Assignment Questions

**Note: -**

1. The tables mentioned in the questions for the references are available in the classic model database.

2. In the questions, if they specifically mention creating the tables, you need to make the tables as per the given specifications.

3. Solve all the assignment questions, organized by topic wise, within a single query tab only. Kindly submit your MySQL assignments in a single file, using a format that suits you best, such as a. SQL file, Notepad, or Word document.

|  |
| --- |
|  |
|  |
| **Q1. SELECT clause with WHERE, AND, DISTINCT, Wild Card (LIKE)**   1. Fetch the employee number, first name and last name of those employees who are working as Sales Rep reporting to an employee with employeenumber 1102 **(Refer employee table)**   **Expected output:**     1. Show the unique productline values containing the word cars at the end from the **products table**.   **Expected output:**    **# Q2 . a. Using a CASE statement, segment customers into three categories based on their country:(Refer Customers table)**  "North America" for customers from USA or Canada  "Europe" for customers from UK, France, or Germany  "Other" for all remaining countries  Select the customerNumber, customerName, and the assigned region as "CustomerSegment".  **Expected output:** |
| **Q3. Group By with Aggregation functions and Having clause, Date and Time functions**   1. Using the **OrderDetails table**, identify the top 10 products (by productCode) with the highest total order quantity across all orders.   **Expected output:**     1. Company wants to analyse payment frequency by month. Extract the month name from the payment date to count the total number of payments for each month and include only those months with a payment count exceeding 20. Sort the results by total number of payments in descending order. **(Refer Payments table).**   Expected output: |
|  |
|  |

**Q4. CONSTRAINTS: Primary, key, foreign key, Unique, check, not null, default**

Create a new database named and **Customers\_Orders** and add the following tables as per the description

1. Create a table named **Customers** to store customer information. Include the following columns:

customer\_id: This should be an integer set as the PRIMARY KEY and AUTO\_INCREMENT.

first\_name: This should be a VARCHAR(50) to store the customer's first name.

last\_name: This should be a VARCHAR(50) to store the customer's last name.

email: This should be a VARCHAR(255) set as UNIQUE to ensure no duplicate email addresses exist.

phone\_number: This can be a VARCHAR(20) to allow for different phone number formats.

Add a NOT NULL constraint to the first\_name and last\_name columns to ensure they always have a value.

1. Create a table named **Orders** to store information about customer orders. Include the following columns:

order\_id: This should be an integer set as the PRIMARY KEY and AUTO\_INCREMENT.

customer\_id: This should be an integer referencing the customer\_id in the Customers table (FOREIGN KEY).

order\_date: This should be a DATE data type to store the order date.

total\_amount: This should be a DECIMAL(10,2) to store the total order amount.

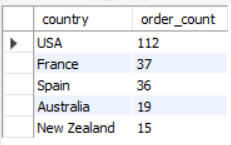
Constraints:

1. Set a FOREIGN KEY constraint on customer\_id to reference the Customers table.
2. Add a CHECK constraint to ensure the total\_amount is always a positive value.

**Q5. JOINS**

a. List the top 5 countries (by order count) that Classic Models ships to. (**Use the Customers and Orders tables**)

**Expected output:**

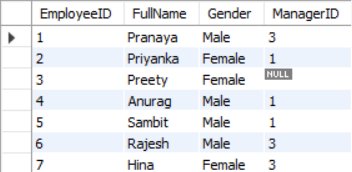
****

**Q6. SELF JOIN**

a. Create a table **project** with below fields.

* EmployeeID : integer set as the PRIMARY KEY and AUTO\_INCREMENT.
* FullName: varchar(50) with no null values
* Gender : Values should be only ‘Male’ or ‘Female’
* ManagerID: integer

Add below data into it.



Find out the names of employees and their related managers.

**Expected output:**



**Q7. DDL Commands: Create, Alter, Rename**

a. Create table facility. Add the below fields into it.

* Facility\_ID
* Name
* State
* Country

i) Alter the table by adding the primary key and auto increment to Facility\_ID column.

ii) Add a new column city after name with data type as varchar which should not accept any null values.



**Q8. Views in SQL**

a. Create a view named product\_category\_sales that provides insights into sales performance by product category. This view should include the following information:

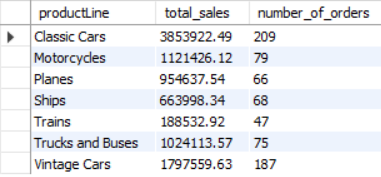
**productLine**: The category name of the product (from the ProductLines table).

