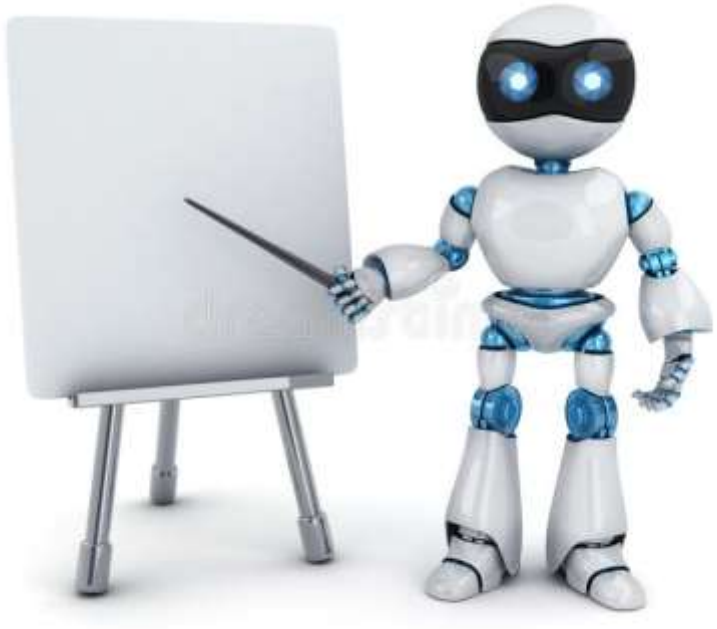


SQL

Datamites

DATABASE

A database is an organized collection of data.

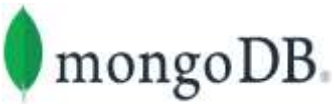
For Example:
A university database organizes
the data about students, faculty, admin staff, etc.

Datamites

DBMS(Data Base Management System)

The software which is used to manage databases is called Database Management System. DBMS allows users to create ,read , update and delete data in database.

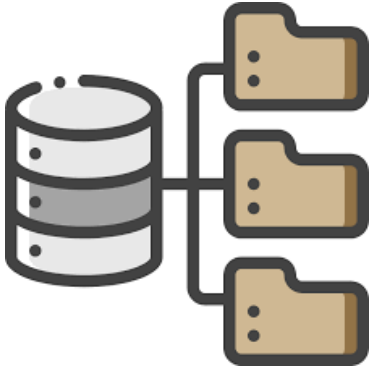
For example:
MySQL, Oracle, Microsoft SQL server, PostgreSQL are the popular commercial DBMS used in different applications.



Datamites

WHY DBMS?

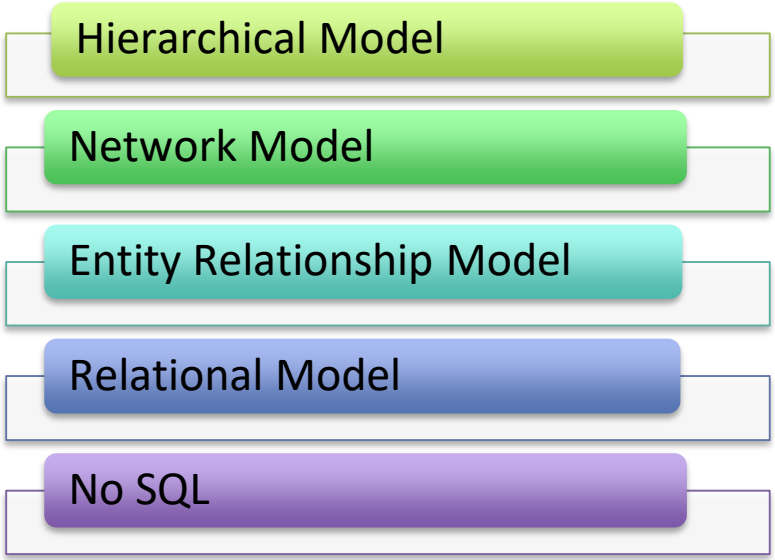
- Data Storage
- Data abstraction
- Controls data redundancy
- Multi - user access and views
- Security



Datamites

DATA MODELS

- Data models define how the logical structure of a database is modelled.
- Data models define how data is connected to each other ,how they are processed and stored inside the system.



Datamites

RELATIONAL MODEL

In this model , the data is organized in the form of two dimensional table. That means data is stored in the form of rows and columns.

For example:
The table here shows "STUDENT DETAILS" with attributes such as Stu_Id, Name and Branch which has 5 records.

Stu_Id	Name	Branch
201	John	CSE
202	Mary	ECE
203	Britto	IS
204	Harry	IT
205	Steve	ME

Datamites

RDBMS

RDBMS is a program which is used to manage Relational databases in which data is stored in the form of rows and columns which can be easily retrieved ,managed and updated.

