SQL [Sequel]

Data: Collection of small volume of information that can be recorded is called as Data.

Eg: Name, age, Ph.no - of single person.

Database: Collection of large volume of information that can be recorded and arranged in a proper manner is called as 'Database'.

Eg: Banking, Hospitals, college, etc.,

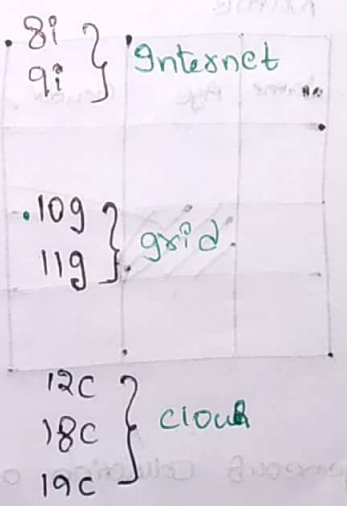
DBMS [Database Management System S/w]: It is used to store, manage, retrive large volume of data.

Management:- 1) Storing the data permanently.

2) Retrieve the data efficiently.

DBMS

- MySQL
- Oracle
- Sqlite
- SystemR
- Sybase



19c – latest

History of Oracle

1977 - Laxry Ellison [SDL] Software Development laboratory.

Oracle.1

1979 - [RSI] Relational Software Incorporation.

Oracle.2

1989 - Oracle System Incorporation.

1995 - Oracle corporation (official) //Present

RDBMS [Relational Database Management System]

* It is a DBMS software but according to RDBMS [Relational Database Management System] the data should be stored in table format only.

* It was developed by E.F codd.

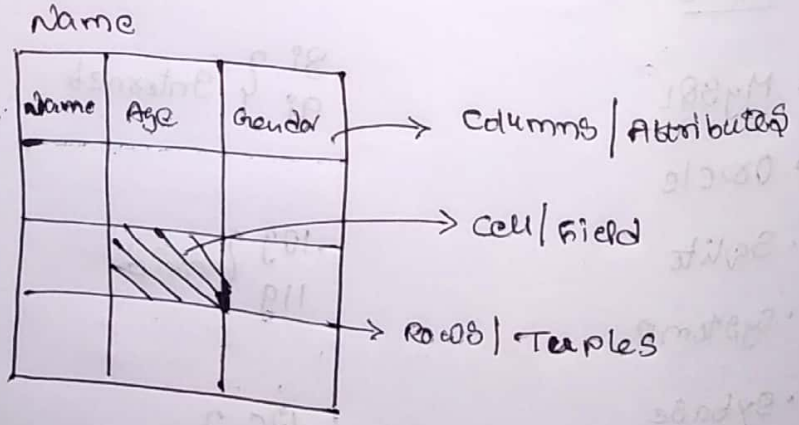


Table :- Homogeneous collection of rows, columns and cells.

Row :- Horizontal lines in a table is called as row or Tuple.

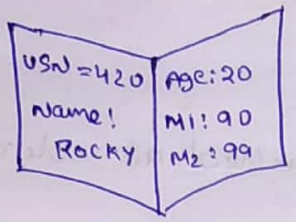
Column :- Vertical sections in a table is called as columns or Attributes.

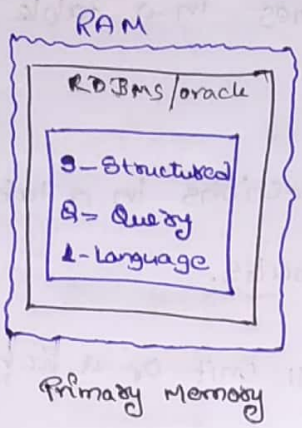
Cell :- It is a small unit of a table where the data is stored.

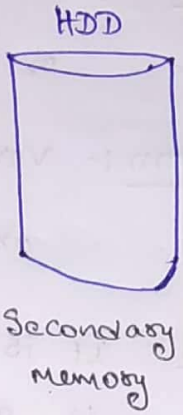
* Difference b/w DBMS and RDBMS.

DBMS	RDBMS
* It is an application which stores data in form of files.	* It is an application which stores data in form of tables only.
* DBMS is used to deal with small organisations which uses small volume of Database.	* It is used to deal with large organisation which uses large volume of database.
* DBMS doesn't use — normalization.	* Normalization is present in RDBMS.
Eg:- Microsoft word, Power point etc,	Eg:- Oracle, mysql, sqlite etc,

* Using RDBMS model we can create relationship b/w multiple tables using key Attribute.







* SQL is a language. It is used to interact with the Store data (Oracle).

* MySQL - is a Software.

Create Table :-

```
Create Table Student.
    USN int,
    Name char(5),
    Age int
    Gender char(1),
    M1 int,
    M2 int,
```

Student

USN	Name	Age	M1	M2

Insert values :-

```
Insert into Student values
(420,'Sunil',20,90,99);
Insert into Student values
(421,'Ravi' 20,90,99);
```

Student

USN	Name	Age	M1	M2
420	SUNIL	20	90	99
421	RAVI	900	90	99

Attributes | Columns

Tuples | Rows

Cell

My SQL :-

Insert into Student (USN, Name, Age, M1, M2) values
(4=0, Sunil, 20, 90, 99)

Eg:- Create a Table for Hospital Database with H.name, P.id, P.name, Bed.no, Word.no, disease, Amt.

Create Table. Hospital
(H.name char(10),
P.id . int,
P.name char(5),
Bed.no int,
Word.no int,
disease char(10),
Amt int);

insert into Hospital values.
(Apollo, 420, Kiran, 20, 21-21, sleeping, 1000),

Entity

Physical
Eg: Student
Employee

Conceptual
Eg: Subject
course

E-R- Diagram :-

* Diagramatic way of representing the data is called as E-R. diagram.

Entity :- Anything which exists in real world is called as entity. Entity devided into two types. They are.

1. Physical entity : Anything which exists physically in a real way is called physical entity. (Student, Car).
2. Conceptual entity : Anything which is in a conceptual manner (Subject).

Rectangle: Entity is always represented in rectangle box.

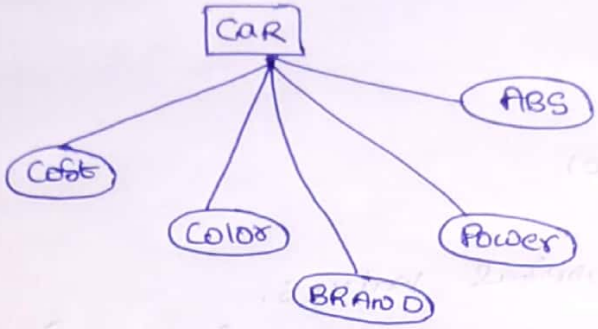
→ It is a table name.

Rectangle → Table name

Attributes: Properties that describes an entity. It is represented by ellipse.

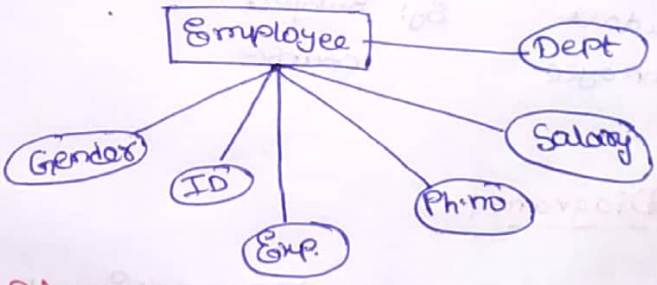
→ Column

Ellipse.



Create Table car

(color,
Speed
Power,
Milage,
ABS,
Cost);



Importance of E-R Diagram:

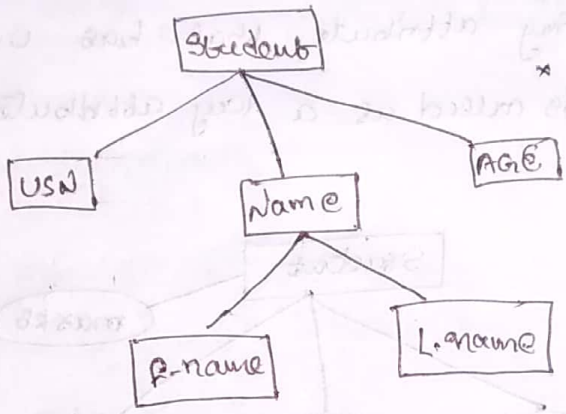
* It is important to plan out the changes carefully.
* By drawing ER diagrams to visualize database design, ideas you have a chance to identify the mistakes and design flaws.
* To make correction before executing the changes in databases [GRD is a tool that helps).

Types of attributes:-

1) Simple v/s Composite
2) Single v/s multivalued.
3) key attribute
4) Derived attribute

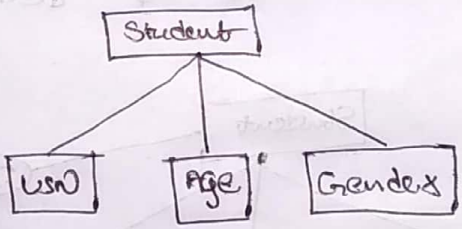
* Any attribute that can be divided further is called as Composite attribute.

Eg:

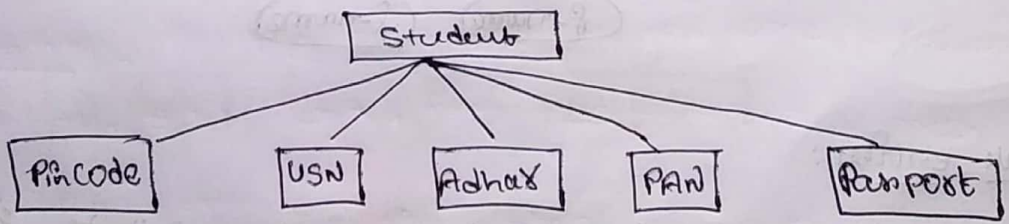


* Any attribute that can't be divided further is called as simple attribute.

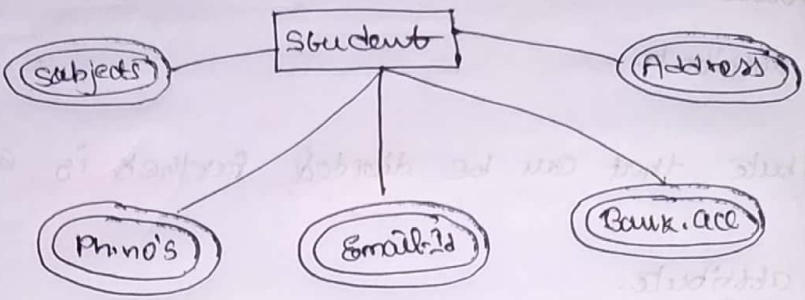
Eg:



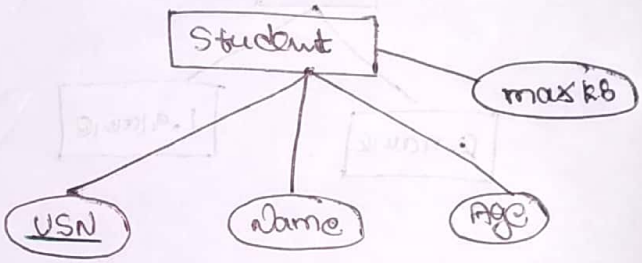
* Single attribute :- Any attribute that has single value.



