MySQL Assignment

28-02-2025

Suneetha.V

Task-1: create a table with name products and entries are

ProductID, Product name, supplier ID, categoryID, Quantity per unit, unit price, units in stock, units on order, reorder level, discontinued.

Indexes: primary key product name, foreign key is products categories & products suppliers.

ans:

```
CREATE TABLE products ( ProductID INT PRIMARY KEY,    ProductName
VARCHAR(255) NOT NULL,    SupplierID INT,  CategoryID INT,
QuantityPerUnit VARCHAR(100),   UnitPrice DECIMAL(10, 2),  UnitsInStock
INT,  UnitsOnOrder INT, ReorderLevel INT,  Discontinued BIT,

-- Indexes

INDEX idx_product_name (ProductName),

-- Foreign Key Constraints

FOREIGN KEY (SupplierID) REFERENCES suppliers(SupplierID),

FOREIGN KEY (CategoryID) REFERENCES categories(CategoryID)

);
```

Queries:

- **Write a mysql query to get Product name and quantity/unit.**

ANS:

```
SELECT ProductName, QuantityPerUnit   FROM products;
```

- **Write a MySQL query to get current Product list (Product ID and name).**

ANS:

```
SELECT ProductID, ProductName
 FROM products
 WHERE Discontinued = 0;
```

- **Write a MySQL query to get discontinued Product list (Product ID and name).**

ANS:

```
SELECT ProductID, ProductName
 FROM products
WHERE Discontinued = 1;
```

- **Write a MySQL query to get most expense and least expensive Product list (name and unit price).**

ANS: **-- Most expensive product**

```
SELECT ProductName, UnitPrice
FROM products
ORDER BY UnitPrice DESC
LIMIT 1;
```

-- Least expensive product

```
SELECT ProductName, UnitPrice

FROM products

ORDER BY UnitPrice ASC

LIMIT 1;
```

- **Write a MySQL query to get Product list (id, name, unit price) where current products cost less than 20 rupees.**

ANS:

```
SELECT ProductID, ProductName, UnitPrice

 FROM products

 WHERE UnitPrice < 20;
```

Task-2:

- **Create a table name departments with primary key column(department_id).**
- **Columns in the table should be department_id, department_name & location_id.**

Hint:

```
CREATE TABLE departments

  ( department_id INTEGER PRIMARY KEY

  , department_name VARCHAR(30)

  , location_id INTEGER

  ) ;
```

ANS:

```
  CREATE TABLE departments (
   department_id INTEGER PRIMARY KEY,
   department_name VARCHAR(30),
   location_id INTEGER
);
```

- **Create another table with name employees with a foreign key.**

Hint:

```
CREATE TABLE employees
  ( employee_id INTEGER
  , first_name VARCHAR(20)
  , last_name VARCHAR(25)
  , email VARCHAR(25)
  , phone_number VARCHAR(20)
  , hire_date DATE
  , job_id VARCHAR(10)
  , salary INTEGER
  , commission_pct INTEGER
  , manager_id INTEGER
  , department_id INTEGER
  , constraint pk_emp primary key (employee_id)
  , constraint fk_deptno foreign key (department_id) references
departments(department_id)
```

```
  ) ;
```

ANS:

```
  CREATE TABLE employees (
  employee_id INTEGER,
  first_name VARCHAR(20),
  last_name VARCHAR(25),
  email VARCHAR(25),
  phone_number VARCHAR(20),
  hire_date DATE,
  job_id VARCHAR(10),
  salary INTEGER,
  commission_pct INTEGER,
  manager_id INTEGER,
  department_id INTEGER,
  CONSTRAINT pk_emp PRIMARY KEY (employee_id), -- Primary key constraint on employee_id
  CONSTRAINT fk_deptno FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

- **Insert 16 Records into departments Table.**

ANS:

```
 INSERT INTO departments (department_id, department_name,
location_id) VALUES
(1, 'Sales', 101),
(2, 'Marketing', 102),
(3, 'HR', 103),
(4, 'IT', 104),
(5, 'Finance', 105),
(6, 'Legal', 106),
(7, 'Operations', 107),
(8, 'Customer Support', 108),
(9, 'R&D', 109),
(10, 'Admin', 110),
(11, 'Production', 111),
(12, 'Quality Assurance', 112),
(13, 'Supply Chain', 113),
(14, 'Product Management', 114),
(15, 'Business Development', 115),
(16, 'Corporate Strategy', 116);
```

Insert 20 Records into employees Table.

ANS:

