MySQL- Part-1

Basics about databases

# Displays the DATABASE present in MySQL

```
SHOW DATABASES;
```

# using the database world database in the MySQL

```
USE world;
```

# Checking the tables present in world database in the MySQL

```
SHOW TABLES;
```

# Selecting the city table present in world database in the MySQL

```
SELECT * FROM city
```

Creating database

# create table in database

```
USE customer;
```

# Using data base customer created earlier to create the table customer_info

```
CREATE TABLE customer_info (
id INTEGER,
first_name VARCHAR(15),
last_name VARCHAR(15));
```

# Displaying customer_info table

```
SELECT * FROM customer_info;
```

# adding data into customer_info table

```
INSERT INTO
customer_info(id,first_name,last_name)
VALUES(1,'Yash','Gowda');
INSERT INTO
customer_info(id,first_name,last_name)
VALUES(2,'Honisha','Gowda');
```

# Deleting customer_info table

```
DROP TABLE customer_info;
```

# Deleting customer database

```
DROP DATABASE customer;
```

NULL VALUES

```
CREATE DATABASE customer;
SHOW DATABASES;
USE customer;
CREATE TABLE customer_info (
id INTEGER AUTO_INCREMENT,
first_name VARCHAR(15),
last_name VARCHAR(15),
salary INTEGER,
PRIMARY KEY (id)
);
INSERT INTO
customer_info(first_name,last_name,salary)
VALUES
('Jhon','Daniel',50000),
('Krish','Naik',60000),
('Darius','Bengali',70000),
('Chandan','Kumar',40000),
('Deepak','Sharma',NULL);
```

# SQL NULL values

```
SELECT * FROM customer_info
WHERE salary IS null;
```

-- *Displays table which contains salary column has null values*

```
SELECT * FROM customer_info
WHERE salary IS NOT null;
```

-- *Displays table which contains salary column has non null values*

#SQL UPDATE statement to replace null values

```
 UPDATE customer_info
 SET salary=75000
 WHERE id=5;
```

#SQL DELETE statement

```
DELETE FROM customer_info
WHERE id=5;
```

#SQL alter statement

## ADD COLUMN in existing column

```
ALTER TABLE customer_info
ADD email VARCHAR(25);
```

--Adding email column in customer_info table

```
ALTER TABLE customer_info
ADD DOB DATE;
--Adding DOB column in customer_info table
##Alter table and MODIFY column
ALTER TABLE customer_info
MODIFY DOB YEAR;
--Modifying DOB column from DATE to YEAR in customer_info table
##Alter Table to DROP column
ALTER TABLE customer_info
DROP COLUMN email;
--Deleting email column from customer_info table
#Describing the information of table
DESC customer_info;
```

SQL constraints

```
/* SQL Constraints
SQL constraints are used to specify any rules for the records in a table.
Constraints can be used to limit the type of data that can go into a table.
It ensures the accuracy and reliability of the records in the table, and if there is any violation between the constraint and the record action, the action is aborted. Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.
*/
USE customer;
--using customer database created earlier
```

NOT NULL

```
#SQL NOT NULL constraint in table
CREATE TABLE student(
id INTEGER NOT NULL,
first_name VARCHAR(25) NOT NULL,
last_name VARCHAR(25) NOT NULL,
age INTEGER);
```

#SQL altering NOT NULL constraint for table

```
ALTER TABLE student
MODIFY age INT NOT NULL;
```

UNIQUE

#creating new table for UNIQUE constraint for table

```
CREATE TABLE person(
id INT NOT NULL,
first_name VARCHAR(25) NOT NULL,
last_name VARCHAR(25) NOT NULL,
age INT NOT NULL,
UNIQUE(id)
);
INSERT INTO person
VALUES(2,'Harini','Gowda',31);
```

#SQL altering UNIQUE constraint for table

```
ALTER TABLE person
ADD UNIQUE(first_name);
```

#SQL altering UNIQUE constraint for two column of table

```
ALTER TABLE person
ADD CONSTRAINT uc_person
UNIQUE(age,first_name);
```

--uc_person is the index name for assigning the UNIQUE constraint

#SQL dropping UNIQUE constraint for two column of table

```
ALTER TABLE person
DROP INDEX uc_person;
```