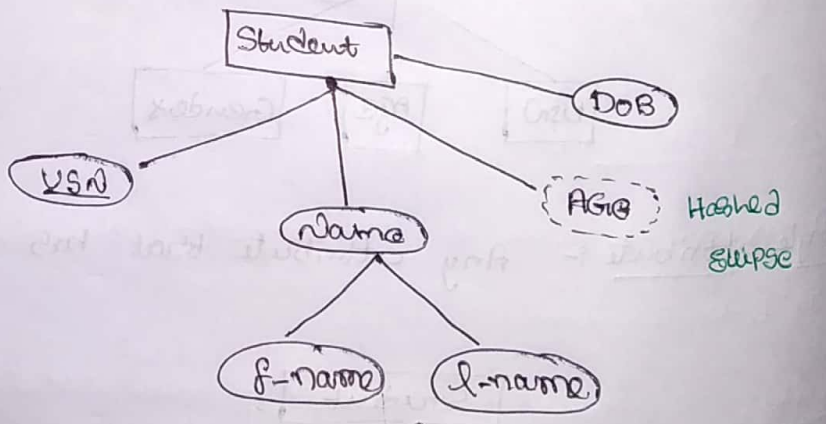
* Multivalue :- Any attribute which has more than one value is called as multivalue.



* key attribute :- Any attribute that has unique value is called as a key attribute.



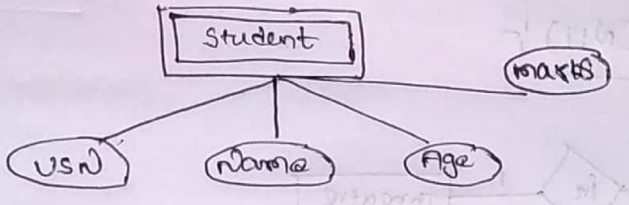
* Desived attribute :- Any attribute that is derived from another attribute is called as derived attribute.



Weak entity :

* Any entity doesn't have a key attribute is called as weak entity.

Eg:



Relationship :- Association between any two entities is called as relationship.

* It is represented through diamond shape symbol.

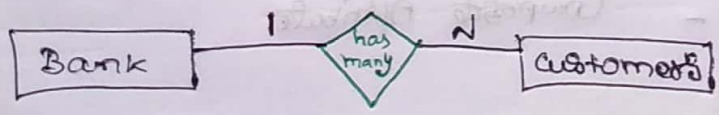
Eg:

Types of Relationships :-

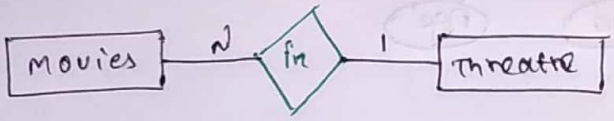
1) One to one (1:1)
2) One to many (1:N)
3) Many to one (N:1)
4) Many to Many (N:N)

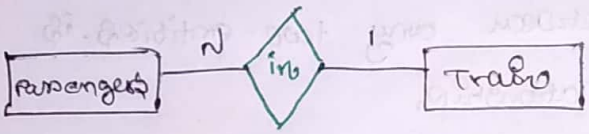
1) One to one (1:1) :-

2) One to many (1:N) :-

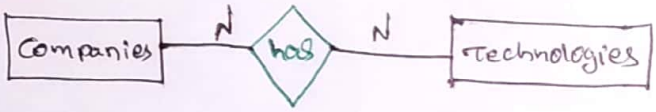


* 3) Many to one (N:1) :-

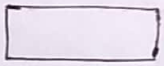


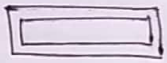


4) Many to Many (N:N) :-



* Representation of entity & Attributes :-

 – Entity

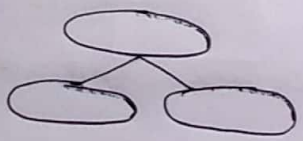
 – Weak Entity

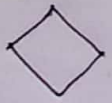
 – Attributes

 – Key Attribute

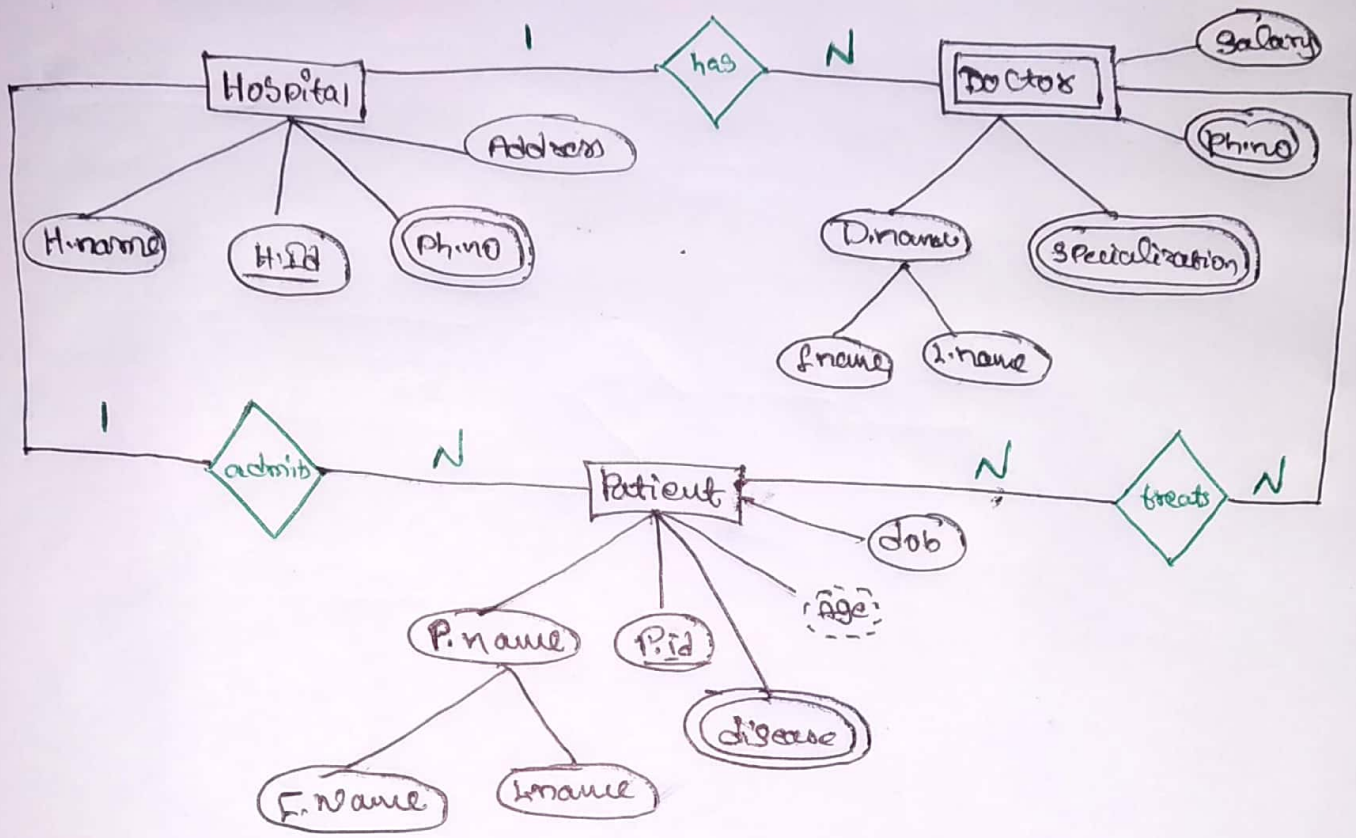
 – Derived Attribute

 – Multivalued

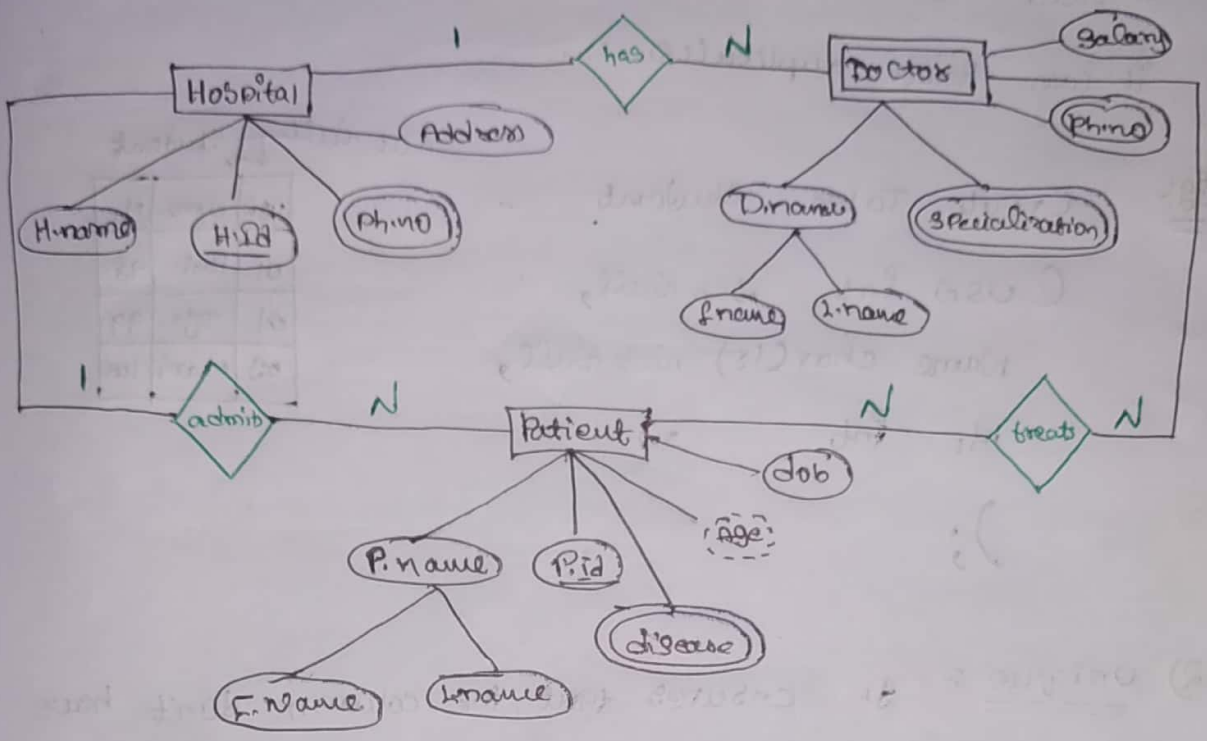
 – Composite Attribute

 – Relationship.

* Create a E-R model for Hospital DB using Hospital, Doctor, Patient as entities.



* Create a E·R model for Hospital DB using Hospital, Doctors, Patient as entities.



Constraints :- Rules / Restrictions / conditions implied (or) applied particular column of a table to standardise the database.

Note :-

* Null is a key word. It is a Invalid data. (or) Not at all Existing.
* Any arithmetic operations performed on a null value is 'null itself'.

Eg: $10 + Null = Null$ $\{+, -, /, *\}$

* In oracle 'one null value' ≠ 'another null value'.

Eg: $Null \neq Null$

1) Not Null Constraint :- It ensures that the column should not have null values. (Empty data) But it can have duplicates.

Eg:
```
Create Table Student
(USN int Not Null,
Name char(15) Not Null,
M1 int
);
```

Not Null → Student

USN	Name	M1
01	Hari	98
01	Teja	99
03	Mari	100

2) Unique :- It ensures that the column don't have duplicate values but it can have Null values.