```
 INSERT INTO employees (employee_id, first_name, last_name, email,
phone_number, hire_date, job_id, salary, commission_pct, manager_id,
department_id) VALUES
(1, 'sudha', 'Doe', 'jdoe@example.com', '555-1234', '2022-01-15', 'SA_REP',
50000, 0.10, 3, 1),
(2, 'suneetha', 'Smith', 'jsmith@example.com', '555-2345', '2021-03-22',
'IT_PROG', 75000, NULL, 4, 2),
(3, 'Alice', 'Johnson', 'ajohnson@example.com', '555-3456', '2020-07-19',
'MK_MAN', 85000, 0.15, 2, 3),
(4, 'Bob', 'Williams', 'bwilliams@example.com', '555-4567', '2019-05-03',
'HR_REP', 60000, NULL, 3, 1),
(5, 'Charlie', 'Brown', 'cbrown@example.com', '555-5678', '2021-11-14',
'FI_ACCOUNT', 95000, NULL, 5, 4),
(6, 'David', 'Davis', 'ddavis@example.com', '555-6789', '2018-08-21',
'IT_PROG', 70000, 0.12, 3, 4),
(7, 'Eve', 'Martinez', 'emartinez@example.com', '555-7890', '2020-10-11',
'SA_MAN', 120000, NULL, 5, 6),
(8, 'Frank', 'Garcia', 'fgarcia@example.com', '555-8901', '2022-02-20',
'HR_REP', 54000, 0.08, 5, 7),
(9, 'Grace', 'Rodriguez', 'grodriguez@example.com', '555-9012',
'2019-09-25', 'IT_PROG', 80000, NULL, 3, 4),
(10, 'Henry', 'Miller', 'hmiller@example.com', '555-1235', '2017-06-18',
'AD_ASST', 55000, NULL, NULL, 10),
(11, 'Ivy', 'Lopez', 'ilopez@example.com', '555-2346', '2021-01-10',
'IT_PROG', 60000, 0.05, 3, 4),
(12, 'Jack', 'Gonzalez', 'jgonzalez@example.com', '555-3457', '2022-07-05',
'FI_ACCOUNT', 78000, NULL, 3, 5),
(13, 'Kim', 'Wilson', 'kwilson@example.com', '555-4568', '2021-04-13',
'SA_REP', 65000, 0.09, 2, 1),
(14, 'Liam', 'Anderson', 'landerson@example.com', '555-5679',
'2018-12-20', 'MK_MAN', 90000, 0.12, 2, 2),
(15, 'Mia', 'Thomas', 'mthomas@example.com', '555-6780', '2021-06-29',
'SA_MAN', 100000, NULL, 1, 7),
(16, 'Nina', 'Jackson', 'njackson@example.com', '555-7891', '2020-04-17',
'HR_REP', 57000, NULL, 4, 3),
(17, 'Oscar', 'White', 'owhite@example.com', '555-8902', '2019-11-02',
'IT_PROG', 78000, NULL, 1, 4),
(18, 'Paul', 'Martinez', 'pmartinez@example.com', '555-9013', '2021-12-06',
'AD_VP', 130000, NULL, 7, 10),
(19, 'Quinn', 'Hernandez', 'qhernandez@example.com', '555-2347',
'2020-05-11', 'R&D_ENGINEER', 95000, NULL, NULL, 9),
(20, 'Rachel', 'Clark', 'rclark@example.com', '555-3458', '2022-04-02',
'BUSINESS_ANALYST', 72000, 0.07, 6, 8);
```

Queries:

Select employees first name, last name, job_id and salary whose first name starts with alphabet S.

ANS:

```
SELECT first_name, last_name, job_id, salary

FROM employees

WHERE first_name LIKE 'S%';
```

Write a query to select employee with the highest salary.

ANS:

```
SELECT first_name, last_name, salary

FROM employees

ORDER BY salary DESC

LIMIT 1;
```

Select employee with the second highest salary

ANS:

```
SELECT first_name, last_name, salary
```

```
FROM employees
ORDER BY salary DESC
LIMIT 1 OFFSET 1;
```

Fetch employees with 2nd or 3rd highest salary.

ANS:

```
SELECT first_name, last_name, salary
FROM employees
WHERE salary IN (
    SELECT DISTINCT salary
    FROM employees
    ORDER BY salary DESC
    LIMIT 2, 1
    UNION
    SELECT DISTINCT salary
    FROM employees
    ORDER BY salary DESC
    LIMIT 3, 1
);
```

Write a query to select employees and their corresponding managers and their salaries.

Now, this is a classic example of **SELF JOIN** in SQL exercises. Also, use the **CONCAT** function to concatenate the first name and last name of each employee and manager.

ANS:

