SQL Basics

Assignment Questions

PW SKILLS

1. Create a table called employees with the following structure:

- **emp_id (integer, should not be NULL and should be a primary key).**
- **emp_name (text, should not be NULL).**
- **age (integer, should have a check constraint to ensure the age is at least 18).**
- **email (text, should be unique for each employee).**
- **salary (decimal, with a default value of 30,000).**

Write the SQL query to create the above table with all constraints.

2. Explain the purpose of constraints and how they help maintain data integrity in a database. Provide examples of common types of constraints.

3.Why would you apply the NOT NULL constraint to a column? Can a primary key contain NULL values? Justify your answer.

4. Explain the steps and SQL commands used to add or remove constraints on an existing table. Provide an example for both adding and removing a constraint.

5. Explain the consequences of attempting to insert, update, or delete data in a way that violates constraints. Provide an example of an error message that might occur when violating a constraint.

6. You created a products table without constraints as follows:

```
CREATE TABLE products (
    product_id INT,
    product_name VARCHAR(50),
    price DECIMAL(10, 2));
```

Now, you realise that:

- **The product_id should be a primary key.**
- **The price should have a default value of 50.00**

7. You have two tables:

- Students:

```
+------------+------------+------------+
| student_id | student_name | class_id  |
+------------+------------+------------+
| 1          | Alice      | 101        |
| 2          | Bob        | 102        |
| 3          | Charlie    | 101        |
+------------+------------+------------+
```

- Classes:

class_id	class_name
101	Math
102	Science
103	History

Write a query to fetch the student_name and class_name for each student using an INNER JOIN.

8. Consider the following three tables:

- Orders:

order_id	order_date	customer_id
1	2024-01-01	101
2	2024-01-03	102

- Customers:

customer_id	customer_name
101	Alice
102	Bob

- Products:

product_id	product_name	order_id
1	Laptop	1
2	Phone	NULL

Write a query that shows all order_id, customer_name, and product_name, ensuring that all products are listed even if they are not associated with an order
Hint: (use INNER JOIN and LEFT JOIN).

9. Given the following tables:

- Sales:

sale_id	product_id	amount
1	101	500
2	102	300
3	101	700

- Products:

product_id	product_name
101	Laptop
102	Phone

Write a query to find the total sales amount for each product using an INNER JOIN and the SUM() function.

10. You are given three tables:

Orders:

order_id	order_date	customer_id
1	2024-01-02	1
2	2024-01-05	2

Customers:

customer_id	customer_name
1	Alice
2	Bob

Order_Details:

order_id	product_id	quantity
1	101	2
1	102	1
2	101	3

Write a query to display the order_id, customer_name, and the quantity of products ordered by each customer using an INNER JOIN between all three tables.

Note - The above-mentioned questions don't require any dataset.

SQL Commands

1-Identify the primary keys and foreign keys in maven movies db. Discuss the differences

2- List all details of actors

3 -List all customer information from DB.

4 -List different countries.

5 -Display all active customers.

6 -List of all rental IDs for customer with ID 1.

7 - Display all the films whose rental duration is greater than 5 .

8 - List the total number of films whose replacement cost is greater than $15 and less than $20.

9 - Display the count of unique first names of actors.

10- Display the first 10 records from the customer table .

11 - Display the first 3 records from the customer table whose first name starts with ‘b’.

12 -Display the names of the first 5 movies which are rated as ‘G’.

13-Find all customers whose first name starts with "a".

14- Find all customers whose first name ends with "a".

15- Display the list of first 4 cities which start and end with ‘a’ .

16- Find all customers whose first name have "NI" in any position.

17- Find all customers whose first name have "r" in the second position .

18 - Find all customers whose first name starts with "a" and are at least 5 characters in length.

19- Find all customers whose first name starts with "a" and ends with "o".

20 - Get the films with pg and pg-13 rating using IN operator.

21 - Get the films with length between 50 to 100 using between operator.

22 - Get the top 50 actors using limit operator.

23 - Get the distinct film ids from inventory table.

Functions

Basic Aggregate Functions:

Question 1:
Retrieve the total number of rentals made in the Sakila database.
Hint: Use the COUNT() function.

Question 2:
Find the average rental duration (in days) of movies rented from the Sakila database.
Hint: Utilize the AVG() function.

