Title: SQL constraints

Objective: To be familiar with constraints in SQL and Implementing it

Theory:

- 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:

NOT NULL - Ensures that a column cannot have a NULL value

UNIQUE - Ensures that all values in a column are different

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - Prevents actions that would destroy links between tables

CHECK - Ensures that the values in a column satisfies a specific condition **DEFAULT**

- Sets a default value for a column if no value is specified

NOT NULL

✓ By default, a column can hold NULL values.
✓ The NOT NULL constraint enforces a column to NOT accept NULL values.
✓ This enforces a field to always contain a value, which means that you cannot insert a new record without adding a value to this field.

➢ **Create table with NOT NULL constraint.**

Example:

```
CREATE TABLE Colleges (   college_id INT NOT NULL,   college_code VARCHAR(20),
college_name VARCHAR(50) );
```

➢ **Add the NOT NULL constraint to a column in an existing table**

Example:

```
ALTER TABLE Colleges
MODIFY COLUMN college_id INT NOT NULL;
```

- **Remove NOT NULL Constraint**

Example:

```
ALTER TABLE Colleges
 MODIFY college_id INT   ;
```

UNIQUE :

- ✓ The **UNIQUE** constraint ensures that all values in a column are different.

- **Create a table with unique constraint**

Example

CREATE TABLE Colleges (college_id INT NOT NULL UNIQUE,   college_code VARCHAR(20) UNIQUE, college_name VARCHAR(50));

- **Add the UNIQUE constraint to an existing column**

For single column

Example

```
ALTER TABLE Colleges
ADD UNIQUE (college_id);
```

For multiple columns

Example

```
ALTER TABLE Colleges
ADD UNIQUE Unique_College (college_id, college_code);
```

- ✦ Here, the SQL command adds the UNIQUE constraint to college_id and college_code columns in the existing Colleges table.
- ✦ Also, Unique_College is a name given to the UNIQUE constraint defined for college_id and college_code columns.

- **DROP a UNIQUE Constraint**

Example

```
ALTER TABLE Colleges
DROP INDEX Unique_College;
```

PRIMARY KEY

- ✓ The PRIMARY KEY constraint uniquely identifies each record in a table.
- ✓ Primary keys must contain UNIQUE values, and cannot contain NULL values.

- ➢ **Create table with PRIMARY KEY constraint Syntax:**

```
CREATE
TABLE
table_name
(    column1
data_type,
  ..........................,
  [CONSTRAINT
constraint_name] PRIMARY KEY
(column1)
);
```

Example

```
CREATE TABLE Colleges (
  college_id INT,
  college_code
VARCHAR(20) ,
college_name
VARCHAR(50),
CONSTRAINT CollegePK  PRIMARY KEY (college_id)
);
```

//Create Colleges table with primary key college_id

- ➢ **Add the PRIMARY KEY constraint to a column in an existing table**

Example

ALTER TABLE Colleges

ADD CONSTRAINT CollegePK PRIMARY KEY (college_id);

- ➢ **DROP a PRIMARY KEY Constraint**

Example

ALTER TABLE Colleges

DROP PRIMARY KEY;

CHECK

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a column it will allow only certain values for this column.

- **CHECK constraint while creating table**

  Example:

  Here we are Applying the CHECK constraint named amountCK the constraint makes sure that amount is greater than 0.

```
CREATE TABLE Orders ( order_id INT PRIMARY KEY,

amount INT,

CONSTRAINT amountCK CHECK (amount > 0)

);
```

- **Add CHECK Constraint in Existing Table**

  Here we add CHECK constraint named amountCK the constraint makes sure that amount is greater than 0.

```
ALTER TABLE Orders

ADD CONSTRAINT amountCK CHECK (amount > 0);
```

- **Remove CHECK Constraint**

```
ALTER TABLE Orders

DROP CONSTRAINT amountCK;
```

DEFAULT

- ✓ the DEFAULT constraint is used to set a default value if we try to insert an empty value into a column.
- ✓ However if the user provides value then the particular value will be stored.