Datamites

MySQL

- MySQL is an open source relational database that uses structured query language to interact with databases.
- It stores data in the form of table and can be modified using SQL.

Datamites

WHY MySQL ?

- Easy to use
- Cost effective
- Secured
- Platform-Friendly

Datamites

SQL

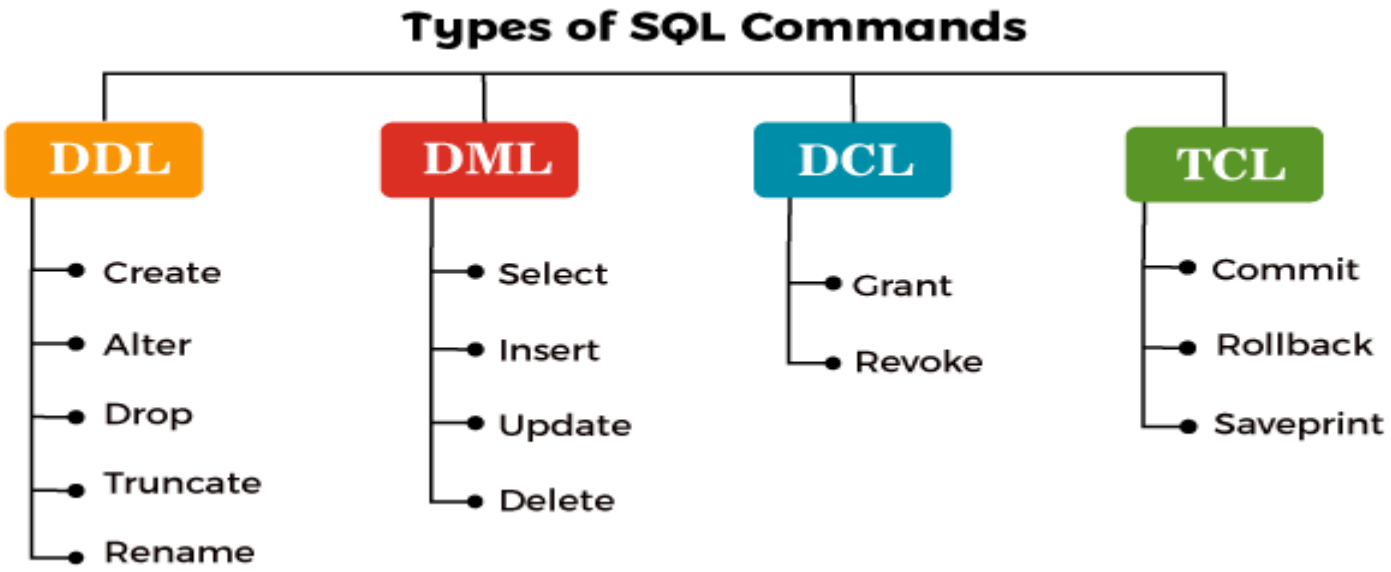
- SQL stands for Structured Query Language.
- SQL is a database programming language designed for the retrieval and management of data in a relational database.
- All relational database management systems like MySQL ,Oracle , MS Access and SQL server uses SQL as their standard database language.

WHAT SQL DOES?

- SQL is used to create new databases and tables.
- SQL execute queries against a database.
- SQL is used to retrieve, update and insert records into a database.
- SQL used to create stored procedures and views in a database.
- SQL can also set permissions on tables, procedures and views.

Datamites

SQL COMMANDS



Datamites

MySQL WORKBENCH

- MySQL workbench is a Graphical tool developed by Oracle , which is used to work with MySQL server and databases.
- MySQL provides data modelling , SQL development and various administration tools for configuration.

Datamites

CREATE DATABASE

CREATE DATABASE <database name>:

```
Create Employee Database

CREATE DATABASE Employee;
```

Datamites

DROP DATABASE

DROP DATABASE <database name>:

```
Drop  Employee database

DROP DATABASE Employee ;
```

Datamites

SHOW DATABASE

SHOW DATABASES:

Return all the databses

```
SHOW DATABASES ;
```

Datamites

TABLE

SQL table is a collection of data which is organized in rows and columns. Each row represents unique record and each column represents field in the record.

emp_id	emp_name	emp_designation	emp_age
E40001	PRADEEP	H.R	36
E40002	ASHOK	MANAGER	28
E40003	PAVAN KUMAR	ASST MANAGER	28
E40004	SANTHOSH	STORE MANAGER	25
E40005	THAMAN	GENERAL MANAGER	26

For example:

Here is a table which contains employee data in which row represents each employee and column represents employee information such as employee number, name, designation and age.

