SQL NOTES

COVER THIS TOPIC FIRST

Link to practice SQL: **https://www.w3schools.com/sql/sql_wildcards.asp#gsc.tab=0**

Content

SQL Introduction

Data:

Data Refers to small volume of facts and Figures.

Ex:- Name, Age, Marks, Gender of one student

Database:

Database refers to Large volume of Facts and figures.

Ex:- Name, Age, Marks, gender of all the students in a class.

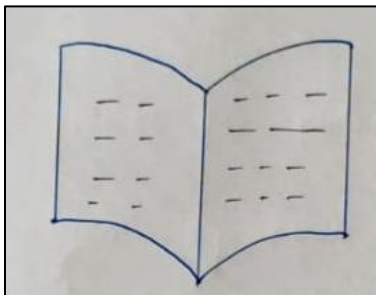## Management of facts and figures:

There are 2 things in managing

→ Storing Permanently

→ Retrieving Efficiently

Note:

Managing the Facts and Figures are very important.

How data was stoned in olden days

In olden days the data was stored on Ledgers.

Ledger

The large volume of facts and figures stored on the ledger should be transfered to computers and managed.

Then people started to demand for a software to manage the large volume of a facts and figures (Database).

So the Database Management Software was invented .

DBMS: DataBase Management Software

These softwares are used to manage the data in the computer.

Ex:- SyBase, Informix, Oracle Database, MySQL, etc
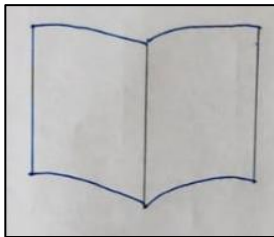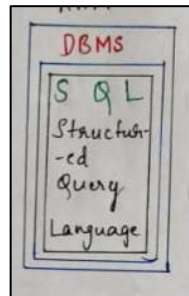
Ledger	RAM	HDD

SQL:

SQL stands for Structured query Language It is a Language. It is a language which is used to communicate with the database.

Rules to follow while storing the data in database.

1. Create a table
2. Specify the columns and its datatype
3. Insert the data

Ledger	RAM	HDD

Ledger
name: Ankit | gender: Male
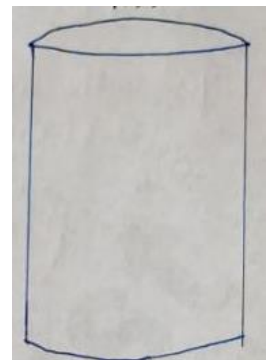age: 22 | percentage: 78
Person

Oracle DBMS
Create Table Person
(
name varchar2 (10),
age number,
gender varchar2 (6),
percentage number
);

name	age	gender	Percent-age
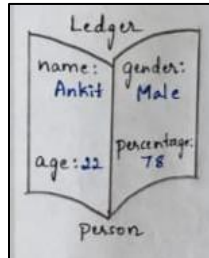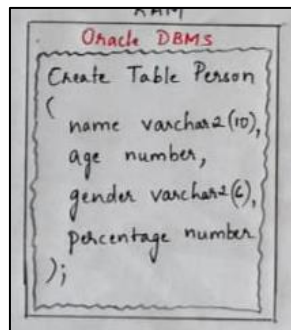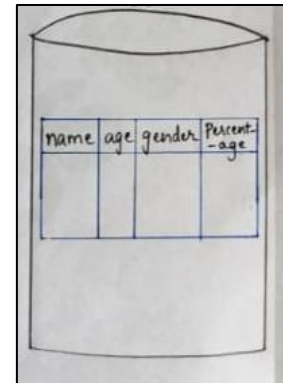
Creating a table

Create table Person

(

   name varchar2(10),

   age number,

   gender varchar2(6),

   percentage number

);

Ex: Create a table for storing the following details of a patient.

P_id: 13 | Treatment: Operation
P_name: Shivani |
P_gender: Female | Address: Bangalore
Patient
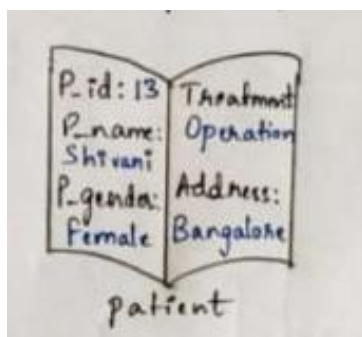
Create table Patient

(

　P_id　number,

　P_name　varchar2(10),

　P_gender varchar2(6),

　Treatment varchar2(15),

　Address varchar2(15)

);

Patient:

P_id	P_name	P_gender	Treatment	Address

Inserting Data into Table
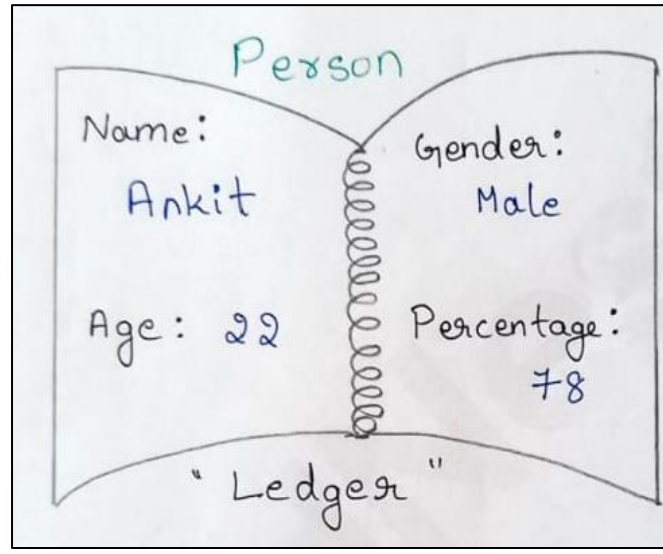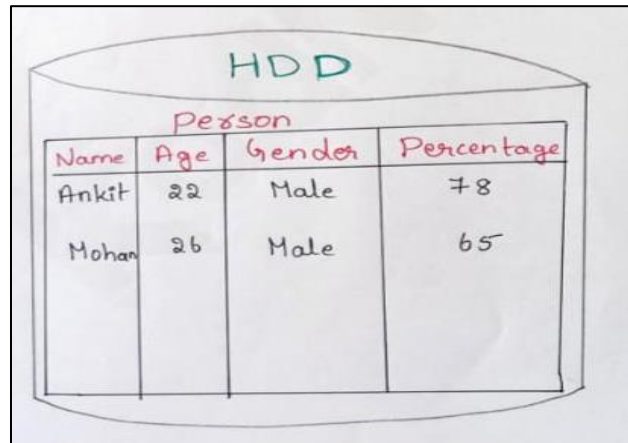
1. Create a table

2 Specify the columns and its datatype

3. Insert the data

RAM

HDD

PERSON

RAM

HDD

Executing on oracle software

- ❖ Go to start.
- ❖ Select oracle database 10g express edition.
- ❖ Select run SQL command line.

SQL >

SQL> conn

Enter user-name: system

Enter password: system // It will be not visible Connected.

SQL> create table Person

 (

 name varchar2(10),

 age number,

 gender varchar2(6),

 percentage number

);

Table created.

SQL> insert into Person values('Ankit', 22; Male', 78);

1 row created.

SQL> insert into Person values ('Mohan', 26, Male', 65);

HDD
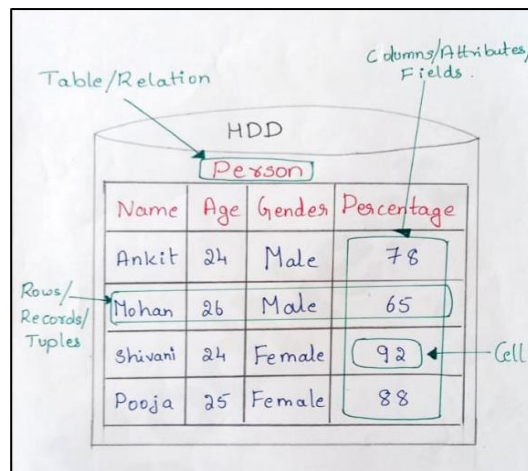
SELECT:

In SQL, the data can be retrieved using select command.

Syntax:

Select column-name from table where condition

Que1) Write a query to display the id of all the students from the student table

ans) select id from student

Output:-

```
ID
---
 1
 2
 3
 4
 5
```

5 rows returned in 0.00 seconds

Que2) Write a query to display id and name of all students from the student table.

Ans) select id, name from student

Output:

```
ID NAME
--- ----------
 1 Ankit
 2 Roshan
 3 Rashmi
 4 Sonal
 5 Chandan
```

5 rows returned in 0.00 seconds

Que3) Write a query to display id, name and age of all the students from the student table.

Ans) select id, name, age from student

Output:-

```
ID NAME             AGE
--- ----------  ----------
  1 Ankit          26
  2 Roshan         24
  3 Rashmi         28
  4 Sonal          22
  5 Chandan        27
```

5 rows returned in 0.00 seconds

Que4) Write a query to display id, name, age and gender of all the students from the student table.

Ans) select id, name, age, gender from student

Output:-

```
ID NAME             AGE GENDER
--- ----------  ---------- ---------
  1 Ankit          26 Male
  2 Roshan         24 Male
  3 Rashmi         28 Female
  4 Sonal          22 Female
  5 Chandan        27 Male
```

5 rows returned in 0.00 seconds

Que5) Write a query to display all the data from student table.

Ans)   select id, name, age, gender, sem from student

                              (Or)

                    select * from student

Output:

```
 ID NAME              AGE GENDER        SEM
 ___ _____   _____   _____   _____
   1 Ankit           26 Male            6
   2 Roshan          24 Male            7
   3 Rashmi          28 Female          7
   4 Sonal           22 Female          6
   5 Chandan         27 Male            8
```

               5 rows returned in 0.00 seconds

Que6) Write a query to display name of students whose age is greater than 25 from the student table.

 Ans)   select name from student where age >25

Output:

```
NAME
_____
Ankit
Rashmi
Chandan
```

          3 rows returned in 0.00 seconds

Que7) Write a query to display id, name, age and gender from student table whose gender is Male.

 Ans) select id, name, age, gender from student where gender = 'Male'

Output:

```
ID NAME                AGE GENDER
----- ---------- ---------- ----------
  1 Ankit           26 Male
  2 Roshan          24 Male
  5 Chandan         27 Male
```

3 rows returned in 0.00 seconds

Que8) Write a query to display all the columns where sem is 6 from the student table.

 Ans) select id, name, age, gender, sem from student where sem=6

(or)

select * from student where sem=6

Output:

```
ID NAME              AGE GENDER                SEM
--- ---------- ---------- ---------- ----------
  1 Ankit          26 Male                 6
  4 Sonal          22 Female               6
```

2 rows returned in 0.00 seconds

Retrieving the data can be done is following two ways:

1. Projection

2. Selection

Projection:

Projection is the process of retrieving all the records from table without restriction. To perform projection select command and from clause is used.

Selection:

Selection is the process of retrieving the specific records from the table with restriction. To perform selection, select command, from, and where clause is used.

Select with Operators

Write query to display all the columns from Employee table.

> Select * from Employee

```
 ID NAME     EMAIL                 AGE GENDER      PHONE     SALARY
---- -------  -------------------- ---------- ------ ---------- ----------
   1 Ashwin   ashwin@gmail.com      23 Male    9988776655      30000
   2 Meghana  meghana@gmail.com     25 Female  9988776655      40000
   3 Bharath  bharath@gmail.com     24 Male    9977886655      55000
   4 Shivani  shivani@gmail.com     27 Female  9988556622      45000
   5 Suraj    suraj@gmail.com       26 Male    9987665544      42000
```

Write a query to display all the Columns from Employee table where salary is greater. than 41000

> select * from Employee where salary>41000

```
 ID NAME     EMAIL                 AGE GENDER      PHONE     SALARY
--- -------  -------------------- ---------- ------ ---------- ----------
   3 Bharath  bharath@gmail.com     24 Male    9977886655      55000
   4 Shivani  shivani@gmail.com     27 Female  9988556622      45000
   5 Suraj    suraj@gmail.com       26 Male    9987665544      42000
```

Write a query to display the id, name, email and salary from employee table by increasing the salary of 1000 rs.

