Day wise Assignment Questions

**Note: -**

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

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

|  |
| --- |
| **Day 1**  No questions |
| **Day 2**  No questions |
| **Day 3**   1. Show customer number, customer name, state and credit limit from customers table for below conditions. Sort the results by highest to lowest values of creditLimit.  * State should not contain null values * credit limit should be between 50000 and 100000   **select customerNumber , customerName , state , creditLimit from customers**  **where state is not null and creditLimit between 50000 and 100000**  **order by creditLimit desc ;**  **Expected output:**     1. Show the unique productline values containing the word cars at the end from products table.   **Expected output:**  **select distinct productline from products**  **where productline like '%cars%' ;**    **Day 4**   1. Show the orderNumber, status and comments from orders table for shipped status only. If some comments are having null values then show them as “-“.   **Expected output:**  **select ordernumber, status, coalesce(comments, "-") as comments from orders where status = "shipped";**     1. Select employee number, first name, job title and job title abbreviation from employees table based on following conditions.   If job title is one among the below conditions, then job title abbreviation column should show below forms.   * President then “P” * Sales Manager / Sale Manager then “SM” * Sales Rep then “SR” * Containing VP word then “VP”   select employeenumber, firstname,jobtitle,  case  when jobTitle = "president" THEN "P"  WHEN jobTitle = "Manager" THEN "SM"  WHEN jobTitle = "Sales Rep" THEN "SR"  WHEN jobTitle = "VP" THEN "VP"  ELSE jobTitle  END AS jobTitle\_Abbreviation  FROM employees;  **Expected output:** |
| **Day 5:**   1. For every year, find the minimum amount value from payments table.   **Expected output:**  select year(paymentdate) as Year,  min(amount) as 'Min Amount'  from payments  group by year  order by year;   1. For every year and every quarter, find the unique customers and total orders from orders table. Make sure to show the quarter as Q1,Q2 etc.   **Expected output:**  SELECT YEAR(orderDate) AS Year,  CONCAT('Q', QUARTER(orderDate)) AS quarter,  COUNT(DISTINCT customerNumber) AS 'unique Customers',  COUNT(orderNumber) AS 'Total Orders'  FROM orders  GROUP BY Year, quarter  ORDER BY Year, quarter;       1. Show the formatted amount in thousands unit (e.g. 500K, 465K etc.) for every month (e.g. Jan, Feb etc.) with filter on total amount as 500000 to 1000000. Sort the output by total amount in descending mode. [ Refer. Payments Table]   **Expected output:**  SELECT  DATE\_FORMAT(paymentDate, '%b') AS Month, CONCAT(ROUND(SUM(amount) / 1000), 'K') AS 'formatted Amount'  FROM payments  GROUP BY month  HAVING SUM(amount) BETWEEN 500000 AND 1000000  ORDER BY SUM(amount) DESC; |
|  |
|  |

**Day 6:**

1. Create a journey table with following fields and constraints.

* Bus\_ID (No null values)
* Bus\_Name (No null values)
* Source\_Station (No null values)
* Destination (No null values)
* Email (must not contain any duplicates)

create table journey

( BUS\_ID int(150) NOT NULL,

BUS\_NAME VARCHAR(150) NOT NULL ,

Source\_Station VARCHAR(150) NOT NULL,

Destination VARCHAR(150) NOT NULL,

Email VARCHAR(150) NOT NULL );

select \* from journey ;

1. Create vendor table with following fields and constraints.

* Vendor\_ID (Should not contain any duplicates and should not be null)
* Name (No null values)
* Email (must not contain any duplicates)
* Country (If no data is available then it should be shown as “N/A”)

create table Vendor (

Vendor\_ID int(150) not null ,

Name varchar(100) not null,

Email varchar(75) not null unique,

Country varchar(100) default "N/A"

);

select \* from Vendor ;

