//view

CREATE VIEW high\_salary\_employees AS

SELECT emp\_id, emp\_name

FROM employees

WHERE emp\_salary > 50000;

//table

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100),

emp\_salary DECIMAL(10, 2),

hire\_date DATE

);

//proc

DELIMITER //

CREATE PROCEDURE get\_employee\_info (IN emp\_id\_param INT)

BEGIN

SELECT \* FROM employees WHERE emp\_id = emp\_id\_param;

END //

DELIMITER ;

//parameters

CREATE PROCEDURE get\_employee\_info (IN emp\_id\_param INT)

BEGIN

SELECT \* FROM employees WHERE emp\_id = emp\_id\_param;

END;

CREATE PROCEDURE get\_employee\_count (OUT emp\_count INT)

BEGIN

SELECT COUNT(\*) INTO emp\_count FROM employees;

END;

CREATE PROCEDURE update\_employee\_salary (INOUT emp\_id\_param INT, IN new\_salary DECIMAL(10, 2))

BEGIN

UPDATE employees SET emp\_salary = new\_salary WHERE emp\_id = emp\_id\_param;

SET emp\_id\_param = emp\_id\_param + 1;

END;

//if else if

DECLARE @num INT = 0;

IF @num > 0 THEN

SELECT 'Positive';

ELSEIF @num < 0 THEN

SELECT 'Negative';

ELSE

SELECT 'Zero';

END IF;

//if loop

DECLARE @counter INT = 1;

LOOP

IF @counter > 10 THEN

LEAVE; -- Exit the loop

END IF;

SELECT @counter;

SET @counter = @counter + 1;

END LOOP;

//case

DECLARE @grade CHAR(1) = 'B';

CASE @grade

WHEN 'A' THEN SELECT 'Excellent'

WHEN 'B' THEN SELECT 'Good'

WHEN 'C' THEN SELECT 'Fair'

ELSE SELECT 'Poor'

END CASE;

//while lopp

DECLARE @counter INT = 1;

WHILE @counter <= 10 DO

SELECT @counter;

SET @counter = @counter + 1;

END WHILE;

//repeat-until

DECLARE @counter INT DEFAULT 1;

REPEAT

SELECT @counter;

SET @counter = @counter + 1;

UNTIL @counter > 10 END REPEAT;

//funtocn

CREATE FUNCTION calculate\_discount(price DECIMAL(10, 2), discount\_rate DECIMAL(4, 2))

RETURNS DECIMAL(10, 2)

BEGIN

DECLARE discounted\_price DECIMAL(10, 2);

SET discounted\_price = price - (price \* discount\_rate / 100);

RETURN discounted\_price;

END;

DROP FUNCTION IF EXISTS calculate\_discount;

SHOW FUNCTION STATUS WHERE Db = 'your\_database\_name';

**TASK 1:**

**Create a stored procedure that accepts an employee ID as input parameter, and returns a result**

**set of all the orders made by that employee, including the order ID, order date, and total order**

**amount.**

DELIMITER //

CREATE PROCEDURE GetEmployeeorders(IN EmployeeID INT)

BEGIN

SELECT

OrderID,

OrderDate,

(SELECT UnitPrice from `order details`)

FROM

Orders

WHERE

EmployeeID = EmployeeID

GROUP BY

OrderID, OrderDate;

END //

DELIMITER ;

**TASK 2:**

**Suppose you want to calculate the total revenue for a given order. You can create a function**

**that takes an order ID as input, calculates the total revenue for that order by multiplying the**

**unit price of each product by the quantity ordered, and returns the result.**

CREATE FUNCTION CalculateOrderRevenue(orderID INT) RETURNS DECIMAL(10,2)

BEGIN

DECLARE totalRevenue DECIMAL(10,2);

SELECT SUM(UnitPrice \* Quantity) INTO totalRevenue

FROM OrderDetails

WHERE OrderID = orderID;

RETURN totalRevenue;

END //

DELIMITER ;

SELECT CalculateOrderRevenue(10260);

**TASK 3:**

**Define a function called square that accepts an integer as input parameter, calculates the square of the integer, and returns the result.**

CREATE FUNCTION square(inputInteger INT) RETURNS INT DETERMINISTIC

BEGIN

DECLARE result INT;

SET result = inputInteger \* inputInteger;

RETURN result;

END //DELIMITER ;

SELECT square(5);

**TASK 4:**

**Define a stored procedure called print\_squares that uses a WHILE loop to loop through the**

**integers 1 to 10, calling the square function for each integer and printing the result to the**

**console.**

DELIMITER //

CREATE PROCEDURE print\_squares()

BEGIN

DECLARE i INT DEFAULT 1;

WHILE i <= 10 DO

SELECT CONCAT('Square of ', i, ' is ', square(i)) AS Result;

SET i = i + 1;

END WHILE;

END //

DELIMITER ;

CALL print\_squares();