Name – Wake Rushikesh Bapusaheb

Mail Id- rushikeshwakte01@gamil.com

SQL Assignment

Q1. Create a table called employees with the following structure?

: emp_id (integer, should not be NULL and should be a primary key)Q

: emp_name (text, should not be NULL)Q

: age (integer, should have a check constraint to ensure the age is at least 18)Q

: email (text, should be unique for each employee)Q

: salary (decimal, with a default value of 30,000).

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

Ans-

```
CREATE TABLE employees (
    emp_id   INTEGER      NOT NULL,
    emp_name TEXT         NOT NULL,
    age      INTEGER      NOT NULL CHECK (age >= 18),
    email    TEXT         NOT NULL UNIQUE,
    salary   DECIMAL(15,2) DEFAULT 30000.00,
    PRIMARY KEY (emp_id)
);
```

Q2. Explain the purpose of constraints and how they help maintain data integrity in a database. Provide examples of common types of constraints.

Ans-

Constraints are rules defined in a database schema that restrict the kinds of data that can be inserted, updated, or deleted. They help ensure that the data remains **correct, consistent, reliable**, and meaningful — in other words, they enforce *data integrity*.

Q3. Why would you apply the NOT NULL constraint to a column? Can a primary key contain NULL values? Justify your answer.

Apply a `NOT NULL` constraint to ensure a column always has a valid value (never missing), and a **primary key** cannot contain `NULL` values because it must uniquely and unambiguously identify each row.

Q 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.

To add a constraint:

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name UNIQUE
(column_name);
```

To remove a constraint:

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

Q5. 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.

If you attempt to insert, update or delete data in a way that violates a constraint, the DBMS will reject the operation (roll it back or abort it) and return an error such as:

```
Violation of PRIMARY KEY constraint 'PK_Employees'. Cannot insert
duplicate key in object 'dbo.Employees'.
```

Q6. 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 keyQ

: The price should have a default value of 50.00

```
-- First, ensure product_id is NOT NULL (required before making it a primary key)
ALTER TABLE products
ALTER COLUMN product_id INT NOT NULL;


-- Add the primary key constraint
ALTER TABLE products
ADD CONSTRAINT pk_products_product_id PRIMARY KEY (product_id);


-- Set default on price column
ALTER TABLE products
ALTER COLUMN price DECIMAL(10,2) SET DEFAULT 50.00;
```

Q7. You have two tables:

- Classes:

```
+------------+------------+
| class_id   | class_name |
+------------+------------+
| 101        | Math       |
| 102        | Science    |
| 103        | History    |
+------------+------------+
```

- Students:

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

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

Ans-

```
SELECT
   s.student_name,
   c.class_name
FROM
   Students s
INNER JOIN
   Classes c
ON
   s.class_id = c.class_id;
```

Q8. 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     |
◦   +------------+-----------+------------+
◦
```

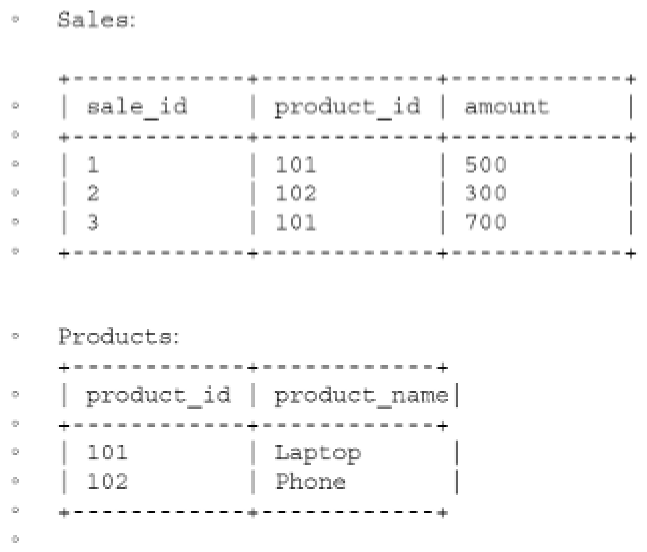
Write a query that shows all order_id, customer_name, and product_name, ensuring that all products arelisted even if they are not associated with an order

Hint: (use INNER JOIN and LEFT JOIN)

Ans-

```
SELECT
    O.order_id,
    C.customer_name,
    P.product_name
FROM
    Products AS P
LEFT JOIN
    Orders AS O ON P.order_id = O.order_id
LEFT JOIN
    Customers AS C ON O.customer_id = C.customer_id;
```

Q9. Given the following tables:



- Sales:

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

- Products:

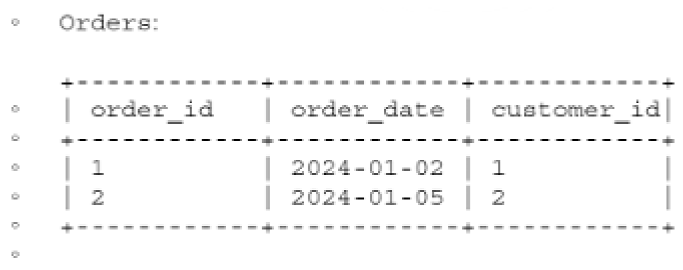
product_id	product_name
101	Laptop
102	Phone

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

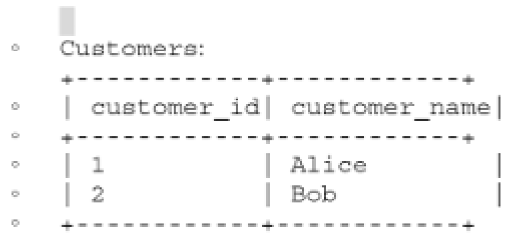
Ans-

```
SELECT
    P.product_name,
    SUM(S.amount) AS total_sales_amount
FROM
    Sales AS S
INNER JOIN
    Products AS P ON S.product_id = P.product_id
GROUP BY
    P.product_name;
```