String Functions:
Question 3:
Display the first name and last name of customers in uppercase.
Hint: Use the UPPER () function.

Question 4:
Extract the month from the rental date and display it alongside the rental ID.
Hint: Employ the MONTH() function.

GROUP BY:

Question 5:
Retrieve the count of rentals for each customer (display customer ID and the count of rentals).
Hint: Use COUNT () in conjunction with GROUP BY.

Question 6:
Find the total revenue generated by each store.
Hint: Combine SUM() and GROUP BY.

Question 7:
Determine the total number of rentals for each category of movies.
Hint: JOIN film_category, film, and rental tables, then use cOUNT () and GROUP BY.

Question 8:
Find the average rental rate of movies in each language.
Hint: JOIN film and language tables, then use AVG () and GROUP BY.

Joins

Questions 9 -
Display the title of the movie, customer s first name, and last name who rented it.
Hint: Use JOIN between the film, inventory, rental, and customer tables.

Question 10:
Retrieve the names of all actors who have appeared in the film "Gone with the Wind."
Hint: Use JOIN between the film actor, film, and actor tables.

Question 11:
Retrieve the customer names along with the total amount they've spent on rentals.
Hint: JOIN customer, payment, and rental tables, then use SUM() and GROUP BY.

Question 12:
List the titles of movies rented by each customer in a particular city (e.g., 'London').
Hint: JOIN customer, address, city, rental, inventory, and film tables, then use GROUP BY.

Advanced Joins and GROUP BY:

Question 13:
Display the top 5 rented movies along with the number of times they've been rented.
Hint: JOIN film, inventory, and rental tables, then use COUNT () and GROUP BY, and limit the results.

Question 14:
Determine the customers who have rented movies from both stores (store ID 1 and store ID 2).
Hint: Use JOINS with rental, inventory, and customer tables and consider COUNT() and GROUP BY.

Windows Function:

1. Rank the customers based on the total amount they've spent on rentals.

2. Calculate the cumulative revenue generated by each film over time.

3. Determine the average rental duration for each film, considering films with similar lengths.

4. Identify the top 3 films in each category based on their rental counts.

5. Calculate the difference in rental counts between each customer's total rentals and the average rentals across all customers.

6. Find the monthly revenue trend for the entire rental store over time.

7. Identify the customers whose total spending on rentals falls within the top 20% of all customers.

8. Calculate the running total of rentals per category, ordered by rental count.

9. Find the films that have been rented less than the average rental count for their respective categories.

10. Identify the top 5 months with the highest revenue and display the revenue generated in each month.

Normalisation & CTE

1. First Normal Form (1NF):

a. Identify a table in the Sakila database that violates 1NF. Explain how you would normalize it to achieve 1NF.

2. Second Normal Form (2NF):

a. Choose a table in Sakila and describe how you would determine whether it is in 2NF. If it violates 2NF, explain the steps to normalize it.

3. Third Normal Form (3NF):

a. Identify a table in Sakila that violates 3NF. Describe the transitive dependencies present and outline the steps to normalize the table to 3N**F.**

4. Normalization Process:

a. Take a specific table in Sakila and guide through the process of normalizing it from the initial unnormalized form up to at least 2NF.

5. CTE Basics:

a. Write a query using a CTE to retrieve the distinct list of actor names and the number of films they have acted in from the `actor` and `film_actor` tables.

6. CTE with Joins:

a. Create a CTE that combines information from the `film` and `language` tables to display the film title, language name, and rental rate.

7. CTE for Aggregation:

a. Write a query using a CTE to find the total revenue generated by each customer (sum of payments) from the `customer` and `payment` tables.

8. CTE with Window Functions:

a. Utilize a CTE with a window function to rank films based on their rental duration from the film table.

9. CTE and Filtering:

a. Create a CTE to list customers who have made more than two rentals, and then join this CTE with the customer table to retrieve additional customer details.

10. **CTE for Date Calculations:**
    a. Write a query using a CTE to find the total number of rentals made each month, considering the rental_date from the rental table.

11. **CTE and Self-Join:**
    a. Create a CTE to generate a report showing pairs of actors who have appeared in the same film together, using the film_actor table.

12. **CTE for Recursive Search:**
    a. Implement a recursive CTE to find all employees in the staff table who report to a specific manager, considering the reports_to column

Datset for assignment.

Access the dataset here - **Mavenmovies.sql**