* WHAT IS DATA?

A DATA CAN BE FACT RELATED TO ANY OBJECT IN CONSIDERATION. (AGE, NAME, EMP\_ID, AADHAAR NO. ETC)

* WHAT IS DATABASE?

A DATABASE IS A COLLECTION OF SYSTEMATICALLY COLLECTION OF DATA BY WHICH WE CAN STORE AND MANIPULATE DATA IN AN EFFICIENCY WAY.

* WHY WE NEED DATABASE?

DBMS IS A SOFTWARE WHICH ALLOWS US TO STORE DATA IN CONVENIENT AND EFFICIENT METHOD OF STORING, RETRIEVING, MANIPULATING OF DATA AND IN CASE OF SECURITY ITS MORE POWERFULL (UNAUTORIZED ACCESS AND SYSTEM CRASH).

* **DBMS : DATABASE MANAGEMENT SYSTEM**
* DBMS IS FIRST IMPLEMENTED IN 1960 BY CHARLES BACHMAN’S AND HE CALLED IT AS IDS (INTEGRETED DATA STORE)
* **Candidate Key -**The candidate keys in a table are defined as the set of keys that is minimal and can uniquely identify any data row in the table.
* **Primary Key -**The primary key is selected from one of the candidate keys and becomes the identifying key of a table. It can uniquely identify any data row of the table.
* **Super Key -**Super Key is the superset of primary key. The super key contains a set of attributes, including the primary key, which can uniquely identify any data row in the table.
* **Composite Key -** If any single attribute of a table is not capable of being the key i.e it cannot identify a row uniquely, then we combine two or more attributes to form a key. This is known as a composite key.
* **Secondary Key -**Only one of the candidate keys is selected as the primary key. The rest of them are known as secondary keys.
* **Foreign Key -**A foreign key is an attribute value in a table that acts as the primary key in another table. Hence, the foreign key is useful in linking together two tables. Data should be entered in the foreign key column with great care, as wrongly entered data can invalidate the relationship between the two tables.
* **SQL : STRUCTURED QUERY LANGUAGE**
* SQL IS A STANDARD LANGUAGE FOR DEALING WITH REALTIONAL DATABASES WHICH CAN BE USED TO CREATE, READ, DELETE, UPDATE DATABASE RECORDS.
* **RDBMS**: RELATIONAL DATABASE MANAGEMENT SYSTEM

ITS REALTIONAL DATABASE IS A TYPE OF DATABASE THAT STORES AND PROVIDE ACCESS TO DATA POINTS THAT ARE REALTED TO ONE ANOTHER.

* IN RELATIONAL DATABASE EACH ROW IN THE TABLE IS RECORD WITH UNIQUE ID CALLED KEY.

**FLAVORS OF SQL**

**MYSQL, ORACLE, MS SQL,**

* **QUERY**

**SELECT \* FROM MY\_DATA WHERE AGE > 20**

**ALL DATABASE CONDITION**

* **APPLICATION OF SQL**
* SQL is used as a **Data Definition Language (DDL)** meaning you can independently create a database, define its structure, use it and then discard it when you are done with it
* SQL is also used as a **Data Manipulation Language (DML)** which means you can use it for maintaining an already existing database. SQL is a powerful language for entering data, modifying data and extracting data with regard to a database
* SQL is also deployed as a **Data Control Language (DCL)** which specifies how you can protect your database against corruption and misuse.
* SQL is extensively used as a **Client/Server language** to connect the front-end with the back-end thus supporting the client/server architecture
* SQL can also be used in the **three-tier architecture of a client, an application server and a database** which defines the Internet architecture.

 DDL commands:

* [**CREATE**](https://www.geeksforgeeks.org/sql-create/): This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
* [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/): This command is used to delete objects from the database.
* [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)**:**This is used to alter the structure of the database.
* [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)**:**This is used to remove all records from a table, including all spaces allocated for the records are removed.
* [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/): This is used to add comments to the data dictionary.
* [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/)**:**This is used to rename an object existing in the database.

DQL :

* [**SELECT**](https://www.geeksforgeeks.org/sql-select-clause/)**:**It is used to retrieve data from the database.

 DML commands:

* [**INSERT**](https://www.geeksforgeeks.org/sql-insert-statement/) : It is used to insert data into a table.
* [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/)**:** It is used to update existing data within a table.
* [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) : It is used to delete records from a database table.