Eg:
```
Create Table student
(USN int Unique,
Name char(15) Not Null,
M1 int
);
```

Unique → Student ← Not Null

USN	Name	M1
01	Hari	98
02	Teja	99
03	Mari	100

3) Check :- It ensures that the column values meets the specified check condition.

Eg:

```
Create Table Student
(USN int unique,
Name char(15) Not Null,
Age int check (Age>18)
);
```

check (Age >= 18)

Student

USN	Name	M1	Age	
01	Hari	99	18	✓
02	Teja	98	17	X
03	Mari	100	16	X

4) **Default** :- It ensures that the column has a default value when no value is given.

Eg:

```
Create Table Student
(USN int unique,
Name char(15) Not Null,
Age int check (Age>18),
City char(15) default 'AP'
);
```

default 'AP'

Student

USN	Name	Age	City
01	Hari	18	AP
02	Teja	17	UP
03	Mari	16	AP

5) **Primary key** :- * It is a constraint.

* It is combination of 'Not Null' and 'Unique' constraints.

unique + Not Null = Primary key

Student

USN	Name	M1	M2
01	Hari	90	95
02	Teja	95	90
03	Mari	98	99

* By having a primary key column we can uniquely identify each row in a table.

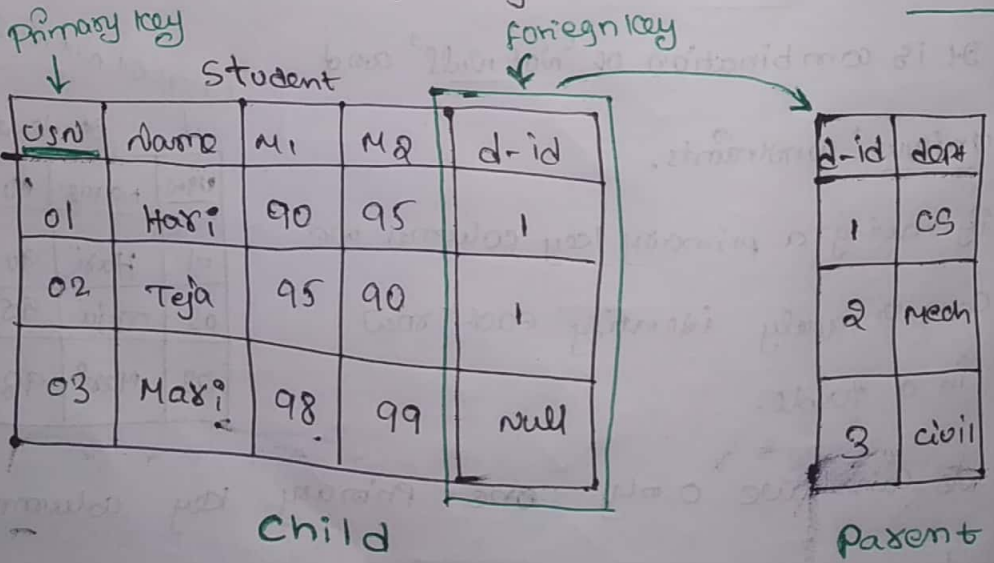
* We can have only one primary key column in a table which is called as **Parent table**.

Eg:

```
Create Table Student
(USN int Primary key,
Name char(15) Not Null,
M1 int,
M2 int
);
```

6) **Foreign key** :- Foriegn key is a constraint.

* Foriegn key is also called as 'referential integrity' constraint. By this we can create relationships b/w multiple tables using foriegn key columns.
* A table can have more than one foriegn key and all those foriegn keys should ~~not~~ and must refer a primary key column only.
* Foriegn key column can have duplicates and null values.
* A table which has a foriegn key is called as 'Child Table'.



```
Create Table department
(d-id int Primary key,
 d-name char(15) Not Null unique
);

Create Table Student
(USN int Primary key,
 Name char(15) Not Null,
 M1 int,
 M2 int,
 d-id int
 foreign key (d-id) References department (d-id)
);
```

Candidate key: Any column which is eligible to become a primary key.

Eg:- USN, PAN, Adhar.NO etc.,

```
Create table Student
(USN int Primary key,
 Adhar int Not Null unique,
 PAN int Not Null unique
);
```

Student

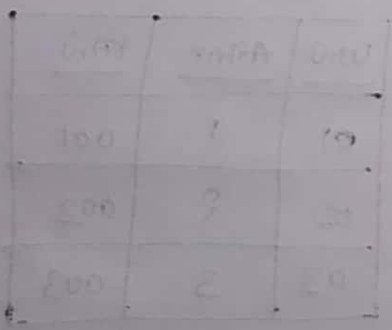
USN	Adhar	PAN
01	1	001
02	2	002
03	3	003

Super key:- Minimum set of attributes where we can uniquely identify each row in a table is called as super key.

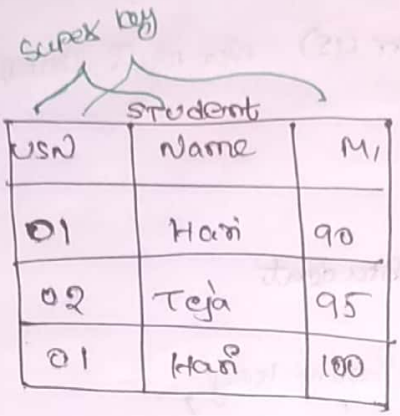
Super key

Student

USN	Name	M1
01	Hari	90
02	Teja	95
01	Hari	100



Super key:- Minimum set of attributes where we can uniquely identify each row in a table is called as super key.



Basic Queries:-

① Data types:

① **Integers:-** It will allocate only numbers either positive and negative values.

Eg:- ±99999

* It will not have decimal values.

② **Number:-** It can allocate numbers either positive & negative along with decimal values also.

Eg: Number(6)

+999999

* It can take upto 38

* Number(6,2)

P,S

P- Precision
S - Scale

+9999.99

* Number (*, 3)
↳ no. of length

if '*' is used,

③ Character :- [char]

It is static in nature. It can hold upto 2000 bytes.

It can have a-z
A-Z
* ^ $ ^ ---
~~1 2 3~~.

Memory is not utilised in char. datatype.

Eg:- Name char(5)

A n u
↳ not utilised

④ Varchar :-

* It is dynamic in nature. It can hold upto 4000 bytes.

* It can have a-z
A-Z
* ^ & $ ---
123 ---

* Memory is effeciently utilised.

Eg:- Name VarChar(5)

Hari
↳ memory is utilised.

⑤ Date :-

→ The standard format for date in oracle

ddmonyy / ddmonyyyy

Eg: 19 Aug 19

* dob Date

6) CLOB :- Character large object.

7) BLOB :- Binary large object.

Eg:-

```
Create Table Employees_data
(
Emapl_id    varchar(25) Primary key,
first_name  varchar(13) Not Null,
Last_name   varchar(13),
Email_id    varchar(50),
Salary      Number(*,2),
Gender      Char(1),
hire_date   date
);
```

Basic Query :-

① WAQ to display USN and Name from Student table.

② select USN, name
① From Student ;

* Student

USN	Name	Age
01	Hari	18
02	Teja	19
03	Sam	20

O/P :-

USN	Name
01	Hari
02	Teja
03	Sam

② WAQ to fetch USN and Name from Student table whose USN is '03'.

③ Select usn, name
① From Student.
② Where usn = 03;

o/p:-

USN	Name
03	Sam

Projection Query: Projection query is that without where condition.

Selection Query: Selection Query is that with 'where' condition.

③ WAQ to display all the details from employees-table

→ ② Select *
① from Employees;

→ ② select *
① from departments;

→ ② Select *
① From Job-grades;

(4) WAQ to display employee_Id, first_name, Last_name from employees table.

```
Select Employee_id, F_Name, L-Name
From Employees;
```

(5) WAQ to display F_name and L_Name, Salary whose F_name sachin.

```
Select F_name, Last name, Salary
From employees
where Last name = 'Sachin';
```

(6) WAQ to display F_name, L_name & Job_Id. whose Job_Id is ST_CLERK.

```
(3) select first name, last name, Jobid
(1) From Employees
(2) where Job_id = 'ST_CLERK';
```

Case-1 :-

Key words in SQL are not case-sensitive

Eg: SeLect FroM WheRe

Case-2 :- Columns in SQL are not case-sensitive.

firsT_name Last_Name

Case-3 :- Table names in SQL are not case-sensitive.

EmployeeS

Case-4:- value inside the cell is sensitive

* SQL IS NOT A SENSITIVE LANGUAGE *

ARITHMETIC OPERATORS :-

		( )	1
*		*	2
/		/	3
+		+	4
-		-	5

*1 WAQ to display Employee_Id, first_name, last_name and salary by incrementing the salary by 1000 Rs.

```
Select Employee_id, first_name, last_name, Salary + 1000
From Employees;
```

*2. WAQ to display first_name and last_name by decrementing the salary by Rs.1000.

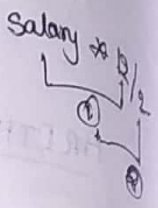
```
select first_name, last_name, Salary - 1000
From Employees;
```

*3. WAQ to display f_name, L_name and Annual_salary from employees table.

```
Select first_name, last_name, Salary*12
from Employees;
```

*4. WAQ to display first_name, last_name and half-yearly salary.

```
Select first_name, last_name, salary *6
from employees;
```



*5. WAQ to display first_name, last_name and salary by hiking the salary by 25 percent.

```
Select firstname, last_name, salary + (salary * 25/100)
from Employees;
```

100*25
2500/100

salary + 25 =

*6. WAQ to display first_name, last_name, salary by deducting 30% from the salary.

```
Select first_name, last_name, salary - (salary * 30/100)
from Employees;
```

*7. WAQ to display f_name, l_name and incrementing salary 15% to annual salary.

```
Select f_name, l_name, salary*12+(salary *12) * 15/100)
from Employee;
```

WAQ to display f_name, last_name, and double salary by employee salary.

```
Select f_name, l_name, salary * 3.
from employees;
```

Concatenation :- (||)

→ Concatenation Operator is used to combine or join multiple columns or multiple data in the given table.

→ It can accept only two arguments either of the sides.

*1. WAQ to Concatenate first_name and last_name from employees

```
Select first_name || last_name
From Employees;
```

*2. WAQ to concatenate first_name, last_name as fullname with a space b/w the full name from employees table.

```
Select first_name ||' '|| last_name as "full name"
from Employees;
```

(① ② ① ②)

*3. WAQ to display firstname, last_name as full name and Salary + Commission Percentage as total salary.

```
Select firstname || last_name as "full name", (salary + commission percent)
as "Total salary"
From Employees;
```

*4. WAQ to concatenate Bahubhali and Kattappa as Actors from ?

```
Select 'bahubali' || 'kattappa' as "Actors"
From Employees // 20 rows
     departments // 8 rows
     Dual // 1 row
```

Column Alias :

* It is used to give alternate name to the columns in SQL.
* Alias name won't be reflected in actual table of the Hard disk.
* It should always be enclosed in double Codes.

Syntax : col-name as "New column"

*1. WAQ to display f_name, L_name and half_yearly Salary by using Alias from employees table.

```
Select first_name, last_name, Salary * 12/2 as "half yearly Salary"
from Employees ;
```

*2 WAQ to display first name as f_name, last_name as L_name and Salary *12 as Annual Salary from employees table.

```
Select first_name as "f_name", last_name as "L_name", Salary *12
as "Annual salary"
From Employees ;
```

* **Dual Table :-** It is a special one row, one column table which is by default present in oracle database.
* The column name is called as dummy column. and the value is 'x'

```
Select * From dual ;
```

DUMMY
X

Varchar	Varchar2
* It can distinguish b/w Null and empty string.	* It can't distinguish b/w Null and empty string.