Datamites

CREATE TABLE

CREATE TABLE < table name > (
< column1> <data type> ,
< column1> <data type> ,
< column1> <data type> ,
…
) ;

```
-- Create employee _details table using SQL.

CREATE TABLE employee_details(
  emp_id VARCHAR(8),
  emp_name VARCHAR(20),
  emp_designation VARCHAR(20),
  emp_age INT);
```

Datamites

DATA TYPES

The data type of a column defines what sort of data that an object can store such as integer, character, money, date and time, binary, and so on.

- NUMERIC
- CHARACTER
- DATE AND TIME

Datamites

NUMERIC

Data Type	Range
bigint dd	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int(s)	-2,147,483,647 to 2,147,483,647
smallint	-32,768 to 32767
tinyint	0 - 255
decimal(s , d)	-10 ^38+1 to 10^38 -1

Datamites

DATE AND TIME

Data Type	Format
Date	YYYY-MM-DD
Time	HH:MM:SS
Year	YYYY

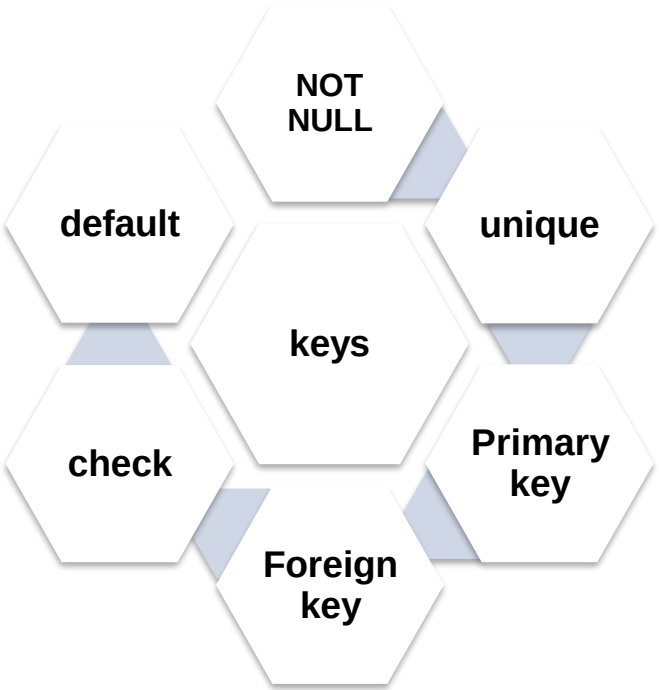
Datamites

CHARACTER

Data Type	Range
Char	0 - 255 characters
Varchar(s)	0 – 65535 characters
Text	65535 characters

SQL CONSTRAINTS

A DBMS keys are an attributes which help you uniquely identify a record or row of data in a relation.



Datamites

SQL CONSTRAINTS

Not null	It ensures that column cannot have null values.
Unique	It ensures that all values in a column are unique.
Primary key	It's a combination of not null and unique keys.
Foreign key	It prevents actions that would destroy link between tables.
Default	Sets a default value for a column if value is specified.
Check	It ensures that the values in a column satisfies the specified condition.

Datamites

DROP TABLE

DROP TABLE < table name > ;

```
- Delete employee_details table

DROP TABLE employee_details ;
```

Datamites

INSERT INTO

Insert into is used to insert new records in the table.

INSERT INTO < table name > (column1 ,column2,column3…)
VALUES (<val1>, <val2> ,<val3>…);

```
INSERT INTO employee_details VALUES
    ('E40001','PRADEEP','H.R',36),
    ('E40002','ASHOK','MANAGER',28),
    ('E40003','PAVAN KUMAR','ASST MANAGER',28),
    ('E40004','SANTHOSH','STORE MANAGER',25),
    ('E40005','THAMAN','GENERAL MANAGER',26);
```

Datamites

UPDATE

Used to modify the existing records in a table using where condition.

UPDATE TABLE < table name >
SET <column1>=<value1> ,<column2>=<value1> …
WHERE <condition> ;

Change employee name and age where empid=E40005

```
UPDATE employee_details SET emp_name='John', emp_age=40  WHERE emp_id='E40005' ;
```

Datamites

DELETE

Used to delete the existing records in a table using where condition.

DELETE FROM < table name >
WHERE <condition> ;

delete details of an employee whose empid=E40005

```
DELETE FROM employee_details  WHERE emp_id='E40005' ;
```

Datamites

ALTER

ADD
It is used to add new column to the table.

ALTER TABLE < table name > **ADD** < column name > < data type > ;

```
add new column emp_experience

ALTER TABLE employee_details ADD emp_experience int ;
```

Datamites

DROP

Used to drop existing column from the table.

ALTER TABLE < table name > **DROP COLUMN** < column name > ;

delete emp_experience column

```
ALTER TABLE employee_details DROP COLUMN emp_experience ;
```

Datamites

MODIFY

Used to modify data type of column.

