MySql_Tasks

Task- 1:

Create two tables: users and orders.

ANS: USERS TABLE

CREATE TABLE users (user_id INT AUTO_INCREMENT PRIMARY KEY, user_name VARCHAR(255) NOT NULL);

ANS: ORDERS TABLE

CREATE TABLE orders (order_id INT PRIMARY KEY, user_id INT, order_date DATE NOT NULL, FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE);

Each user can have multiple orders.

Write a SQL query to fetch the names of users along with the total number of orders they have placed.

ANS:

```
SELECT   u.user_name,   COUNT(o.order_id) AS total_orders   FROM   users u

LEFT JOIN    orders o ON u.user_id = o.user_id

GROUP BY    u.user_id;
```

Task-2:

You are working with a database that stores information about students and their courses. There are three tables: students, courses, and enrollments.

ANS:table creation for students

CREATE TABLE students (student_id INT PRIMARY KEY, student_name VARCHAR(255) NOT NULL);

ANS: table creation for courses

```
CREATE TABLE courses (course_id INT  PRIMARY KEY,   course_name
VARCHAR(255) NOT NULL );
```

ANS:table creation for enrollments

```
CREATE TABLE enrollments (enrollment_id INT  PRIMARY KEY,  student_id
INT,   course_id INT,    FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE
CASCADE, FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE CASCADE  );
```

Write a SQL query to display the names of students along with the courses they have enrolled in.

ANS:

```
SELECT  s.student_name,   c.course_name

FROM     students s

JOIN  enrollments e ON s.student_id = e.student_id

JOIN   courses c ON e.course_id = c.course_id;
```

ANOTHER WAY:

```
SELECT   students.student_name,  courses.course_name

FROM   students, courses, enrollments

WHERE   students.student_id = enrollments.student_id

            AND courses.course_id = enrollments.course_id;
```

Task-3:

You need to retrieve data from a database that tracks product sales. There are tables for products, sales, and customers.

ANS:creation of products table

```
CREATE TABLE products (product_id INT PRIMARY KEY,  product_name VARCHAR(255) NOT
NULL,  category VARCHAR(255) NOT NULL,    price DECIMAL(10, 2) NOT NULL  );
```

ANS:creation of sales table

```
CREATE TABLE sales (sale_id INT PRIMARY KEY,  product_id INT,   quantity_sold INT NOT NULL,
```

sale_date DATE NOT NULL, FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE);

ANS:creation of customers table

CREATE TABLE customers (customer_id INT PRIMARY KEY, customer_name VARCHAR(255) NOT NULL);

Write a SQL query to show the total sales amount for each product category.

ANS:

SELECT   p.category,

 SUM(p.price * s.quantity_sold) AS total_sales_amount

FROM   products p

JOIN   sales s ON p.product_id = s.product_id

GROUP BY   p.category

ORDER BY   total_sales_amount DESC;

Task-4:

You have a database containing information about employees in a company.

ANS:

CREATE TABLE employees (employee_id INT PRIMARY KEY, employee_name VARCHAR(255) NOT NULL,   manager_id INT);

Write a SQL query to list the names of employees along with their respective managers' names.

ANS:

SELECT   e.employee_name AS employee,   m.employee_name AS manager

FROM employees e

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

Task-5:

You are managing a database for an online store.

(I have created 3 tables to join and perform this query)

ANS:creation of products table

CREATE TABLE products (product_id INT PRIMARY KEY, product_name VARCHAR(255) NOT NULL, price DECIMAL(10, 2) NOT NULL);

ANS:creation of orders table

CREATE TABLE orders (order_id INT PRIMARY KEY, order_date DATE NOT NULL);

ANS:creation of order_items table

CREATE TABLE order_items (order_item_id INT PRIMARY KEY, order_id INT, product_id INT, quantity INT NOT NULL);

Write a query to retrieve the top 10 bestselling products based on the total number of units sold.

ANS:

```
SELECT  p.product_name,   SUM(oi.quantity) AS total_units_sold

FROM  products p

JOIN  order_items oi ON p.product_id = oi.product_id

GROUP BY   p.product_name

ORDER BY  total_units_sold DESC  LIMIT 10;
```

Task-6:

You have tables for students, courses, and grades.

ANS:creation of students table

CREATE TABLE students (student_id INT PRIMARY KEY, student_name VARCHAR(255) NOT NULL);

ANS: creation of courses table

CREATE TABLE courses (course_id INT PRIMARY KEY, course_name

VARCHAR(255) NOT NULL);

ANS:creation of grades table

```
                    CREATE TABLE grades (grade_id INT  PRIMARY KEY,   student_id INT,  course_id
INT,  grade DECIMAL(5,2) );
```

Write a SQL query to display the average grade for each student.

ANS:

```
SELECT  s.student_name,

 AVG(g.grade) AS average_grade

FROM    students s

JOIN  grades g ON s.student_id = g.student_id

GROUP BY  s.student_name;
```

Task-7:

You are working with a database for a social media platform.

