Video-5

Topics to cover:

- SQL Constraints

SQL Constraints

- SQL constraints are used to specify rules for the data in a table.

- Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

The following constraints are commonly used in SQL:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT

Why we need Constraints ?

Example: - table without Constraints

```
Create table employee (
        id int,
        name varchar(30),
        salary double
);

DESC employee;

# insert data without name
insert into employee (id, salary)
Values
(1,3000);

# insert data without salary and name
insert into employee (id)
Values
(1);
```

SQL Constraints

1. **NOT NULL** - Ensures that a column cannot have a NULL value.

Example: - In this example, every student must have a name.

```
Create table employee1 (
        id int,
        name varchar(30) not null,
        salary double
);

DESC employee1;

# Column name can not be null
insert into employee1(id, salary)
Values
(1,3000);

# Column name can not be null
insert into employee1
Values
(1,null,3000);
```

SQL Constraints

2. UNIQUE - Ensures that all values in a column are different.

Example: - In this example, every student must have a name.

```
Create table employee2 (
        id int Unique,
        name varchar(30),
        salary double
);

Select * from employee2;
```

 # It will allow this

```
insert into employee2
Values
(1,"Rajan",3000);
```

 # ID can not be duplicate

```
insert into employee2
Values
(1,"Ankit",4000);
```

SQL Constraints

3. Check- Ensures values meet a specific condition.

Example: - In this example, every student must have a name.

```
Create table employee3 (
        id int,
        name varchar(30),
        salary double check (salary>1000)
);

Select * from employee3;
```

 # It will allow this

```
insert into employee3
Values
(1,"Sumit",3000);
```

 # salary can not be less than 1000

```
insert into employee3
Values
(1,"Alok",500);
```

SQL Constraints

4. Default- Assigns a default value if none is provided.

Example: - In this example, every student must have a name.

```
Create table employee4 (
        id int,
        name varchar(30),
        hiring_date Date default "2023-08-11"
);

Select * from employee4;

# It will allow this
insert into employee4 (id, name, hiring_date)
Values
(1,"Sumit","2022-01-01");

# it will also allow this and take default value in hiring_date column
insert into employee4 (id, name)
Values
(1,"Akash");
```

SQL Constraints

5. **Primary Key** - Uniquely identifies each row in a table.

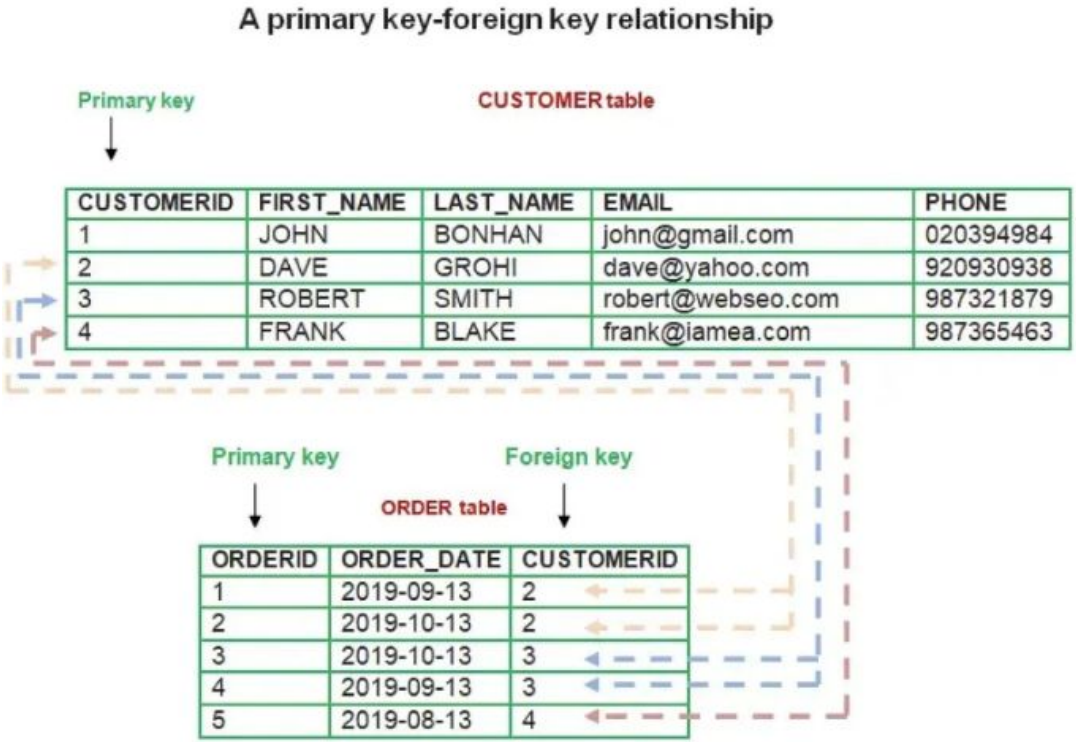
- It is a combination of NOT NULL and UNIQUE.
- A table can have only one primary key.

Example:

```
create table customer (
    cust_id int PRIMARY KEY,
    name VARCHAR(50),
);
```

6. **Foreign Key** - foreign key creates a link between two tables by referencing the primary key of another table.

Primary Key - foreign key Relationship



Primary Key & Foreign Key

```
create table customer (
    cust_id int,
    name VARCHAR(50),
    age int,
    constraint pk1 Primary Key (cust_id)
);
DESC customer;

create table orders (
    order_id int,
    order_num int,
    customer_id int,
    constraint pk2 Primary Key (order_id),
    constraint fk1 Foreign Key (customer_id) REFERENCES customer(cust_id)
);
DESC orders;
```

 # Insert record in customer table

```
insert into customer values(1,"Akash",29);
insert into customer values(2,"Rahul",30);

select * from customer;
```

 # Insert record in orders table

```
insert into orders values(1001, 20, 1);
insert into orders values(1002, 30, 2);
insert into orders values(1003, 40, 2);

select * from orders;
```

 # try to insert a record in foreign key table with a id which is not present in primary key table

```
insert into orders values(1004, 35, 3);
```

Two ways to declare the constraints

1. At the time of variable declaration

```
Create table employee_with_constraints (
        id int UNIQUE,
        name varchar(30),
        salary double CHECK (salary >1000)
);
```

2. In the last line

```
Create table employee_with_constraints1 (
        id int,
        name varchar(30),
        salary double,
        CONSTRAINT unique_emp_id UNIQUE (id),
        CONSTRAINT salary_check CHECK (salary > 1000)
);
```

Note: we can omit the constraint keyword and its name but we have to provide full naming to our constraints so it is easy to debug in case of error