DCL commands:

* [**GRANT:**](https://www.geeksforgeeks.org/mysql-grant-revoke-privileges/)This commandgives users access privileges to the database.
* [**REVOKE:**](https://www.geeksforgeeks.org/difference-between-grant-and-revoke/)This command withdraws the user’s access privileges given by using the GRANT command.

TCL commands: 

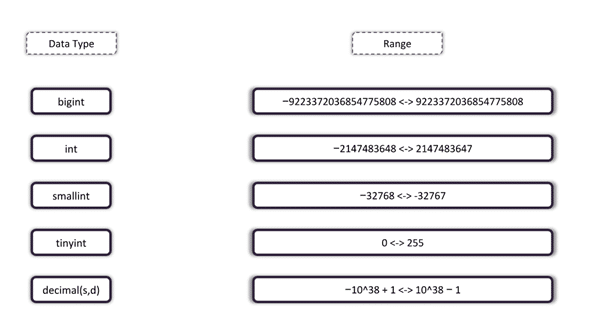
* [**COMMIT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Commits a Transaction.
* [**ROLLBACK**](https://www.geeksforgeeks.org/sql-transactions/)**:**Rollbacks a transaction in case of any error occurs.
* **BASIC OF SQL TABLES**

A TABLE IS A CAOLLECTION OF ROWS AND COLUMNS IN SQL

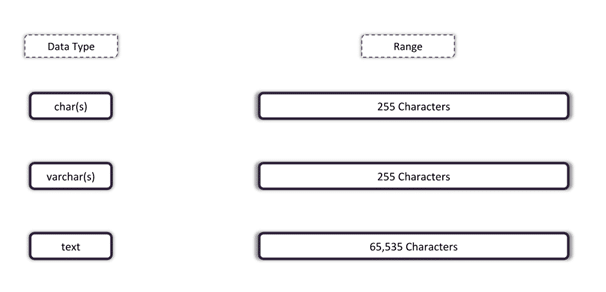
COLUMNS ARE KNOWN AS FIELDS AND ROWS ARE CALLED AS RECORDS.

* **DATATYPES**

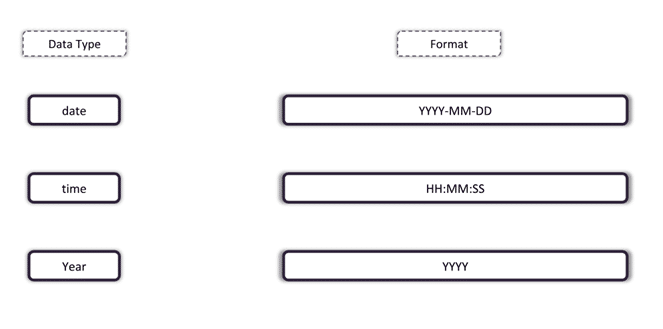
NUMERICS:



CHARACTERS:



DATE AND TIME:



SQL CONSTRAINTS: **NULL** (ENSURES NULL VALUE PRESENT IN THE TABLE), **NOT** **NULL** (ENSURES NULL VALUE NOT PRESENT), **UNIQUE** (ALL THE CONSTRAINTS INSIDE THE TABLE ARE UNIQUE VALUE), **PRIMARY KEY** (NOT NULL + UNIQUE).

**LET’S START WITH SQL COMMANDS**

**CREATING A DATABASE:**

CREATE DATABASE database\_name

**USE A DATABASE:**

USE database\_name;

**DROP DATABASE:**

DROP DATABASE database\_name;

**CREATE TABLE:**

CREATE TABLE table\_name (

COLUMN\_NAME DATATYPE ,

……,

…..,

PRIMARY KEY(COLUMN\_NAME));

**INSERT INTO TABLE:**

INSERT INTO table\_name VALUES(VALUE1,VALUE2 ….);

**EXTRACT DATA FROM TABLE USING SELECT STATEMENT :**

SELECT column\_name FROM table\_name; { extract single column

SELECT column\_name1, column\_name2, column\_name3 from table\_name { extract multiple column

**EXTRACT THE DISTINCT VALUES FROM TABLE:**

SELECT DISTINCT column\_name FROM table\_name;

{ It will extract all the distinct value that means repeated value like a table having gender column and we need to extract those then it will shows only MALE and FEMALE

**WHERE STATEMENT (CONDITIONAL STATEMENT):**

WHERE CLAUSE IS USED IN THE STATEMENT WHERE WE NEED TO EXTRACYT VALUE WITH CONDITION.

