**SQL ASSIGNMENT**

**Q2. Ans: Constraints** are rules applied to columns or tables in a database to enforce the correctness, validity, and consistency of the data stored. They help ensure **data integrity** by restricting the type of data that can be entered, preventing errors and maintaining reliable, trustworthy data.

How Constraints Help Maintain Data Integrity

* Prevent invalid data entry: Constraints stop incorrect or incomplete data from being saved.
* Enforce business rules: For example, ensuring unique identifiers or mandatory fields.
* Maintain relationships: Such as ensuring foreign keys correspond to existing records.
* Improve data consistency: So the data stays accurate across the database.
* **Common types of constraints** **in a database:** Include **NOT NULL**, which ensures a column cannot be empty; **UNIQUE**, which ensures all values in a column are different; **PRIMARY KEY**, which uniquely identifies each record; **FOREIGN KEY**, which enforces relationships between tables; **CHECK**, which enforces specific conditions on data; and **DEFAULT**, which assigns a default value when none is provided.

**Q3. Ans:** The **NOT NULL** constraint is applied to ensure that a column **must always have a value**—it cannot be left empty (NULL). This is important when the data in that column is **required** for the record to be meaningful or valid. For example, you might require every user to have an email address or every product to have a name, so you apply NOT NULL to those columns to prevent missing information. **Primary Key cannot contain NULL values.**

Justification: A primary key uniquely identifies each row in a table. If it allowed NULLs, it would mean that some rows have no identifier or an unknown identifier, which defeats the purpose of uniquely identifying records.

**Q4. Ans:** Adding key: ALTER TABLE table\_name

ADD CONSTRAINT constraint\_name PRIMARY KEY (column\_name);

Removing key: ALTER TABLE Employees

DROP CONSTRAINT CHK\_Age;  
  
**Q5. Ans**: When you attempt to insert, update, or delete data in a database in a way that violates constraints, the database management system (DBMS) will **reject the operation** and raise an error. Constraints are rules defined on the data to maintain data integrity and consistency. Violating them means the operation would compromise these rules, so the DBMS prevents it from happening.

Primary key violation:

ERROR: duplicate key value violates unique constraint "users\_pkey"

DETAIL: Key (user\_id)=(123) already exists.

**SQL COMMANDS**

**1. Primary Keys and Foreign Keys in Maven Movies DB**

| **Table** | **Primary Key** | **Foreign Key** |
| --- | --- | --- |
| actor | actor\_id | None |
| film | film\_id | language\_id |
| inventory | inventory\_id | film\_id |
| customer | customer\_id | address\_id |
| rental | rental\_id | inventory\_id, customer\_id, staff\_id |
| address | address\_id | city\_id |
| city | city\_id | country\_id |
| country | country\_id | None |

* **Primary Key :** Unique identifier for each record in a table.
* **Foreign Key :** Field in one table that refers to the primary key in another table, creating relationships between tables.

**2. List all details of actors**

SELECT \* FROM actor;

**3. List all customer information from DB**

SELECT \* FROM customer;

**4. List different countries**

SELECT DISTINCT country FROM country;

**5. Display all active customers**

SELECT \* FROM customer WHERE active = 1;

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

SELECT rental\_id FROM rental WHERE customer\_id = 1;

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

SELECT \* FROM film WHERE rental\_duration > 5;

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

SELECT COUNT(\*) FROM film WHERE replacement\_cost > 15 AND replacement\_cost < 20;

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

SELECT COUNT(DISTINCT first\_name) FROM actor;

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

SELECT \* FROM customer LIMIT 10;

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

SELECT \* FROM customer WHERE first\_name LIKE 'b%' LIMIT 3;

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

SELECT title FROM film WHERE rating = 'G' LIMIT 5;

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

SELECT \* FROM customer WHERE first\_name LIKE 'a%';

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

SELECT \* FROM customer WHERE first\_name LIKE '%a';

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

SELECT city FROM city WHERE city LIKE 'a%' AND city LIKE '%a' LIMIT 4;

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

SELECT \* FROM customer WHERE first\_name LIKE '%NI%';

(Note: The query is case-sensitive depending on the DB. Use ILIKE in PostgreSQL or convert to uppercase/lowercase if needed.)

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

SELECT \* FROM customer WHERE first\_name LIKE '\_r%';

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

SELECT \* FROM customer WHERE first\_name LIKE 'a%' AND LENGTH(first\_name) >= 5;

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

SELECT \* FROM customer WHERE first\_name LIKE 'a%' AND first\_name LIKE '%o';

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

SELECT \* FROM film WHERE rating IN ('PG', 'PG-13');

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

SELECT \* FROM film WHERE length BETWEEN 50 AND 100;

**22. Get the top 50 actors using LIMIT operator**

SELECT \* FROM actor LIMIT 50;

**23. Get the distinct film ids from inventory table**

SELECT DISTINCT film\_id FROM inventory;

**Basic Aggregate Functions**