> select id, name, email, salary + 1000 from Employee.

```
 ID NAME     EMAIL                   SALARY+1000
---- ------- -------------------- ------------
   1 Ashwin   ashwin@gmail.com            31000
   2 Meghana meghana@gmail.com           41000
   3 Bharath bharath@gmail.com           56000
   4 Shivani shivani@gmail.com           46000
   5 Suraj   suraj@gmail.com             43000
```

Write a query to display the id, name, email and Salary from employee table by reducing the Salary of 1000 rs.

> select id, name, email, salary - 1000 from Employee

```
 ID NAME     EMAIL                   SALARY-1000
---- ------- -------------------- ------------
   1 Ashwin   ashwin@gmail.com            29000
   2 Meghana meghana@gmail.com           39000
   3 Bharath bharath@gmail.com           54000
   4 Shivani shivani@gmail.com           44000
   5 Suraj   suraj@gmail.com             41000
```

Write a query to display the id, name, email and salary from employee table by giving hike of 10 percent

> select id, name, email, salary + (salary * 10/100) from Employee

```
 ID NAME     EMAIL                SALARY+(SALARY*10/100)
------- ------- -------------------- --------------------
   1 Ashwin   ashwin@gmail.com                    33000
   2 Meghana meghana@gmail.com                   44000
   3 Bharath bharath@gmail.com                   60500
   4 Shivani shivani@gmail.com                   49500
   5 Suraj   suraj@gmail.com                     46200
```

Write a query to display the id, name, email and Salary from employee table by reducing the salary by 20 percent

> select id, name, email, salary- (salary*20/100) from Employee

```
 ID NAME     EMAIL                 SALARY-(SALARY*20/100)
--- -------  --------------------  ----------------------
  1 Ashwin   ashwin@gmail.com                       24000
  2 Meghana  meghana@gmail.com                      32000
  3 Bharath  bharath@gmail.com                      44000
  4 Shivani  shivani@gmail.com                      36000
  5 Suraj    suraj@gmail.com                        33600
```

Write a query to display the half yearly Salary from Employee table.

> select salary*6 from Employee

```
SALARY*6
---------
   180000
   240000
   330000
   270000
   252000
```

Precedence of Operators:-

OPERATORS	PRECEDENCE
*	1
/	2
+	3
-	4

UPDATE

To update or modify the existing record or cell in a table is called as updation. I can be achieved by using update command in SQL

Syntax :-

Update table_name set column_name = new_value

      where condition;

Eg: Write a query to update the email to ' vivek123@g mail.com' from customer with name Vivek in Bank table

> update bank set email='vivek123@gmail.com' where name = 'Vivek'

```
 ACCNO NAME     EMAIL                      PHONE     BALANCE PASSWORD
------- ------- -------------------- ---------- ---------- ----------
  1111 Akshay   akshay@gmail.com     9988725655      40000 akshay1111
  2222 Darshan  darshan@gmail.com    9988776655      65000 darsh2222
  3333 Pooja    pooja@gmail.com      9977886865      21000 pooja3333
  4444 Vivek    vivek123@gmail.com   9988525622      83000 vivek4444
  5555 Darshan  darshan@gmail.com    9987865544      62000 darsh5555
```

Write a query to update the password to 'pooja7777' for the customer with name 'pooja' in Bank table

> update bank set password = 'pooja7777' where name = 'pooja'

```
 ACCNO NAME     EMAIL                      PHONE     BALANCE PASSWORD
---------- ------- -------------------- ---------- ---------- ----------
  1111 Akshay   akshay@gmail.com     9988725655      40000 akshay1111
  2222 Darshan  darshan@gmail.com    9988776655      65000 darsh2222
  3333 Pooja    pooja@gmail.com      9977886865      21000 pooja7777
  4444 Vivek    vivek123@gmail.com   9988525622      83000 vivek4444
  5555 Darshan  darshan@gmail.com    9987865544      62000 darsh5555
```

Write a query to update the balance to 91000 for the customer with name 'Darshan' and accno 5555 in Bank table

> update bank set balance = 91000 where name =' Darshan' and accno=5555

```
 ACCNO NAME     EMAIL                      PHONE      BALANCE PASSWORD
------- -------- -------------------- ---------- ---------- ----------
  1111 Akshay   akshay@gmail.com     9988725655      40000 akshay1111
  2222 Darshan  darshan@gmail.com    9988776655      65000 darsh2222
  3333 Pooja    pooja@gmail.com      9977886865      21000 pooja7777
  4444 Vivek    vivek123@gmail.com   9988525622      83000 vivek4444
  5555 Darshan  darshan@gmail.com    9987865544      91000 darsh5555
```

DELETE

To delete the existing records in the table, delete command is used.

Syntax of delete command :-

Delete from table_name where condition

Write a query to delete all the records from Amazon table

> delete from Amazon

```
5 row(s) deleted.
```

Write a query to delete the record of Customer whose name is 'priya' from table.

> delete from Amazon where name = 'priya'

    1 Row'(s) deleted.

```
    ID NAME     EMAIL                  PLAN
------ -------  -------------------- ----------
    11 Sunil    sunil@gmail.com        Basic
    12 Ravi     ravi@gmail.com         Basic
    14 Anusha   anusha@gmail.com       Premium
    15 Ravi     ravi@gmail.com         Premium
```

4 rows returned in 0.00 seconds

Write a query to delete the record of customer where name is 'Ravi' and plan is ' Premium'

> delete from Amazon where name='Ravi'

   and plan 'Premium'

  1 Row'(s) deleted

```
    ID NAME      EMAIL                      PLAN
 ----- -------   -------------------------  ----------
    11 Sunil     sunil@gmail.com            Basic
    12 Ravi      ravi@gmail.com             Basic
    14 Anusha    anusha@gmail.com           Premium
```

FUNCTIONS

Que1) Write a query to display names in uppercase from student table.

Ans) select upper (name) from student

Output-

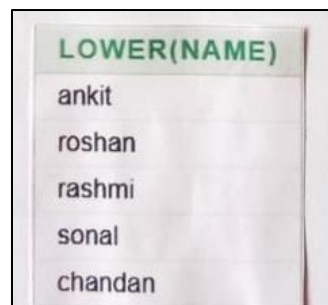| UPPER(NAME) |
|---|
| ANKIT |
| ROSHAN |
| RASHMI |
| SONAL |
| CHANDAN |

Que2) Write a query to display names in lowercase from student table.

ans) select lower(name) from student

Output:-

LOWER(NAME)
ankit
roshan
rashmi
sonal
chandan

Que3) Write a query to display length of all name from student table.

Ans) select length (name) from student

Output:

LENGTH(NAME)
5
6
6
5
7

Que4) Write a query to concentrate the data of name and age from student table.

Ans: select concat (name, age) from student

Output:

CONCAT(NAME,AGE)
Ankit26
Roshan24
Rashmi28
Sonal22
Chandan27

Que5) Write a query to the substring of name from 2nd position in student table.

Ans) Select substr(name, 2) from student

Output:

SUBSTR(NAME,2)
nkit
oshan
ashmi
onal
handan

Que6) Write a query to the substring of name from 2nd position and extract 3 characters in student table.

Ans) select substr (name, 2, 3) from student

Output:

SUBSTR(NAME,2,3)
nki
osh
ash
ona
han

Que7) Write a query to display the position of a the Character 'a' in all the names in student table.

Ans) select instr(name, 'a') from student

Output:-

INSTR(NAME,'A')
0
5
2
4
3

Que8) Write a query to trim leading '2' in all age in student table.

ans) select trim (leading '2' from age) from student

Output:-

TRIM(LEADING'2'FROMAGE)
6
4
8
-
7

Que9) Write a query to the trim trailing 'e' in all gender in student table.

ans) select trim (tralling ' e' from gender) from student

Qutput:

TRIM(TRAILING'E'FROMGENDER)
Mal
Mal
Femal
Femal
Mal

Que10) Write a query to display the minimum age of student table.

ans) select min(age) from student
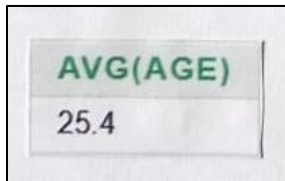
Output:

MIN(AGE)
22

Que11) Write a query to display the average of all the ages of student.

Ans) select avg (age) from student

Output:
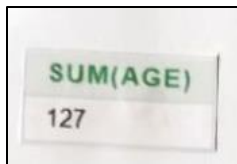
AVG(AGE)

25.4

Que12) Write a query to display the sum of all the ages of student table.

ans) select sum(age) from student

Output:

SUM(AGE)

127

Que13) Write a query to display the maximum of age of student table.

Ans) select max(age) from student

Output:
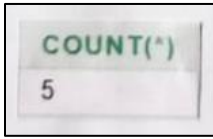
MAX(AGE)

28

Que14) Wnite a query to display the total number of rows of student table.

Ans)select count(*) from student

Output:-

Single Row Functions:

Single row functions are such functions which accepts single row or multiple rows as input but produces one result per input.

Single Rows Functions examples:-

upper()

lower()

length ()

concat()

substr()

instr()

trim()

Multiple Row Functions:-

Multiple row functions are such functions accepts single row or multiple rows as input but produces one result per group.

Multiple Row Functions examples:-

min()

max()

sum()

avg ()

count()

Group by, Having, & Order by Clause

Que1) Write a query to display the team and sum of salaries in each team from Employee table.

ans) select team, sum(salary)

        from Employee

            group by team

Output:-

TEAM	MAX(SALARY)
Support	45000
Operations	60000
Testing	65000
Developer	90000

Que 2) Write a query to display the team and maximum salary of the employees in each team from Employee table.

Ans) select team, max(salary)

      from Employee

          group by team

Output:

TEAM	MAX(SALARY)
Developer	90000
Testing	65000
Operations	60000
Support	45000

Que3) Write a query to display the team & sum of salaries in each team from Employee table where sum of salary is greater than 100000.

Ans) select team, sum(salary)

from Employee

group by team

having sum (salary)> 100000

Output:

TEAM	SUM(SALARY)
Developer	240000
Testing	115000

Que4) Write a query to display the team and maximum Salary in each team where maximum salary in greater than 40000 in Employee table.

Ans) Select team, max(salary)

from Employee

group by team

having max(salary) > 40000

Output:

TEAM	MAX(SALARY)
Support	45000
Developer	90000
Testing	65000
Operations	60000

Que5) WAQ to display the team and maximum salary in each team where maximum salary is greater than 40000 in Employee

ans) select team, max (salary)

from employee

group by team

having max salary) >10000

order by max(salary) asc

Output:

TEAM	MAX(SALARY)
Support	45000
Operations	60000
Testing	65000
Developer	90000

Que6) WAQ to display the team and maximum salary in each team where maximum salary is greater than 40000 in Employee table in descending order.

Ans) select team, max(salary)

from employee

group by team

having max(salary)>4000 .

Order by max (salary) desc

Output:

TEAM	MAX(SALARY)
Developer	90000
Testing	65000
Operations	60000
Support	45000

Que7) WAQ to display the team and maximum salary in each team Where age is greater than 23 and the maximum salary than 46000 from employee table in descending order.

Ans) select team, max (salary)

  from Employee

   where age>23

    group by team

     having max (salary)>46000

      order by max(salary) desc

Output:-

TEAM	MAX(SALARY)
Developer	90000
Testing	50000

Structure of select

Select → What data to be displayed

from→ from which table

where → to put condition on normal column

group by→ to group the data

having→ to put condition on aggregate function

order by→ to sort the result in ascending/descending order.

Sub Queries

Subquery:-

- A subquery is a query within another SQL query and embedded with the where clause.
- A subquery is used to return the data that will be used in the main query as a condition to further restrict the data to be retrieved.

Que1) WAQ to display the id, name and salary of the employees whose salary is greater than Suraj's salary from the Employee table.

Ans) select id, name, salary

from Employee

where salary>(select salary from Employee

where name ='Suraj')

Output:

ID	NAME	SALARY
11	Ankit	90000
21	Rimjhim	65000
14	Lokesh	60000
23	Naveen	70000

Que2) Write a query to display the id, name, age and salary of the employees whose age is equal to Kavya's age from the Employee table.