SELECT COLUMN1,COL\_2.. FROM table\_name WHERE [CONDITION]

Ex: SELECT EMP\_NAME,AGE,CITY FROM EMPLOYEE WHERE AGE >= 21;

The following constraints are commonly used in SQL:

* [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables
* [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified
* [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly

**OPERATORS:**

**AND OPERATOR :** IT GRABS YOU THE RECORDS WHICH STATISFIES THE SEPARATED CONDTION BY AND.

SELECT column\_name FROM table\_name WHERE [CONDITION1] AND [CONDITION2]

Ex: SELECT \* FROM EMPLOYEE WHERE AGE >=21 AND CITY ='PUNE';

**OR OPERATOR :** IT GRABS YOU THE RECORDS WHICH STATISFIES THE SEPARATED CONDTION BY OR.

SELECT column\_name FROM table\_name WHERE [CONDITION1] AND [CONDITION2] OR [CONDIOTN3]

Ex: SELECT \* FROM EMPLOYEE WHERE AGE >=21 AND CITY ='P' OR CITY='M';

**NOT OPERATOR :** NOT OPERATOR USED TO GRAB THE RECORDS WHERE CONDITION NEED TO BE FALSE.

SELECT column\_name from table\_ name WHERE NOT [CONDITION]

Ex: SELECT \* FROM EMPLOYEE WHERE NOT AGE >= 21;

**LIKE OPERATOR :** LIKE OPERATOR IS USED TO EXTRACT THE RECORDS WHERE PARTICULAR PATTERN IS PRESENT.

WE USED TWO OPERATOR FOT THIS `%` AND `\_`

% : USED FOR ONE OR MULTIPLE CHARCTERS AND ZERO

\_: USED FOR ONLY SINGLE CHARACTER

SELECT \* FROM table\_name WHERE NAME LIKE ‘A%’

SELECT \* FROM table\_name WHERE NAME LIKE ‘%A’

SELECT \* FROM table\_name WHERE NAME LIKE ‘A\_’

SELECT \* FROM table\_name WHERE NAME LIKE ‘\_A’

**BETWEEN OPERATOR:**

BETWEEN OPERATOR IS USED TO GRAB THE REOCRDS HAVING A RANGE

SELECT \* FROM table\_name WHERE AGE BETWEEN RANGE\_1 AND RANGE\_2;

Ex: SELECT \* FROM EMPLOYEE WHERE AGE BETWEEN 21 AND 23;

**AGGREGATE FUNCTIONS:**

**MIN( ) AND MAX ( ) FUNCTION:**

IT WILL EXTRACT THE VALUE FROM THE COLUMN OF A TABLE.

SELECT MIN(column\_name) from table\_name;

SELECT MAX(column\_name) FROM table\_name;

Ex:

SELECT MIN(AGE) FROM EMPLOYEE;

SELECT MAX(AGE) FROM EMPLOYEE;

**COUNT( ) FUNCTION:**

IT WILL COUNT THE NUMBER OF RECORDS FROM A GIVEN COLUMN OR TABLE.

SELECT COUNT(\*) FROM table\_name;

Ex:

SELECT COUNT(\*) FROM EMPLOYEE

SELECT COUNT(\*) FROM EMPLOYEE WHERE AGE >= 21

**SUM( ) FUNCTION:**

IT WILL CALCULATE THE SUM OF THE VALUES IN A GIVEN COLUMN.

SELECT SUM(column\_name) FROM TABLE\_NAME;

Ex: SELECT SUM(SALARY) FROM EMPLOYEE;

SELECT SUM(SALARY) FROM EMPLOYEE WHERE DEPT = ‘TESTING’.

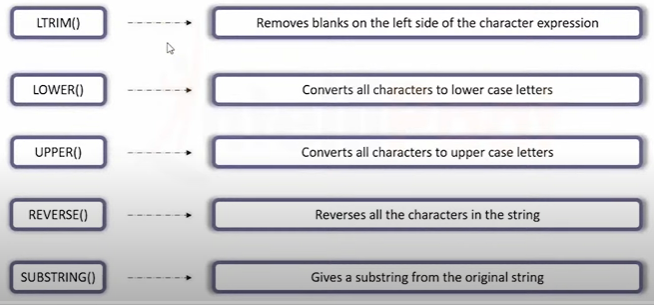
**AVG( ) FUNCTION:**

IT WILL CALCULATE THE AVERAGE OF A PARTICUALR COLUMN FROM A TABLE.