ALTER TABLE < table name > **MODIFY** < column name > < data type > ;

Modify data type of emp_name as varchar(50)

```
ALTER TABLE employee_details MODIFY  emp_name varchar(50) ;
```


RENAME

Used to rename the existing column name.

ALTER TABLE < table name > **RENAME COLUMN** < old_column_name >
to <new_column_name> ;

Change emp_age as age_of_employee

```
ALTER TABLE employee_details RENAME COLUMN emp_age to age_of_employee ;
```

Datamites

RENAME

Used to rename table.

ALTER TABLE < table name > **RENAME TO** <new_table_name> ;

Change table name employee_details to employee_data

```
ALTER TABLE employee_details RENAME TO employee_data ;
```


TRUNCATE

Truncate used to clear the records inside table.

TRUNCATE TABLE< table name > ;

Delete all the records from employee_data

```
TRUNCATE TABLE employee_data ;
```

Datamites

IMPORT TABLE

- Select database
- Right click
- Select Table data import wizard
- Browse to select the file
- Check on create new table
- Click next
- Import

IMPORT DATABASE

- Click on Server
- Select data import
- Self contained file
- Open Import progress
- Start import

Datamites

EXPORT TABLE

- Select Table
- Right click
- Select Table data export wizard
- Browse to select destination
- Export

Datamites

EXPORT DATABASE

- Click on Server
- Select data export
- Select the database
- Open export progress
- Start export

SELECT

Select statement is used to fetch all the records from the table or to fetch subset of columns.

SELECT <column1>, < column2 > **FROM** < table name > ;
SELECT < * > **FROM** < table name > ;

```
Fetch all the records from customer table
SELECT * FROM customers ;

Fetch " OrderId , OrderDate , Country , Region" from customers
SELECT OrderId ,OrderDate ,Country , Region FROM customers ;
```

Datamites

SELECT DISTINCT

Used to fetch unique records from the table.

SELECT <distinct(column1)> **FROM** < table name > ;

```
Fetch unique categories from customer table
SELECT DISTINCT(Category) FROM customers ;
```

Datamites

ALIASES

SQL Aliases are used to give temporary name for column and table .

SELECT <column_name> **AS** <alias_name> **FROM** < table name > ;

```
Change OrderId as id
SELECT OrderId as id FROM customers ;
```

Datamites

WHERE CLAUSE

Used to filter records based on condition.

Conditions are of two types:

Comparison : = , > ,< >= ,<= ,!=

Logical : and , or ,not

WHERE CLAUSE

Used to filter records based on condition.

SELECT * FROM < table name > where <condition> ;

```
Fetch order details related to technology
SELECT * FROM customers WHERE category = 'Technology' ;

Fetch order details where Quantity is greater than 10
SELECT * FROM customers where Quantity>10 ;

Fetch order details where Sales is less than 5000
SELECT * FROM customers where Sales<5000 ;
```

Datamites

AND

Return records which satisfies both the conditions.

SELECT * FROM < table name > where <condition 1> AND <condition 2> ;

Fetch all records from customers where State is Texas and Category is Technology.

```
SELECT * FROM customers WHERE State= 'Texas' AND Category = 'Technology';
```

Fetch all records from customers where Quantity is greater than 5 ,Region is East and Subcategory is Phones.

```
SELECT * FROM customers WHERE Quantity > 5 AND Region = 'Technology' AND Subcategory = 'Phones';
```

Datamites

OR

Return records which satisfies either of the conditions.

SELECT * **FROM** < table name > where <condition 1> OR <condition 2> ;

```
Fetch all the records related to Technology and Office Supplies
SELECT * FROM customers WHERE Category = ' Technology ' OR Category = 'Office Supplies' ;

Fetch all the records related to Phones, Paper and Art
SELECT * FROM customers WHERE Subcategory = ' Phones ' OR Subcategory = ' Paper ' OR Subcategory =' Art ' ;
```

Datamites

NOT

Return records if condition is true.

SELECT * **FROM** < table name > **where not** <condition > :

```
Fetch all the records except records related to technology
SELECT * FROM customers WHERE NOT Category = 'Technology' ;

Fetch all the records except records related to central region
SELECT * FROM customers WHERE NOT Region='Central' ;
```

Datamites

ORDER BY

Order by keyword is used to sort the records in Ascending or descending order.

SELECT * FROM < table name > where <condition > order by <column> asc/desc :

```
Fetch all the records related to Technology and order by orderdate in ascending order
SELECT * FROM customers WHERE Category = ' Technology ' ORDER BY OrderDate ASC ;

Fetch all the records related to Technology and order by orderdate in ascending order
SELECT * FROM customers WHERE State= 'Texas' ORDER BY Orderdate DESC ;
```

Datamites

IS NULL/IS NOT NULL

Is null / Is not null are used to check whether null values exists in the table or not.