Description of a Table :-

Syntax :- desc Tab_name ;

Eg:- desc Employees ;

Relational Opexators :-

>
<
>=
<=
!=
<>
^=

} Not equal to

* WAQ to display first_name, last_name and Salary whose is earning more than 32,500.

```
Select first_name, last_name, salary
From Employees
Where Salary > 32500;
```

* WAQ to display first_name, Salary, hire_date who joined the Company on 17 Aug 2010.

```
Select first_name, Salary, hire_date
From Employees
Where hire_date = '17-Aug-2010';
```

* WAQ to display first_name, last_name and salary whose salary is more or equal to 17,000.

```
Select first_name, last_name, Salary
From Employees
Where Salary >= 17000;
```

* WAQ to display first_name, last_name and salary who is not earning 19,500 as salary.

```
Select first_name, last_name, salary
From Employees
Where Salary != 19500;
        <>
        ^=
```

* WAQ to display first_name, last_name and hire_date who when hired after the date 15Oct2008.

```
Select first_name, last_name, hire_date
From Employees
where hire_date > '15Oct2008';
```

* WAQ to display first_name, last_name and salary whose salary is lesses than 24000.

```
Select first_name, last_name, salary
From Employees
where salary < 24000;
```

* WAQ to display all the details of employee who is not earning the salary 17,500.

```
Select *
from Employee
where salary != 17500;
```

* Wap to concatenate the following

salary details
Sachin gets 24000 as salary
Vinod gets 17000 as salary
Ram gets 1900 as salary

```
Select first_name || ' ' || 'gets' || ' ' || salary || ' ' || 'as salary' as "salary
details"
From employees;
```

* Hi A is working in B department.

```
select 'Hi_' || first_name || '_is working in_' || dept_Id || '_department' as
"Message"
From Employees;
```

* Employee A was joined on B and is working in C department

A = f_name

B = hire_date

C = dept_Id

```
Select 'Employee_' || first_name || '_was joined on_' || hire_date
|| '_and is working in_' || dept_Id || '_department'
From Employees;
```

Distinct :- It is used to remove duplicate values from the given column. irrespective of any data type.

* Difference b/w Unique and Distinct.

Unique	Distinct
* Unique is a constraint	* Distinct is a key word.
* Unique is used while creating the table in a database.	* Distinct is used while selecting the values from the database.

* WAQ to display unique salary from Employees table

```
select distinct salary
from Employees;
```

* WAQ to display distinct salary and last_name.

```
Select distinct salary, last_name
From Employees;
```

Between and :-

* It is used to select values which lies on the given range.
* Always values should start from lower limit to upper limit.

* WAQ to display first_name, last_name and Salary whose salary is in the range 36,000 to 24,000.

```
Select first_name, lastname, Salary
from Employees
where Salary between 24000 and 36000;
```

* WAQ to display all the details of Employees who where hired after 2010 and before 2013.

```
Select *
from Employees
where hire_date between '01-Jan-2011' and '31-dec-2012';
```

* WAQ to fetch all the details of employees who where hired on any date excluding the 2011.

```
Select *
from Employees
where hire_date not between '01-Jan-2011' and '31-dec-11';
```

* WAQ to fetch employee_id, first name and Salary who is not getting the salary from 45000 to 84000

```
Select Employee_id, first_name, Salary
from Employees.
where Salary not between 45000 and 84000.
```

IN :-

Whenever comparison has to be made with a set of values we use IN as the key word.

* WAQ to display first name, last name and salary of all are getting the salary 24,000 (or) 18,500 (or) 32,500.

```
Select first name, last name, salary.
from employees
where salary IN (24000, 18500, 32500);
```

* WAQ to display all the details of employees whose every first name is Sachin, Hari, Samantha or Teja.

```
Select *
from Employees
where firstname in ('sachin', 'Hari', 'samantha', 'Teja');
```

* WAQ to display first-name and salary who are not getting the salary 24000 & 18400 or 32,500.

```
Select firstname, salary
from Employees
where salary not in (24000, 18400, 32500);
```

LIKE :- (wild cards, Pattern matching)

1) % (Modules) :- Match 0 or More characters.

2) _ (underScore) :- Match exactly one character.

Like :- It is used to match of patterns in a strings.

* WAQ to display first_name and last_name. whose last name starts with 'ku'.

last name
Kiran ✗
ku ✓
kunal ✓
kuku ✓

```
select first name, last_name
from Employees
where last_name like 'ku%' ;
```

* WAQ to display first name and last name whose last names last two character are 'ms'.

```
Select first name, last name
From Employees
where last name like '%ms' ;
```

* WAQ to display first_name, last_name from employees whose first name contains a sub string 'as'.

```
select first name, last name
from Employees
where first name like '%as%';
```

→ WAQ to display first name and last name whose first names, second and last character is 'a'.

```
Select first name, last name
From Employees
Where first name like '_a%a';
```

Mangsa
Radha

→ Write a query to display all the details of employee whose last names third character is 'n'.

```
select *
from Employees
where last name like '__n%';
```

Man_
Ma_nasa

→ WAQ to display all the details of the employees whose first name doesn't start with 's'

```
Select *
from Employees
where First name not like 'S%';
```

→ WAQ to display All the details of employees whose last names last third character is 'r'.

```
Select *
From Employees
where last name like '%r__';
```

→ WAQ to fetch all the details of employees which contains substring 'mad'.

```
Select *
From Employees
where last name like '%mad%';
```

→ WAQ to display 'e' whose salary is ending with

```
Selat *
from employees
where ~~last name~~ like '%500';
      Salary
```

* WAQ to display '*' whose first name has consecutive 'L'.

```
Select *
from Employees
where firstname like '%LL%';
```

* WAQ to display '*' whose last name contains exactly '5' characters in its length.

```
Select *
from Employees
where lastname like '_____';
```

* WAQ to display '*' whose first name starts with 'A' (or) 'S'.

```
Select *
from Employees
where firstname like 'A%' OR First.name like 'S%';
```

* WAQ to display all the details of the employees which starts with 'a' and atleast it should have '3' characters in it.

```
select *
from Employees
where first name like 'a_%_%';
```

a _ _ %

Is Null :-

It is used to select the values which has null values in the columns.

→ 'Is' keyword should be used only with null.

* WAQ to display commission_percentage whose commission_pct is null from employees table.

```
Select  Commission_pct
From    Employees
Where   Commission_pct  is  null ;
```

* WAQ to display commission pct who is having commission pct for the company.

```
Select Commission_pct
From   Employees
Where  Commission_pct  is not null ;
```

* WAQ to display '*' who are not having job_id and who is having commission pct also 25% of his salary should be lesser than 2000.

```
Select  *
From    Employees
Where   Job_id is null and Commission_pct is not null and
        salary * 25/100 < 2000 ;
```

Logical Operators :-

AND	1
OR	2
NOT	3

AND:- We use AND if both conditions are satisfying to fetch a row.

OR:- We use OR if any one of the conditions are satisfying to fetch a row.

NOT:- This an uninary operater which used to negate the value.

* WAQ to display all the details of employees whose job-id is Stationclerk and he must be working in dept.Id- 50.

```
Select *
From Employees
where Job-Id = 'STC' and dept Id = 50;
```

* WAQ to display first-name, last-name and salary whose salary is greater than 20,000 and their first name should start with 'H'.

```
Select first name, last name, Salary
From employees.
where Salary > 20000 and first name like 'S%';
```

* WAQ to display if an employee working as Station clerk or sales representative but must be earning salary more than 36,000.

```
Select first name *
From Employees.
where (Job id = 'STC' OR Job-Id = 'SA-Rep') and Salary > 36000;
```

* WAQ to display '*'. whose salary is in the range 10000 to 30000 and his last name's 3rd and last char should be 'a'.

```
Select *
From employees
where salary between 10000 and 30000 AND last name
like '__a%a';
```

→ WAQ to display all the details of employee whose first name should not start with 'M' and he should be working in dept Id. 40 or 10 or 50 or 112.

```
Select *
From employees
where firstname not like 'M%' AND dep-Id in (40,10,50,112);
```

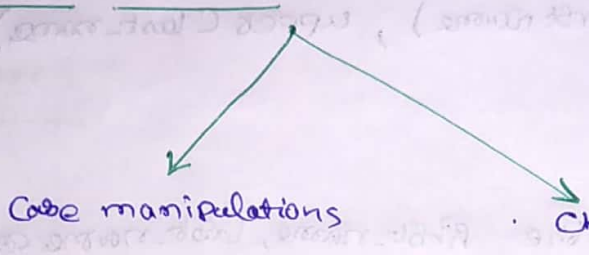
→ WAQ to display all the details of employees who is having job-id and not having commission pect also his name should end with 'as'. make sure that employee was hired during 2011.

```
Select *
from employees
where Job-Id isnotNull AND Commission pect is null
and first name like '%as' and
hire date between '1-Jan-2011' and '31-Dec-2011';
```

Functions

1) Single row functions :- Single row functions are such functions where for every input there is an output.

Character Functions



1) upper(); converts all I/p characters into upper case (A-Z)

2) lower(): converts all I/p characters into lower case (a-z)

3) Initcap(); converts all I/p characters into Initcap case (India)

* WAQ to convert the following into upper case, lower case and initcap case.

→ Hi sachin, how are you ??

* Select upper('hi sachin, how are you ??') as "message" from dual; // HI SACHIN, HOW ARE YOU??

* Select lower('HI SACHIN, HOW ARE YOU??') from dual; // hi sachin, how are you??

* Select Initcap ('hi sachin, how are you??') from dual; // Hi Sachin, How Are You ??

WAQ to convert all the first name and last name into upper case.

* Select upper (first_name, last name)
  from employees;
  } invalid no. of arguments.

* Select upper (first name), upper (last_name)
  from employees;

* WAQ to convert the first_name, last_name as full name into upper case.

  Select upper(first name || last name) as "full name"
  from employees;

* WAQ to convert first_name in upper case and last name in lower case and fetch only which has numbers or special characters in last name.

  select upper (first name), lower (last name)
  from Employees
  where upper(last name) = lower (last name);

  POOJA ✗ = pooja
  SRI ✗ = sri
  123 ✓ = 123
  %. # ✓ = %. #

* WAQ to display customer number first name in lowercase and last name in upper case for all customers whose customer number is in the range 80 to 150.

* Concat()

* WAQ to concatenate first_name and last name as full name using function.

Select concat (firstname, lastname) as 'full name'
① ②
from Employees ;

→ WAQ to concatenate first name middle name and last name as full name from employees tables.

Select Concat (Concat (firstname, mid name) , last name)
① ② ②
from employees;
①

→ WAQ to concatenate first name last name with a space as full name (use both operators & functions).

→ Select concat (concat (firstname, ' '), lastname) as "full name"
from Employees ;

→ select concat (firstname, ' ') || last name as "full name".
from employees;

→ select first name || concat(' ', last name)
from employees;

* Length() :-
It is used to count the number of characters in the given input.