```
SELECT CONCAT(e.first_name, ' ', e.last_name) AS employee_name,

CONCAT(m.first_name, ' ', m.last_name) AS manager_name,

e.salary AS employee_salary, m.salary AS manager_salary

FROM employees e   LEFT JOIN employees m ON e.manager_id = m.employee_id;
```

Write a query to show count of employees under each manager in descending order.

ANS:

```
    SELECT manager_id, CONCAT(m.first_name, ' ', m.last_name) AS manager_name,

     COUNT(e.employee_id) AS employee_count

     FROM employees e

     LEFT JOIN employees m ON e.manager_id = m.employee_id

    GROUP BY manager_id

   ORDER BY employee_count DESC;
```

Find the count of employees in each department.

ANS:

```
SELECT department_id, COUNT(employee_id) AS employee_count
```

```
FROM employees
GROUP BY department_id;
```

Get the count of employees hired year wise.

ANS:

```
SELECT YEAR(hire_date) AS hire_year, COUNT(employee_id) AS employee_count
FROM employees
GROUP BY YEAR(hire_date)
ORDER BY hire_year;
```

Find the salary range of employees.

ANS:

```
SELECT MIN(salary) AS min_salary, MAX(salary) AS max_salary
FROM employees;
```

Write a query to divide people into three groups based on their salaries.

ANS:

```
   SELECT first_name, last_name, salary,
    CASE
   WHEN salary <= (SELECT AVG(salary) FROM employees) THEN 'Low Salary'
```

```
    WHEN salary > (SELECT AVG(salary) FROM employees) AND salary <=
(SELECT   MAX(salary) FROM employees) * 0.5 THEN 'Medium Salary'

    ELSE 'High Salary'

    END AS salary_group

    FROM employees;
```

or

```
SELECT first_name, last_name, salary,

CASE

 WHEN salary < 50000 THEN 'Low Salary'

 WHEN salary BETWEEN 50000 AND 100000 THEN 'Medium Salary'

  ELSE 'High Salary'

  END AS salary_group

  FROM employees;
```

Select the employees whose first_name contains “an”.

ANS:

```
SELECT first_name, last_name

FROM employees

WHERE first_name LIKE '%an%';
```

Select employee first name and the corresponding phone number in the format (_ _ _)-(_ _ _)-(_ _ _ _).

ANS:

```
    SELECT first_name,

     CONCAT('(', SUBSTRING(phone_number, 1, 3), ')-', SUBSTRING(phone_number, 4, 3), '-', SUBSTRING(phone_number, 7, 4)) AS formatted_phone

     FROM employees;
```

Find the employees who joined in August, 1994.

ANS:

```
SELECT first_name, last_name, hire_date

FROM employees

WHERE hire_date BETWEEN '1994-08-01' AND '1994-08-31';
```

Write an SQL query to display employees who earn more than the average salary in that company.

ANS:

```
SELECT first_name, last_name, salary

FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);
```

Find the maximum salary from each department.

ANS:

```
SELECT department_id, MAX(salary) AS max_salary

FROM employees

GROUP BY department_id;
```

Write a SQL query to display the 5 least earning employees.

ANS:

```
SELECT first_name, last_name, salary

FROM employees

ORDER BY salary ASC

LIMIT 5;
```

Find the employees hired in the 80s.

ANS:

```
SELECT first_name, last_name, hire_date

FROM employees

WHERE hire_date BETWEEN '1980-01-01' AND '1989-12-31';
```

Display the employees first name and the name in reverse order.

ANS:

```
   SELECT first_name,

    CONCAT(REVERSE(first_name), ' ', REVERSE(last_name)) AS reversed_name

    FROM employees;
```

Find the employees who joined the company after 15th of the month.

ANS:

```
SELECT first_name, last_name, hire_date

FROM employees
```

```
WHERE DAY(hire_date) > 15;
```

Display the managers and the reporting employees who work in different departments.

ANS:

```
   SELECT CONCAT(m.first_name, ' ', m.last_name) AS manager_name,
    CONCAT(e.first_name, ' ', e.last_name) AS employee_name,
    e.department_id AS employee_department,
    m.department_id AS manager_department
    FROM employees e
    JOIN employees m ON e.manager_id = m.employee_id
    WHERE e.department_id != m.department_id;
```