SELECT * FROM < table name > where <column1 > is null / not null :

```
Check are there any null values in category column
SELECT * FROM customers WHERE Category IS NULL ;

SELECT * FROM customers WHERE Category IS NOT NULL ;
```

Datamites

LIMIT

It is used to return the specified number of records.

SELECT * FROM < table name > WHERE <condition> LIMIT <value> :

```
Fetch top 5 orders with highest sales in Technology
SELECT * FROM customers WHERE Category='Technology' ORDER BY Sales DESC LIMIT 5 ;
```

Datamites

IN

It is used to return records by specifying multiple conditions.

SELECT * FROM < table name > WHERE <column1> IN (val1 ,val2,val3) :

```
Fetch all the records related Paper ,phones and Art Subcategories
SELECT * FROM customers WHERE Subcategory IN ( 'Paper' ,'Phones' ,'Art' ) ;
```

Datamites

BETWEEN

It is used to return records within a certain range. It can be values ,dates or text.

SELECT * FROM < table name > WHERE <column1> BETWEEN <val1> AND <val2A> ;

```
Fetch all the records between 04-01-2019 and 29-12-2022
SELECT * FROM customers WHERE OrderDate BETWEEN '04-01-2019' AND '29-12-2022' ;
```

Datamites

LIKE

It is used to return records based on the pattern.

SELECT * FROM < table name > WHERE <column1> LIKE <PATTERN> ;

```
Fetch all the records related to west region
SELECT * FROM customers WHERE Region LIKE ' W% ' ;

Fetch all the records related to Technology
 SELECT * FROM customers WHERE Category='%no%' ;
```

Datamites

LIKE

% – It represents zero ,one or multiple characters.

_ – It is used represent one or single character.

'a%' – It returns values that start with a .

'%a' – It returns values that end with a .

% or % – It returns values that have or in any position .

a %b – It returns values that start with a and end with b .

'a_' – It returns values that start with a and are at least 2 characters in length.

'_s' – It returns values that has s in the second position.

Datamites

AGGREGATE FUNCTIONS

SQL aggregate functions are used to perform the calculations on multiple rows of a single column and returns a single value.

TYPES OF AGGREGATE FUNCTIONS ARE:

- Count()
- Sum()
- Avg()
- Min()
- Max()

Datamites

COUNT

Used to return number of rows in a table.

SELECT COUNT(*) **FROM** table_name **WHERE** <condition>;
SELECT COUNT(column1) **FROM** table_name **WHERE** <condition>;

```
Find number of orders related to Technology
SELECT COUNT(*) FROM customers WHERE Category=' Technology' ;

Find count of unique categories
SELECT COUNT(DISTINCT Category ) FROM customers ;
```

Datamites

AVG()

Used to find average value of numeric column .

SELECT AVG(column_name) **FROM** table_name **WHERE**<condition>;

Find average of Sales

```
SELECT AVG(Sales) FROM customers ;
```

Datamites

SUM()

Used to find total sum of column .

SELECT SUM(column_name) **FROM** <table_name> **WHERE** <condition>;

```
Find Sum of sales in Texas
SELECT SUM(Sales) FROM customers WHERE State = 'Texas';
```

Datamites

MIN()

Used to find smallest value in selected column .

SELECT MIN(column_name) **FROM** table_name **WHERE** <condition>;

```
Find least sales in Technology
SELECT MIN(Sales) FROM customers WHERE Category= 'Technology';
```

Datamites

MAX()

Used to find largest value in selected column

SELECT MAX(column_name) **FROM** table_name **WHERE** <condition>;

Find highest sales in Technology

```
SELECT MAX(Sales) FROM customers WHERE Category = 'Technology ' ;
```

Datamites

GROUP BY

The group by statement in SQL is used to arrange identical data into groups with the help of aggregate functions.

For eg:
1) What if I want to find total sales in different categories?
2) What if I want to find highest sales for each segment?

Datamites

GROUP BY SINGLE COLUMN

SELECT column1 , **function**(column2) **FROM** <table_name> **GROUP BY** <column1> **ORDER BY** <column1>;

Find number of orders based on category

```
SELECT COUNT(OrderID) , Category FROM customers GROUP BY Category ;
```

Find Total sales over different region

```
SELECT SUM(Sales) , Region FROM customers GROUP BY  Region ORDER BY SUM(Sales)  DESC ;
```

Datamites

HAVING CLAUSE

Having clause is used to filter the result obtained by the group by clause based on specific condition.

SELECT column1 ,**function**(column2) **FROM** <table_name> **GROUP BY** <column1 ,column2> **HAVING** <condition> **ORDER BY** <column1, column2>;

Find subcategories which has got orders greater than 1000

```
SELECT COUNT(OrderID) , Subcategory FROM customers GROUP BY Subcategory HAVING COUNT(OrderID)> 1000 ;
```

Datamites

SUBQUERY

SQL subquery is a query within another SQL query. Subquery is used to fetch data from two tables.