* WAQ to count the length of the string 'Banglore is beautiful!'.

select length ('Banglore is Beautiful!') as "length"
from dual;

O/P = 23

* WAQ to count the length of the string Sachin.

select length ('Sachin')
from dual; O/P :- 6

* WAQ to find the length of all the first name from employees table.

select length (first name)
from employees;

* WAQ to obtain all the details of employees if their salary is more than 4 digits.

select *
from employees
where length (salary) > 4;

* WAQ to select all the details of employees whose first name has more than six characters.

```
Select *
from employees
where length (first name) > 6;
```

* WAQ to display all the details of employes whose salary is for exactly '4' digits.

```
Select *
From Employees
where length (salary) = 4;
```

100 3 X
2000 4 X
50000 5 ✓

Substring :- Substr()

→ It is used to fetch a part of a string.

→ Substring can accept two parameters or 3 parameters.

* WAQ to fetch a substring from the given string 'Tendulkar' from second position.

```
Select Substr ('Tendulkar', 2)
from dual;
```

-9 -8 -7 -6 -5 -4 -3 -2 -1
Tendulkar
1 2 3 4 5 6 7 8 9

Position

endulkar → o/p

Source string

* WAQ to fetch a substring from the string Tendulkar from second position extract '5' character.

```
Select Substr (Tendulkar, 2, 5)
from dual;
```

ss p length

o/p — endul.

* WAQ to display 'oha' from the string rajaram mohanroy

```
Select SubStr ('Rajaram mohanroy', 9, length ('oha'))
from dual;
```

(3 — position of 'o' in 'mohanroy'; 1234567 8 9 10 11 12 13 14 15)

imp

* WAQ to fetch '3' letter from first_name.

```
Select SubStr (first name, 3, 1)
From Employees;
```

* WAQ to fetch first character from the first name.

```
select SubStr (first_name, 1, 1)
from employees;
```

* WAQ to fetch last character from the first name.

```
Select SubStr (first_name, -1, 1)
from employees;
```

(-1 → length (first name))

* WAQ to fetch last third character from last name should be 'R'. (any case).

```
Select last name
from Employees
where SubStr (last name, -3, 1) in ('r', 'R');
```

* WAQ to fetch all the first name whose first name is starting from vowels.

```
select first name
from employees
where SubStr (first name, 1, 1) IN ('A', 'E', 'I', 'O', 'U');
```

* WAQ to fetch all first name whose first_name starts with consonants.

```
select first name
from employees
where SubStr (first name, 1, 1) not in ('A', 'E', 'I', 'O', 'U');
```

* WAQ to display half of the first name in upper case and rest in lower case.

~~Select (first name~~

~~From~~

```
Select upper (substr (firstname, 1, length (firstname)/2)) ||
                    TEND
       lower (substr (firstname, length (firstname)/2+1))
                    ulkas
from employees;
```

* WAQ to find the position of character 'A' from the string sachin.

```
Select instr ('sachin', 'a')        o/p — 2
from dual;
```

* WAQ to find the position of 'o' from first name.

```
Selece instr (first_name, 'o')
from employees;
```

* WAQ to find the position of 'A' from the string 'malayalam'.

```
Select instr ('malayalam', 'a')        o/p — 2
from dual;
```

* WAQ to find the third occurence of A from second position from the string malayalam.

```
select instr ('malayalam', 'a', 2, 3)
                2  4  6  8   chr p occurence
from dual;                    o/p — 6
```

* WAQ to find 3rd occurence of 'a' from first position of the string 'papaya'.

```
select instr ('papaya', 1, 3).
from dual;
```

O/P:- 6

* WAQ to find the substring 'dul' from the last name from employee table.

```
Select substr (last_name, instr (last_name, 'dul'),
                length ('dul'))
from employees;
```

→ Select substr ('abcdefghij', instr ('123232123', '2', 2, 4), length ('defg'))

from dual;

O/P:- hij

→ Select substr ('I like Bangalore', instr ('Bangalore', 'a', 1, 2), 2)

from dual;

O/P:- ke

* WAQ to fetch first-name before space from the column full-name from employee table.

```
select substr ( full_name, 1, instr ( full_name, ' ' ) -1 )
                SS                P
from employees;
```

7 - 1 = 6
6 - 1 = 5

Sachin Tendulkar
Vinod Kumar

→ WAQ to fetch last name after space from fullname column.

```
select substr (full_name, -1, instr (full_name, ' ' ) -1)
                                { instr (fullname, ' ' ) +1)
from employees;
```

Replace:

It is used to replace a substring with the given string to the column.

→ if 3rd parameter is not mentioned it will be replaced by nothing.

* WAQ to replace 'B' with M from the string Bangloxe.

```
Select Replace ('Bangalore', 'B', 'M')
from dual table;
```

Mangaloxe

* WAQ to replace 'J' with 'BL' from the string Jack and Jue.

```
Select Replace ('Jack and Jue', 'J', 'Bl')
       Black and Blue
from dual;
```

* WAQ to replace 'is' with '15' from the string

Mississippi

```
select Replace ('Mississippi', 'is', 15)
       M15s15sippi
```

* WAQ to count the numbers of from the string 'A,B,C,D,E'.

```
Select length('a,b,c,d,e') - length (Replace ('a,b,c,d,e', ',', ''))
       9                                  5
from dual;
       4
```

Translate() :-

'o'

* WAQ to replace 'i' with '1' & 's' with '5'. using encryption & decryption method.

```
Select Translate ('MISSISSIPI', 'IS', 15)
from dual;      M15515515IPI
```

Reverse :-

It is used to reverse a given string or a column.

→ WAQ to reverse 'Sony'

```
Select Reverse ('Sony')
From dual ;
```

→ WAQ to reverse first_name, last_name as full_name from employees table.

```
Select reverse ( first_name|| last_name) as "full name".
From employee ;
```

→ WAQ to find all the palindrome names from the given last name.

```
Select last_name
from Employees
where Reverse ( last_name ) = last name ;
```

refer = refer ✓

liril = liril ✓

AJET = TEJA ✗

→ WAQ to trim leading 'M' from the string 'Madam'.

```
Select trim ( 'leading 'M' from 'madam' )
From dual ;
```

→ WAQ to trim trailing 'M' from the string 'Madam'.

```
Select trim ( 'trailing 'M' from 'madam' )
From dual ;
```

WAQ to trim both the 'M' from the string 'Madam'.

```
Select trim ('M' from 'madam')
from dual;
```

M m M m M

Case sensitive

WAQ to trim all the leading 's' from first_name.

```
Select ltrim ('leading 's' from first_name)
From employees;
```

* WAQ to trim left hand side space from the string 'SQL is interesting!'.

```
Select ltrim ('   SQL is interesting!')
From dual;
```

* WAQ to trim RHS from the string 'SQL is interesting!!'.

```
Select Rtrim ('   SQL is interesting !   ')
fromdual;
```

* WAQ to trim both the space from the string 'SQL is interesting!'

```
select trim ('   SQL is interesting !   ')
From dual;
```

* WAQ to add numbers of characters to the given string.

```
select lpad ('Bangalore', 15, '*')
from dual;
```

9 → 6

******Bangalore

15

* Select Rpad ('Bangalore', 15, '*')
  from dual; Bangalore *******

→ Regular expression count

regexp_count

* WAQ to count no. of a in the string ("Bangalore is very very Beautiful").

Select regexp_count ('Bangalore is very very Beautiful', 'a')
from dual; (3)

Number functions :-

* WAQ to round of the number 45.326 upto second decimal.

Select Round (45.326, 2) o/p :- 45.33
from dual;

* Round off 1234.356 upto 1st decimal.

Select Round (1234.356, 1) o/p :- 1234.4
from dual;

* Round off 2678.345 upto -2 decimal place.

Select Round (2678.345, -2)
from dual;

Eliminate the decimal value

Back tracking in Reverse order

o/p :- 2700

* Round off 123 upto -2 decimal place.

Select Round (123, -2)
from dual; o/p :- 100

* WAQ to truncate the number 45.326 upto second decimal.

```
Select Trunc (45.326, 2)
from dual;        o/p:- 45.32
```

* Truncate 1234.356 upto 1st decimal.

```
Select Trunc (1234.356, 1)
from dual;        o/p:- 1234.3
```

* Truncate the number 2678.345 upto '-2' decimal place.

```
Select Trunc (2678.345, -2)
from dual;        o/p:- 2600
```

* Truncate 123 upto '-2' decimal place.

```
Select (123, -2)
from dual;        o/p:- 100
```

[54, -2]
-2 -1
0

* (54, -2)
-2 -1
100

* (45.44, -2)
-2 -1
0

} Round

→ WAQ to Round off the numbers upto highest integers. {ceil}

Select ceil (45.326)
from dual;

O/P :- 46

→ ceil (0.1) is 1.

→ ceil (-0.23) is 0.

→ ceil (5/2) is 3.

* WAQ to round off the numbers upto lowest integers. (Floor)

Select floor (45.326)
from dual;

O/P :- 45

→ floor (5/2) is 2.

→ floor (-2.5) is -3.

* WAQ to round off the column commission percentage upto first decimal.

0.5 – 0.5
0.75 – 0.8

Select round (commission_pct, 1)
from employees;

* WAQ to find the modulus of the numbers 120/2.

Select mod (120,2)
from dual;

O/P :- 0

* WAQ to find odd employee Id from employees table.

```
Select Employee Id oo *
from employees
where mod (employee-Id, 2) = 1 ;
```

* WAQ to fetch even employee Id from employees table.

```
Selece *
from employees
where mod (employee_Id, 2) = 0 ;
```

* write a query to find the power of 5^2 (square).

```
select power (5, 2)
from dual;
```

o/p :- 25

* WAQ to find the square root of 25. ($\sqrt{25}$):

```
Select sqrt (25)
from dual ;
```

o/p :- 5

* General functions :-

NVL ( ) null value

NVL2 ( ) null value 2

→ WAQ to display commission pct. if commission pct is having null values then replace it with 0.

```
Select NVL (commision_pct, 0)
from employees ;
```

* WAQ to display commission_pct if commission_pct is having null value replace with '0' else replace with '5'.

```
select NVL2 (commission_pct, 5, 0)
                              else if
from employees;
```

* WAQ to display employee_Id first name and salary + commission_pct as total salary from employees table.

```
Select employee_Id, firstname, (salary + NVL(commission_pct, 0) as
                                        "total salary"
From employees;
```

*

```
Select sysdate
From dual;
```

* Select systimestamp

```
From dual;
```

→ WAQ to display the no. of months b/w the dates 14-Nov-2018 to 14-Feb-2018.

```
Select months_between ('14-nov-2018', '14-feb-2018')
                        present          past
From dual;
```

o/p :- 9

→ WAQ to find age of a person

```
Select months_between ('sysdate', '22-Apr-1995') / 12
From dual;
```

o/p :- 24

→ WAQ in order to add 9 months to the date 14 feb.

```
Select add_months ('14-feb-2019', 9)
from dual;
```

O/P :- 14-NOV-19

→ WAQ to subtract 9 months from the date 14 NOV 2019.

```
Select add_months ('14-NOV-2019', -9)
from dual;
```

O/P :- 14-feb-19

→ WAQ to find the date of next Fri'day from the current date.

sunday → 8

```
select next_day (Sysdate, 'Friday')
                                5
from dual;
```

O/P :- 6-Sep-19

Monday - 1
Tuesday - 2
Wednesday - 3
Thursday - 4
Friday - 5
Saturday - 6
Sunday - 7

→ WAQ to find the last day from the current month.

```
Select last_day (Sysdate)
from dual;
```

O/P :- 31 Aug 2019

* WAQ to find current date, last day of the current month and new date of the next month. using alias.

```
select sysdate as "current date", last_day(sysdate) as
"last_day", last_day(sysdate)+1 as "new month".
from dual;
```

o/p :-

current Date	Last Day	New Month
30-AUG-19	31-AUG-19	01-Sep-19

* WAQ to find no of. days left until the end of the month.

```
select sysdate last_day(sysdate) - sysdate
from dual;
```

o/p :- 1

* WAQ to select the numbers of months an employee worked for the company.

```
select round(months_between(sysdate, hired date))
from Employees;
```

o/p :-

* Extract function

eg.