- **Default constraint while creating table**

  The following example set default value of college_country column to 'Nepal'
  <u>Example:</u>

```
CREATE TABLE College (   college_id INT PRIMARY KEY,   college_code
VARCHAR(20),

  college_country VARCHAR(20) DEFAULT 'Nepal'

);
```

- **Add the DEFAULT constraint to an existing column**

Example:

```
ALTER TABLE College

ALTER college_country SET DEFAULT 'Nepal';
```

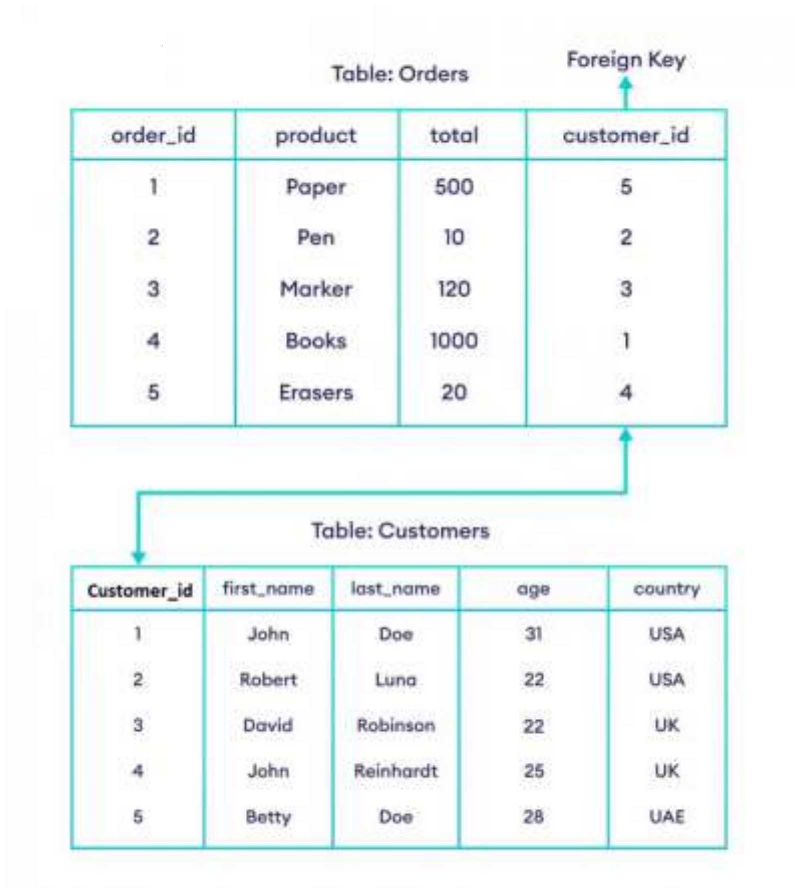
- **Remove Default Constraint**

Example:

```
ALTER TABLE College
ALTER college_country DROP DEFAULT;
```

FOREIGN KEY

The FOREIGN KEY constraint in SQL establishes a relationship between two tables by linking columns in one table to those in another.



order_id	product	total	customer_id
1	Paper	500	5
2	Pen	10	2
3	Marker	120	3
4	Books	1000	1
5	Erasers	20	4

Customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

- ✓ Here, the customer_id field in the Orders table is a FOREIGN KEY that references the customer_id field in the Customers table.
- ✓ This means that the value of the customer_id (of the Orders table) must be a value from the customer_id column (of the Customers table).

The syntax of the SQL FOREIGN KEY constraint is:

```
CREATE TABLE table_name (
column1 data_type,
column2 data_type,
  ………………………….,
 [CONSTRAINT CONSTRAINT_NAME] FOREIGN KEY (column_name)
  REFERENCES referenced_table_name (referenced_column_name)
);
```

Here,

- ✓ table_name is the name of the table where the FOREIGN KEY constraint is to be defined
- ✓ column_name is the name of the column where the FOREIGN KEY constraint is to be defined
- ✓ referenced_table_name and referenced_column_name are the names of the table and the column that the FOREIGN KEY constraint references
- ✓ [CONSTRAINT CONSTRAINT_NAME] is optional