- Subquery must be enclosed within parenthesis.
- Order by cannot be used in a subquery.
- Between operator cannot be used.
- A subquery can have only single column in subquery.

```
SELECT * FROM customers WHERE Sales = (SELECT MIN(SALES) FROM CUSTOMERS ) ;

executes subquery first ; finds minimum sales from customers
executes outerquery ; select rows where sales is equal to the result of subquery
```

Datamites

SUBQUERY

Customer_id	f_name	l_name	age	country
1	John	Doe	32	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	David	25	UK
5	Betty	Doe	28	UAE

```
SELECT * FROM customers
WHERE age =
(SELECT MIN(age)
FROM CUSTOMERS ) ;
```

Customer_id	f_name	l_name	age	country
2	Robert	Luna	22	USA
3	David	Robinson	22	UK

Datamites

SUBQUERY

Customer_id	f_name
1	John
2	Robert
3	David
4	John
5	Betty

```
SELECT customer_id ,f_name
from customers
where customer_id in
(SELECT customer_id from orders ) ;
```

Order_id	amount	Customer_id
1	200	4
2	500	10
3	300	3
4	800	1
5	150	2

customer_id	f_name
1	John
2	Robert
3	John
4	David

Datamites

RULES TO USE GROUP BY

- The group by clause is used with select statement.
- Group by is placed before having clause.
- Group by clause is placed before order by.
- Having clause is placed before order by.
- Conditions are specified in having clause.

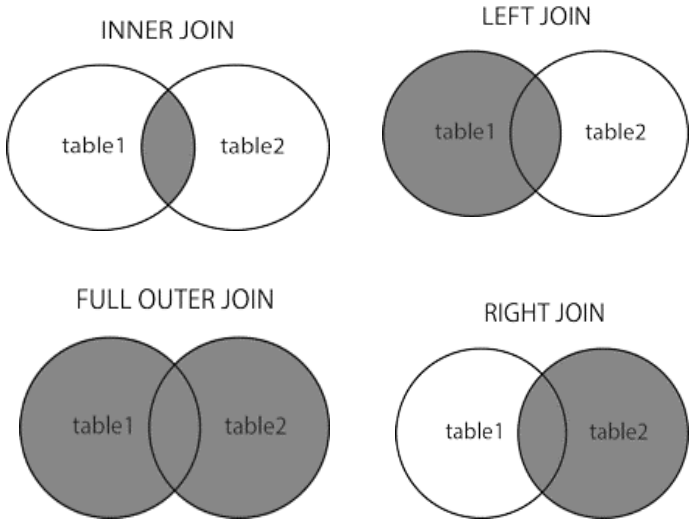
Datamites

JOINS

SQL Joins are used combine two tables based on common column, and selects records that have matching values in these columns.

Types of Joins in SQL

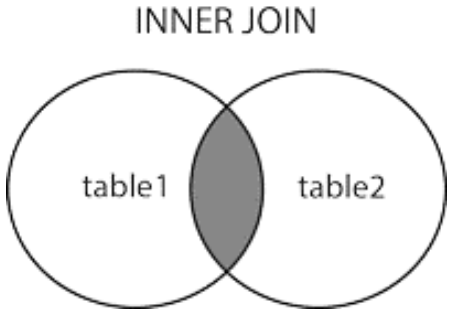
- INNER JOINS
- LEFT JOIN
- Right join
- Full outer join



Datamites

INNER JOIN

Inner join return those records which a have matching values in both tables.




```
SELECT <column_names> FROM  Table1
 INNER JOIN Table2
ON Table1.matching_column=Table2.matching_column ;
```

Datamites

empid	ename	Salary	depid
E1	John	45000	D1
E2	Mary	60000	D2
E3	Steve	73000	D3
E4	Helen	86000	D4
E5	Joe	35000	D7

```
SELECT e.empid , e.ename ,e.salary , d.depid ,d.dname
FROM employee as e INNER JOIN department as d
ON e.depid=d.depid ;
```

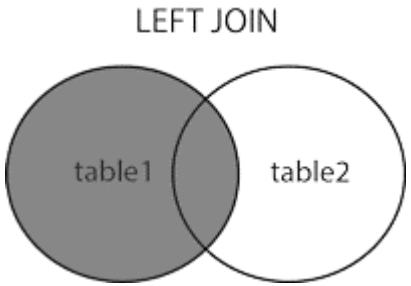
depid	dname
D1	IT
D2	HR
D3	Admin
D4	Finance
D5	Sales

empid	ename	Salary	depid	dname
E1	John	45000	D1	IT
E2	Mary	60000	D2	HR
E3	Steve	73000	D3	Admin
E4	Helen	86000	D4	Finance