→ WAQ to display first name, last name and hire date from employees who was joined in the month July.

```
select first_name, last_name, hire_date
from employees
where extract(month from hire_date) = 7;
```

o/p: 07-july-20
01-Jul-2000

* WAQ to find all the details of employees who was hired in the year 2012.

```
Select *
from employees
where extract (year from hire_date) = 2012;
```

O/P :- 10-May-12
11-June-12
13-Dec-12

* WAQ to find all the details of employees who was hired on the date 10th day.

```
select *
from employees
where extract (day from hire_date) = 10;
```

From P

* WAQ to display employee-Id, first name, date of joining, month of joining, year of joining.

```
select employee_Id, first_name, extract (day from hire_date)
                as "day", extract (month from hire_date)
                as "month", extract (year from hire_date)
                as "year".
from employees;
```

Multirow Functions / Group / Aggregate :-

Multirow functions are such functions where multirow input or single row input there is only single row output.

1. Count()
2. Sum()
3. Avg()
4. min()
5. max()

1. **Count() :-** only one argument

It is used to count no. of rows in a column or a table. It can accept column or * keyword as argument.

2. **Sum() :-**

It is used to add the values in the column.

3. **Avg() :-**

It is used to get the average of a particular column.

i.e., $\frac{\text{Sum()}}{\text{Count()}} = \text{Avg()}$

4. **Min() :-**

It is used to get the lowest value in the column.

5. **Max() :-**

It is used to get the highest value in the column.

* WAQ to count the no. of employees in a table.

Select count (*) → no. of inputs giving 1 row as o/p

From employees ; // 20

* WAQ to find the total salary.

Select sum (salary)

From employees ; // 4,48000

* WAQ to find the avg of salary.

Select avg (salary)

From employees ; // 22400

$\frac{\text{sum()}}{\text{count()}} = \frac{4,48000}{20}$

$= 22400$

(or)

Select sum (salary) / count (salary)

* WAQ to find min and max of a salary.

Select min (salary), max (salary)

From employees ; // 8500 (↑ max) 5300 (↑ min)

* Min and max() can also gives characters → o/p will come according to alphabets.

Select Min (first_name), Max (first_name)

Brown Vinod

* WAQ to count f_name from employees table.

select count (f_name)

from employees ; // 20

WAQ to count no. of employees and total salary whose working in dept-Id 20.

```
Select count (*) , Sum (salary)
from employees
where dept_id = 20; // 2  39000
```

Count salary

WAQ to display half no. of employees working in a company

```
Select count (*)/2
from employees ;
```

o/p: 20/2 = 10

WAQ to find total salary, min salary, highest salary and avg salary who is having managers-id.

```
Select Sum(salary), min (salary), max (salary), avg (salary)
from employees.
where managers_id . IS NOT NULL;
```

(or)

Round (avg (salary), 2)

Output

235500
12000
53000
26166.67

WAQ to display first hired employee.

```
Select min (hired_date)
from employees; // 16-APR-07
```

WAQ to display recently hired employee.

```
Select max (hired_date)
from employees; // 27-Aug-12.
```

WAQ to display number of first name & total salary who is having 2 consecutive 'L' in their F_name.

```
Select Count (first_name), Sum (salary)
from employees
where first name like '%. LL %.' ;
```

Group By clause :-

It is used to group the identical rows in the column. group by clause is used after where clause. Group by clause can be used without where clause as well.

WAQ to display dept-id and total salary for all employ in each department.

```
select department_id, sum(salary)
from employees
group by department_id;
```

O/P :-

dept-id	Salary
90	60000
20	39000
110	39000
50	135000
80	95500
60	60000
10	19500

* WAQ to display lowest salary and dept-id for each department.

```
select department_id, min(salary)
from employees
group by department_id;
```

* Display dept-id, highest salary for each dept whose dept id is greater than 50.

```
select dept-id, max(salary)
from employees
where department_id > 50
group by department_id;
```

O/P

90	24000
110	19500
80	36000
60	26000

→ WAQ to count no. of employees working as station clerk in each dept.

```
select count(*)
from employees
where job-id = 'station clerk'
group by department-id;
```

* WAQ to find min and max wages given to employees in each dept.

```
Select min(salary), max(salary)
from employees
group by dept-id;
```

O/P

min	max	dept-id
17000	24000	90
19500	195000	20
⋮	⋮	⋮

Having clause :-

* It is used to filter the group function.
* Having clause is used to write a condition on a group function.
* Having clause executes after group by clause.

* Having WAQ to find dept-id and highest salary for all the employees whose max-salary is greater than 20,000.

* WAQ to display who is working as 'ST-CLERK' in each dept incase same dept no. of employees should be more than 2.

* WAQ to obtain highest salary for an employee only if the no. of employee working in the company is more than 10.

```
* Select dept-id, max(salary)
  from employees
  group by department-id
  having max(salary) > 20,000.
```

difference b/w where and Having.

where	Having
* Executes row by row.	* Executes group by group.
* It is used to check the condition before grouping on a normal column.	* It is used to check the condition after group by on a group function.
* cannot use multirow function in where clause.	* can use multirow function in having clause.

* Select dept_id
from employees
where Job_id = 'ST_CLERK'
group by dept_id
having count(*) > 2;

* select max(salary)
from employees
group by dept_id
having count(*) > 10;

Order By clause :-

It is used to arrange the result set in either ascending or descending order.

* It should always be used in the last statement of query.
* We can order the table using multiple columns.
* By default order by clause sorts the result in ascending order.

> WAQ to display f_name in ascending order.
> WAQ to display f_name in ascending & l_name in descending order.
> WAQ to display salary in descending order.

* select f_name
from employees
Order by f_name ASC;

* select salary
from employees
Order by salary ASC;

* select f_name, last_name
from employees
Order by f_name ASC, last_name desc;

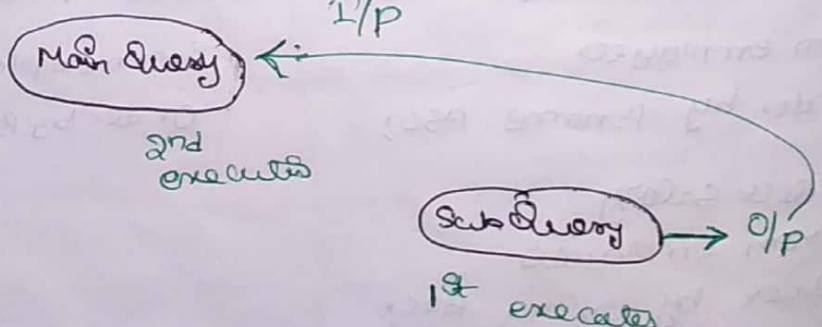
* Write a query to display dept-Id, highest salary for all employees whose dept Id is greater than '10' and max. salary greater than 20,000 for each dept. while displaying data display ~~result~~ dept-Id ~~it~~ in descending order wrt to dept-Id.

(5) Select dept-Id, max(Salary) // columns
(1) From Employees // Tables
(2) where department_id > 10 // condition on a NG
(3) group by department_id // group rows
(4) Having max(Salary) > 20,000 // condition on a Group by func
(6) Order By department_id Desc; // Arrange the Result

↳ Execution order.

* Select firstname, Salary * 12 as "Annual Salary"
From employees
where Salary * 12 > 20000
Order by "Annual Salary" ASC;

→ Sub-Query :- / Nested Query / Inner Query :-



→ Output of sub query is given as an input to the main query that is main query is always depended on sub query.

→ subquery is a query in another query which is embedded in where class.

→ Always sub-query should be enclosed within Paranthesis (Brackets).

→ 'Order by' class can't be inside a sub. query.

→ A sub query can have only column in the select clause unless multiples columns are in the main query.

→ If a subquery return more than one row we can use operators such as IN, NOT IN, All, ANY.

→ WAQ to display first_name, last_name and salary for all employees who are earning more than Tendulkar.

```
Select first_name, last_name, salary
From employees
where salary > ( select salary
                 from employees
                 where last_name = 'Tendulkar' );
```

= Any	Both (IN)
> Any	Min
< Any	Max

= All	
> All	max
< All	min

→ WAQ to display dept_id, firstname for all employees who work in the same department in which vinod works.

```
select department_id, first_name
from employees
where department_id ( select department_id
                      from employees
                      where first name = 'vinod');
```

→ WAQ to display dept_id, first name and job_id for all employees who work in administration department.

```
select dept_id, first_name, job_id.
from employees
where dept_id = (select department_id
                 from department
                 where dept_name = 'Administration');
```

→ WAQ to display first-name, last-name and Job-Id whose job-id is similar to king and whose salary is greater than singh salary

```
select first-name, last name, job-id
from employees
where salary > (select salary
                from employees
                where last name = 'singh')
and job-id = (select job-id
              from employees
              where last name = 'king');
```

(1) (2)

→ WAQ to display employee-Id. for all employees whose dept. Id in employees table = dept. Id in department table.

```
select employee id
from employees
where deptmart id in (select deptment id
                      from deptment);
```

→ WAQ to obtain all the details from the employees table whose salary is greater than 'Adam'.

```
select *
from employees
where salary > (select salary
                from employees
                where ename = 'adam');
```

→ WAQ to display all details of employees who were hired after Miller and salary greater than Smith salary

```
Select *
From employees
Where hire_date > (Select hire_date
                   From employees
                   Where ename = (miller) and salary > (Select salary
                                                        From employees
                                                        Where ename =
                                                        'Smith');
```

→ WAQ to display first name, last name and job_id who is having employee Id 7521.

```
select first_name, last_name, job_id
From employees
Where employee_id = &7521;
```

→ WAQ to display all the details of employees who earn second amounts of salary which is the smallest salary for any department.

```
select *
From employees
Where salary >Any (Select min(Salary)
                   From employes
                   group by department_id);
```

→ WAQ to display all the details of employees whose salary is within the range of smallest salary and 50,000.

```
select *
from employees
where salary between (select min(salary) and 50,000;
                      from employees)
```

JOINS :-

It is used to combine or compare multiple rows from two or more tables using a common column between them.

1) Cartesian Join / cross Join
2) Inner Join / Equi Join
3) Outer Join
   - Left outer Join
   - Right outer Join
   - Full outer Join
4) Natural Join
5) Self Join

1) **Cartesian Join / Cross Join :-** It is a cross product of two tables.

2. Works on the principle that each and every record of one table matches with all the records of another

table (including matching and non matching data).

3. Cross Join is always found without a where clause

Tab 1

A	B
1	2
3	4
5	6
7	8

m

m*n

Tab 2

B	C
2	1
8	1
9	1

n

4x3
12

Cross Join

```
Select * / col
from Tab1 CrossJoin Tab2;
```

Cartesian Join

```
select * / col
From Tab1, Tab2
where ..Condition... ;
```

A	B	B	C
1	2	2	1
1	2	8	1
1	2	9	1
3	4	2	1
3	4	8	1
3	4	9	1
5	6	2	1
5	6	8	1
5	6	9	1
7	8	2	1
7	8	8	1
7	8	9	1

Tab1

Physics		
2	3	4
Hindi		
1	5	8

Find Hindi's 5th book and kannada's 1st book?