**total\_sales**: The total revenue generated by products within that category (calculated by summing the orderDetails.quantity \* orderDetails.priceEach for each product in the category).

**number\_of\_orders**: The total number of orders containing products from that category.

(Hint: Tables to be used: Products, orders, orderdetails and productlines)

The view when read should show the output as:



**Q9. Stored Procedures in SQL with parameters**

a. 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)

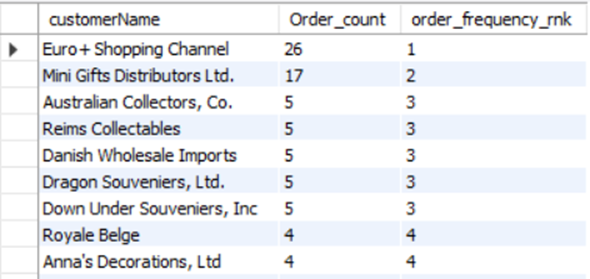
Tables: Customers, Payments

**Expected output:**



**Q10. Window functions - Rank, dense\_rank, lead and lag**

a) Using customers and orders tables, rank the customers based on their order frequency



b) Calculate year wise, month name wise count of orders and year over year (YoY) percentage change. Format the YoY values in no decimals and show in % sign.

Table: Orders

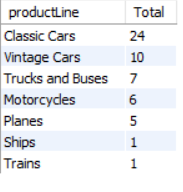
**Expected output:**



**Q11.Subqueries and their applications**

a. Find out how many product lines are there for which the buy price value is greater than the average of buy price value. Show the output as product line and its count.

**Expected output:**



**Q12. ERROR HANDLING in SQL**

Create the table Emp\_EH. Below are its fields.

* EmpID (Primary Key)
* EmpName
* EmailAddress

Create a procedure to accept the values for the columns in Emp\_EH. Handle the error using exception handling concept. Show the message as “Error occurred” in case of anything wrong.

**Q13. TRIGGERS**

Create the table Emp\_BIT. Add below fields in it.

* Name
* Occupation
* Working\_date
* Working\_hours

Insert the data as shown in below query.

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 before insert trigger to make sure any new value of Working\_hours, if it is negative, then it should be inserted as positive.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
  
show databases;

use classicmodels;

# Q1. a. Fetch the employee number, first name and last name of those employees who are working as Sales Rep reporting to employee with employeenumber 1102 (Refer employee table)

select \* FROM classicmodels.employees;

select employeenumber, firstname, lastname from employees WHERE jobtitle="sales rep" AND REPORTSTO = 1102;

# b. Show the unique productline values containing the word cars at the end from the products table.

#distinct for remove duplicate

select \* from products;

select distinct productline from products where productLine like "%cars" ;

# Q2. a.

select \* from customers;

select customerNumber, customername,

case

when country in ("USA" , "canada") then "North America"

when country in ("uk","france" ,"germany") then "Europe"

else "other"

end as "CustomerSegment"

from customers;

# Q3. a.

SELECT \* FROM orderdetails;

SELECT productCode, sum(quantityOrdered) as Total\_Ordered from orderdetails group by productcode order by Total\_Ordered desc limit 10;

# Q3. b.

select \* from payments;

select date\_format(paymentdate, "%M") as Payment\_Month, count(\*) as Num\_Payment from payments group by Payment\_Month having Num\_Payment >20 order by Num\_Payment desc;

# Q4. a.

SHOW databases;

create database Customers\_Order;

use Customers\_Order;

Create table Customers(Customer\_Id int primary key auto\_increment, First\_Name varchar(50) not null, Last\_Name varchar(50) not null, Email varchar (255) unique, Phone\_Number varchar(20));

desc customers;

# Q4. b.

Create table Orders(Order\_Id int primary key auto\_increment, Customer\_Id int , Order\_date date, Total\_Amount decimal (10,2),

foreign key(Customer\_Id) references customers(Customer\_Id),check (Total\_Amount >0));

desc Orders;

# Q5

use classicmodels;

select \* from Customers;

select \* from Orders;

SELECT c.country, COUNT(o.ordernumber) AS order\_count

FROM Customers c

JOIN Orders o ON c.customernumber = o.customernumber

GROUP BY c.country

ORDER BY order\_count DESC

LIMIT 5;