Ans) select id, name, age, salary from Employee

Where age= (select age from Employee

Where name ='kavya')

Output:

ID	NAME	AGE	SALARY
19	Pooja	24	40000
20	Suraj	24	50000
9	Kavya	24	30000

Que3) WAQ to display name, age and team of all the employees whose age is greater than 'Rimjhim' age and who is working in the same team where Naveen is working from the Employee table.

Ans) select name, age, team

from Employee

where age> (select age from Employee

where name ='Rimjhim')

and

team =(select team from Employee where name ='Naveen')

Output:

NAME	AGE	TEAM
Ankit	28	Developer
Pooja	24	Developer
Naveen	26	Developer

Joins:

Joins: A join clause is used to combine the records from two or more tables by comparing them.

Types of joins:-

1. Equi join or Inner join

2. Natural join

3. Outer Join

> ➢ Left Outer join
> ➢ Right Outer join
> ➢ Full Outer join

4. Cross join

Equi Join: It is used to compare the tables and returns all the rows when there is atleast one match in both tables. It is also called as Inner join

Inner Join: It is used to compare the tables and returns all the rows when there is atleast one match in both table but it maintains only one copy of the column common

Que) Write a query to display all the rows from table 1 and table2 where table1 column B is equal to table2 column B using inner join keyword..

ans) select *

from table1 inner join table2

on table1. B=table 2.B

Output:-

A	B	B	C
3	4	4	1
5	6	6	8

Que) Write a query to display all the rows from table1 and table2 using natural join keyword.

Ans) Select *

from table1 natural join table2

Output:

B	A	C
4	3	1
6	5	8

Que) Write a query to display the emp_ld, name and salary from Employee table whose dept_id in Employee table is equal to dept_id in Department table using the equi join

ans) select emp_id, emp_name, emp_salary from Employee, Department where Employee. dept_id= Department.dept_id

Output:

EMP_ID	EMP_NAME	EMP_SALARY
11	Pooja	30000
12	Darshan	40000
13	Soumya	50000
14	Lokesh	60000
15	Kavya	70000
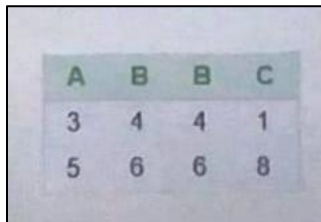
Que) Write a query to display all the rows from tables1 & table2 where table1 column column B is equal to table2 column B using equi join.

Ans) select *

     from table1, table2

       where table 1.B =table 2. B

Output:-

A	B	B	C
3	4	4	1
5	6	6	8

Left Outer Join: It returns all the rows from the left table & matched rows from the right table.

Right Outer Join: It returns all the rows from the right talbe & matched rows from the left table.

Full Outer Join: It returns all the rows(matched and unmatched) from the both the table.

Que) Waite a query to display all the rows from table1 and table2 using left outer join keyword.

Ans) select *

     fram table1 left outer join table 2

       on table1.B= table 2.B

Output:-

A	B	B	C
3	4	4	1
5	6	6	8
7	8	-	-
1	2	-	-
-	-	3	5

Que) Write a quely to display all the rows from table1 and table2 using right outer join keyword.

Ans) select *

      from table1 right outer join table 2

         on table1.B=table 2.B

Output:-

A	B	B	C
3	4	4	1
5	6	6	8
-	-	3	5

Que) Write a query to display all the rows from table1 and table2 using full outer join keyword.

Ans) select *

      from table1 full outter join table2

         on table1. B=table2.B

Output:

A	B	B	C
3	4	4	1
5	6	6	8
7	8	-	-
1	2	-	-
-	-	3	5
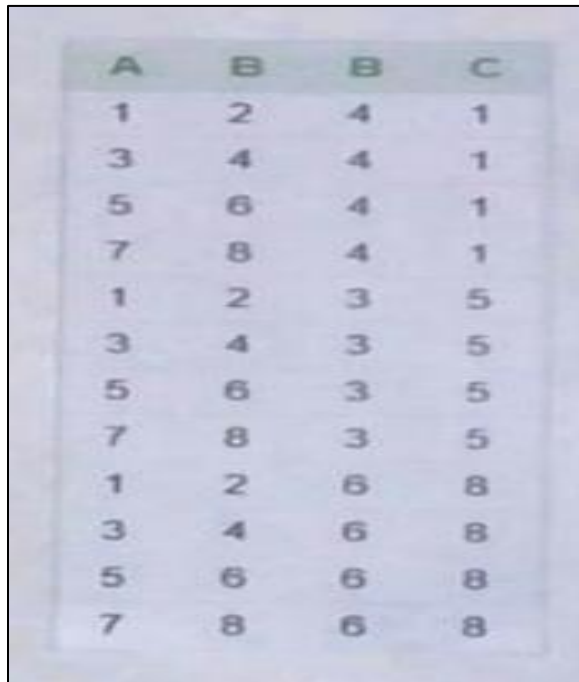
Cross Join: It returns cartesian product of the records from both the tables. It is also called as cartesian join.

Que) Write a query to display all the rows from tablet and table 2 using cross join keyword.

Ans) Select *
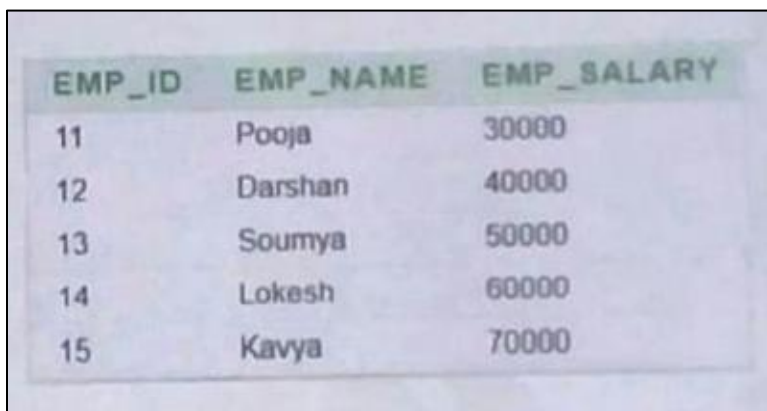
       from tablet1 cross join table 2

Output:

A	B	B	C
1	2	4	1
3	4	4	1
5	6	4	1
7	8	4	1
1	2	3	5
3	4	3	5
5	6	3	5
7	8	3	5
1	2	6	8
3	4	6	8
5	6	6	8
7	8	6	8

Que) Write a query to display all the rows from employee table and Department table whose dept_id in Employee table is equal to dept_id in Department table by uing Inner join keyword.

Ans) select * from Employee inner join Department on Employee.dept_id = Department. dept-id

Output:

EMP_ID	EMP_NAME	EMP_AGE	EMP_SALARY	DEPT_ID	DEPT_ID	DEPT_NAME	DEPT_HEAD
11	Pooja	23	30000	1	1	Development	Sachin
12	Darshan	22	40000	1	1	Development	Sachin
13	Soumya	24	50000	3	3	Support	Chandan
14	Lokesh	23	60000	2	2	Testing	Gourav
15	Kavya	25	70000	2	2	Testing	Gourav

Que) Write a query to display all the rows from Employee table and Department table by uing natural join keyword.

Ans) select * from Employee natural join Department

Output:

DEPT_ID	EMP_ID	EMP_NAME	EMP_AGE	EMP_SALARY	DEPT_NAME	DEPT_HEAD
1	11	Pooja	23	30000	Development	Sachin
1	12	Darshan	22	40000	Development	Sachin
3	13	Soumya	24	50000	Support	Chandan
2	14	Lokesh	23	60000	Testing	Gourav
2	15	Kavya	25	70000	Testing	Gourav

Que) Write a query to perform the left outer join on employee table and Department table.

Ans) select *

from Employee left outer join Department

on Employee.dept_id = Department.dept_id

Output:-

EMP_ID	EMP_NAME	EMP_AGE	EMP_SALARY	DEPT_ID	DEPT_ID	DEPT_NAME	DEPT_HEAD
12	Darshan	22	40000	1	1	Development	Sachin
11	Pooja	23	30000	1	1	Development	Sachin
15	Kavya	25	70000	2	2	Testing	Gourav
14	Lokesh	23	60000	2	2	Testing	Gourav
13	Soumya	24	50000	3	3	Support	Chandan

Que) Write a query to perform the right outer join on Employee table & Department table.

Ans) select *

from Employee right outer join Department

on Employee.dept_id = Department.dept_id

Output:

EMP_ID	EMP_NAME	EMP_AGE	EMP_SALARY	DEPT_ID	DEPT_ID	DEPT_NAME	DEPT_HEAD
11	Pooja	23	30000	1	1	Development	Sachin
12	Darshan	22	40000	1	1	Development	Sachin
13	Soumya	24	50000	3	3	Support	Chandan
14	Lokesh	23	60000	2	2	Testing	Gourav
15	Kavya	25	70000	2	2	Testing	Gourav
					4	Operations	Seema

Que) Write a query to perform the full outer join on Employee table and Department table.

Ans) select *

from Employee full outer Join Department

on Employee.dept_id = Department.dept_id

Output:

EMP_ID	EMP_NAME	EMP_AGE	EMP_SALARY	DEPT_ID	DEPT_ID	DEPT_NAME	DEPT_HEAD
11	Pooja	23	30000	1	1	Development	Sachin
12	Darshan	22	40000	1	1	Development	Sachin
13	Soumya	24	50000	3	3	Support	Chandan
14	Lokesh	23	60000	2	2	Testing	Gourav
15	Kavya	25	70000	2	2	Testing	Gourav
-	-	-	-	-	4	Operations	Seema

NORMALIZATION

Normalization is the process of decomposing or splitting the tables to resolve the problems.

Problems like:-

1. Spurious Tuples

2. Anomalies

3. Data Redundancy

1. Spurious Tuples: Spurious Tuples are such tuples which are not present in original table but gets generated. When table is decomposed and further joined.
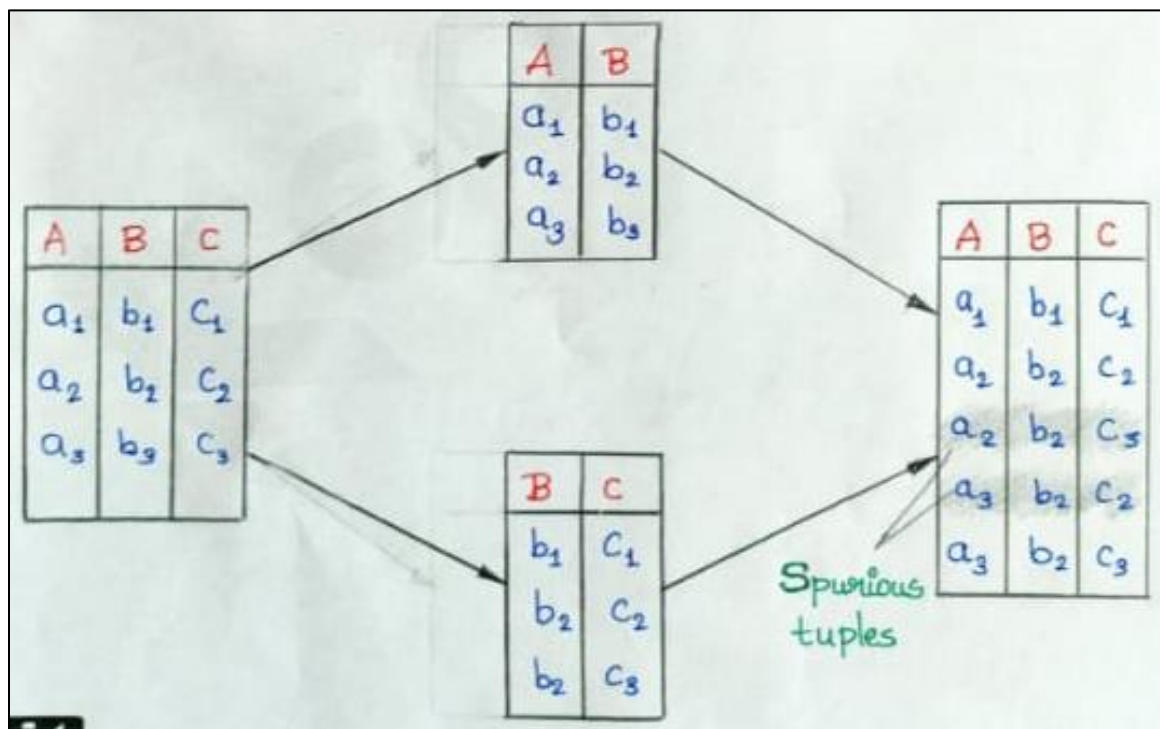
2. Anomalies: The problems that occur in table while performing insertion, updation or deletion operation.
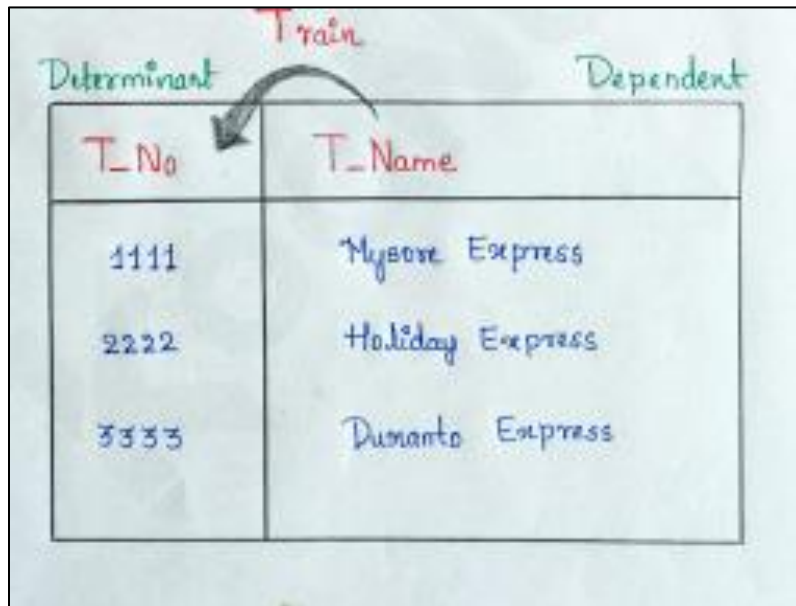
3. Data Redundancy: The repetition of same data within the table which leads to memory wastage is called data redundancy.

