What is a database:- A collection of data and holds this data in the form of tables

What are tables:- hold the data in form of rows and columns.

It is similar to an excel spreadsheet.

The database provides us the capability to access and manipulate this data.

2 types of databases

1. **Relational databases:-** data stored in the form of rows and columns and tables have relations between them.
   MySQL
   SQL Server
   SQLite
   MariaDB
2. **NOSQL databases:-** key values, document, graph, and table have no relation between them.
   Hbase
   MongoDB
   Cassandra

What is SQL:- It Stands for Structured Query Language. SQL is a way to interact with a database and is used to query a relational DB.

MySQL VS SQL

MySQL is a database while SQL is a way to interact with databases.

SQL Commands

1. Show databases;
2. Create database databasename;
3. Drop database databasename;
4. Use database;:- to go inside a database.
5. Select database();:- to know in which database we are.
6. Create table table_name (name varchar(50) ,age INT ,salary INT)
   Int for numeric
   Varchar for string up to 255 characters
7. Show tables;
8. Describe table_name;:- To describe a table and in place of describe we also can use desc.
9. Drop table table_name;

Session 2

CRUD Operations:- Create, Read, Update, Delete.

1. Create table tablename(firstname varchar(30) ,lastname varchar(20) ,age int ,salary int ,location varchar(29))
2. Select * From table;

3. Insert into table(firstname varchar(30) ,lastname varchar(20) ,age int ,salary int ,location varchar(29)) values(“kapil”, “kumar”, 28, 1000, “banglore”) .
4. **To insert multiple values**:- Insert into table(firstname varchar(30) ,lastname varchar(20) ,age int ,salary int ,location varchar(29)) values(“kapil”, “kumar”, 28, 1000, “banglore”) , (“Tushar\’Sonp”, “kumar”, 28, 1000, “banglore”), (“Tushar’Sonp”, “kumar”, 28, 1000, “banglore”).
5. **NOT NULL:-** Create table tablename(firstname varchar(30) Not NULL ,lastname varchar(20) Not NULL ,age int Not NULL ,salary int Not NULL ,location varchar(29) Not NULL).
6. **Default Values:-** Create table tablename(firstname varchar(30) Not NULL ,lastname varchar(20) Not NULL ,age int Not NULL ,salary int Not NULL ,location varchar(29) default “banglore”).
7. **Default with not null :-** Create table tablename(firstname varchar(30) Not NULL ,lastname varchar(20) Not NULL ,age int Not NULL ,salary int Not NULL ,location varchar(29) not null default “banglore”).

Topic:-

1. CRUD Operations
2. Simple Insert
3. Multiple Insert
4. Datatype Mismatch
5. Null and not null
6. Default values
7. Not null and default

Session 3

Topic:-

1. **Primary Key:-** Help to uniquely identifiy each record and in primary key null and repeated values are not allowed.
   Create table tablename(id int Primary Key, firstname varchar(30) Not NULL ,lastname varchar(20) Not NULL ,age int Not NULL ,salary int Not NULL ,location varchar(29) not null default “banglore”).
2. **Auto Increment Key:-**
   Create table tablename(id int Auto_Increment, firstname varchar(30) Not NULL ,lastname varchar(20) Not NULL ,age int Not NULL ,salary int Not NULL ,location varchar(29) not null default “banglore”, PRIMARY KEY(id)).
3. **Unique key:-**
   Create table tablename(id int Unique Key, firstname varchar(30) Not NULL ,lastname varchar(20) Not NULL ,age int Not NULL ,salary int Not NULL ,location varchar(29) not null default “banglore”).
4. **Primary Key VS Unique Key:-**
   1. You can have only one primary key.
   2. The primary key cannot hold any null.
   3. We should use primary key when we have to uniquely identify each record.
   4. Unique key can hold any number of null values.
   5. So the purpose of unique key is to make sure the values don’t duplicate
   6. We can have only one primary key but multiple unique key in a table.

Session 4

1. **To Select All the columns**:- Select * from tablename;
2. **Select specific columns:-** Select Firstname, lastname from tablename;
3. **Condition:**- Select * From table where age > 29;
4. **For making case sensitive:-** select * from tablename where binary firstname = “tushAr”;
   The above statement will match the exact case and is case sensitive.
5. Select firstname as name, lastname as surname from tablename;
6. **Update:-** update tablename set lastname= “Sinha” where firstname= “Maneesh”;
   update tablename set salary = salary + 5000;
7. **Delete:-** delete from tablename where id = 3;
   To delete all record:- delete from tablename;
8. **Alter:-** for structure changes use alter command. In other words, alter deals with structure manipulation.
   Alter table tablename add column newcol varchar(50);
9. **To drop a columns:-** alter table tablename drop column columnname;
10. **To modify columns:**- alter table tablename modify column firstname varchar(60);
11. **To drop primary key:**- alter table tablename primary key;
12. **To add primary key:**- alter table tablname primary key(id);
13. Truncate table tablename;

DDL VS DML

1. DDL stands for data definition language
2. DML stands for data manipulation language.
3. DDL Deal with table structure while DML deal with data directly.
4. DML has insert, update, delete.
5. DDL create, alter, drop, truncate
6. Truncate internally drops the table and recreates it.