SELECT AVG(coulumn\_name) FROM table\_name;

Ex: SELECT AVG(SALARY) FROM EMPLOYEE;

**STRING FUNCTION:**



**LTRIM ():** SELECT ' EMPLOYEE' -> WHEN YOU EXECUTE THIS STATEMENT IT WILL NOT REMOVE THE SPACE

SELECT LTRIM (' EMPLOYEE') -> BY USING LTRIM IT WILL REMOVE THE SPACE.

**LOWER ():** SELECT LOWER('WELCOME TO SQL'):

**UPPER ():** SELECT UPPER('welcome to sql')

**REVERSE ():** SELECT reverse('welcome to sql')

**SUBSTRING ():** SELECT'WELCOME TO SQL'

SELECT SUBSTRING('WELCOME TO SQL',12,3)

#SQL

**ORDER BY :**

IT WILL SORT THE COLUMN AS ASCENDING OR DESCENDING ORDER.

SYNTAX : **SELECT column\_list FROM table\_name ORDER BY column\_name ASC |DESC**

SELECT \* FROM EMPLOYEE ORDER BY AGE DESC

Imp: get 3 rd highest salary

SELECT \*

FROM

(

Select \*, ROW\_NUMBER() OVER(Order by SALARY Desc) as Salary\_Order

from DEVELOPMENT

) DT

WHERE DT. Salary\_Order = 3;

**TOP CLAUSE :**

**SELECT TOP X \* FROM** table\_name; # X NUMBER OF COUNTS FROM TOP

BUT IN MY SQL TOP CLUASE IS NOT SUPPORTED SO WE USE **LIMIT**

**LIMIT ():**

**SELECT \* FROM** table\_name **LIMIT X;** # X NUMBER OF COUNTS FROM TOP

**GROUB BY CLUASE:**

GROUP BY CLAUSE IS USE TO FIND THE AGGREGATE VALUE OF A GROUP.

**SELECT column\_list FROM table\_name GROUP BY col\_name(s);**

**SELECT column\_list FROM table\_name WHERE CONDITION GROUP BY column\_name(s)**

**ORDER BY column\_names(s);**

EXAMPLE:

SELECT AVG(EMP\_SALARY),DEPT\_NAME FROM KEYBANK GROUP BY DEPT\_ID.

SELECT AVG(EMP\_AGE),DEPT\_NAME FROM KEYBANK GROUP BY DEPT\_ID ORDER BY AVG(EMP\_AGE) DESC

**HAVING CLAUSE:**

HAVING CLUASE IS USED TO DEFINE A AGGREGATE CONDITION OF A GROUP.

**SELECT column\_list from table\_name**

**GROUP BY col\_name**

**HAVING condition**

**ORDER BY col\_name;**

Example:

SELECT DEPT\_NAME,AVG(EMP\_SALARY) AS AVG\_SALARY

FROM KEYBANK

GROUP BY DEPT\_NAME

HAVING AVG\_SALARY > 45000

ORDER BY AVG\_SALARY DESC;

**UPDATE STATEMENT:**

UPDATE STATEMENT IS USED TO MODIFY THE EXISTING RECORDS FROM THE TABLE.

**UPDATE table\_name**

**SET col\_name = val1, col\_name2 = val2**

**WHERE [CONDITION…]**

UPDATE KEYBANK SET EMP\_SALARY =50000; # CHANGE ALL THE SLARY TO 50K

UPDATE KEYBANK SET EMP\_AGE = 31 WHERE EMP\_NAME='GUDU';

**DELETE STATEMENT:**

DELETE STATEMENT IS USED TO DELETE THE RECORD FROM TABLE

**DELETE FROM table\_name**

**WHERE CONDITION…**

DELETE FROM KEYBANK WHERE ‘ EMP\_AGE’ = 34;

**TRAUNCATE :**

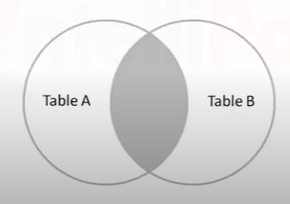
TRAUNCATE IS USED TO DELETE ALL OF DATA FROM TABLE,BUT STRUCTURE IS THERE.

**TRAUNCATE TABLE table\_name;**

J**OINS:**

**INNER JOIN:**

INNER JOINS RETURNS THE REORDS THAT ARE MATCHING THE VALUE FROM BOTH THE RECORDS.