1. Create movies table with following fields and constraints.

* Movie\_ID (Should not contain any duplicates and should not be null)
* Name (No null values)
* Release\_Year (If no data is available then it should be shown as “-”)
* Cast (No null values)
* Gender (Either Male/Female)
* No\_of\_shows (Must be a positive number)

create table movies (

Movies\_ID int not null ,

Name varchar(100) not null,

Release\_Year int,

Cast varchar(50) not null,

Gender enum("Male","Female") not null,

No\_of\_Shows int check (no\_of\_shows > 0)

);

select \* from movies ;

1. Create the following tables. Use auto increment wherever applicable

a. Product

* product\_id - primary key
* product\_name - cannot be null and only unique values are allowed
* description
* supplier\_id - foreign key of supplier table

create table Suppliers(

Supplier\_id int (150) auto\_increment primary key,

supplier\_name varchar(250) not null,

location varchar(250)

);

select \* from Suppliers;

b. Suppliers

* + supplier\_id - primary key
  + supplier\_name
  + location

create table product(

product\_Id int (150) auto\_increment primary key,

Product\_Name varchar(250) not null unique,

description text ,

supplier\_id1 int ,

foreign key (supplier\_id1) references Suppliers(supplier\_id)

);

select \* from product;

c. Stock

* + id - primary key
  + product\_id - foreign key of product table
  + balance\_stock

create table Stock(

id int (150) auto\_increment primary key,

product\_id1 int ,

balance\_stock varchar(50),

foreign key (product\_id1) references product(product\_id)

);

select \* from Stock ;

**Day 7**

1. Show employee number, Sales Person (combination of first and last names of employees), unique customers for each employee number and sort the data by highest to lowest unique customers.

Tables: Employees, Customers

select \* from Employees ;

select\* from Customers ;

SELECT

e.employeeNumber AS employeeNumber,

CONCAT(e.firstName, ' ', e.lastName) AS SalesPerson,

COUNT(DISTINCT c.customerNumber) AS uniqueCustomers

FROM Employees e

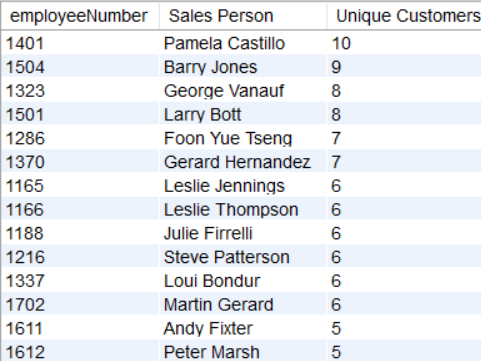
LEFT JOIN

Customers c ON e.employeeNumber = c.salesRepEmployeeNumber

GROUP BY e.employeeNumber, SalesPerson

ORDER BY uniqueCustomers DESC;

**Expected output:**



1. Show total quantities, total quantities in stock, left over quantities for each product and each customer. Sort the data by customer number.

Tables: Customers, Orders, Orderdetails, Products

select \*from customers ;

select \*FROM ORDERS ;

select \*from ORDERDETAILS ;

select\*from PRODUCTS ;

SELECT

c.customerNumber AS CustomerNumber,

c.customerName AS CustomerName,

p.productCode AS ProductCode,

p.productName AS ProductName,

SUM(od.quantityOrdered) AS Ordered\_Qty,

IFNULL(p.quantityInStock, 0) AS Total\_inventory,

IFNULL(p.quantityInStock - SUM(od.quantityOrdered), 0) AS Left\_Qty

FROM Customers c

JOIN Orders o ON c.customerNumber = o.customerNumber

JOIN OrderDetails od ON o.orderNumber = od.orderNumber

JOIN Products p ON od.productCode = p.productCode

LEFT JOIN products s ON p.productCode = s.productCode

GROUP BY c.customerNumber, p.productCode

ORDER BY c.customerNumber;

**Expected output:**

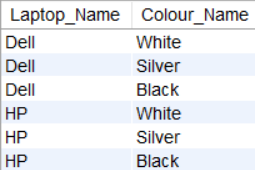


1. Create below tables and fields. (You can add the data as per your wish)

* Laptop: (Laptop\_Name)
* Colours: (Colour\_Name)

Perform cross join between the two tables and find number of rows.