Tab2

Chemistry		
2	6	1
kannada		
5	1	6

Tab1

Physics	Hindi
2	1
3	(5)
4	8

Tab2

chemistry	kannada
2	5
6	(1)
1	6

o/p:-

H	K
(5)	(1)

Select Hindi, kannada

from Tab1, Tab2

where Hind=5 and kannada = 1 ;

physics	Hindi	Chemistry	kannada
2	1	2	5
2	1	6	(1)
2	1	1	6
3	5	2	5
3	5	6	(1)
3	5	1	6
4	8	2	5
4	8	6	(1)
4	8	1	6

2. Equi Join / Inner Join :-

* An Equi Join is used to match two columns from two tables using explicit operator 'equal to'.
* if not 'equal to' operator that is greater than equal to. i.e., >=, <= becomes non Equi Join

Inner Join :

Returns all the rows when there is atleast one match in both the tables.

*

Tab1

A	B
1	2
3	4
5	6
7	8

Tab2

B	C
2	1
8	1
9	1

Equi Join :-

```
Select * / col
From Tab1, Tab2
Where Tab1.B = Tab2.B
```

A	B	B	C
1	2	2	1
7	8	8	1

Inner Join :-

```
select * / col
from Tab1 Inner Join Tab2
On Tab1.B = Tab2.B
where condition . . . ;
```

Tab 1

A	B

Tab2

B	C

Tab3

C	D

Equi Join :-

```
select A, Tab2.B, Tab3.c
From Tab1, Tab2, Tab3
where Tab1.B=Tab2.B and Tab2.C=Tab3.C;
```

inner Join :-

```
select A, Tab2.B, Tab3.c
from Tab1 Inner Join Tab2
On Tab1.B = Tab2.B inner Join Tab3 on Tab2.c =
Tab3.C;
```

→ WAQ to display employee_id & department name for all employees whose department_id in employees table = depart_id in departments table and department name should be administration.

* select Employee_id, d_name
From Employee E, Department D
where E.department_id = D.Department_Id
and department.name = 'Administration';

* Select employee_id, d_name
From employees E Inner Join Departments D.
On E.department_id = D.department_id
where departname = 'Administration';

3. Outer Join :-

It gives the non-matching along with matching records.

* Left outer Join :-
It returns all the left table but only matching records from right table

Tab1

A	B
1	2
3	4
5	6
7	8

left

Tab2

B	C
2	1
8	1
9	1

right

Tab

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

left

Select * /col
From Tab1 left Outer join Tab2
on Tab1.B = Tab2.B;

Right Outer Join: Returns all the rows from right table but only matching records from left table.

Tab1

A	B
1	2
3	4
5	6
7	8

left

Tab2

B	C
2	1
8	1
9	1

right

A	B	B	C
1	2	2	1
7	8	8	1
null	null	9	1
null	null		

right

```
Select */col
From Tab1 right outer join Tab2
On Tab1.B = Tab2.B;
```

Full Outer Join: It results in valid rows from all the tables and also invalid rows from left and right table. i.e; full outer join is combination of left outer join & right outer join.

Tab1

A	B
1	2
3	4
5	6
7	8

Tab2

B	C
2	1
8	1
9	1

A	B	B	C
1	2	2	1
3	4	null	null
5	6	null	null
7	8	8	1
null	null	9	1

```
Select */col
From Tab1. full outer join Tab2
On Tab1.B = Tab2.B;
```

4. Natural Join :-

It eliminates identical paired of columns in the given table. (Inner join).

Tabl

A	B	C
1	2	1
7	8	1

Natural join

```
select * / col
from Tab1 natural join Tab2;
```

A	B	B	C
1	2	2	1
7	8	8	1

} Inner Join

5. Self Join :-

Self Join is a such join where a table joins to itself.

→ At run time two copies of table will be created

Emp.a

Emp_Id	Name	Mgr_Id
1	Mohan	2
2	Das	2
3	Maathy	1

Emp.b

Emp_Id	Name	Mgr_Id
1	mohan	2
2	Das	2
3	Maathy	1

Emp.name	Mgr.name
Mohan	Das
Das	Das
Maathy	Mohan

select a.Name as "Emp-name", b.Name as "mgr-name"

① from Emp a, Emp b

② where a.mgr-Id = b.Emp_Id

⑨ Student (St_id, name, class)

Marks (St_id, Percentage, Elective-Id)

Elective (Elective-Id, Elective name)

Print Elective_name and average percentage

Select elective_name, Avg (Percentage)

From marks, Elective

table table2

where marks. Elective id = Elective. Elective_Id.

group by Elective_name;

⑩ Employees (Emp_id, Emp_name)

departments (dep_id, dep_name, Emp_id)

Salary (Emp.id, salary)

Print ~~Employee~~ name dep_name, Salary.

select Emp Name, dept_name, salary

From ~~departments, salary~~ Employees E Inner join

departments d

On E. Emp_id = d. Emp_id inner join Salary S

on d. Emp_id = S. emp_id;

```
Select Emp_name , dep_name, Salary
from Employees E, Departments D, Salary S
where E. Emp_id = D. Emp_id and
      d. Emp_id = S. Emp_id ;
```

(*) Employees (Emp_id , First_name , last_name , Salary , depart name, genders)

Project (Project_id , Emp_id , Proj_name)

1. Get employees first_name and project name from both the tables for those employees who has been assigned project already. Arrange first_name in ascending order.

```
Select first name , Proj_name
From Employees , Project
where Employees . Emp_id = Project. Emp_id
Order by Proj_name Asc ;
```

O/P

Mohan N/A
Raj 01A0 null
Ram N/A

2. get employees first name & project name order by first name from employees and project for all employees even if they have not assigned any project.

```
Select first name , Project_name
From Employees left outer Join Project
on Employees . Emp_id = Project . Emp. id
Order by first_name ;
```

3. get employees first_name and project name from both the tables even if they don't have any matching employee id. Arrange the first name in descending order.

```
Select first_name, Project_name
from employees right outer join project
on employees.Emp_id = project.emp_id
order by first_name desc;
```

4. get complete record from both the tables even if there is no match in any table.

```
select *
from employees full outer join project
On Employees.Emp_id = Project.Emp_id;
```

Write down each employees first name & project name who has been assigned more than one project.

City

CP-ID	From_city	To-city
BLR-HYD	BLR	HYD
NAG-PUN	NAG	PUN
HYD-BLR	HYD	BLR

Routes.

RS_ID	CP_ID	fare	Operater
1	Blr-Hyd	100	SRS
2	Blr-Hyd	110	VRL
3	Nag-Pun	220	KPN
4	Hyd-Blr	210	VRL

1) write a Query to find all operator name with lowest available fare for the given CP-ID Blr-Hyd.

```
Select Operator
from Routes
where fare = (select min(fare)
              from Routes
              where CP_ID = 'BLR-HYD');
```
100

O/P:- SRS

2) WAQ to find all from_city, to_city and operator name having route fare is greater than 110.

```
Select from_city, To_city, Operator
from city, Routes
where city CP_ID = Routes (CP_ID) and fare > 110;
```

3) What will be the O/P?

```
select Operator, Sum (fare) as "fare"
from Routes
where RS_ID > 1
group by operator
order by fare ASC;
```

Routes

operator	faxe
KPN	220
VRL	320

4) WAQ to find the 3rd letter from Routes CP_ID.

```
Select Substr ( CP_ID, 3, 1)
From Routes ;
```

* Commands :-

⇒ Data Control Language (DCL) :-

DCl is used to access the data stored in the database (Authentication).

1) Grant
2) Revoke

1) Grant :- It allows specified users to perform specified task.

2) Revoke :- To cancel the previously granted permissions

* WAQ to create user by the name ABC_Student and assign necessary privilages.

```
Create user ABC_STUDENT identified by ABC123;
Grant All priviliges to ABC_Student ;
```

2) WAQ to take back the permissions from the user ABC-Student.

* Revoke All Privilages to ABC-Student;

3) WAQ to assign dba previllage to ABC-Student.

* Grant DBA to 'ABC-STUDENT.';

rows

DML (Data Manipulation Language):- imp

→ It is used to modify the data present in the table.

→ DML actions are w.r.t rows of a table not w.r.t columns.

→ DML actions performed are not permanent actions.

1) Insert :- It is used to add rows to a table

Eg: Insert into Tab-name(values----);

Insert into tab-names (col1, col2,---) values (---);

2) Update :- imp

It is used to modify the data present in a particular coleumn of the table.

Eg: update Tab-name

Syntax

set column-name = value

where ------ ;

* update Student

set name = 'Mahi', Age = 28,

DoJ = '06-Sep-19'.

where USN = 3 ;

Student

USN	Name	Age	DoJ
1	Sam	23	01-Feb-19
2	Hari	24	25-May-19
3	Mahi	40	20-Apr-19
3	Mahi	28	06-Sep-19

* update Student

set Age = 23

where USN between 3 and 6 ;

*WAQ to set dept-name = tourism where dept-Id is 110.

update dept

set dept-name = 'tourism'

where dept-Id = 110;

* WAQ to set dept-name to sales executive and also set location Id to 999 where dept-Id is 202.

update Departments

set dept-name = 'sales executive', location Id = 999

where dept-Id = 202 ;

3) Delete & imp

It is used to remove the rows from a table.

1→ All the rows

2→ Only a particular row.

syntax

delete from Tab_name ; // All the rows from a Table will be deleted But Structure Remains same.

where dept-Id =20 condition ; // only a particular row will be deleted.

⇒ WAQ to delete rows from employees table where dept-id is 90 and last name is taylor.

delete from Employees

where dept-Id = 20 and last_name = 'taylor';

⇒ WAQ to delete rows from employees table whose employee id is 2,6,10 (or) his last name is kumar.

delete from employees

where employee id in (2,6,10) or last name = 'kumar';

⇒ what happens when no condition is specified in delete statement with an example.

* Structure remains the same, but all rows will be deleted.

Delete from employees ;

TCL (Transaction Control Language)

→ It is used to control the transaction processing in a database.

→ As DML commands are not permanent, it can be made permanent using TCL commands

1) Commit :- It is used to save the DML actions performed on the table.

2) Rollback :- to UNDO all the DML actions performed on the database.

3) Savepoint :- Save point is used to go to a particular DML command.

```
Savepoint X ;
Rollback to X ;
```

DDL (Data Definition Language) :-

→ It is used to modify the data present in the table. i.e., creating a table, Deleting a table or renaming a table etc.,

→ DDL commands are always w.r.t columns of a table.

1) Create :- It is used to create a table of users.

* WAQ to create a table by copying the structure of existing table.

Create Table copy_Employees
As
Select * Employees;

* WAQ to copy the structure of the existing table.

Select * From Employees.
where 1=100; only structure of a Table will be copied.

2) Alter :- It is used to alter the data in the database. i.e., by adding a column, renaming a column or deleting a column.