s_id	s_name	s_age	p_id	p_name
1	Ankit	22	11	Train Management System
2	Vikram	23	13	University Management System
3	Pooja	21	14	Hospital Management System
1	Ankit	22	13	University Management System

Dependencies

1. Functional Dependency

2. Partial Dependency

3. Transitive Dependency

1. Functional Dependency: If an attribute is dependent on another attribute then it is said to be functionally dependent. The attribute on which it is dependent is called as Determinant.

Patient

R_id	P_Name	P_Age
001	Ankit	29
001	Rohit	55
003	Uday	30

Determinant → P_id

Dependent → P_Age

2. Partial Dependency: If an attribute is functionally dependent on a part of candidate key then it is called as Partial Dependency.
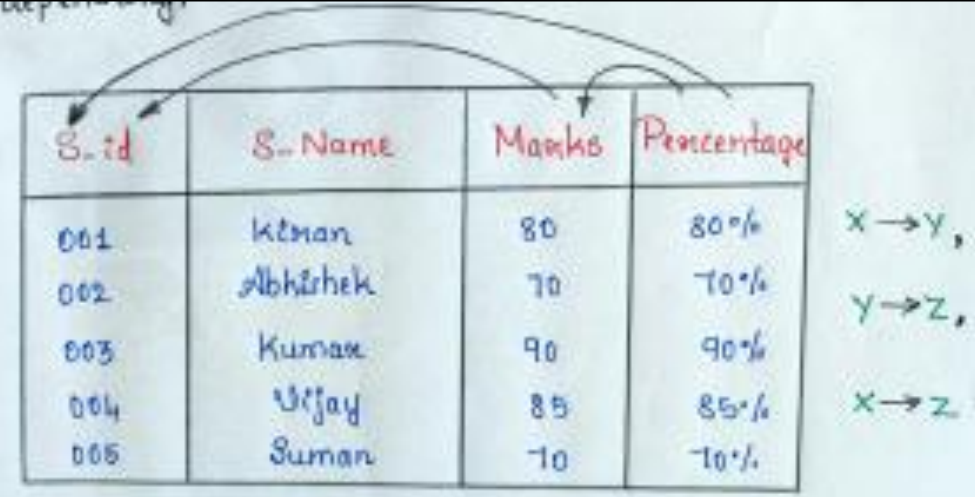
(or)

If in a table, more than 1 determinants are present, then it will lead to partial dependency.

S_no	P_no	S_Name	P_Name
001	P1	Varun	Train Booking
002	P2	Akshata	Bus Booking
003	P3	Rachana	Flight Booking
004	P2	Arun	Bus Booking
004	P3	Arun	Flight Booking
001	P2	Varun	Bus Booking
005	P4	Kiran	Taxi Booking

3. Transitive Dependency: When an indirect relation causes functional dependency then it is called as transitive dependency.

S_id	S_Name	Marks	Percentage
001	Kiran	80	80%
002	Abhishek	70	70%
003	Kumar	90	90%
004	Vijay	85	85%
005	Suman	70	70%

$X \rightarrow Y,$

$Y \rightarrow Z,$

$X \rightarrow Z$

Types of Normalization:

1. INF (First Normal Form)

2. 2NF(Second Normal Form)

3. 3NF (Third Normal Form)

4. BCNF (Boyce-Codd Normal Form)

Raw Data For Employee-Project Details:

PNo:	PName:	ENo:	EName:	Rating:	Hourly Charges
1001	Java Project	10,	Sowmya,	A,	100.
		11,	Anu,	B,	90,
		12,	Nidhi	C,	80,
		13	Darshan	D	70
1002	Python Project	14,	Rohit,	E,	60,
		10	Sowmya	A	100

Employee-Project

PNo	PName	ENo	EName	Rating	Hourly charges
1001	Java Project	10, 11, 12, 13	Sowmya, Anu, Nidhi, Darshan	A, B, C, D	100, 90, 80, 70
1002	Python Project	14, 10	Rohit, Sowmya	E, A	60, 100

1NF (First Normal Form);

A relation or table is said to be in 1NF when

- Each attribute should have separate columns.
- Every cell should be atomic.

Employee-Project

PNo	PName	ENo	EName	Rating	Hourly charges
1001	Java Project	10	Sowmya	A	100
1001	Java Project	11	Anu	B	90
1001	Java Project	12	Nidhi	C	80
1001	Java Project	13	Darshan	D	70
1002	Python Project	14	Rohit	E	60
1002	Python Project	10	Sowmya	A	100

2NF (Second Normal Form)

A relation or table is said to be in 2NF when

- The relation should be in 1NF.
- There should be no partial dependency.

Project

PNo	PName
1001	Java Project
1002	Python Project

Employee

ENo	EName	Rating	Hourly Charges
10	Sowmya	A	100
11	Anu	B	90
12	Nidhi	C	80
13	Darshan	D	70
14	Rohit	E	60

Project Employee

PNo	ENo
1001	10
1001	11
1001	12
1001	13
1002	14
1002	10

3NF (Third Normal Form):

A relation or table is said to be in 3Nf when

- The relation should be in 2NF.
- There should be no transitive dependency.

Project

PNo	PName
1001	Java Project
1002	Python Project

Employee

ENo	EName	Rating
10	Soumya	A
11	Anu	B
12	Nidhi	C
13	Darshan	D
14	Rohit	E

Rating

Rating	Hourly Charges
A	100
B	90
C	80
D	70
E	60

Project Employee

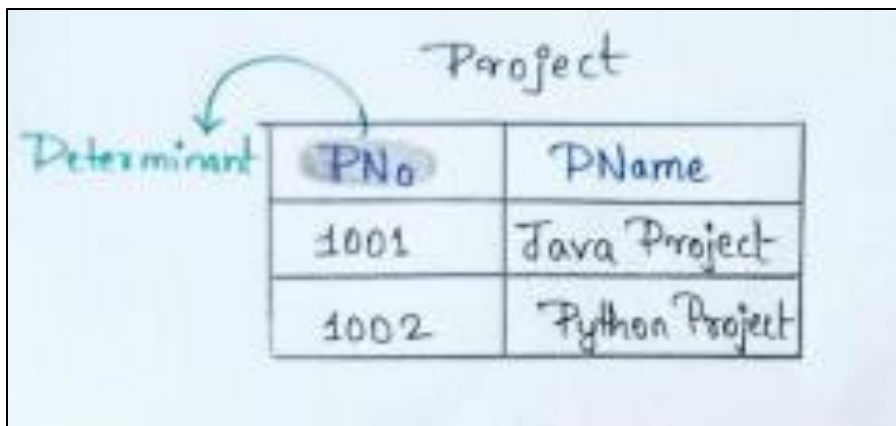| PNo	ENo
1001	10
1001	11
1001	12
1001	13
1002	14
1002	10

BCNF-(Boyce-Codd Normal Form)

A relation or table is said to be in BCNF when
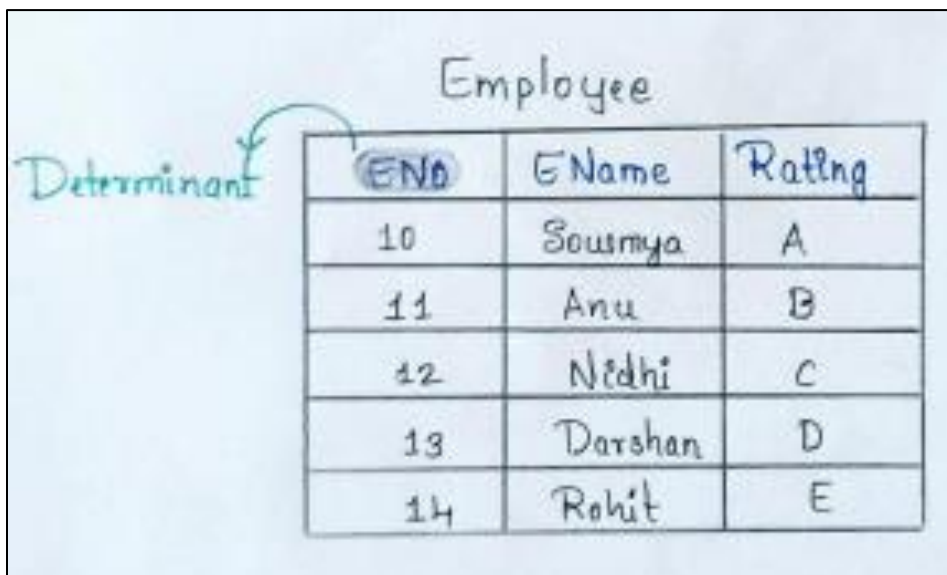
- A relation should be in 3NF.
- All determinants should be keys.

Project

Employee

Rating

Rating

Rating	Hourly Charges
A	100
B	90
C	80
D	70
E	60

Determinant → Rating

Project _Employee

Project_Employee

PNo	ENo
1001	10
1001	11
1001	12
1001	13
1002	14
1002	10

Determinant → PNo   ENo ← Determinant

Create Project table:

Project

PNo	PName
1001	Java Project
1002	Python Project

Create table Project

(

   PNo numbes primary Key,

   PName varchar2(80)

);

Creating Rating Table:-

Rating

Rating	Hourly Charges
A	100
B	90
c	80
D	70
E	60

Create table Rating

(

Rating varchar2(2) primary key,

Hourlycharges number

);

Creating Employee Table:-

Employee

ENo	EName	Rating
10	Sowmya	A
11	Anu	B
12	Nidhi	C
13	Darshan	D
14	Rohit	E

Create table Employee

(

ENo number primary key,

EName varchar2 (20),

Rating varchar2(2),

Foreign key (Rating) references

Rating (Rating)

);

Creating Project _Employee Table:-

Project_Employee

PNo	ENo
1001	10
1001	11
1001	12
1001	13
1002	14
1002	10

Create table Project_Employee

(

PNo number,

ENo number,

Foreign Key (PNo) references Project (PNo),

Foreign Key(ENo) references Employee (Eno),
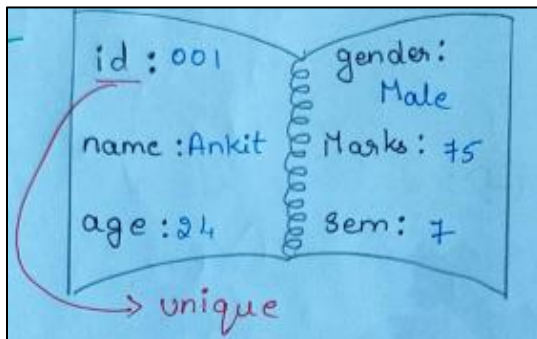
Primary Key (PNo, ENo)

);

CONSTRAINTS

Constraints are the rules or restrictions which are applied on the columns.