Let us see with following example

- ✓ This table doesn't have a foreign key
- ✓ add foreign key to the customer_id field
- ✓ the foreign key references the id field of the Customers table

```
   -- this table doesn't have a foreign key

CREATE TABLE Customers (
customer_id INT,   first_name
VARCHAR(40),
  last_name VARCHAR(40),   age
INT,
  country VARCHAR(10),
  CONSTRAINT CustomersPK PRIMARY KEY (customer_id)
);
-- add foreign key to the customer_id field
-- the foreign key references the id field of the Customers table
CREATE TABLE Orders (
order_id INT,   product
VARCHAR(40),
  total INT,
  customer_id INT,
  CONSTRAINT OrdersPK PRIMARY KEY (order_id),
  CONSTRAINT CustomerOrdersFK FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

Add the FOREIGN KEY constraint to an existing table

- ✓ add foreign key to the **customer_id** field of Orders the foreign key references the **customer_id** field of Customers

```
ALTER TABLE Orders

ADD FOREIGN  KEY (customer_id) REFERENCES Customers(customer_id);
```

Remove a FOREIGN KEY Constraint

```
ALTER TABLE Orders

DROP FOREIGN KEY CustomerOrdersFK;
```

Problem:

1) create a database named customers_db

```
create database customer_db;

use customer_db;
```

2) create a table named customers with following columns adding constraints following constraints

Customer_id: Primary key

Name: not null

customer_id	**Name**	**Email**	**Age**	**Address**
Int	**varchar(30)**	**varchar(30)**	**Int**	**varchar(30)**

```
CREATE TABLE customers(customer_id int,name varchar(30) NOT NULL,email varchar(255),age int,address varchar(30),PRIMARY KEY(customer_id));
```

OR

```
CREATE TABLE customers(customer_id int,name varchar(30) NOT NULL,email varchar(255),age int,address varchar(30),CONSTRAINT customer_pk PRIMARY KEY(customer_id));
```

OR

```
CREATE TABLE customers(customer_id int PRIMARY KEY,name varchar(30) NOT NULL,email varchar(255),age int,address varchar(30));
```

3) We have already added two constraints (PRIMARY KEY and NULL) while creating table customers. We can add constraints after creating table as well.

Now add the following constraint to the above table named customers

Email must be unique of each customer

```
alter table customers add unique(email);
```

Age must be greater than 18

```
alter table customers  add constraint age_check  CHECK(age>18); OR

alter table customers add CHECK(age>18);
```

The default value of address must be bhaktapur