• Adding :-

imp

Alter Table Tab-name
Add Col-name datatype constraints;

Eg:

Alter Table Student
Add (phone_no number(10) unique,
email_Id varchar(35) unique not null);

Rename :-

Alter Table Tab.name

Rename column old column to new_column ;

⇒ Alter Table Student

Rename column phone_no to Mobile_no ;

Drop :- (Delete) :-

Alter Table Tab.name

Drop Column Col-name ;

⇒ Alter Table Student

Drop column mobile_no ;

3) Rename :- It is used to rename a existing table.

Rename old_Table to New_Table ;

⇒ Rename Student to Power_rangers ;

4) Truncate :- → It is used to delete all the rows from a table in one shot. But structure remains same.

→ can't be rolled back.

→ Uses less memory space.

⇒ Truncate Table Students ;

Truncate Table Tab-name ;

Difference blw Delete & Truncate :-

Delete	Truncate
* It is DML command	* It is DDL command
* All rows will be deleted but structure remains same	* All rows will be deleted but structure remains same
* uses more memory	* uses less memory
* Rollback can be performed.	* Rollback can't be performed.
* Slower	* faster

5) Drop :- It is used to delete all the rows including the structure of the table. i.e., whole table will be delete.

→ Can't perform rollback.

Drop Table Tab-name ;

⇒ Drop Table Employees ;

* Given the city & country table Query the sum of Population for all cities where continent is Asia.

City (ID, name, countrycode, District., population)

Country (code, Name, continent, region, Population, Capital)

Ans Select sum (city.population)
from city, country
where city.countrycode = Country.code and Continent = 'Asia';

* Given the city & country table Query the names of all the cities where continent is 'Africa'.

Select cityname
from city
~~where city.continent = 'Africa';~~
where city.country code = country.code and continent is 'Africa';

* Query the names of all the continents and their respective average city population rounded down to nearest integer.

Select continent, floor(Avg(city.Population))
from city, country
where city.countrycode = country.code
group by Continent;

* If there are two tables Employee1 and Employee2, and both common records how can? fetch all the records but common records only once.

```
(select * from Employee 1)
union
(select * from Employee 2);
```

$U = \{1,2,3,4,5\}$

$E_1 = \{1,2,3,4\}$

$E_2 = \{3,4,5\}$

$E_1 \cup E_2 = \{1,2,3,4,5\}$

→ how to fetch only common records from both the tables.

```
(select * from Emp1)
intersect
(select * from Emp2);
```

$E_1 \cap E_2 = \{3,4\}$

→ how can I retrive (fetch) those should not present in Emp 2.

$E_1 - E_2 = \{1,2\}$

```
(select * from Emp1)
Minus
(select * from Emp2);
```

→ how to fetch all the common records along with duplicates

```
(select * from Emp1)
Union All
(select * from Emp2);
```

→ write a difference b/w Union and Union All.

Union	Union All
Union always returns distinct rows. i.e, eliminates duplicate rows from the result set.	→ It includes duplicate records along with other records as well.

* Select Top 2 Salaries from Employes Tables.

```
Select Salary
From ( Select Salary
       From employees
       order by Salary Desc)
where rownum < 3;
```

	Salary
1	54,000
2	45,000
3	5400
4	4500

* WAQ to find second highest salary from employees table.

```
Select min (Salary)
From ( Select Salary
       From employees
       order by Salary Desc)
where rownum < 3;
```

* Display 7th highest salary.

```
Select min (Salary)
From (Select salary
      From employees
      order by Salary Desc)
where rownum < 8
```

* Display 5th lowest salary

```
select max (Salary)
From ( Select Salary
       From employees
       order by Salary asc.)
where rownum < 6;
```

* Display 5th recent hired Employee Data.

```
select max(hire_date)
from (select hired date
      from employees
      order by hire_date desc)
where rownum < 6
```

* Display 5th highest salary and 8th highest salary.

```
(Xselect salary
 from (select salary
       from employees
       order by salary desc)
 where rownum in (5,8); X)

( select min(salary)
  from (select salary
        from employees
        order by salary desc)
  where rownum < 6)

UNION

(select min(salary)
 from (select salary
       from employees
       order by salary desc)
 where rownum<9);
```

* WAQ to fetch odd rows from employees table

```
Select *
From ( Select rownum as "rno", Employees. *
       From employees)
where mod ("rno", 2) = 1;
```

* WAQ to display even rows from Employees table :-

```
Select *
From (Select rownum as "rno", Employees. *
      From employees)
where mod ("rno", 2) = 0;
```

Views :-

View is a virtual table based on the base table.

Users of views :-

1) To restrict data access.
2) To make complex query easy.
3) To provide data independence.

* WAQ to create a view that contains first name, last name, Emp-Id and salary from employees table. whose department Id is 80.

Syntax

```
Create view View_name
As
( Query ).
```

Create view Emp80

As

(Select first_name, last_name, Emp_id, Salary

from employees

where dep_id = 80);

⇒ Select * from view_name;

⇒ ~~Select * from~~ Drop view view_name;

Index:

Syntax

Create index index_name On Tab_name (col1, col2, --);

→ It is used to get the direct access to the rows in a table.

→ It is used by oracle server to speedup the retrival of rows using a pointer.

⇒ WAQ to create index value on first_name of Employees table.

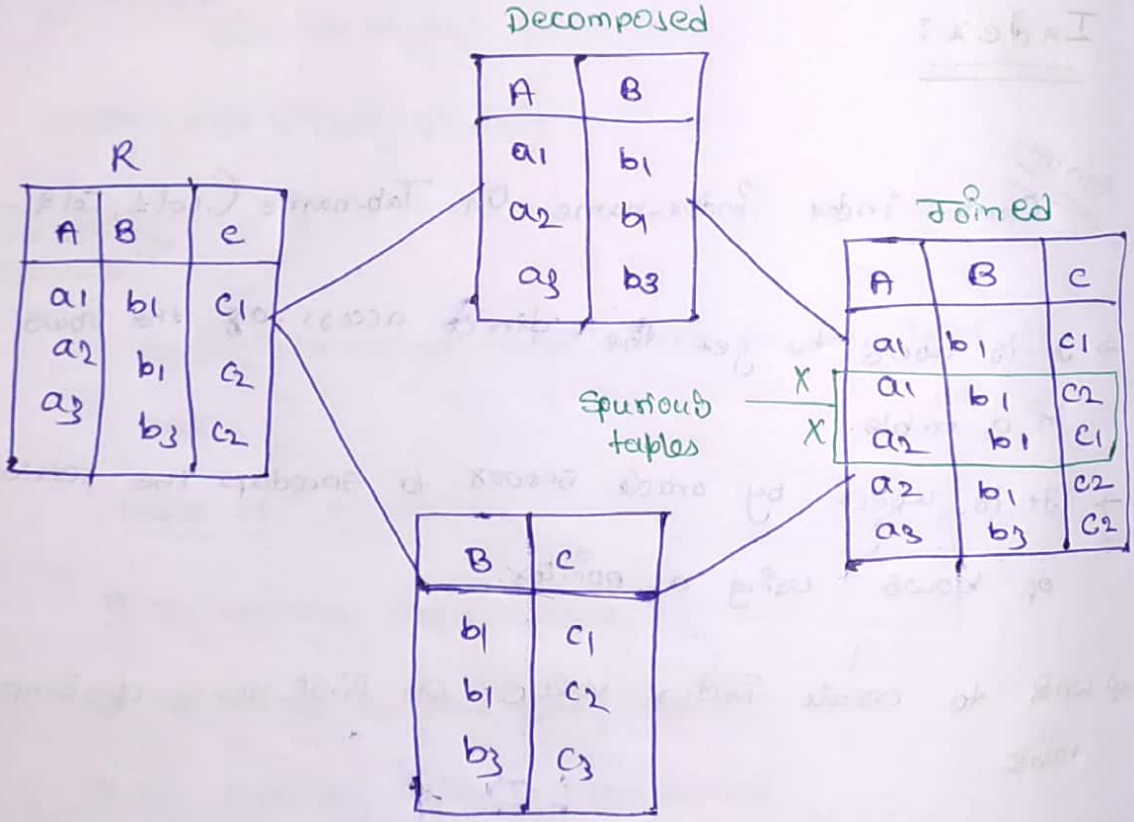
Sol:- Create index first_n On Employees (f_name);

⇒ Drop index first_n;

Normalisation :-

Spurious Tuples :- Spurious Tuples are such tuples that are generated unwantedly by decomposing the table and future Joins.

→ In order to avoid spurious tuples we have to use key attribute (NotNull & Unique).



Anamolies :-

1) Insertion Anamoly
2) Deletion Anamoly
3) Updation Anamoly

Anamolies :-

Anamolies are problems that are occuring in the table unknowingly.

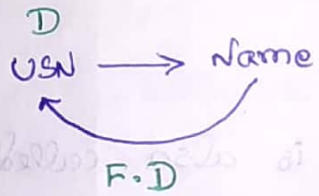
Note:- In order to remove anamolies we have to use Normalisation.

Determinent :- Determinent are such attribute by which we can access the value of an other attribu -te.

Eg:- USN, Name

D - Determinent

F.D - Functional Dependent



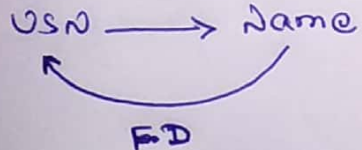
D D

Student (S-nos, S-name, S.age, S-address, S-marks, S-grade)

↳ key attribute

Studentno = Possesful key attribute.

Functional Dependency :- Those attributes which are functionally dependent on another attribute.



1) Partial Dependency
2) Total Dependency
3) Transitive Dependency

R(S_nos. C_nos, S_name, C_name, S_address, C_duration, S_marks, S_grades)

PD :- S_name, C_name, S_address, C_duration

TD :- S_marks

TrD :- S_grades

Normalisation :

Systematic approach of decomposing the table inorder to eliminate anamolies is refered as the Normalisation.

Organizing the table is also called as Normalizations.

* It was developed by E.F Codd.
* According to E.F Codd we have to follow some rules called as "forms". i.e.,

1NF (1st Normal Form)

2NF (2nd Normal Form)

3NF (3rd Normal Form)

BCNF (Boyce Codd Form)

P.NO	P.name	Eno	Ename	Rate_category	Hourly charges
1023	Android	101	Sonu	A	60
1023	Android	102	Konkona	B	70
1023	Android	103	Pappu	C	80
1024	Network	101	sonu	A	60
1024	Network	104	Kavya	D	90

1st Rule :- 1NF

A Relation is said to be in 1NF if.

1) Each attribute should have separate column. (atomic).

2. Every cell should have single value.

Rule :-2

2NF :- A relation is said to be in 2NF. if.

i) The table should be in 1NF.

ii) It should not have partial dependency.

R1

P.NO (D)	P.Name
1023	Android
1024	Network

R2

E.no (D)	Ename	Rate category	Hourly charges
101	sonu	A	60
102	konkona	B	70
103	Pappu	C	80
104	Kavya	D	90

RR

P.no	E.no
1023	101
1023	102
1023	103
1024	101
1024	104

Candidate keys

Rule 3

3NF :- A relation is said to be in 3NF. if

i) the table should be in 2NF.

ii) It should not have transitive dependency.

R1

P.no (D)	P.name
1023	Android
1024	Network

R211

E.no	E.name	Rate category
101	Sonu	A
102	konkona	B
103	Pappu	C
104	Kavya	D

R12

Rate_category	Hourly charges
A	60
B	70
C	80
D	90

P.no	E.no
1023	101
1023	102
1023	103
1024	101
1024	104

candidate keys

BCNF :-

A relation is said to be in BCNF if all the determinants are made as key attributes.