There are following constraints in SQL:

1. Unique

2. Not Null

3. Primary key

4. Foreign key

5. Check

6. Default

1. Unique :- It ensures that the column of a table will have unique values that is all values will be different.

Example :-

Create table Student(

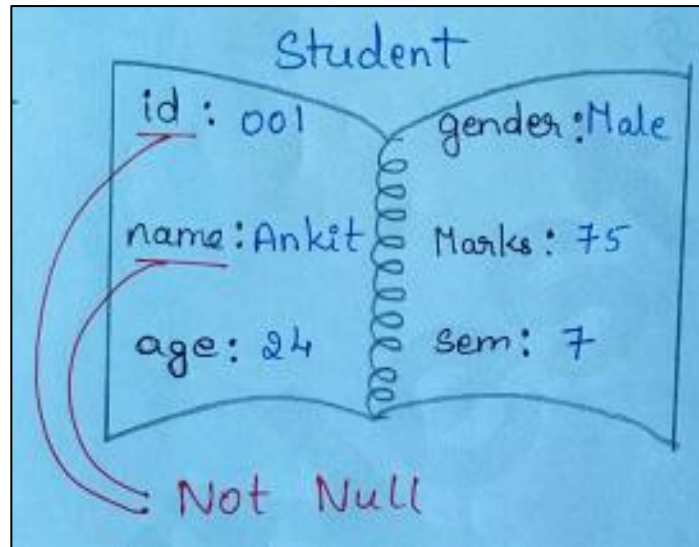id number Unique,

name varchar2 (10),

age number,

gender varchar2(6),

marks number,

Sem numbers);

2. Not Null :- It ensures that the column of a table will not have null values.

Example :-

Create table Student

(

Id number Not Null,

name varchar2(10) Not Null,

age number,

gender varchar2(6),

marks number,
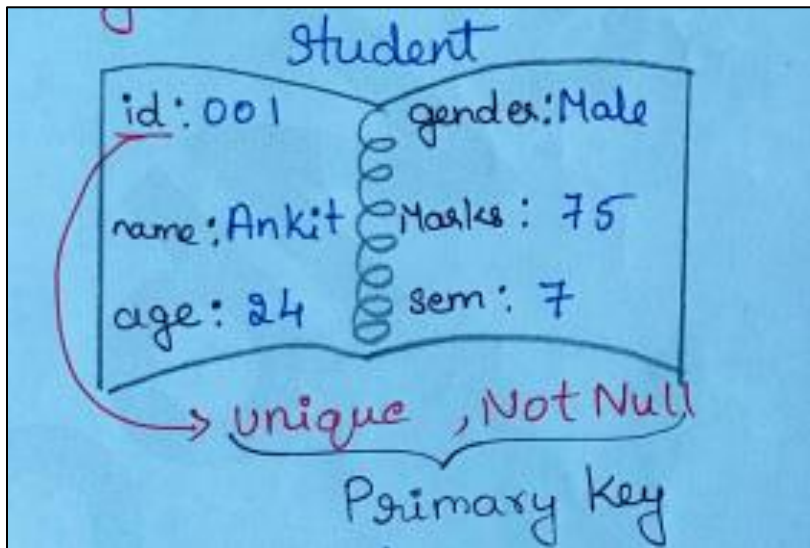
Sem number

);

3. Primary Key :- It is combination of unique and not null constraints.

- It ensures that column of a table cannot have duplicate values and also cannot have null values.
- A table can have only one primary key column.

Example :-

Create table Student

(

id number Primary key,

name varchar2 (10),

age number,

gender varchar2(6),

marks number,

sem number

);

4. Foreign key:- It ensures that data present in a column of a table refers to the data of primary Key column of parent table.

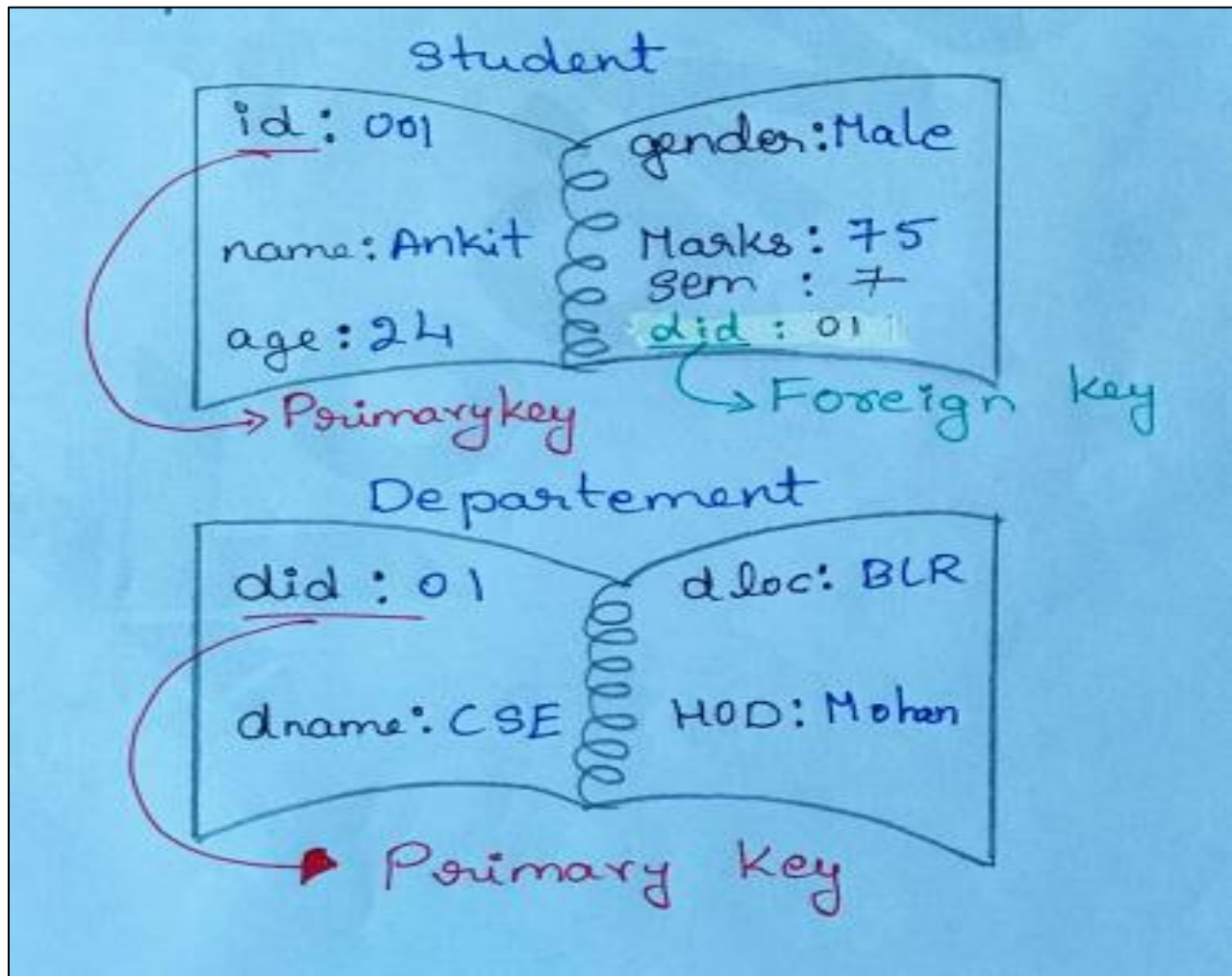- It is also called as referential integrity as constraint.

Example:-

```sql
Create table Student
(
id number primary key,
name varchar2 (10),
age number,
gender varchar2(6),
marks number,
sem number,
did number,
Foreign key(did) references Department(did)
);


Create table Department
(
did number primary key,
dname varchar2 (10),
dloc varchar2 (10),
HOD varchar2 (10)
);
```

5. Check-It ensures that the Column of a table will contain only such values which satisfy the specified check condition.

Example:-

Create table Student

(
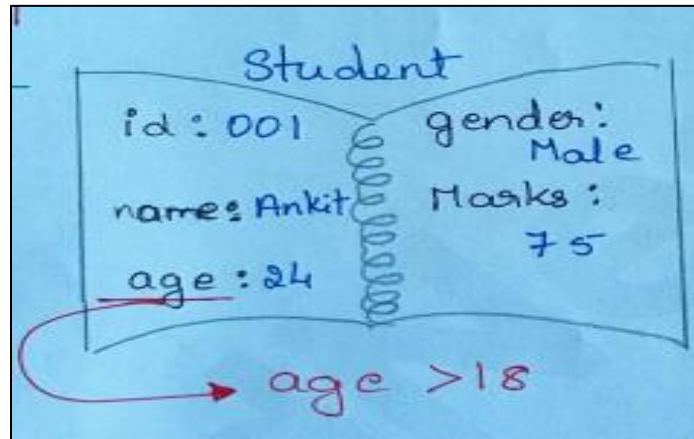
id number,

name varchar2(10),

age number Check (age > 18),

gender varchar2(6),

marks number

);

6. Default:- It ensures that Column of a table will have a specified default value if no value is given by the user.

Example:-

Create table Student
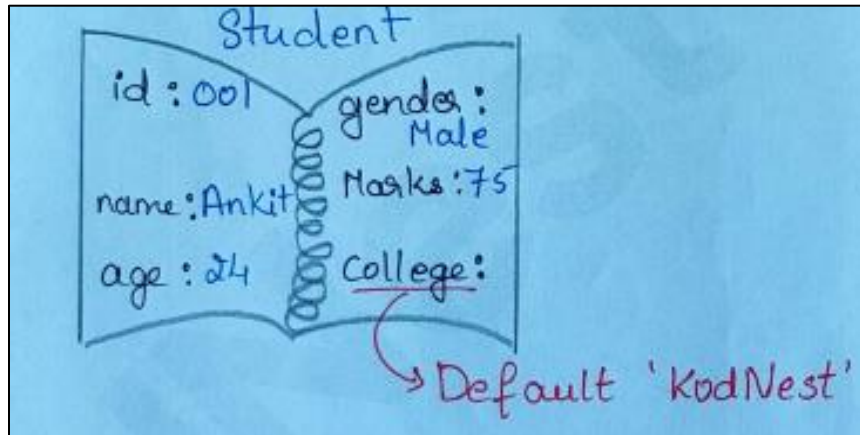
(

id number,

name varchar2 (10),

age number,

gender varchar2(6),

marks number,

College varchar(20) Default 'kodnest'

);

Important Interview Questions:

Practice to get perfect

Link to practice SQL: https://www.w3schools.com/sql/sql_wildcards.asp#gsc.tab=0

1. What is Database?
2. What is DBMS?
3. What are Different Types in DBMS?
4. What is RDBMS? How is it different from DBMS?
5. What is SQL?
6. What is the difference between SQL and MySQL?
7. What are Tables and Fields?
8. What are Constraints in SQL?
9. What is a Primary Key?
10. What is a UNIQUE constraint?
11. What is a Foreign Key?
12. What is a Join? List its different types.
13. What is a Self-Join?
14. What is a Cross-Join?
15. What is an Index? Explain its different types.
16. What is the difference between Clustered and Non-clusteredindex?
17. What is Data Integrity?
18. What is a Query?
19. What is a Subquery? What are its types?
20. What is the SELECT statement?
21. What are some common clauses used with SELECT query in SQL?
22. What are UNION, MINUS and INTERSECT commands?
23. What is Cursor? How to use a Cursor?
24. What are Entities and Relationships?
25. List the different types of relationships in SQL.
26. What is an Alias in SQL?