Datamites

LEFT JOIN

Left join returns all the records from left table and also matching records from right table.




```
SELECT <column_names> FROM  Table1
 LEFT JOIN Table2
ON Table1.matching_column=Table2.matching_column ;
```

Datamites

empid	ename	Salary	depid
E1	John	45000	D1
E2	Mary	60000	D2
E3	Steve	73000	D3
E4	Helen	86000	D4
E5	Joe	35000	D7

```
SELECT e.empid , e.ename ,e.salary , d.depid ,d.dname
FROM employee as e LEFT JOIN department as d
ON e.depid=d.depid ;
```

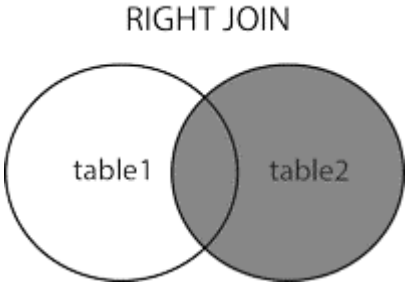
depid	dname
D1	IT
D2	HR
D3	Admin
D4	Finance
D5	Sales

empi	ename	Salary	depid	dname
E1	John	45000	D1	IT
E2	Mary	60000	D2	HR
E3	Steve	73000	D3	Admin
E4	Helen	86000	D4	Finance
E5	Joe	35000	D7	NULL

Datamites

RIGHT JOIN

Right join returns all the records from right table and also matching records from left table.




```
SELECT <column_names> FROM  Table1
RIGHT JOIN Table2
ON Table1.matching_column=Table2.matching_column ;
```

Datamites

empid	ename	Salary	depid
E1	John	45000	D1
E2	Mary	60000	D2
E3	Steve	73000	D3
E4	Helen	86000	D4
E5	Joe	35000	D7

```
SELECT e.empid , e.ename ,e.salary , d.depid ,d.dname
FROM employee as e RIGHT JOIN department as d
ON e.depid=d.depid ;
```

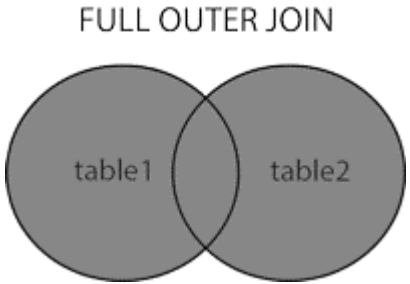
depid	dname
D1	IT
D2	HR
D3	Admin
D4	Finance
D5	Sales

depid	dname	empid	ename	Salary
D1	IT	E1	John	45000
D2	HR	E2	Mary	60000
D3	Admin	E3	Steve	73000
D4	Finance	E4	Helen	86000
D5	Sales	NULL	NULL	NULL

Datamites

FULL OUTER JOIN

Full outer join returns all the matching records from both the tables.




```
SELECT <column_names> FROM  Table1
LEFT JOIN Table2
ON Table1.matching_column=Table2.matching_column
UNION
SELECT <column_names> FROM  Table1
LEFT JOIN Table2
ON Table1.matching_column=Table2.matching_column ;
```

Datamites

empid	ename	Salary	depid
E1	John	45000	D1
E2	Mary	60000	D2
E3	Steve	73000	D3
E4	Helen	86000	D4
E5	Joe	35000	D7

```
SELECT e.empid , e.ename ,e.salary , d.depid ,d.dname
FROM employee as e LEFT JOIN department as d
ON e.depid=d.depid
UNION
SELECT e.empid , e.ename ,e.salary , d.depid ,d.dname
FROM employee as e RIGHT JOIN department as d
ON e.depid=d.depid ;
```

depid	dname
D1	IT
D2	HR
D3	Admin
D4	Finance
D5	Sales

empid	ename	Salary	depid	dname
E1	John	45000	D1	IT
E2	Mary	60000	D2	HR
E3	Steve	73000	D3	Admin
E4	Helen	86000	D4	Finance
E5	Joe	35000	D7	Null
NULL	NULL	NULL	D5	Sales

