SQL Constraints

NOT NULL

Ensures that a column cannot have a NULL value.

Example:

```
CREATE TABLE employees (
  employee_id INT NOT NULL,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL
);
```

UNIQUE

Ensures that all values in a column are different.

Example:

```
CREATE TABLE employees (
  employee_id INT NOT NULL UNIQUE,
  email VARCHAR(100) UNIQUE
);
```

PRIMARY KEY

A combination of NOT NULL and UNIQUE. Uniquely identifies each row in a table. Only one primary key can be assigned to a table, but it can consist of multiple columns (composite primary key).

Example:

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50)
);
```

FOREIGN KEY

Ensures the referential integrity of the data in one table to match values in another table.

Example:

```
CREATE TABLE departments (
   department_id INT PRIMARY KEY,
   department_name VARCHAR(50)
);

CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   department_id INT,
   FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

CHECK

Ensures that all values in a column satisfy a specific condition.

Example:

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   salary DECIMAL(10, 2),
   CHECK (salary > 0)
);
```

DEFAULT

Provides a default value for a column when none is specified.

Example:

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   hire_date DATE DEFAULT CURRENT_DATE
);
```

INDEX

Improves the performance of queries by creating indexes on columns. Not exactly a constraint but is used to enhance search operations.

Example:

```
CREATE INDEX idx_last_name ON employees(last_name);
```