**SELECT COULMNS**

**FROM TABLE\_1**

**INNER JOIN TABLE\_2**

**ON WHERE CONDITION….**

EXAMPLE

SELECT DISTINCT KEYBANK.EMP\_NAME, KEYBANK.DEPT\_NAME, KEYDEPT.D\_NAME, KEYDEPT.D\_HEAD

FROM KEYBANK

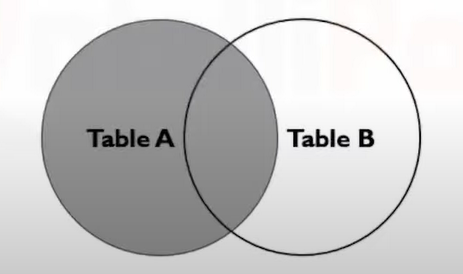
INNER JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME

**LEFT JOIN:**

LEFT JOIN RETURNS ALL THE RECORDS FROM LEFT TABLE AND THOSE RECORDS WHICH

ARE MATCHING WITH RIGHT TABLE.



**SELECT COULMNS**

**FROM TABLE\_1**

**LEFT JOIN TABLE\_2**

**ON CONDITION….**

EXAMPLE:

SELECT DISTINCT KEYBANK.EMP\_NAME, KEYBANK.EMP\_AGE, KEYDEPT.D\_NAME

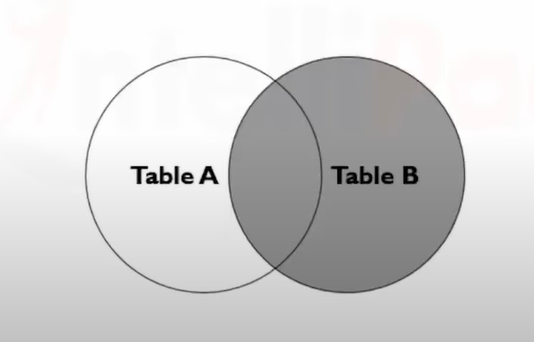
FROM KEYBANK

LEFT JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME;

**RIGHT JOIN:**

RIGHT JOIN RETURNS ALL THE RECORDS FROM THE RIGHT TABLE AND THE RECORDS WHICH ARE MATCHING WITH LEFT TABLE.



**SELECT COLUMNS**

**FROM TABLE1**

**RIGHT JOIN TABLE2**

**ON CONDITION;**

EXAMPLE:

SELECT DISTINCT KEYBANK.EMP\_NAME, KEYBANK.EMP\_AGE, KEYDEPT.D\_NAME,KEYDEPT.D\_HEAD

FROM KEYBANK

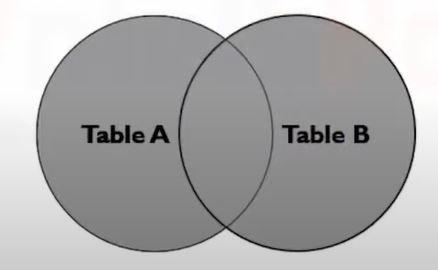
RIGHT JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME;

**FULL JOIN:**

FULL JOIN RETURNS THE RECORDS FROM LEFT AND RIGHT TABLE WITH NULL VALUES WHERE THE CONDITION NOT MET.

**MY SQL IDE DOESN’T SUPPORT FULL JOIN, SO WE CAN USE IT LIKE LEFT JOIN UNION RIGHT JOIN.**



**SELECT COLUMNS**

**FROM TABLE1**

**FULL JOIN TABLE2**

**ON CONDTION**

EXAMPLE:

SELECT DISTINCT KEYBANK.EMP\_NAME, KEYBANK.EMP\_AGE, KEYDEPT.D\_NAME,KEYDEPT.D\_HEAD

FROM KEYBANK

FULL JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME;

EXAMPLE: (MY SQL)

SELECT DISTINCT KEYBANK.EMP\_NAME, KEYBANK.EMP\_AGE, KEYDEPT.D\_NAME,KEYDEPT.D\_HEAD

FROM KEYBANK

LEFT JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME

union

SELECT DISTINCT KEYBANK.EMP\_NAME, KEYBANK.EMP\_AGE, KEYDEPT.D\_NAME,KEYDEPT.D\_HEAD

FROM KEYBANK

RIGHT JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME;

**UPDATE :**

WE CAN UPDATE THE VALUE PRESENT IN THE RECORDS USING UPDATE.