Datamites

SELF JOIN

A self join is a regular join which helps you to join a table to itself

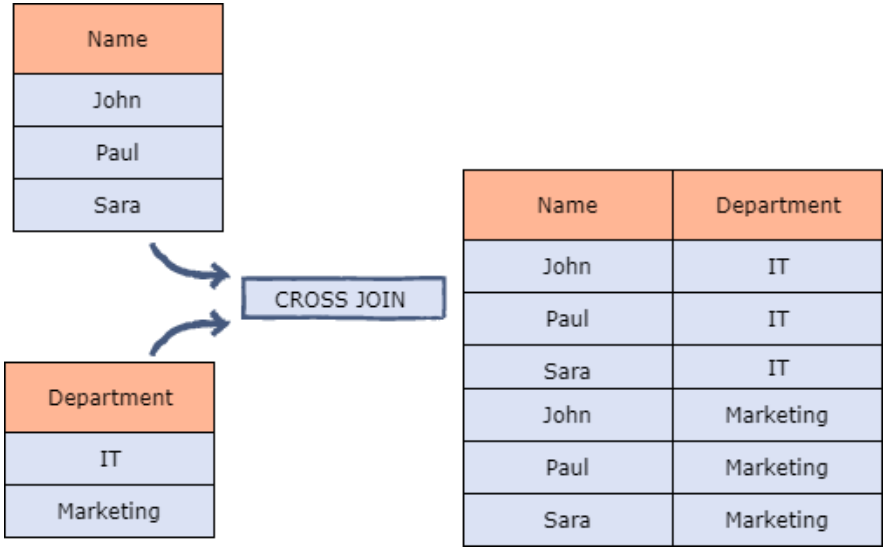
SELECT column_name(s) **FROM** table1 T1, table T2 **WHERE** <condition> ;

Datamites

CROSS JOIN

The cross join returns all the records from both the tables.

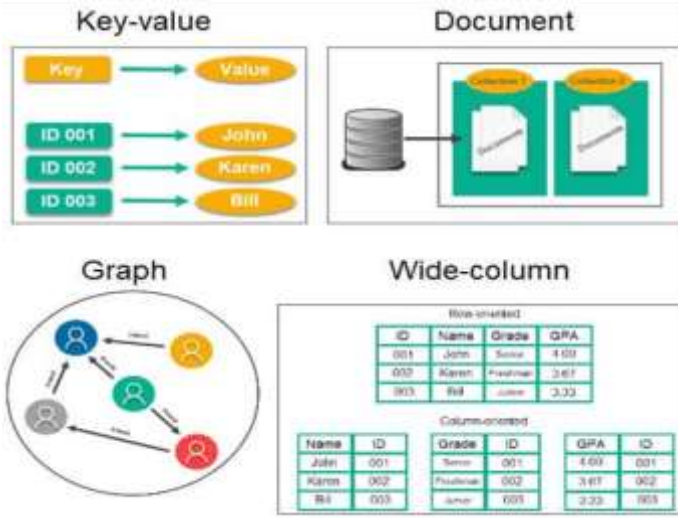
SELECT column_name(s) **FROM** <Table 1> **CROSS JOIN** <Table 2>:



Datamites

NO SQL

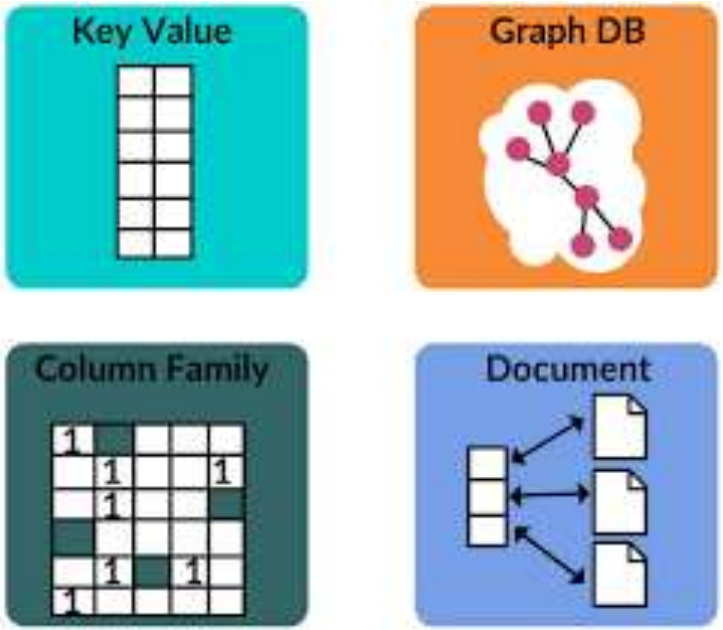
- No SQL stands for “Not only SQL”.
- No SQL database is non-relational database management system.
- No SQL database stores information in JSON documents instead of rows and columns.
- No SQL databases are flexible ,scalable and capable of rapidly responding to the data management demands of modern businesses.



Datamites

POPULAR NO SQL DATABASES

- Document databases
- Key-value stores
- Wide-column databases
- Graph databases



Datamites

APPLICATIONS OF NO SQL

- Support large number of concurrent users.
- Deliver highly responsive experience to a globally distributed base of users.
- It can store structured , semi-structure ,unstructured and polymorphic data.
- No SQL is used for Big data and real-time web apps.
- Companies like Twitter , Facebook and Google collect terabytes of user data every single day.