```
alter table customers alter address SET DEFAULT 'bhaktapur';
```

4) Now insert records of customer with and without violating constraints. PRIMARY KEY

insert into customers

values(1,'ram','ram@gmail.com',22,'kathmandu'); insert into

customers values(2,'sita','sita@gmail.com',25,'pokhara');

Insertion can be performed successfully.

insert into customers values(2,'gita','gita@gmail.com',26,'butwal');

This cannot inserted because **customer_id is** primary key which should be unique and not null **NOT NULL**

insert into customers(customer_id,email,age,address) values (3,'gita@gmail.com',26,'butwal');

select *from customers;

//Now we cannot see NULL in column named name which we have left empty

Note:

If we have a column in a database table with a "NOT NULL" constraint, it means that a value must be provided for that column when inserting a new row, and the database will not allow the insertion of a row with a NULL value in that column.

If we are experiencing a situation where a row is being inserted without providing a value for a column that is marked as "NOT NULL," there may be error in xampp configuration in our case.

But Here, NULL is not seen in column which we left empty.

CHECK

insert into customers values (4,'hari','hari@gmail.com',27,'lalitpur');

// this will works insert into customers values

(11,'ganesh','ganesh@gmail.com',15,'dolkha');

//this will not works because age must be greater than 18 **UNIQUE**

insert into customers values (16,'ramesh','ram@gmail.com',24,'chitwan');

//this will not works because Duplicate entry 'ram@gmail.com' for key 'email' **DEFAULT**

insert into customers(customer_id,name,email,age) values(5,'gopal','gopal@gmail.com',20);

Now,

select * from customers;

//Although we have not inserted the address, we can see that the address has been set to Kathmandu.

FOREIGN KEY

We have already created table **customers** with following columns

customer_id	Name	Email	age	address
Int	varchar(30)	varchar(30)	Int	varchar(30)

Now we have to create table **orders** with following columns.

Order_id	Product_name	Price	Customer_id

and **order_id** is primary key and **customer_id** as foreign key for customer

create table orders(order_id int,product_name varchar(30),price int,customer_id int,primary key(order_id),foreign key (customer_id) references customers(customer_id));

OR

create table orders(order_id int,product_name varchar(30),price int,customer_id int,CONSTRAINT order_pk primary key(order_id),CONSTRAINT custorder_fk foreign key (customer_id) references customers(customer_id));

(Note: CONSTRAINT constraint_name is optional)

Now let update customer table, update customers set name='gita' where customer_id=3; Again insert some rows in orders table

insert into orders values(1,'iron',1200,1); insert into orders values(2,'guitar',4500,2); insert into orders values (3,'phone',57000,1); insert into orders values(4,'printer',88000,3); insert into orders values(5,'drone',125000,2); insert into orders values(6,'smartwatch',7500,4);

insert into orders values(7,'oven',14500,5);

Now, table looks like

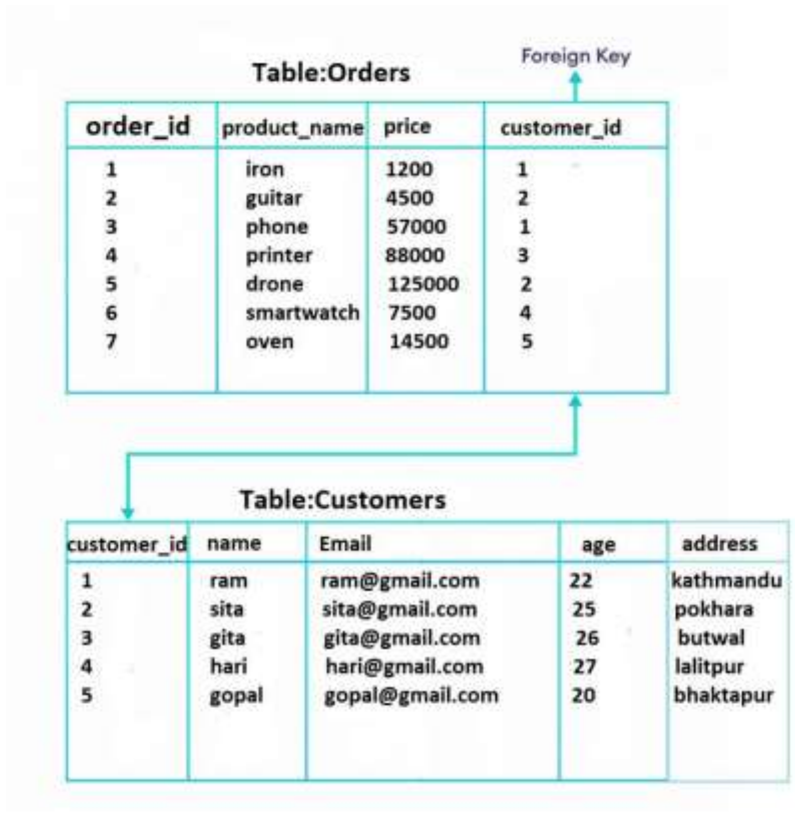

Foreign Key

Table:Orders

order_id	product_name	price	customer_id
1	iron	1200	1
2	guitar	4500	2
3	phone	57000	1
4	printer	88000	3
5	drone	125000	2
6	smartwatch	7500	4
7	oven	14500	5

Table:Customers

customer_id	name	Email	age	address
1	ram	ram@gmail.com	22	kathmandu
2	sita	sita@gmail.com	25	pokhara
3	gita	gita@gmail.com	26	butwal
4	hari	hari@gmail.com	27	lalitpur
5	gopal	gopal@gmail.com	20	bhaktapur

Try to insert customer_id in orders table that does not exist in customers table

insert into orders values(8,'Laptop',158000,6);

This cannot be inserted, because it violates foreign key consraints.

Discussion: (This portion is left for student)

Conclusion: (This portion is left for student)

*******THE END*******