PRIMARY KEY

#SQL assigning PRIMARY KEY constraint for table

```
CREATE TABLE person1(
id INT NOT NULL,
first_name VARCHAR(25) NOT NULL,
last_name VARCHAR(25),
age INT,
CONSTRAINT pk_person PRIMARY
KEY(id,last_name)
);
```

#SQL dropping PRIMARY KEY constraint for table

```
ALTER TABLE person1
DROP PRIMARY KEY;
```

#SQL adding PRIMARY KEY constraint for a column in a table

```
ALTER TABLE person1
ADD PRIMARY KEY(id);
```

FOREIGN KEY

```
CREATE TABLE person(
id INT NOT NULL,
first_name VARCHAR(25) NOT NULL,
last_name VARCHAR(25) NOT NULL,
age INT,
Salary INT,
PRIMARY KEY (id)
);
CREATE TABLE department(
id INT NOT NULL,
dept_id INT NOT NULL,
dept_name VARCHAR(25) NOT NULL,
PRIMARY KEY(dept_id),
CONSTRAINT fk_pd FOREIGN KEY(id) REFERENCES person(id)
);
-- Referencing the foreign key for the table department
```

# Update the Foreign Key

```
 DROP TABLE department ;
-- dropping previous table to create the new table without foreign key
CREATE TABLE department(
id INT NOT NULL,
dept_id INT NOT NULL,
dept_name VARCHAR(25) NOT NULL,
PRIMARY KEY(dept_id)
);
ALTER TABLE department
ADD FOREIGN KEY(id)
REFERENCES person(id);
```

--altering the foreign key for the department table with referring the table person having id as primary key

# CHECK constraint

```
CREATE TABLE person_1(
id INT NOT NULL,
first_name VARCHAR(25) NOT NULL,
last_name VARCHAR(25) NOT NULL,
age INT,
salary INT,
PRIMARY KEY(id),
CHECK(salary<50000)
);
```

--Added constraint of salary < 50000

```
INSERT INTO person_1
VALUES
( '1','Yash','Gowda', 33, 40000);
```

--when tried to give the salary value > 50000 constraint error occurs

#DEFAULT constraint

```
CREATE TABLE student_1(
id INT NOT NULL,
first_name VARCHAR(25) NOT NULL,
last_name VARCHAR(25) NOT NULL,
city_name VARCHAR(25) DEFAULT
'Bengaluru'
);
```

--Setting the default value of city_name as Bengaluru

#Dropping DEFAULT constraint from the student_1 table

```
ALTER TABLE student_1
ALTER city_name DROP DEFAULT;
```

SQL Indexes

/* MY SQL Indexes

CREATE INDEX statement in SQL is used to create indexes in tables.

The indexes are used to retrive data from the database more quickly than others.

The user can not see the indexes, and they are just used to speed up queries/searches.

Note: Updating the table with indexes takes a lot of time than updating a table without indexes.

It is because the indexes also need an update.

so, only create indexes on those columns that will be frequently searched against.

*/

# Creating the INDEX for the table department

```
SELECT * FROM department;
```

--selecting the department table to create index

```
CREATE INDEX index_dept_name

ON department(dept_name);
```

--Creating dept_name as index

```
DESC department;
```

--To check the dept_name as index in the table

# Creating the INDEX for the table department for multiple columns

```
ALTER TABLE department

ADD COLUMN books_name

VARCHAR(25);
```

--Added new column book_name for the table for selecting the two columns as index

```
ALTER TABLE department

MODIFY books_name VARCHAR(25)

NOT NULL;
```

--Modifying the books_name column for not containing null values for selecting the two columns as index

```
CREATE INDEX

index_dept_name_books_name

ON department(dept_id,books_name);
```

```
#Dropping INDEX from the department
table
ALTER TABLE department
DROP INDEX
index_dept_name_books_name
```