Session 5

Foreign Key Constraint :- The foreign key constraint is used to prevent actions that would destroy links between two tables.

A foreign key is a field in one table that refers to the primary key in another table.

Example:-

```
Create table courses(

Course id int Not Null,

Course_name varchar(30) Not Null,

Course_dourantion_month int Not Null,

Course_fee int Not Null,

Primary_key(course_id)

);
```

```
Create table students(
Student_id int Auto_increment,
Student_fname varcha(30) Not Null,
Student_lname varchar(30) Not Null,
Student_mname varchar(30),
Selected_course int not null default 1,
Student_email varchar(30) Not Null,
Student_phone varchar(15) not null,
Student_Alternate_phone varchar(15),
Enrollment_date TimeStamp Not Null,
Years_of_exp int Not Null,
Student_company varchar(30) Not Null,
Batch_date varchar(30) Not Null,
Source_of_joining varchar(30) Not Null,
Location varchar(30) Not Null,
PRIMARY KEY (student_id),
UNIQUE KEY (Student_email)
);
```

Now here we created two tables first is containing courses which we offer and second contains the details of students which has enrolled for our course, here is problem is that in the second table anyone can fill wrong selected course id i.e 5 which is not in our courses table. Hence the foreign key constraint has come to overcome this problem.

```
Create table students(
Student_id int Auto_increment,
Student_fname varcha(30) Not Null,
Student_lname varchar(30) Not Null,
Student_mname varchar(30),
Selected_course int not null default 1,
```

Student_email varchar(30) Not Null,

Student_phone varchar(15) not null,

Student_Alternate_phone varchar(15),

Enrollment_date TimeStamp Not Null,

Years_of_exp int Not Null,

Student_company varchar(30) Not Null,

Batch_date varchar(30) Not Null,

Source_of_joining varchar(30) Not Null,

Location varchar(30) Not Null,

PRIMARY KEY (student_id),

UNIQUE KEY (Student_email),

FOREIGN KEY (Selected_course) REFRENCES courses(course_id)

);

Parent table:- Courses

Child table:- Students

- So here whenever anyone tries to enter a selected_course number that is not in the courses table then it will give an error.
- But we can't delete the record from course details because of foreign key constraints.
- Selected course is a foreign key in the students table which refers to course_id (primary key) in courses table.
- The table with the foreign key is called the child table.
- The table with primary key is called the parent or referenced table.

Constraints:-

- NOT NULL
- PRIMARY KEY
- UNIQUE KEY
- FOREIGN KEY
- CHECK CONSTRAINT (not supported in mysql)

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

Session 6

- **Distinct:-** Select DISTINCT location from Students;
- **Order By:-** Select Student_fname from students order by years_of_exp desc;
  Select student_fname , years_of_exp from students order by years_of exp, student_fname asc;
- **Limit:-** select * from students limit3;
  Select * from order by entrollment_data limit 0,3; (0 is the starting row and 3 means go though next 3 rows)
  It will give the data with the given limit.
- **Like:-** select * from students where student_fname like '%ra%';
  Select * from studenst where student_fname like '_____';
  % and _ is a wildcard character.
  \ and single ' is use as a except character.

Session 7

Select DISTINCT source_of_joining from students ORDER BY enrollment data desc; (This query is fail while executing.)

DISTINCT & ORDER By would not work together

Order of Execution is :-

1. From (LOADING THE TABLE)
   Select * from students;
2. Select (Projecting Source_of_Joining)
   Select source_of_joining, enrollment_data from students;
3. DISTINCT
   Select distinct source_of_joining, enrollment_date from students;
4. Order by (based on enrollment date it will order by)
   Select source_of_joining from students order by enrollment_date;

Session 8

Aggregate Function (Count, max, sum, min, average)

Here the input can be n numbers of a row but the output can be one number of row.

Count

- Select Count(*) from students; (it will give the count of the number of rows)
- Select count(student_company) from Students;
- Select count(distinct students_company) as num_companies from students;
- Select count(*) from students where batch_date like '%-02-%';

Group By

To know how many people have joined my course got to know about me?

- Select source_of_joining , count(*) from students Group By source_of_joining;

The group by columns is always in select

Select location, count(*) From students group by source_of_joining; (this will not work).

Min & Max

Select min(year_of_exp) from students;

Select max(year_of_exp) from students;

Select source_of_joining, max(years_of_exp) from students group by source_of_joining;

Session 9

Whenever try to insert float or decimal in int data type column then it round of the decimal value to an integer. So for this, we have a new data type called decimal(number of digits, and one of the digits after decimal.

Example:-

```
Create table courses_new(
Course_id int Not Null,
Course_name varchar(30) Not Null,
Course_duration_months decimal(3,1) Not Null,
Course_fee int Not Null,
Primary Key (course_id)
);
```

Use of Timestamp:- To record the time while the data is inserted or updated

Example:-

```
Create table courses_new(
Course_id int Not Null,
Course_name varchar(30) Not Null,
Course_duration_months decimal(3,1) Not Null,
Course_fee int Not Null,
Changed_at TIMESTAMP Default Now() ON UPDATE NOW(),
Primary Key (course_id)
);
```

Note:- Inplace of Now we can also use Current_Timestamp.

Session 10

- Select * from student where location= “bangalore”;
- Select * from student where location != “bangalore”;
- Select * from courses where course_name like “%data%”;
- Select * from courses where course name not like “ %data%”;
- Select * from students where year_of_exp < 8 and source_of_joining and location = ‘bangalore’;
- Select * from students where year_of_exp < 8 or year_of_exp > 12;
- Select * from students where year_of_exp not between 8 and 12;
- Select * from students where student_company in (“Flipkart”, “Walmart”, “Microsoft”);

Case Statements

```
Select course_id, course_name, course_fee,
        Case
                When course_duration_months >4 then ”masters”;
                Else “diploma”;
        End as course_type
From courses;
```

Session 11

Joins

Inner Join:-

- By default join is an inner join.
- Only the matching is considered. Non-matching records are discarded.

Select students.student_fname, students.student_lname, courses.course_name from students join courses on students.selected_course = courses.course_id;

Left Outer Join:-

- All the matching records from left and right are considered.
- All the non-matching records in the left table which does not have the match in the right padded with null.

Right Outer Join:-

- All the matching record from left and right table are considered.
- All the non matching records in the right table which does not have the match in the left padded with null.

Full Outer Join:-

- Its is used as union.
- All the matching records.

- Non matching records from left.
- Non matching records from right.

Session 12

Where Vs Having Clause in SQL

- Where clause is used to filter the individual records before aggregation.
- Having clause is used to filter the individual records after aggregation.
- Where clause is used before group by and do filtering on individual records.
- Having clause is used after group by and do filtering on aggregated records.
- We can use where and having in the same query also.
- Where is more performant than having.
- Select source_of_joining, count(*) as total from students group by source_of_joining where total > 1; (this query is not work bcz of where clause)
    - Select source_of_joining, count(*) as total from students group by source_of_joining having total > 1; (this query will work)
- Select source_of_joining, count(*) as total from students where source_of_joining = “linkedin” group by source_of_joining;

  Select source_of_joining, count(*) as total from students group by source_of_joining having source_of_joining = “linkedin”;

Session 13

Over Partition By Clause

Select firstname, lastname, employee.location, total_count, avg_salary from employee join (select location, count(location), avg(salary) from employee group by location) temptable on employee.location = temptable.location;

Or the other way of writing this query by Partition By Clause is:-

Select firstname, lastname, location, count(location) over(Partition By Location), avg(salary) over(Partition By location) as average from employee;

Session 14

Row_number()

Select firstname, lastname, salary, row_number() over (order by salary desc) from employee;

To find the 5th highest salary:- select * from (select firstname, lastname, salary, row_number() over(order by salary desc) as rownum from employee) temptable where rownum = 5;

This problem statement is to assign row numbers for partitions based on each locaton.

Select firstname, lastname, salary, row_number() over(partition by location order by salary desc) from employee;

Find the highest salary getters at each locations.

Select * from (Select firstname, lastname, salary, row_number() over(partition by location order by salary desc) as rownum from employee;) temptable where rownum = 1;

Session 15

Rank & Dense Rank

- If there are no duplicates then row_number, rank, and dense rank lead to similar results
- Only the difference comes when there are duplicates.
- **Rank:-** For duplicates same rank is assigned and the for the next entry it skips the ranks.
- **Dense Rank:-** It does not skip any ranks in between and for duplicate it will assign the same rank to both.
- Whenever you do not have duplicates use row_number.