27. What is a View?

28. What is Normalization?

29. What is Denormalization?

30. What are the various forms of Normalization?

31. What are the TRUNCATE, DELETE and DROP statements?

32. What is the difference between DROP and TRUNCATE statements?

33. What is the difference between DELETE and TRUNCATE statements?

34. What are Aggregate and Scalar functions?

35. What is User-defined function? What are its various types?

36. What is OLTP?

37. What are the differences between OLTP and OLAP?

38. What is Collation? What are the different types of Collation Sensitivity?

39. What is a Stored Procedure?

40. What is a Recursive Stored Procedure?

41. How to create empty tables with the same structure as anothertable?

42. What is Pattern Matching in SQL?

1. What is Database?

A database is an organized collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modeling approaches.

2. What is DBMS?

DBMS stands for Database Management System. DBMS is a system software responsible for the creation, retrieval, updation, and management of the database. It ensures that our data is consistent, organized, and is easily accessible by serving as an interface between the database and its end-users or application software.

3. What are Different Types in DBMS?

Centralized Database

It is the type of database that stores data at a centralized database system. It comforts the users to access the stored data from different locations through several applications. These applications contain the authentication process to let users access data securely. An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.

Distributed Database

Unlike a centralized database system, in distributed systems, data is distributed among different database systems of an organization. These database systems are connected via communication links. Such links help the end-users to access the data easily. Examples of the Distributed database are Apache Cassandra, HBase, Ignite, etc.

Relational Database

This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation). A relational database uses SQL for storing, manipulating, as well as maintaining the data. E.F. Codd invented the database in 1970. Each table in the database carries a key that makes the data unique from others. Examples of Relational databases are MySQL, Microsoft SQL Server, Oracle, etc.

NoSQL Database

Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets. It is not a relational database as it stores data not only in tabular form but in several different ways. It came into existence when the demand for building modern applications increased. Thus, NoSQL presented a wide variety of database technologies in response to the demands. We can further divide a NoSQL database into the following four types:

Cloud Database

A type of database where data is stored in a virtual environment and executes over the cloud computing platform. It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database. There are numerous cloud platforms, but the best options are:

o Amazon Web Services(AWS)

o Microsoft Azure

o Kamatera

o PhonixNAP

o ScienceSoft

o Google Cloud SQL, etc.

Object-oriented Databases

The type of database that uses the object-based data model approach for storing data in the database system. The data is represented and stored as objects which are similar to the objects used in the object-oriented programming language.

Hierarchical Databases

It is the type of database that stores data in the form of parent-children relationship nodes.
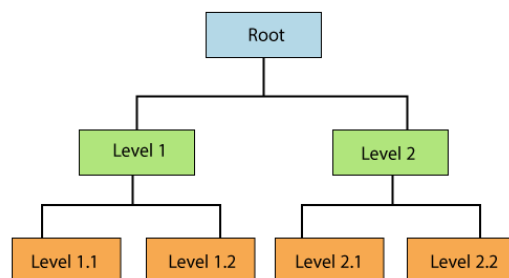
Hierarchical Database

It is the database that typically follows the network data model. Here, the representation of data is in the form of nodes connected via links between them. Unlike the hierarchical database, it allows each record to have multiple children and parent nodes to form a generalized graph structure.

Personal Database

Collecting and storing data on the user's system defines a Personal Database. This database is basically designed for a single user.

4. What is RDBMS? How is it different from DBMS?

RDBMS stands for Relational Database Management System. The key difference here, compared to DBMS, is that RDBMS stores data in the form of a collection of tables, and relations can be defined between the common fields of these tables. Most modern database management systems like MySQL, Microsoft SQL Server, Oracle, IBM DB2, and Amazon Redshift are based on RDBMS.

5. What is SQL?

SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data comprised of entities (variables) and relations between different entities of the data.

6. What is the difference between SQL and MySQL?

SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

7. What are Tables and Fields?

A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

8. What are Constraints in SQL?

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command. The constraints are:

- **NOT NULL** - Restricts NULL value from being inserted into a column.
- **CHECK** - Verifies that all values in a field satisfy a condition.
- **DEFAULT** - Automatically assigns a default value if no value has been specified for the field.
- **UNIQUE** - Ensures unique values to be inserted into the field.
- **INDEX** - Indexes a field providing faster retrieval of records.
- **PRIMARY KEY** - Uniquely identifies each record in a table.
- **FOREIGN KEY** - Ensures referential integrity for a record in another table.

9. What is a Primary Key?

The PRIMARY KEY constraint uniquely identifies each row in a table. It must contain UNIQUE values and has an implicit NOT NULL constraint.
A table in SQL is strictly restricted to have one and only one primary key, which is comprised of single or multiple fields (columns).

```
CREATE TABLE Students (    /* Create table with a single field as primary key
*/
    ID INT NOT NULL
    Name VARCHAR(255)
    PRIMARY KEY (ID)
);

CREATE TABLE Students (    /* Create table with multiple fields as primary key
*/
    ID INT NOT NULL
    LastName VARCHAR(255)
    FirstName VARCHAR(255) NOT NULL,
    CONSTRAINT PK_Student
    PRIMARY KEY (ID, FirstName)
);

ALTER TABLE Students    /* Set a column as primary key */
ADD PRIMARY KEY (ID);
ALTER TABLE Students    /* Set multiple columns as primary key */
ADD CONSTRAINT PK_Student   /*Naming a Primary Key*/
PRIMARY KEY (ID, FirstName);
```

10.   What is a UNIQUE constraint?

A UNIQUE constraint ensures that all values in a column are different. This provides uniqueness for the column(s) and helps identify each row uniquely. Unlike primary key, there can be multiple unique constraints defined per table. The code syntax for UNIQUE is quite similar to that of PRIMARY KEY and can be used interchangeably.

```
CREATE TABLE Students (    /* Create table with a single field as unique */
    ID INT NOT NULL UNIQUE
    Name VARCHAR(255)
);

CREATE TABLE Students (    /* Create table with multiple fields as unique */
    ID INT NOT NULL
    LastName VARCHAR(255)
    FirstName VARCHAR(255) NOT NULL
    CONSTRAINT PK_Student
    UNIQUE (ID, FirstName)
);

ALTER TABLE Students    /* Set a column as unique */
ADD UNIQUE (ID);
ALTER TABLE Students    /* Set multiple columns as unique */
ADD CONSTRAINT PK_Student   /* Naming a unique constraint */
UNIQUE (ID, FirstName);
```

11. What is a Foreign Key?

A FOREIGN KEY comprises of single or collection of fields in a table that essentially refers to the PRIMARY KEY in another table. Foreign key constraint ensures referential integrity in the relation between two tables.
The table with the foreign key constraint is labeled as the child table, and the table containing the candidate key is labeled as the referenced or parent table.

```
CREATE TABLE Students (          /* Create table with foreign key - Way 1 */
```

```
    ID INT NOT NULL
    Name VARCHAR(255)
    LibraryID INT
    PRIMARY KEY (ID)
    FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

CREATE TABLE Students (   /* Create table with foreign key - Way 2 */
    ID INT NOT NULL PRIMARY KEY
    Name VARCHAR(255)
    LibraryID INT FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

ALTER TABLE Students   /* Add a new foreign key */
ADD FOREIGN KEY (LibraryID)
REFERENCES Library (LibraryID);
```

12. What is a Join? List its different types.

The SQL Join clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.
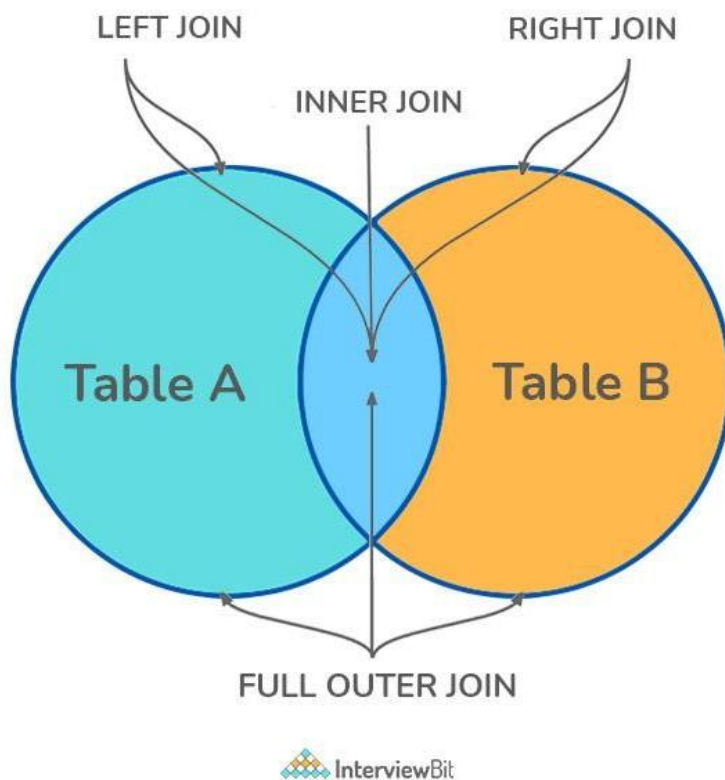
There are four different types of JOINs in SQL:

- **(INNER) JOIN:** Retrieves records that have matching values in both tables involved in the join. This is the widely used join for queries.

SELECT *

```
FROM Table_A
JOIN Table_B;
SELECT *
FROM Table_A
INNER JOIN Table_B;
```

- **LEFT (OUTER) JOIN:** Retrieves all the records/rows from the left and the matched records/rows from the right table.

```
SELECT *
FROM Table_A A
LEFT JOIN Table_B B
ON A.col = B.col;
```

- **RIGHT (OUTER) JOIN:** Retrieves all the records/rows from the right and the matched records/rows from the left table.

```
SELECT *
FROM Table_A A
RIGHT JOIN Table_B B
ON A.col = B.col;
```

- **FULL (OUTER) JOIN:** Retrieves all the records where there is a match in either the left or right table.

```
SELECT *
FROM Table_A A
FULL JOIN Table_B B
ON A.col = B.col;
```

13. What is a Self-Join?

A **self JOIN** is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

```
SELECT A.emp_id AS "Emp_ID",A.emp_name AS "Employee",
B.emp_id AS "Sup_ID",B.emp_name AS "Supervisor"
FROM employee A, employee B
WHERE A.emp_sup = B.emp_id;
```

14. What is a Cross-Join?

Cross join can be defined as a cartesian product of the two tables included in the join. The table after join contains the same number of rows as in the cross-product of the number of rows in the two tables. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

```
SELECT stu.name, sub.subject
FROM students AS stu
CROSS JOIN subjects AS sub;
```

SELF JOIN   CROSS JOIN

Table A

Table A   Table B
- A   - X
- B   - Y
- C   - Z

InterviewBit
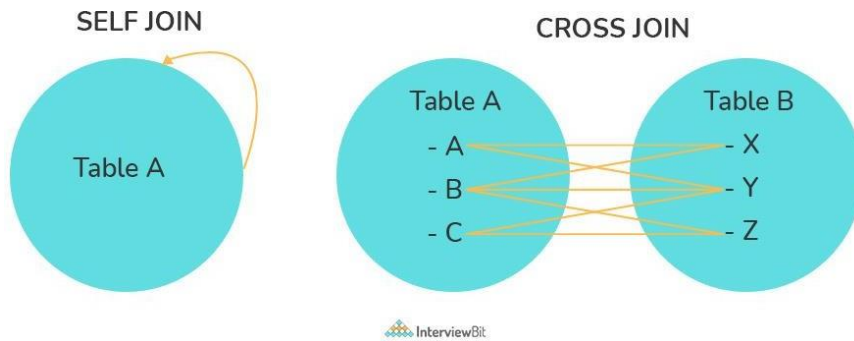
15. What is an Index? Explain its different types.

A database index is a data structure that provides a quick lookup of data in a column or columns of a table. It enhances the speed of operations accessing data from a database table at the cost of additional writes and memory to maintain the index data structure.

```
CREATE INDEX index_name    /* Create Index */
ON table_name (column_1, column_2);
DROP INDEX index name;    /* Drop Index */
```

There are different types of indexes that can be created for different purposes:

- **Unique and Non-Unique Index:**

  Unique indexes are indexes that help maintain data integrity by ensuring that no two rows of data in a table have identical key values. Once a unique index has been defined for a table, uniqueness is enforced whenever keys are added or changed within the index.

```
CREATE UNIQUE INDEX myIndex
ON students (enroll_no);
```

  Non-unique indexes, on the other hand, are not used to enforce constraints on the tables with which they are associated. Instead, non- unique indexes are used solely to improve query performance by maintaining a sorted order of data values that are used frequently.