**UPDATE** TABLE\_NAME

**SET** UPDATION\_CONDITION

**WHERE CONDITION,**

EXAMPLE:

UPDATE KEYBANK

SET EMP\_AGE = EMP\_AGE + 4

WHERE DEPT\_NAME = 'FINANCE';

**UPDATE USING DUAL TABLE :**

UPDATE KEYBANK

JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME

SET EMP\_AGE = EMP\_AGE + 4

WHERE D\_HEAD = 'ANIL';

**DELETE:**

**DELETE FROM** TABLE\_NAME

**WHERE CONDITION;**

EXAMPLE: IN DUAL TABLE

DELETE KEYBANK

FROM KEYBANK

JOIN KEYDEPT

ON KEYBANK.DEPT\_NAME = KEYDEPT.D\_NAME

WHERE D\_HEAD = 'MR MARK'; // DELETES THE RECORS WHERE D\_HEAD IS MARK.

**UNION:**

**UNION** OPERATOR IS USED TO COMBINE THE RESULT SET OF TWO OR MORE STATEMENT

DULICATION NOT ALLOWED.



**SELECT column\_list FROM table\_1**

**UNION**

**SELECT column\_list FROM table\_2**

Example:

select \* from stud1

union

select \* from stud2

**UNION ALL:**

BY USING UNION ALL WE CAN EXTRACT THE DUPLICATE VALUE.

Example:

select \* from stud1

union all

select \* from stud2

**INTERSECT:**

Example:

select \* from stud1

union all

select \* from stud2

**THUS, INTERSECT IS NOT PRESENT IN MYSQL INSTEAD OF THAT WE CAN USE INNER JOIN AND IN OPERATOR.**

**CREATE VIEW:**

BY USING THIS **CREATE VIEW** OPERATOR WE CAN EXTRACT THE PARTICULAR RECORDS FROM THE TABLE.

**CREATE VIEW view\_name AS**

**SELECT \* FROM TABLE\_NAME**

**WHERW CONDITION;**

EXAMPLE:

CREATE VIEW AGE AS

SELECT \* FROM STUD1

WHERE STD\_AGE = 21

SELECT \* FROM AGE

**DROP VIEW:**

**DROP VIEW view\_name;**

**ALTER:**

ALTER TABLE IS USED TO ADD DELETE AND MODIFY THE COLUMNS IN THE TABLE.

**ADD COLUMN:**

**ALTER TABLE table\_name**

**ADD column \_name DATATYPE;**

EXAMPLE:

ALTER TABLE STUD1

ADD CITY CHAR;

**DROP COLUMN:**

**ALTER TABLE table\_name**

**DROP COLUMN column\_name;**

IMPORTANT:

FIND THE NTH HIGHEST SAALRY FROM TABLE ?

**SELECT \* FROM DEVELOPMENT**

**ORDER BY**

**SALARY DESC**

**LIMIT N-1,1;**

**IN THE PLACE OF N ENTER YOUR NTH NUMBER . IF 3RD HIEGHEST NUMBER**

**THEN QUERY SHOULD BE**

**SELECT \* FROM DEVELOPMENT**

**ORDER BY**

**SALARY DESC**

**LIMIT 2,1;**

NORMALIZATION :

DATA IN THE DATABASE ARE NOT ORGANISED MANNER, BY THE USE OF NORMALIZATION WE CAN ORGANISE THE DATA AND REDUNDANT THE DATABASE.

Divides big table into small tables.

PROBLEMS FACED / need of anomalies:

INSERTION ANAMALIES: a emp is on training, no dept allotted so it is takes Null

UPDATE ALAMALIES: address changed of an emp who is present in two diff dept

DELETION ANAMALIES: if dept is closed so it deleted all the emp details associated with it.

1NF: if the values in the domain of each attribute are atomic.

2NF: No non prime attribute should be determined by past of Candidate key

3NF: X->Y

* 1. X-> Y TRIVIAL F.P
  2. X-> Y X IS A SUPERKEY
  3. X->Y THEN (Y-X) IS A PRIME ATTRIBUTE

BCNF : X->Y TRIVIAL

X->Y X IS A SUPERKEY

-- LEFT JOIN

-- LEFT JOIN = INNER JOIN + ADDITIONAL RECORDS LEFT IN LEFT TABLE

-- RIGHT JOIN

-- RIGHT JOIN = INNER JOIN + ADDTIONAL RECORDS LEFT IN RIGHT TABLE

FULL JOIN = INNER JOIN+

-- + ADDTIONAL RECORDS LEFT IN LEFT TABLE

-- + ADDTIONAL RECORDS LEFT IN RIGHT TABLE

SELF JOIN:

