INTO Sales (sale\_id, product\_id, quantity\_sold, sale\_date, total\_price) VALUES

(1, 101, 5, '2024-01-01', 2500.00),

(2, 102, 3, '2024-01-02', 900.00),

(3, 103, 2, '2024-01-02', 60.00),

(4, 104, 4, '2024-01-03', 80.00),

(5, 105, 6, '2024-01-03', 90.00);

-- Products Table

create table products(

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

category VARCHAR(50),

unit\_price DECIMAL(10 , 2 )

);

INSERT INTO Products (product\_id, product\_name, category, unit\_price) VALUES

(101, 'Laptop', 'Electronics', 500.00),

(102, 'Smartphone', 'Electronics', 300.00),

(103, 'Headphones', 'Electronics', 30.00),

(104, 'Keyboard', 'Electronics', 20.00),

(105, 'Mouse', 'Electronics', 15.00);

-- Retrieve all columns from the Sales table.

SELECT

\*

FROM

sales;

-- Retrieve the product\_name and unit\_price from the Products table.

SELECT

product\_name, unit\_price

FROM

products;

-- Retrieve the sale\_id and sale\_date from the Sales table.

SELECT

sales\_id, sales\_date

FROM

sales;

-- Filter the Sales table to show only sales with a total\_price greater than $100.

SELECT

\*

FROM

sales

WHERE

total\_price > 100;

-- Filter the Products table to show only products in the 'Electronics' category.

SELECT

\*

FROM

products

WHERE

category = 'electronics';

-- Retrieve the sale\_id and total\_price from the Sales table for sales made on January 3, 2024.

SELECT

sales\_id, total\_price

FROM

sales

WHERE

sales\_date = '2024-01-03';

-- Retrieve the product\_id and product\_name from the Products table for

-- products with a unit\_price greater than $100.

SELECT

product\_id, product\_name

FROM

products

WHERE

unit\_price > 100;

-- Calculate the total revenue generated from all sales in the Sales table.

SELECT

SUM(total\_price) AS total\_revenue

FROM

sales;

-- Calculate the average unit\_price of products in the Products table.

SELECT

AVG(unit\_price) AS avrage\_unit\_price

FROM

products;

-- Calculate the total quantity\_sold from the Sales table.

SELECT

SUM(quantity\_sold) AS total\_quantity\_sold

FROM

sales;

-- Count Sales Per Day from the Sales table

SELECT

sales\_date, COUNT(\*) AS sales\_count

FROM

sales

GROUP BY sales\_date

ORDER BY sales\_date;

-- retrieve product\_name and unit\_price from the Products table with the Highest Unit Price

SELECT

product\_name, unit\_price

FROM

products

ORDER BY unit\_price DESC

LIMIT 1;

-- Retrieve the sale\_id, product\_id, and total\_price from the Sales table

-- for sales with a quantity\_sold greater than 4.

SELECT

sales\_id, product\_id, total\_price

FROM

sales

WHERE

quantity\_sold > 4;

-- Retrieve the product\_name and unit\_price from the Products table,

-- ordering the results by unit\_price in descending order.

SELECT

product\_name, unit\_price

FROM

products

ORDER BY unit\_price DESC;

-- Retrieve the total\_price of all sales, rounding the values to two decimal places.

SELECT

ROUND(SUM(total\_price), 2) AS total\_sales

FROM

sales;

-- Calculate the average total\_price of sales in the Sales table.

SELECT

AVG(total\_price) AS avrage\_\_total\_price

FROM

sales;

-- Retrieve the sale\_id and sale\_date from the Sales table, formatting the sale\_date as 'YYYY-MM-DD'.

SELECT

DATE\_FORMAT(sales\_date, '%y-%m-%d') AS formatted\_sales\_date

FROM

sales;

-- Calculate the total revenue generated from sales of products in the 'Electronics' category.

SELECT

SUM(sales.total\_price) AS total\_revenue

FROM

sales

JOIN

products ON sales.product\_id = products.product\_id

WHERE

products.category = 'electronics';

-- Retrieve the product\_name and unit\_price from the Products table,

-- filtering the unit\_price to show only values between $20 and $600.

SELECT

product\_name, unit\_price

FROM

products

WHERE

unit\_price BETWEEN 20 AND 600;

-- Retrieve the product\_name and category from the Products table,

-- ordering the results by category in ascending order.

SELECT

product\_name, category

FROM

products

ORDER BY category ASC;

-- Calculate the total quantity\_sold of products in the 'Electronics' category.

SELECT

SUM(quantity\_sold) AS total\_quatity\_sold

FROM

sales

JOIN

products ON products.product\_id = sales.product\_id

WHERE

products.category = 'electronics';

SELECT

product\_name, quantity\_sold \* unit\_price AS total\_price

FROM

sales

JOIN

products ON products.product\_id = sales.product\_id;

SELECT

product\_id, COUNT(\*) AS sales\_count

FROM

sales

GROUP BY product\_id

ORDER BY sales\_count DESC

LIMIT 1;

SELECT

product\_id, product\_name

FROM

products

WHERE

product\_id NOT IN (SELECT DISTINCT

product\_id

FROM

sales);

SELECT

products.category, SUM(total\_price) AS total\_revenue

FROM

sales

JOIN

products ON products.product\_id = sales.product\_id

GROUP BY products.category;

SELECT

category

FROM

products

GROUP BY category

ORDER BY AVG(unit\_price) DESC

LIMIT 1;

SELECT

products.product\_name

FROM

sales

JOIN

products ON products.product\_id = sales.product\_id

GROUP BY products.product\_name

HAVING SUM(sales.total\_price) > 30;

SELECT

DATE\_FORMAT(sales.sales\_date, '%y-%m') AS month,

COUNT(\*) AS sales\_count

FROM

sales

GROUP BY month;

SELECT

sales\_id, total\_price, product\_name

FROM

sales

JOIN

products ON products.product\_id = sales.product\_Id

WHERE

product\_name LIKE '%smart%';

SELECT

AVG(sales.quantity\_sold) AS avrage\_quantity\_sold

FROM

sales

JOIN

products ON sales.product\_id = products.product\_id

WHERE

unit\_price > 100;

SELECT

products.product\_name,

SUM(sales.total\_price) AS total\_revenue

FROM

sales

JOIN

products ON sales.product\_id = products.product\_id

GROUP BY product\_name;

-- List all sales along with the corresponding product names.

SELECT

sales\_id, product\_name

FROM

sales

JOIN

products ON products.product\_id = sales.product\_id;

-- Retrieve the product name and total sales revenue for each product.

SELECT

products.category,

SUM(sales.total\_price) AS category\_revenue,

(SUM(sales.total\_price) / (SELECT

SUM(total\_price)

FROM

sales)) \* 100 AS revenue\_percentage

FROM

sales

JOIN

products ON products.product\_id = sales.product\_id

GROUP BY products.category

ORDER BY revenue\_percentage DESC

LIMIT 3;

select products.product\_name,

sum(total\_price) as sales\_revenue ,

rank() over (order by sum(sales.total\_price) desc) as revenue\_rank

from sales

join products on sales.product\_id =products.product\_id

group by products.product\_name;

select products.category, products.category, sales.sales\_date,

sum(sales.total\_price) over (partition by products.category order by sales.sales\_date) running\_total\_revenue

from sales

join products on products.product\_id=sales.product\_id;