- **Clustered and Non-Clustered Index:**

  Clustered indexes are indexes whose order of the rows in the database corresponds to the order of the rows in the index. This is why only one clustered index can exist in a given table, whereas, multiple non-clustered indexes can exist in the table.

  The only difference between clustered and non-clustered indexes is that the database manager attempts to keep the data in the database in the same order as the corresponding keys appear in the clustered index.

  Clustering indexes can improve the performance of most query operations because they provide a linear-access path to data stored in the database.

16. What is the difference between Clustered and Non-clusteredindex?

As explained above, the differences can be broken down into three small factors -

- Clustered index modifies the way records are stored in a database based on the indexed column. A non-clustered index creates a separate entity within the table which references the original table.

- Clustered index is used for easy and speedy retrieval of data from the database, whereas, fetching records from the non-clustered index is relatively slower.
- In SQL, a table can have a single clustered index whereas it can have multiple non-clustered indexes.

17. What is Data Integrity?

Data Integrity is the assurance of accuracy and consistency of data over its entire life-cycle and is a critical aspect of the design, implementation, and usage of any system which stores, processes, or retrieves data. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

18. What is a Query?

A query is a request for data or information from a database table or combination of tables. A database query can be either a select query or an action query.

```
SELECT fname, lname    /* select query */
FROM myDb.students
WHERE student_id = 1;
UPDATE myDB.students    /* action query */
SET fname = 'Captain', lname = 'America'
WHERE student_id = 1;
```

19. What is a Subquery? What are its types?

A subquery is a query within another query, also known as a **nested query** or **inner query**. It is used to restrict or enhance the data to be queried by the main query, thus restricting or enhancing the output of the main query respectively. For example, here we fetch the contact information for students who have enrolled for the maths subject:

```
SELECT name, email, mob, address
FROM myDb.contacts
WHERE roll_no IN (
 SELECT roll_no
 FROM myDb.students
 WHERE subject = 'Maths');
```

There are two types of subquery - **Correlated** and **Non-Correlated**.

- A **correlated** subquery cannot be considered as an independent query, but it can refer to the column in a table listed in the FROM of the main query.
- A **non-correlated** subquery can be considered as an independent query and the output of the subquery is substituted in the main query.

20. What is the SELECT statement?

SELECT operator in SQL is used to select data from a database. The data returned is stored in a result table, called the result-set.

SELECT * **FROM** myDB.students;

21. What are some common clauses used with SELECT query in SQL?

Some common SQL clauses used in conjuction with a SELECT query are as follows:

- **WHERE** clause in SQL is used to filter records that are necessary, based on specific conditions.
- **ORDER BY** clause in SQL is used to sort the records based on some field(s) in ascending (**ASC**) or descending order (**DESC**).

```
SELECT *
FROM myDB.students
WHERE graduation_year = 2019
ORDER BY studentID DESC;
```

- **GROUP BY** clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
- **HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

```
SELECT COUNT(studentId), country
FROM myDB.students
WHERE country != "INDIA"
GROUP BY country
HAVING COUNT(studentID) > 5;
```

22. What are UNION, MINUS and INTERSECT commands?

The **UNION** operator combines and returns the result-set retrieved by two or more SELECT statements.
The **MINUS** operator in SQL is used to remove duplicates from the result-set obtained by the second SELECT query from the result-set obtained by the first SELECT query and then return the filtered results from the first.
The **INTERSECT** clause in SQL combines the result-set fetched by the two SELECT statements where records from one match the other and then returns this intersection of result-sets.

Certain conditions need to be met before executing either of the above statements in SQL -

- Each SELECT statement within the clause must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement should necessarily have the same order

```
SELECT name FROM Students    /* Fetch the union of queries */
UNION
SELECT name FROM Contacts;
SELECT name FROM Students    /* Fetch the union of queries with duplicates*/
UNION ALL
SELECT name FROM Contacts;
SELECT name FROM Students    /* Fetch names from students */
MINUS      /* that aren't present in contacts */
SELECT name FROM Contacts;
SELECT name FROM Students    /* Fetch names from students */
INTERSECT    /* that are present in contacts as well */
SELECT name FROM Contacts;
```

23. What is Cursor? How to use a Cursor?

A database cursor is a control structure that allows for the traversal of records in a database. Cursors, in addition, facilitates processing after traversal, such as retrieval, addition, and deletion of database records. They can be viewed as a pointer to one row in a set of rows.

Working with SQL Cursor:

1. **DECLARE** a cursor after any variable declaration. The cursor declaration must always be associated with a SELECT Statement.

2. Open cursor to initialize the result set. The **OPEN** statement must be called before fetching rows from the result set.
3. **FETCH** statement to retrieve and move to the next row in the result set.
4. Call the **CLOSE** statement to deactivate the cursor.
5. Finally use the **DEALLOCATE** statement to delete the cursor definition and release the associated resources.

```sql
DECLARE @name VARCHAR(50)   /* Declare All Required Variables */
DECLARE db_cursor CURSOR FOR   /* Declare Cursor Name*/
SELECT name
FROM myDB.students
WHERE parent_name IN ('Sara', 'Ansh')
OPEN db_cursor   /* Open cursor and Fetch data into @name */
FETCH next
FROM db_cursor
INTO @name
CLOSE db_cursor   /* Close the cursor and deallocate the resources */
DEALLOCATE db_cursor
```

24. What are Entities and Relationships?

Entity: An entity can be a real-world object, either tangible or intangible, that can be easily identifiable. For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

Relationships: Relations or links between entities that have something to do with each other. For example - The employee's table in a company's database can be associated with the salary table in the same database.
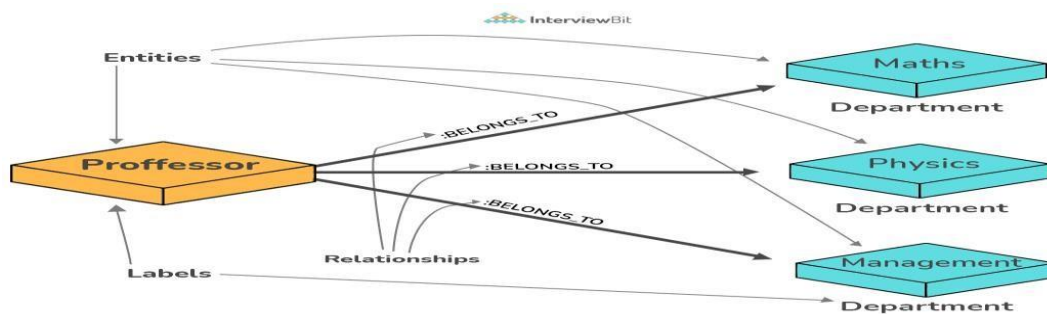
25. List the different types of relationships in SQL.

- **One-to-One** - This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
- **One-to-Many & Many-to-One** - This is the most commonly used relationship where a record in a table is associated with multiple records in the other table.
- **Many-to-Many** - This is used in cases when multiple instances on both sides are needed for defining a relationship.
- **Self-Referencing Relationships** - This is used when a table needs to define a relationship with itself.

26. What is an Alias in SQL?

An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name.

An alias is represented explicitly by the AS keyword but in some cases, the same can be performed without it as well. Nevertheless, using the AS keyword is always a good practice.

```
SELECT A.emp_name AS "Employee" /* Alias using AS keyword */
B.emp_name AS "Supervisor"
FROM employee A, employee B   /* Alias without AS keyword */
WHERE A.emp_sup = B.emp_id;
```

27. What is a View?

A view in SQL is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
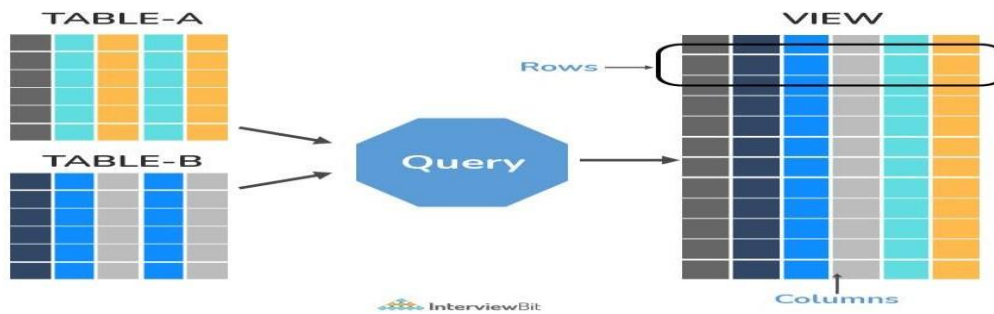
28. What is Normalization?

Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database.

29. What is Denormalization?

Denormalization is the inverse process of normalization, where the normalized schema is converted into a schema that has redundant information. The performance is improved by using redundancy and keeping the redundant data consistent. The reason for performing denormalization is the overheads produced in the query processor by an over-normalized structure.

30. What are the various forms of Normalization?

Normal Forms are used to eliminate or reduce redundancy in database tables. The different forms are as follows:

- **First Normal Form:**
  A relation is in first normal form if every attribute in that relation is a **single-valued attribute**. If a relation contains a composite or multi- valued attribute, it violates the first normal form. Let's consider the following **students** table. Each student in the table, has a name, his/her address, and the books they issued from the public library -

Students Table

Student	Address	Books Issued	Salutation
Sara	Amanora ParkTown 94	Until the Day I Die (Emily Carpenter), Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho), Inferno (DanBrown)	Mr.

Student	Address	Books Issued	Salutation
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward), Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street777	Dracula (Bram Stoker)	Mr.

As we can observe, the Books Issued field has more than one value per record, and to convert it into 1NF, this has to be resolved into separate individual records for each book issued. Check the following table in 1NF form -

Students Table (1st Normal Form)

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter)	Ms.
Sara	Amanora Park Town 94	Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho)	Mr.
Ansh	62nd Sector A-10	Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward)	Mrs.
Sara	24th Street Park Avenue	Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

- **Second Normal Form:**

A relation is in second normal form if it satisfies the conditions for the first normal form and does not contain any partial dependency. A relation in 2NF has **no partial dependency**, i.e., it has no non-prime attribute that depends on any proper subset of any candidate key of the table. Often, specifying a single column Primary Key is the solution to the problem. Examples -

Example 1 - Consider the above example. As we can observe, the Students Table in the 1NF form has a candidate key in the form of [Student, Address] that can uniquely identify all records in the table. The field Books Issued (non-prime attribute) depends partially on the Student field. Hence, the table is not in 2NF. To convert it into the 2nd Normal Form, we will partition the tables into two while specifying a new *Primary Key* attribute to identify the individual records in the Students table. The *Foreign Key* constraint will be set on the other table to ensure referential integrity.

Students Table (2nd Normal Form)

Student_ID	Student	Address	Salutation
1	Sara	Amanora Park Town 94	Ms.
2	Ansh	62nd Sector A-10	Mr.
3	Sara	24th Street Park Avenue	Mrs.
4	Ansh	Windsor Street 777	Mr.

Books Table (2nd Normal Form)

Student_ID	Book Issued

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Example 2 - Consider the following dependencies in relation to R(W,X,Y,Z)

```
WX -> Y     [W and X together determine Y]
XY -> Z     [X and Y together determine Z]
```

Here, WX is the only candidate key and there is no partial dependency, i.e., any proper subset of WX doesn't determine any non-prime attribute in the relation.

- **Third Normal Form**

A relation is said to be in the third normal form, if it satisfies the conditions for the second normal form and there is **no transitive dependency** between the non-prime attributes, i.e., all non-prime attributes are determined only by the candidate keys of the relation and not by any other non-prime attribute.