Write a query to show the users who have the most friends.

ANS:creation of users, friends table to join.

```
    CREATE TABLE users (user_id INT  PRIMARY KEY,  username VARCHAR(255) NOT NULL);

CREATE TABLE friends (user_id INT,   friend_id INT );
```

Write a query to show the users who have the most friends

ANS:

```
SELECT   u.username,  COUNT(DISTINCT f.friend_id) + COUNT(DISTINCT f.user_id) - 1 AS
total_friends

FROM   users u

JOIN  friends f ON u.user_id = f.user_id OR u.user_id = f.friend_id

GROUP BY  u.user_id
```

ORDER BY total_friends DESC LIMIT 1;

Task-8:

You have tables for employees and departments.

ANS: creation of departments table

CREATE TABLE departments (department_id INT PRIMARY KEY,

department_name VARCHAR(255) NOT NULL);

ANS:creation of employees table

CREATE TABLE employees (employee_id INT   PRIMARY KEY,

employee_name VARCHAR(255) NOT NULL,   department_id INT,   FOREIGN KEY (department_id) REFERENCES departments(department_id));

Write a query to display the department names along with the total number of employees in each department.

ANS:

```
SELECT  d.department_name,

COUNT(e.employee_id) AS total_employees

FROM   departments d

LEFT JOIN   employees e ON d.department_id = e.department_id

GROUP BY  d.department_name;
```

Task-9:

You need to retrieve data from a database tracking product inventory.

ANS: creation of products table

CREATE TABLE products (product_id INT PRIMARY KEY, product_name VARCHAR(255) NOT NULL, stock_quantity INT, price DECIMAL(10, 2));

Write a query to display products with low stock (less than 10 units).

ANS:

```
SELECT  product_id, product_name, stock_quantity,   price

FROM  products

WHERE  stock_quantity < 10;
```

Task-10:

You have tables for customers and orders.

ANS: creation of customer table:

```
CREATE TABLE customers (customer_id INT PRIMARY KEY,customer_name VARCHAR(255) NOT
NULL,  email VARCHAR(255) UNIQUE );
```

ANS:creation of orders table

```
CREATE TABLE orders (order_id INT  PRIMARY KEY,   customer_id INT,  order_date DATE,
total_amount DECIMAL(10, 2),   FOREIGN KEY (customer_id) REFERENCES
customers(customer_id) );
```

Write a query to show the average order value for each customer.

ANS:

```
SELECT  c.customer_id,  c.customer_name,

AVG(o.total_amount) AS average_order_value

FROM  customers c

JOIN   orders o ON c.customer_id = o.customer_id

GROUP BY   c.customer_id, c.customer_name;
```

another way:(simple way)

```
SELECT  customer_id,  AVG(total_amount) AS average_order_value

FROM   orders
```

```
GROUP BY   customer_id;
```

another way:

```
SELECT  customer_id,  AVG(total_amount) AS average_order_value

FROM    orders

GROUP BY  customer_id

HAVING   AVG(total_amount) > 100;
```

Task-11:

In a database storing movie information,

ANS: creation of movies table:

```
   CREATE TABLE movies (movie_id INT  PRIMARY KEY,  movie_name VARCHAR(255) NOT NULL,

   release_date DATE,  rating DECIMAL(3, 2) );
```

Write a query to show the top 5 highest-rated movies by users.

ANS:

```
SELECT  movie_id,   movie_name,  release_date,  rating

FROM   movies

ORDER BY   rating DESC LIMIT 5;
```

Task-12:

You have tables for invoices and payments.

ANS: creation of Table for invoices:

```
   CREATE TABLE invoices (

   invoice_id INT   PRIMARY KEY,

   invoice_date DATE,
   amount DECIMAL(10, 2),
   status ENUM('paid', 'unpaid') DEFAULT 'unpaid'
);
```

ANS: Creation of tables for payments:

```
   CREATE TABLE payments (
   payment_id INT  PRIMARY KEY,
   invoice_id INT,
   payment_date DATE,
   payment_amount DECIMAL(10, 2),
   FOREIGN KEY (invoice_id) REFERENCES invoices(invoice_id)
);
```

Write a query to show the unpaid invoices and their total amount.

ANS:

we can write it in many ways:

1 way:

```
                         SELECT    i.invoice_id,    i.invoice_date,   i.amount AS total_invoice_amount,
   COALESCE(SUM(p.payment_amount), 0) AS total_paid_amount,
(i.amount - COALESCE(SUM(p.payment_amount), 0)) AS remaining_balance
FROM  invoices i
LEFT JOIN   payments p ON i.invoice_id = p.invoice_id
WHERE     i.status = 'unpaid'
GROUP BY   i.invoice_id
```

```
HAVING    remaining_balance > 0;
```

Another way:(simple way)

```
  SELECT  invoice_id,   invoice_date,  amount AS total_invoice_amount

  FROM    invoices

  WHERE    status = 'unpaid';
```