MYSQL ASSIGNMENTS

Q8. Views in SQL

1. Create a view named product\_category\_sales that provides insights into sales performance by product category.

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `product\_category\_sales` AS

SELECT

`p`.`productLine` AS `productLine`,

SUM((`od`.`quantityOrdered` \* `od`.`priceEach`)) AS `total\_sales`,

COUNT(DISTINCT `o`.`orderNumber`) AS `number\_of\_orders`

FROM

(((`productlines` `p`

JOIN `products` `pr` ON ((`p`.`productLine` = `pr`.`productLine`)))

JOIN `orderdetails` `od` ON ((`pr`.`productCode` = `od`.`productCode`)))

JOIN `orders` `o` ON ((`od`.`orderNumber` = `o`.`orderNumber`)))

GROUP BY `p`.`productLine`

SELECT \* FROM classicmodels.product\_category\_sales;

Q9. Stored Procedures in SQL with parameters

1. Create a stored procedure Get\_country\_payments which takes in year and country as inputs and gives year wise, country wise total amount as an output. Format the total amount to nearest thousand unit (K)

CREATE DEFINER=`root`@`localhost` PROCEDURE `Get\_country\_payments`(

IN input\_year INT,

IN input\_country VARCHAR(50)

)

BEGIN

SELECT

input\_year AS Year, -- Explicitly use the input year

c.country AS Country,

CONCAT(ROUND(SUM(p.amount) / 1000), 'K') AS TotalAmount

FROM

Payments p

JOIN

Customers c ON p.customerNumber = c.customerNumber

WHERE

YEAR(p.paymentDate) = input\_year

AND c.country = input\_country

GROUP BY

c.country; -- Group only by country

END

call classicmodels.Get\_country\_payments(2003, 'France');

Q12. ERROR HANDLING in SQL

CREATE TABLE Emp\_EH (

EmpID INT PRIMARY KEY,

EmpName NVARCHAR(100),

EmailAddress NVARCHAR(255)

);

CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertEmp\_EH`(

IN p\_EmpID INT,

IN p\_EmpName NVARCHAR(100),

IN p\_EmailAddress NVARCHAR(255)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

-- Error handling block

SELECT 'Error occurred' AS ErrorMessage;

END;

-- Insert the data into the table

INSERT INTO Emp\_EH (EmpID, EmpName, EmailAddress)

VALUES (p\_EmpID, p\_EmpName, p\_EmailAddress);

-- Success message

SELECT 'Insert successful' AS SuccessMessage;

END

call classicmodels.InsertEmp\_EH(1, 'Sahana', 'iamsahana@gmail.com');

call classicmodels.InsertEmp\_EH(1, 'Sahana', 'iamsahana@gmail.com');

Q13. TRIGGERS

CREATE TABLE Emp\_BIT (

Name NVARCHAR(100),

Occupation NVARCHAR(100),

Working\_date DATE,

Working\_hours INT

);

INSERT INTO Emp\_BIT VALUES

('Robin', 'Scientist', '2020-10-04', 12),

('Warner', 'Engineer', '2020-10-04', 10),

('Peter', 'Actor', '2020-10-04', 13),

('Marco', 'Doctor', '2020-10-04', 14),

('Brayden', 'Teacher', '2020-10-04', 12),

('Antonio', 'Business', '2020-10-04', 11);

CREATE DEFINER=`root`@`localhost` TRIGGER `emp\_bit\_BEFORE\_INSERT` BEFORE INSERT ON `emp\_bit` FOR EACH ROW BEGIN

-- Check and modify Working\_hours to be positive

IF NEW.Working\_hours < 0 THEN

SET NEW.Working\_hours = ABS(NEW.Working\_hours);

END IF;

END

-- Insert data with negative Working\_hours to test the trigger

INSERT INTO Emp\_BIT VALUES ('Lucy', 'Artist', '2020-10-04', -8);

-- Verify the result

SELECT \* FROM Emp\_BIT;