**Expected output:**



1. Create table project with below fields.

* EmployeeID
* FullName
* Gender
* ManagerID

Add below data into it.

INSERT INTO Project VALUES(1, 'Pranaya', 'Male', 3);

INSERT INTO Project VALUES(2, 'Priyanka', 'Female', 1);

INSERT INTO Project VALUES(3, 'Preety', 'Female', NULL);

INSERT INTO Project VALUES(4, 'Anurag', 'Male', 1);

INSERT INTO Project VALUES(5, 'Sambit', 'Male', 1);

INSERT INTO Project VALUES(6, 'Rajesh', 'Male', 3);

INSERT INTO Project VALUES(7, 'Hina', 'Female', 3);

Find out the names of employees and their related managers.

**Expected output:**



**Day 8**

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.

**Expected output:**



**Day 9**

Create table university with below fields.

* ID
* Name

Add the below data into it as it is.

INSERT INTO University

VALUES (1, " Pune University "),

(2, " Mumbai University "),

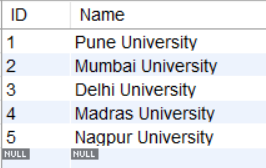
(3, " Delhi University "),

(4, "Madras University"),

(5, "Nagpur University");

Remove the spaces from everywhere and update the column like Pune University etc.

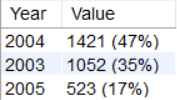
**Expected output:**



**Day 10**

Create the view products status. Show year wise total products sold. Also find the percentage of total value for each year. The output should look as shown in below figure.

**Expected output:**



**Day 11**

1. Create a stored procedure GetCustomerLevel which takes input as customer number and gives the output as either Platinum, Gold or Silver as per below criteria.

Table: Customers

* Platinum: creditLimit > 100000
* Gold: creditLimit is between 25000 to 100000
* Silver: creditLimit < 25000

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)

Tables: Customers, Payments

**Expected output:**



**Day 12**

1. 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:**



1. Create the table emp\_udf with below fields.

* Emp\_ID
* Name
* DOB

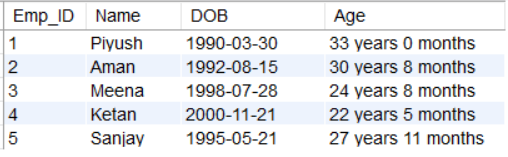
Add the data as shown in below query.

INSERT INTO Emp\_UDF(Name, DOB)

VALUES ("Piyush", "1990-03-30"), ("Aman", "1992-08-15"), ("Meena", "1998-07-28"), ("Ketan", "2000-11-21"), ("Sanjay", "1995-05-21");

Create a user defined function calculate\_age which returns the age in years and months (e.g. 30 years 5 months) by accepting DOB column as a parameter.

**Expected output:**



**Day 13**

1. Display the customer numbers and customer names from customers table who have not placed any orders using subquery

Table: Customers, Orders

**Expected output:**



1. Write a full outer join between customers and orders using union and get the customer number, customer name, count of orders for every customer.

Table: Customers, Orders

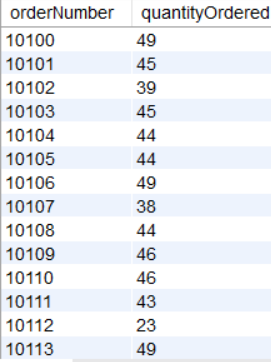
**Expected output:**



1. Show the second highest quantity ordered value for each order number.

Table: Orderdetails

**Expected output:**



1. For each order number count the number of products and then find the min and max of the values among count of orders.

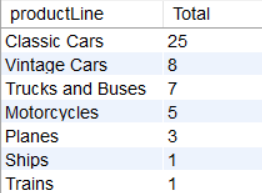
Table: Orderdetails

**Expected output:**



1. 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:**



**Day 14**

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.

**Day 15**

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.