Example 1 - Consider the Students Table in the above example. As we can observe, the Students Table in the 2NF form has a single candidate key Student_ID (primary key) that can uniquely identify all records in the table. The field Salutation (non-prime attribute), however, depends on the Student Field rather than the candidate key. Hence, the table is not in 3NF. To convert it into the 3rd Normal Form, we will once again partition the tables into two while specifying a new *Foreign Key* constraint to identify the salutations for individual records in the Students table. The *Primary Key* constraint for the same will be set on the Salutations table to identify each record uniquely.

Students Table (3rd Normal Form)

Student_ID	Student	Address	Salutation_ID
1	Sara	Amanora Park Town 94	1
2	Ansh	62nd Sector A-10	2
3	Sara	24th Street Park Avenue	3
4	Ansh	Windsor Street 777	1

Books Table (3rd Normal Form)

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)

Student_ID	Book Issued
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Salutations Table (3rd Normal Form)

Salutation_ID	Salutation
1	Ms.
2	Mr.
3	Mrs.

Example 2 - Consider the following dependencies in relation to R(P,Q,R,S,T)

```
P -> QR      [P together determine C]
RS -> T      [B and C together determine D]
Q -> S
T -> P
```

For the above relation to exist in 3NF, all possible candidate keys in the above relation should be {P, RS, QR, T}.

- **Boyce-Codd Normal Form**

A relation is in Boyce-Codd Normal Form if satisfies the conditions for third normal form and for every functional dependency, Left-Hand- Side is super key. In other words, a relation in BCNF has non-trivial functional dependencies in form X –> Y, such that X is always a super key. For example - In the above example, Student_ID serves as the sole unique identifier for the Students Table and Salutation_ID for the Salutations Table, thus these tables exist in BCNF. The same cannot be said for the Books Table and there can be several books with common Book Names and the same Student_ID.

31. What are the TRUNCATE, DELETE and DROP statements?

DELETE statement is used to delete rows from a table.

```
DELETE FROM Candidates
WHERE CandidateId > 1000;
```

TRUNCATE command is used to delete all the rows from the table and free the space containing the table.

TRUNCATE TABLE Candidates;

DROP command is used to remove an object from the database. If you drop a table, all the rows in the table are deleted and the table structure is removed from the database.

DROP TABLE Candidates;

32. What is the difference between DROP and TRUNCATE statements?

If a table is dropped, all things associated with the tables are dropped as well. This includes - the relationships defined on the table with other tables, the integrity checks and constraints, access privileges and other grants that the table has. To create and use the table again in its original form, all these relations, checks, constraints, privileges and relationships need to be redefined. However, if a table is truncated, none of the above problems exist and the table retains its original structure.

33. What is the difference between DELETE and TRUNCATE statements?

The **TRUNCATE** command is used to delete all the rows from the table and free the space containing the table.
The **DELETE** command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.

34. What are Aggregate and Scalar functions?

An aggregate function performs operations on a collection of values to return a single scalar value. Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

- **AVG()** - Calculates the mean of a collection of values.
- **COUNT()** - Counts the total number of records in a specific table or view.
- **MIN()** - Calculates the minimum of a collection of values.
- **MAX()** - Calculates the maximum of a collection of values.
- **SUM()** - Calculates the sum of a collection of values.
- **FIRST()** - Fetches the first element in a collection of values.
- **LAST()** - Fetches the last element in a collection of values.

Note: All aggregate functions described above ignore NULL values except for the COUNT function.

A scalar function returns a single value based on the input value. Following are the widely used SQL scalar functions:

- **LEN()** - Calculates the total length of the given field (column).
- **UCASE()** - Converts a collection of string values to uppercase characters.
- **LCASE()** - Converts a collection of string values to lowercase characters.
- **MID()** - Extracts substrings from a collection of string values in a table.
- **CONCAT()** - Concatenates two or more strings.
- **RAND()** - Generates a random collection of numbers of a given length.
- **ROUND()** - Calculates the round-off integer value for a numeric field (or decimal point values).
- **NOW()** - Returns the current date & time.
- **FORMAT()** - Sets the format to display a collection of values.

35. What is User-defined function? What are its various types?

The user-defined functions in SQL are like functions in any other programming language that accept parameters, perform complex calculations, and return a value. They are written to use the logic repetitively whenever required. There are two types of SQL user-defined functions:

- Scalar Function: As explained earlier, user-defined scalar functions return a single scalar value.
- Table-Valued Functions: User-defined table-valued functions return a table as output.
- **Inline:** returns a table data type based on a single SELECT statement.
- **Multi-statement:** returns a tabular result-set but, unlike inline, multiple SELECT statements can be used inside the function body.

36. What is OLTP?

OLTP stands for Online Transaction Processing, is a class of software applications capable of supporting transaction-oriented programs. An essential attribute of an OLTP system is its ability to maintain concurrency. To avoid single points of failure, OLTP systems are often decentralized. These systems are usually designed for a large number of users who conduct short transactions. Database queries are usually simple, require sub-second response times, and return relatively few records. Here is an insight into the working of an OLTP system *[Note - The figure is not important for interviews] -*
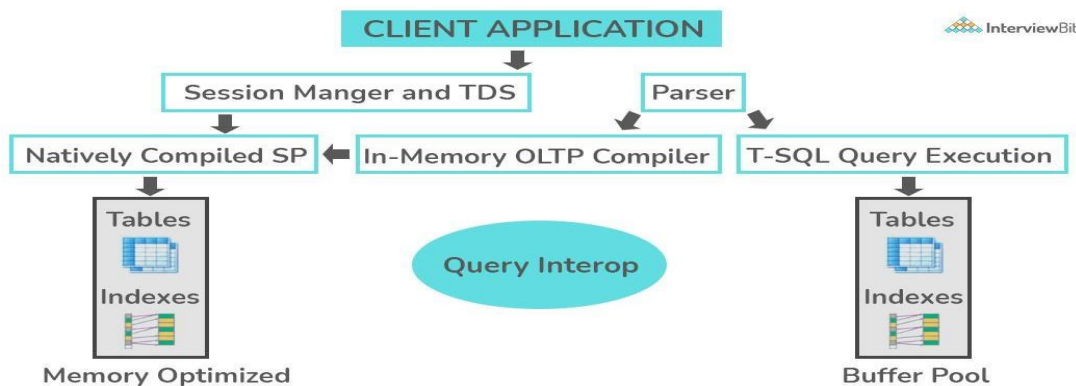
37. What are the differences between OLTP and OLAP?

OLTP stands for **Online Transaction Processing**, is a class of software applications capable of supporting transaction-oriented programs. An important attribute of an OLTP system is its ability to maintain concurrency. OLTP systems often follow a decentralized architecture to avoid single points of failure. These systems are generally designed for a large audience of end-users who conduct short transactions.
Queries involved in such databases are generally simple, need fast response times, and return relatively few records. A number of transactions per second acts as an effective measure for such systems.

OLAP stands for **Online Analytical Processing**, a class of software programs that are characterized by the relatively low frequency of online transactions. Queries are often too complex and involve a bunch of aggregations. For OLAP systems, the effectiveness measure relies highly on response time. Such systems are widely used for data mining or maintaining aggregated, historical data, usually in multi- dimensional schemas.

38. What is Collation? What are the different types of Collation Sensitivity?

Collation refers to a set of rules that determine how data is sorted and compared. Rules defining the correct character sequence are used to sort the character data. It incorporates options for specifying case sensitivity, accent marks, kana character types, and character width.
Below are the different types of collation sensitivity:

- **Case sensitivity: A** and **a** are treated differently.

- **Accent sensitivity: a** and **á** are treated differently.
- **Kana sensitivity:** Japanese kana characters Hiragana and Katakana are treated differently.
- **Width sensitivity:** Same character represented in single-byte (half-width) and double-byte (full-width) are treated differently.

39. What is a Stored Procedure?

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. The sole disadvantage of stored procedure is that it can be executed nowhere except in the database and occupies more memory in the database server. It also provides a sense of security and functionality as users who can't access the data directly can be granted access via stored procedures.

```
DELIMITER $$
CREATE PROCEDURE FetchAllStudents()
BEGIN
SELECT *  FROM myDB.students;
END $$
DELIMITER ;
```
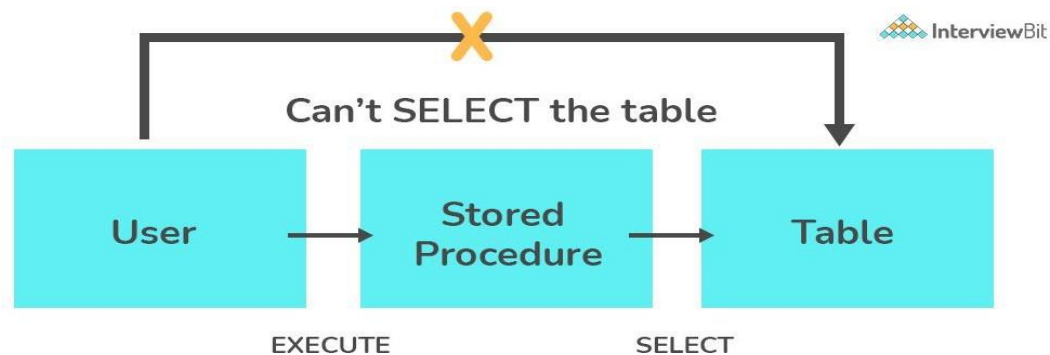

40. What is a Recursive Stored Procedure?

A stored procedure that calls itself until a boundary condition is reached, is called a recursive stored procedure. This recursive function helps the programmers to deploy the same set of code several times as and when required. Some SQL programming languages limit the recursion depth to prevent an infinite loop of procedure calls from causing a stack overflow, which slows down the system and may lead to system crashes.

```
DELIMITER $$      /* Set a new delimiter => $$ */
CREATE PROCEDURE calctotal( /* Create the procedure */
   IN number INT,   /* Set Input and Ouput variables */
   OUT total INT
) BEGIN
DECLARE score INT DEFAULT NULL;   /* Set the default value => "score" */
SELECT awards FROM achievements   /* Update "score" via SELECT query */
WHERE id = number INTO score;
IF score IS NULL THEN SET total = 0;   /* Termination condition */
ELSE
CALL calctotal(number+1);   /* Recursive call */
SET total = total + score;   /* Action after recursion */
END IF;
END $$      /* End of procedure */
DELIMITER ;      /* Reset the delimiter */
```

41. How to create empty tables with the same structure as anothertable?

Creating empty tables with the same structure can be done smartly by fetching the records of one table into a new table using the INTO operator while fixing a WHERE clause to be false for all records. Hence, SQL prepares the new table with a duplicate structure to accept the fetched records but since no records get fetched due to the WHERE clause in action, nothing is inserted into the new table.

```
SELECT * INTO Students_copy
FROM Students WHERE 1 = 2;
```

42. What is Pattern Matching in SQL?

SQL pattern matching provides for pattern search in data if you have no clue as to what that word should be. This kind of SQL query uses wildcards to match a string pattern, rather than writing the exact word. The LIKE operator is used in conjunction with **SQL Wildcards** to fetch the required information.

- **Using the % wildcard to perform a simple search**

  The % wildcard matches zero or more characters of any type and can be used to define wildcards both before and after the pattern. Search a student in your database with first name beginning with the letter K:

```
SELECT *
FROM students
WHERE first_name LIKE 'K%'
```

- **Omitting the patterns using the NOT keyword**

  Use the NOT keyword to select records that don't match the pattern. This query returns all students whose first name does not begin with K.

```
SELECT *
FROM students
WHERE first_name NOT LIKE 'K%'
```

- **Matching a pattern anywhere using the % wildcard twice**

  Search for a student in the database where he/she has a K in his/her first name.

```
SELECT *
FROM students
WHERE first_name LIKE '%Q%'
```

- **Using the _ wildcard to match pattern at a specific position**

  The _ wildcard matches exactly one character of any type. It can be used in conjunction with % wildcard. This query fetches all students with letter K at the third position in their first name.

```
SELECT *
FROM students
WHERE first_name LIKE '__K%'
```

- **Matching patterns for a specific length**

The _ wildcard plays an important role as a limitation when it matches exactly one character. It limits the length and position of the matched results. For example -

```sql
SELECT *    /* Matches first names with three or more letters */
FROM students
WHERE first_name LIKE '___%'

SELECT *    /* Matches first names with exactly four characters */
FROM students
WHERE first_name LIKE '____'
```