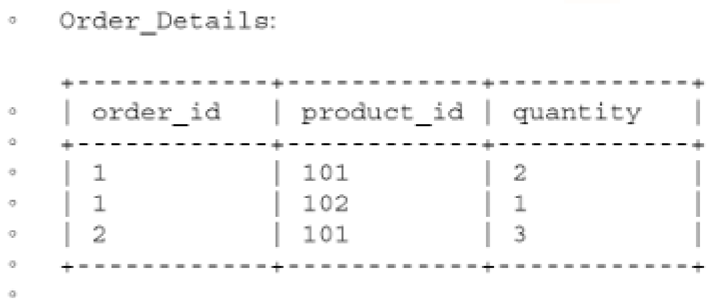
Q10. You are given three tables:


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.

Ans-

```
SELECT
    O.order_id,
    C.customer_name,
    P.product_name
FROM
    Products AS P
LEFT JOIN
    Orders AS O ON P.order_id = O.order_id
LEFT JOIN
    Customers AS C ON O.customer_id = C.customer_id;
```

Functions-

Q1. Retrieve the total number of rentals made in the Sakila database.

Hint: Use the COUNT() function.

Ans-

SELECT COUNT(*) FROM rental;

Q2. Find the average rental duration (in days) of movies rented from the Sakila database. Hint: Utilize the AVG() function. String Functions:

Ans-

SELECT AVG(rental_duration) AS 'Average Rental Duration' FROM film;

Q3. Display the first name and last name of customers in uppercase. Hint: Use the UPPER () function.

Ans-

SELECT UPPER(first_name) AS 'First Name', UPPER(last_name) AS 'Last Name' FROM customer;

Q4. Extract the month from the rental date and display it alongside the rental ID. Hint: Employ the MONTH() function. GROUP BY:

Ans-

SELECT rental_id, MONTH(rental_date) AS 'Rental Month' FROM rental;

Q5. Retrieve the count of rentals for each customer (display customer ID and the count of rentals). Hint: Use COUNT () in conjunction with GROUP BY.

SELECT customer_id, COUNT(*) AS 'Rental Count' FROM rental GROUP BY customer_id;

Q6. Find the total revenue generated by each store. Hint: Combine SUM() and GROUP BY.

Ans-

SELECT store_id, SUM(amount) AS 'Total Revenue' FROM payment GROUP BY store_id;

Q7 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.

Ans-

```
SELECT c.name AS 'Category', COUNT(*) AS 'Rental Count'

FROM category c

JOIN film_category fc ON c.category_id = fc.category_id

JOIN film f ON fc.film_id = f.film_id

JOIN inventory i ON f.film_id = i.film_id

JOIN rental r ON i.inventory_id = r.inventory_id

GROUP BY c.category_id;
```

Q8. Find the average rental rate of movies in each language. Hint: JOIN film and language tables, then use AVG () and GROUP BY.(Need one sentence answer for SQL)

Ans-

```
SELECT l.name AS 'Language', AVG(f.rental_rate) AS 'Average Rental Rate'

FROM language l

JOIN film f ON l.language_id = f.language_id

GROUP BY l.language_id;
```

Joins-

Q9. 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.

Ans-

```
SELECT f.title AS 'Movie Title', c.first_name AS 'First Name', c.last_name AS 'Last Name'

FROM film f

JOIN inventory i ON f.film_id = i.film_id

JOIN rental r ON i.inventory_id = r.inventory_id

JOIN customer c ON r.customer_id = c.customer_id;
```

Q10. 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.

Ans-

```
SELECT a.first_name AS 'First Name', a.last_name AS 'Last Name'

FROM actor a

JOIN film_actor fa ON a.actor_id = fa.actor_id

JOIN film f ON fa.film_id = f.film_id

WHERE f.title = 'Gone with the Wind';
```

Q11. 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.

Ans-

```
SELECT CONCAT(c.first_name, ' ', c.last_name) AS 'Customer Name', SUM(p.amount) AS 'Total
Spent'

FROM customer c

JOIN payment p ON c.customer_id = p.customer_id

GROUP BY c.customer_id;
```

Q12. 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.(Need one sentence answer for SQL)

Ans-

```
SELECT CONCAT(c.first_name, ' ', c.last_name) AS 'Customer Name', f.title AS 'Movie Title'

FROM customer c

JOIN address a ON c.address_id = a.address_id

JOIN city ci ON a.city_id = ci.city_id

JOIN rental r ON c.customer_id = r.customer_id

JOIN inventory i ON r.inventory_id = i.inventory_id

JOIN film f ON i.film_id = f.film_id

WHERE ci.city = 'London'

ORDER BY c.last_name, f.title;
```

Advanced Joins and GROUP BY:

Q13. 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.

Ans-

```
SELECT f.title AS 'Movie Title', COUNT(r.rental_id) AS 'Rental Count'

FROM film f

JOIN inventory i ON f.film_id = i.film_id

JOIN rental r ON i.inventory_id = r.inventory_id

GROUP BY f.film_id

ORDER BY COUNT(r.rental_id) DESC

LIMIT 5;
```

Q14. 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.(Need one sentence answer for SQL)

Ans-

```
SELECT c.customer_id, CONCAT(c.first_name, ' ', c.last_name) AS 'Customer Name'

FROM customer c

JOIN rental r ON c.customer_id = r.customer_id

JOIN inventory i ON r.inventory_id = i.inventory_id

WHERE i.store_id IN (1, 2)

GROUP BY c.customer_id

HAVING COUNT(DISTINCT i.store_id) = 2;
```