**Task 5:**

Discuss the primary key constraint and its significance in ensuring dataquality. Provide an example of how you would use a primary key to enforce uniqueness and non-null entries in a table.

**Primary Key Constraint in a Nutshell**

Imagine you have a collection of unique keys for your house. The primary key is like your main house key. It's special because it uniquely identifies your house. In a database, a primary key is a unique and important identifier for each row in a table.

**Significance in Ensuring Data Quality:**

1. **Uniqueness:**

Just like your main house key is unique to your house, a primary key ensures that each row in a database table has a unique identifier. This prevents confusion and mix-ups, making sure each piece of information is distinct**.**

1. **Non-Null Entries:**

Your main house key is essential; you can't leave it at home. Similarly, a primary key ensures that there are no "empty" or missing key values in the database. Each row must have a key, just like you always need your main key to enter your house**.**

**Example of Using a Primary Key:**

Imagine you have a table for keeping track of students in a school. Each student needs a unique identifier, like a student ID, to avoid any mix-ups. Here's how you'd use a primary key:

1. **Create a Student Table:**

You create a table to store student information. Each row will represent a student.

1. **Add a Student ID Column:**

In that table, you add a column called "Student ID" and make it the primary key.

1. **Enforce Uniqueness:**

Each student is assigned a unique Student ID. This ensures that no two students have the same ID in the table. It's like each student having a unique number for identification.

1. **Prevent Null Entries:**

The Student ID column is set as a primary key, meaning it cannot be left blank. Every student must have a Student ID, just like you can't have an empty main key to your house.

1. **Data Quality Assurance:**

With the primary key constraint, you ensure data quality by avoiding duplicate IDs and missing values. It's like making sure each student is uniquely identified and no student is missing their identification**.**

**Task 6:**

Explain the role of foreign keys in maintaining referential integrity. Provide an example of a scenario where foreign keys are essential, and describe how they can be implemented.

**Role of Foreign Keys in a Nutshell:**

Imagine you have a library with books and borrowing cards. The borrowing card has a special number that links it to a specific book. In a database, a foreign key is like that special number. It connects one table to another, ensuring that relationships between different pieces of information are accurate and reliable.

**Scenario where Foreign Keys are Essential:**

Picture a library database with two tables: one for books and another for borrowing cards. Each borrowing card should be linked to a specific book. Without foreign keys, you might end up with borrowing cards that have numbers but don't match any book. Foreign keys prevent this by making sure that each borrowing card number connects to a real book.

**How Foreign Keys Can be Implemented:**

1. **Create the Books Table:**

Imagine you have a table called "Books" with information about each book, including a unique identifier called "BookID."

1. **Create the Borrowing Cards Table:**

Now, you create another table called "Borrowing Cards" to track who borrowed which book. In this table, you have a column called "BookID" as well.

1. **Link with Foreign Key:**

You declare "BookID" in the "Borrowing Cards" table as a foreign key that references the "BookID" in the "Books" table. This is like saying, "The BookID in Borrowing Cards points to the BookID in Books."

1. **Enforce Relationships:**

With the foreign key relationship, you ensure that every "BookID" in the "Borrowing Cards" table corresponds to a real "BookID" in the "Books" table. It's like making sure each borrowing card number matches an actual book on the shelf.

1. **Maintain Referential Integrity:**

If someone tries to enter a borrowing card with a non-existing "BookID," the database will say, "Hold on! This doesn't match any book in the Books table." This helps maintain referential integrity by preventing disconnected or orphaned data.

1. **Cascade Actions (Optional):**

You can set up rules to automatically handle changes. For example, if a book is deleted from the "Books" table, you might want the related borrowing card entries to be automatically removed too. This is called cascading actions.

**Task 7:**

Define the CHECK constraint and its purpose in data validation. Provide a real-world example of when you would use the CHECK constraint to validate a column’s value.

**CHECK Constraint:**

A CHECK constraint is like a rule you set for a column in a database table. It ensures that the data entered into that column meets a specific condition or set of conditions.

**Purpose in Data Validation:**

The main purpose of a CHECK constraint is to validate the data that goes into a column. It helps ensure that only the right kind of data is allowed, based on certain conditions you define.

**Real-World Example:**

Imagine you have a database table to store information about employees, and you have a column named **age** to store their ages. Now, you know that the age of an employee can't be negative. To make sure nobody accidentally enters a negative age, you can use a CHECK constraint on the **age** column, specifying that the age must be greater than or equal to zero.

**Task 8:**

Describe the Unique constraint and how it ensures the uniqueness of valuesin a column. Provide an example of a situation where you would applythe Unique constraint.

**Unique Constraint:**

A Unique constraint is like a special rule for a column in a database table. It ensures that every value entered in that column is unique, meaning there are no duplicate values.

**Ensuring Uniqueness:**

When you apply a Unique constraint to a column, it checks that no two rows in the table have the same value in that specific column. It helps maintain distinctiveness, making sure each value is different from the others in that column

**Real-World Example:**

Consider a database table that stores information about students, and it has a column named **student\_id** to store a unique identification number for each student. To make sure that no two students have the same ID, you would apply a Unique constraint to the **student\_id** column. This way, every student is assigned a distinct and unique ID, preventing

**Task 9:**

Explain the NOT NULL constraint and its significance in data validation. Describe when and why you would use the NOT NULL constraint on columns.

**NOT NULL Constraint:**

The NOT NULL constraint is like a rule for a column in a database table that says the column must always have a value. In other words, it prevents the column from being left empty or having a null value.

**Significance in Data Validation:**

The NOT NULL constraint is crucial for ensuring that essential information is always provided. It avoids situations where important details are missing, making sure that every row in the table has a value for the specified column.

**When and Why to Use NOT NULL:**

We can use the NOT NULL constraint when a certain piece of information is mandatory and should be present for every record in the table. For example, in a table storing customer data, we might have a column for the customer's email address. Applying NOT NULL to this column ensures that we always have an email address for every customer entry, and we won't end up with incomplete or unreliable data.