# Q6.

use Classicmodels;

create table Project (EmployeeID int primary key auto\_increment, FullName varchar(50) not null, Gender varchar (6) check ( gender in ("Male" , "Female")), ManagerID int);

drop table project;

create table Project (EmployeeID int primary key auto\_increment, FullName varchar(50) not null, Gender enum ("Male" , "Female") not null, ManagerID int);

desc project;

truncate table project;

insert into project (Fullname, Gender, ManagerID) values ( "Pranaya", "Male",3),

('Priyanka', 'Female', 1),

('Preety', 'Female', null),

('Anurag', 'Male', 1),

('Sambit', 'Male', 1),

('Rajesh', 'Male', 3),

('Hina', 'Female', 3);

select \* from project;

select m.fullname as "Manager Name", e.Fullname as "Emp Name" from Project e

join project m on e.ManagerId = m.EmployeeId order by "Manager Name", "Emp name";

# Q7.

create table Facility (Facility\_ID int, Name varchar (100), State varchar (100), Country varchar(100));

desc facility;

alter table facility modify column Facility\_ID int primary key auto\_increment not null;

alter table facility add column City varchar (100) not null after Name;

#

alter table facility modify facility\_id varchar (50) primary key auto\_increment ;

# Q8.

CREATE VIEW product\_category\_sales AS

SELECT

pl.productLine AS productLine,

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

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

FROM

Products p

JOIN

ProductLines pl ON p.productLine = pl.productLine

JOIN

OrderDetails od ON p.productCode = od.productCode

JOIN

Orders o ON od.orderNumber = o.orderNumber

GROUP BY

pl.productLine;

select \* from product\_category\_sales;

# Q9.

select \* from customers;

select \* from payments;

DELIMITER //

CREATE PROCEDURE Get\_country\_payments(IN input\_year INT, IN input\_country VARCHAR(50))

BEGIN

SELECT

YEAR(p.paymentDate) AS Year,

c.country AS Country,

CONCAT(FORMAT(SUM(p.amount) / 1000, 0), '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

YEAR(p.paymentDate), c.country;

END //

DELIMITER ;

call Get\_country\_payments(2003, "France");

# Q10. a.

select \* from customers;

select \* from orders;

SELECT

c.customerName,

COUNT(o.ordernumber) AS order\_count,

dense\_rank() OVER (ORDER BY COUNT(o.ordernumber) DESC) AS order\_frequency\_rnk

FROM

customers c

JOIN

orders o ON c.customernumber = o.customerNumber

GROUP BY

c.customerName

ORDER BY

order\_frequency\_rnk;

# Q10. b.

select \* from orders;

select

year(orderDate) as order\_year,

Monthname(orderdate) as order\_month,

count(orderNumber) as order\_count,

concat(round(((count(ordernumber)-lag(count(orderNumber),1)over())/lag(count(orderNumber),1)over())\*100),"%")as "% YOY change"

from orders

group by year(orderDate),monthname(orderdate);

# Q11.

select \* from products;

SELECT productline, COUNT(\*) AS count

FROM products

WHERE buyprice > (SELECT AVG(buyprice) FROM products)

GROUP BY productline;

# Q12.

use classicmodels;

create table emp\_eh (

empid int primary key,

empname varchar(100),

emailaddress varchar(100)

);

Select \* from emp\_eh;

delimiter //

create procedure insertintoemp\_eh (

in p\_empid int,

in p\_empname varchar(100),

in p\_emailaddress varchar(100)

)

begin

declare exit handler for sqlexception

begin

-- handle the error

rollback;

select 'error occurred' as errormessage;

end;

start transaction;

-- attempt to insert the values

insert into emp\_eh (empid, empname, emailaddress)

values (p\_empid, p\_empname, p\_emailaddress);

commit;

end //

delimiter ;

call insertintoemp\_eh(1, 'john doe', 'john.doe@example.com');

drop table emp\_eh;

drop procedure insertintoemp\_eh;

# Q13.

drop table emp\_bit;

create table Emp\_BIT( Name varchar(100), Occupation varchar (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);

select \* from emp\_bit;

DELIMITER //

CREATE TRIGGER before\_insert\_emp\_bit

BEFORE INSERT ON Emp\_BIT

FOR EACH ROW

BEGIN

IF NEW.Working\_hours < 0 THEN

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

END IF;

END //

DELIMITER ;

insert into emp\_bit (name, occupation, working\_date, working\_hours) values

("Mahesh", "Sales Person" , "2020-10-04", -10);

select \* from emp\_BIT;