VIEWS

```
/* VIEWS
Virtual table based on the result of set
of an SQL query
*/
#Creating new table for the VIEW
CREATE TABLE customer_data (
customer_id INT AUTO_INCREMENT,
first_name VARCHAR(24) NOT NULL,
last_name VARCHAR(24) NOT NULL,
quantity INT NOT NULL,
cost INT NOT NULL,
PRIMARY KEY(customer_id)
);
```

```
INSERT INTO customer_data
VALUES
(1,'Yash','Gowda',10,5000),
(2,'Kishore','Dhora',5,500),
(3,'Khirod','Kumar',20,10000);

SELECT * FROM customer_data;
--To check the table
CREATE TABLE customer_adress (
customer_id INT AUTO_INCREMENT,
City VARCHAR(24) NOT NULL,
State VARCHAR(24) NOT NULL,
FOREIGN KEY(customer_id)
REFERENCES
customer_data(customer_id)
);

INSERT INTO customer_adress
VALUES
(1,'Bengaluru','Karnataka'),
(2,'Warangal','Telengana'),
(3,'Vizak','Andra pradesh');
```

```
SELECT * FROM customer_adress;

SELECT * FROM customer_data;

--To check the table

SELECT * FROM customer_data

INNER JOIN customer_adress

USING (customer_id);

#Creating view for the

customer_information by inner join of

the two table data

CREATE VIEW customer_information

as

SELECT * FROM customer_data

INNER JOIN customer_adress

USING (customer_id);

SELECT * FROM

customer_information ;

#Droping customer_information VIEW

DROP VIEW customer_information

SELECT * FROM student;
```

Joins in SQL

```
# Creating new table 1 to perform SQL

joins

CREATE TABLE student(

id INTEGER AUTO_INCREMENT,

first_name VARCHAR(15),

last_name VARCHAR(15),

age INT

);

ALTER TABLE student

ADD PRIMARY KEY(id);

# inserting values toTable 1 to perform

SQL joins

INSERT INTO student

VALUES

(01,'Yash','Gowda',33),

(02, 'Khirod', 'Kumar',29),

(03,'Kishore','Dhora',27),

(04, 'Abihesk','Kumar',34);

 (05,'Nivi','Gowda',28),

(06, 'Haini', 'Gowda',31),

(07,'Pallavi','M S',29),
(08,'Suprithe','R',29),
(09,'Mohan','D P',35),
(10,'Shareef','Raja',35);
# Creating new table 2 to perform SQL joins
CREATE TABLE department(
student_id INT AUTO_INCREMENT,
department_name VARCHAR(25) NOT NULL,
FOREIGN KEY(student_id)
REFERENCES student(id)
);
/*Note: to check tables
DESC department;
SELECT * FROM student;
SELECT * FROM department;) */
# inserting values toTable 2 to perform SQL joins
INSERT INTO department
VALUES
(01,'Mechanical Engineering'),
(02,'Mechanical Engineering'),
(03,'Chemical Engineering'),
(04, 'Chemical Engineering'),
(06, 'Electronics'),
(07, 'Computer science'),
(08, 'Computer science'),
(10,'Nanotechnology');
```

SQL JOIN

#INNER JOINT

```
SELECT * FROM student INNER JOIN department
ON student.id=department.student_id;


SELECT id, student.first_name,
student.last_name, student.age,
department.department_name
FROM student
INNER JOIN department
ON student.id=department.student_id;
```

#LEFT JOIN

```
SELECT * FROM student LEFT JOIN
department
ON student.id=department.student_id;
--selecting all table names
SELECT id, student.first_name,
student.last_name, student.age,
department.department_name
FROM student
LEFT JOIN department
ON student.id=department.student_id;
```

#RIGHT JOIN

```
SELECT * FROM student RIGHT JOIN
department
ON student.id=department.student_id;
--selecting all table names
SELECT id, student.first_name,
student.last_name, student.age,
department.department_name
FROM student
RIGHT JOIN department
ON student.id=department.student_id;
```

# FULL OUTER JOIN

```
SELECT id, student.first_name,
student.last_name, student.age,
department.department_name
FROM student
LEFT JOIN department
ON student.id=department.student_id
UNION
SELECT id, student.first_name,
student.last_name, student.age,
department.department_name
FROM student
RIGHT JOIN department
ON student.id=department.student_id;
```

# CROSS JOIN

```
SELECT id, student.first_name,
student.last_name, student.age,
department.department_name
FROM student
```

```
CROSS JOIN department;
```

# NATURAL JOIN

```
SELECT id, student.first_name,

student.last_name, student.age,

department.department_name

FROM student

NATURAL JOIN department;
```