CREATE DATABASE advanced\_sql\_mlc65;

USE advanced\_sql\_mlc65;

CREATE TABLE scores (

id INTEGER PRIMARY KEY,

player\_name TEXT NOT NULL,

score INTEGER NOT NULL

);

INSERT INTO scores (id,player\_name, score) VALUES

(1,'Alice', 100),

(2,'Bob', 90),

(3,'Charlie', 80),

(4,'Dave', 80),

(5,'Eve', 80),

(6,'Frank', 50),

(7,'Gloria', 40),

(8,'Henry', 30),

(9,'Igor', 20),

(10,'Jane', 10);

SELECT id,player\_name,score,

RANK() OVER(ORDER BY score DESC) as rank\_score

FROM scores;

SELECT id,player\_name,score,

DENSE\_RANK() OVER(ORDER BY score DESC) as dense\_rank\_score

FROM scores;

SELECT id,player\_name,score,

rank() over(ORDER BY score DESC) as rank\_score,

dense\_rank() over(ORDER BY score DESC) as dense\_rank\_score

from scores;

CREATE TABLE sales\_table (

date DATE,

product VARCHAR(255),

sales\_price DECIMAL(10,2),

product\_rating INT

);

INSERT INTO sales\_table (date, product, sales\_price, product\_rating)

VALUES ('2020-01-01', 'Product A', 100.00, 4),

('2020-01-02', 'Product A', 150.00, 3),

('2020-01-03', 'Product A', 200.00, 5),

('2020-01-04', 'Product A', 250.00, 2),

('2020-01-05', 'Product B', 100.00, 3),

('2020-01-06', 'Product B', 150.00, 4),

('2020-01-07', 'Product B', 200.00, 5),

('2020-01-08', 'Product C', 100.00, 2),

('2020-01-09', 'Product C', 150.00, 3),

('2020-01-10', 'Product C', 200.00, 4);

-- Write a sql query to rank the sales\_table basis increasing order of sales\_price of the product

SELECT date, product, sales\_price, product\_rating,

rank() over(order by sales\_price)

FROM sales\_table;

SELECT date, product, sales\_price, product\_rating,

rank() over(partition by product order by sales\_price)

FROM sales\_table;

-- Write a query to calculate the running total of sales for each product, ordered by date.

SELECT date,product, sales\_price,

SUM(sales\_price) OVER (PARTITION BY product ORDER BY date) as running\_total

FROM sales\_table;

SELECT date,product, sales\_price,

avg(sales\_price) OVER (PARTITION BY product ORDER BY date) as running\_total

FROM sales\_table;

-- Write a query to calculate the moving average sales price for each product,

-- ranked by product rating.

SELECT date, product, sales\_price, product\_rating,

AVG(sales\_price) OVER (PARTITION BY product ORDER BY product\_rating) as avg\_sales\_price

FROM sales\_table;

-- Write a sql code to find the count of player name whose dense rank is greator than 4

WITH CTE1 AS

(SELECT id,player\_name,score,

dense\_rank() over(ORDER BY score DESC) as dense\_rank\_score

from scores)

SELECT count(id) from CTE1 where dense\_rank\_score > 4 ;

CREATE TABLE sales (

id INT PRIMARY KEY,

product\_name VARCHAR(50),

quantity INT,

price DECIMAL(10, 2),

date\_purchased DATE

);

INSERT INTO sales (id, product\_name, quantity, price, date\_purchased) VALUES

(1, 'Product A', 5, 10.99, '2023-06-01'),

(2, 'Product B', 2, 5.99, '2023-06-02'),

(3, 'Product C', 3, 8.99, '2023-06-03'),

(4, 'Product A', 4, 10.99, '2023-06-04'),

(5, 'Product B', 1, 5.99, '2023-06-05'),

(6, 'Product C', 2, 8.99, '2023-06-06'),

(7, 'Product A', 6, 10.99, '2023-06-07'),

(8, 'Product B', 3, 5.99, '2023-06-08'),

(9, 'Product C', 5, 8.99, '2023-06-09'),

(10, 'Product A', 2, 10.99, '2023-06-10'),

(11, 'Product B', 1, 5.99, '2023-06-11'),

(12, 'Product C', 4, 8.99, '2023-06-12'),

(13, 'Product A', 3, 10.99, '2023-06-13'),

(14, 'Product B', 2, 5.99, '2023-06-14'),

(15, 'Product C', 6, 8.99, '2023-06-15'),

(16, 'Product A', 4, 10.99, '2023-06-16'),

(17, 'Product B', 3, 5.99, '2023-06-17'),

(18, 'Product C', 1, 8.99, '2023-06-18'),

(19, 'Product A', 5, 10.99, '2023-06-19'),

(20, 'Product B', 2, 5.99, '2023-06-20');

-- Case Statements

-- Q: Create a column tagging against every row whether the quantity is high,medium or low

-- using below criteria: high if quantity >=5, medium if quantity <5 and >2 else low quantity.

SELECT \*,

CASE

WHEN quantity >=5 THEN 'high'

WHEN quantity >2 and quantity <5 THEN 'medium'

ELSE 'low'

END AS category

FROM sales;

-- Q: Calculate the final price of the product after adding the taxes:

-- if price>6 and price<=9 then tax is 10% of the price

-- if price>9 then tax is 8%

-- if else it is 15%

SELECT \*,

CASE

WHEN price >9 THEN price\*1.08

WHEN price>6 AND price<=9 THEN price\*1.1

ELSE price\*1.15

END AS final\_price

FROM sales;

-- UDFs --> User Defined Functions

DELIMITER $$

CREATE FUNCTION summation (product VARCHAR(50))-- DATATYPE OF INPUT

RETURNS DECIMAL -- DATATYPE OF OUTPUT

DETERMINISTIC

BEGIN

DECLARE output DECIMAL;

SELECT sum(price) INTO output FROM sales where product\_name = product;

RETURN output;

END $$

DELIMITER ;

SELECT summation('Product B');

-- Stored Procedure

DELIMITER $$

CREATE PROCEDURE quant (product VARCHAR(50))

BEGIN

SELECT quantity FROM sales where product\_name = product;

END $$

DELIMITER ;

CALL quant('Product B');

-- Q: Write a sql query to create the stored procedure which will take month as an input

-- and return the product names for given month. Also write procedure call

-- June month.

DELIMITER $$

CREATE PROCEDURE prod\_name (MON VARCHAR(50))

BEGIN

SELECT product\_name FROM sales WHERE MONTHNAME(date\_purchased) = MON;

END $$

DELIMITER ;

CALL prod\_name ('June');