**DBMS**

**NISHANTHINI V**

EBEON0323765163

**DBMS**

Database Management Systems (DBMS) are software systems

used to store, retrieve, and run queries on data. A DBMS serves as an

interface between an end-user and a database, allowing users to

create, read, update, and delete data in the database.

**SQL**

Structured query language (SQL) is a programming language for

storing and processing information in a relational database. A

relational database stores information in tabular form, with rows and

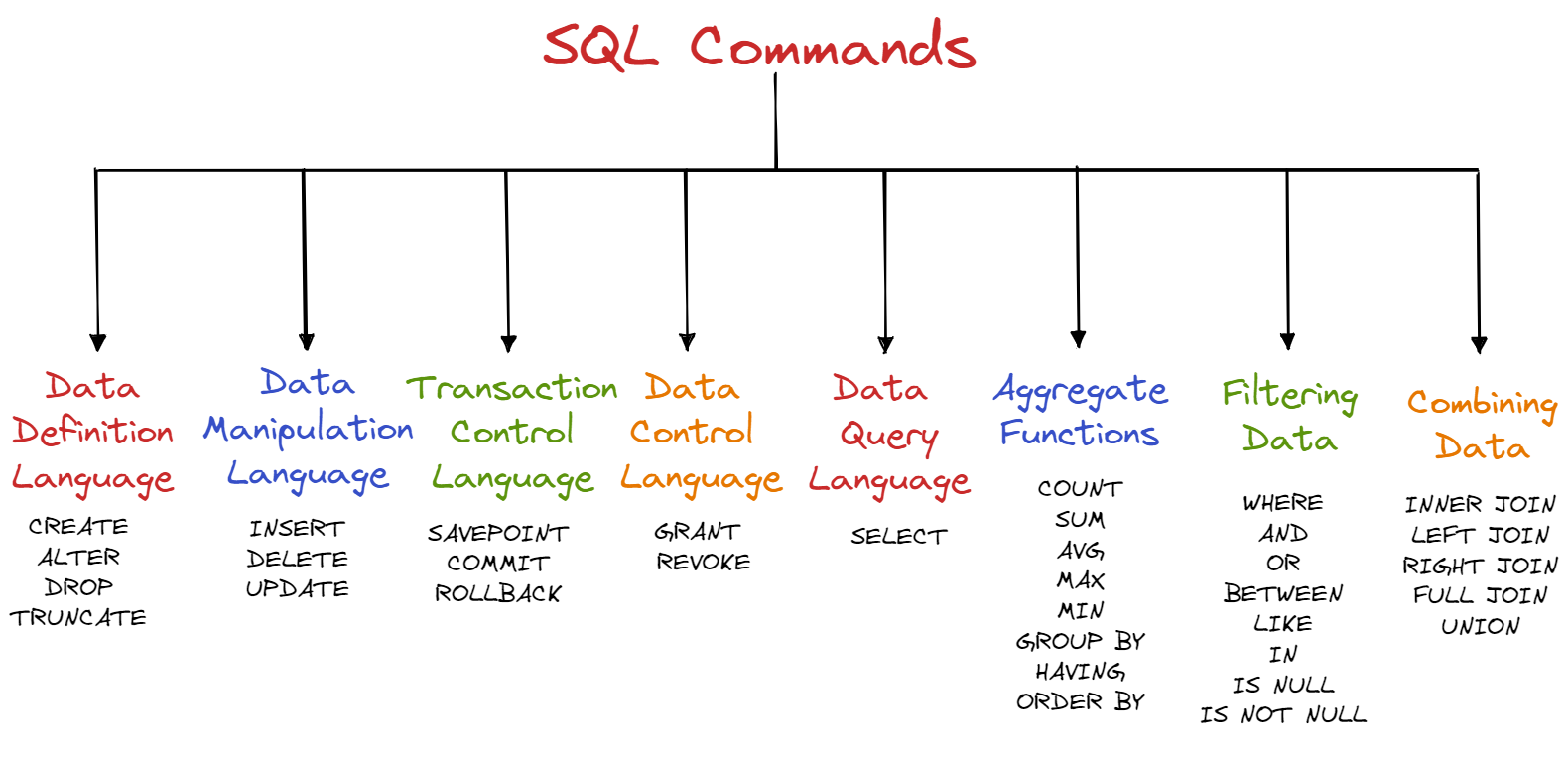
columns representing different data attributes and the various

relationships between the data values. You can use SQL statements

to store, update, remove, search, and retrieve information from the

database. You can also use SQL to maintain and optimize database

Performance.



**DDL COMMANDS**

**CREATE**

CREATE TABLE st1(REG\_NO VARCHAR(20) PRIMARY KEY,NAME VARCHAR(20) NOT NULL,MARKS INT);

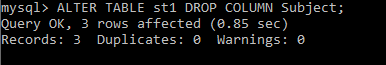


**ALTER**

**●** ALTER TABLE st1 ADD Subject varchar(50);



**●** ALTER TABLE student DROP COLUMN subject;



**●** ALTER TABLE st1 RENAME TO Student\_marks;



**TRUNCATE**

TRUNCATE TABLE student;

**RENAME**

● ALTER TABLE student RENAME COLUMN Email to EmailId;

● ALTER TABLE student RENAME TO studentsInformation;

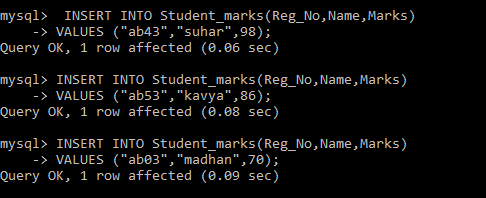
**DROP**

DROP TABLE student

**DML COMMANDS**

**INSERT**

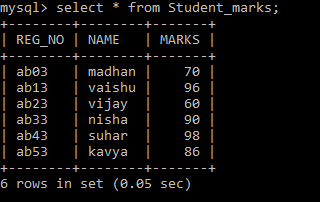
* INSERT INTO Student\_marks(Reg\_No,Name,Marks) VALUES (“ab13”,”vaishu”,96);
* INSERT INTO Student\_marks(Reg\_No,Name,Marks) VALUES (“ab23”,”vijay”,60); INSERT INTO Student\_marks(Reg\_No,Name,Marks) VALUES (“ab33”,”nisha”,76);
* INSERT INTO Student\_marks(Reg\_No,Name,Marks) VALUES (“ab43”,”suhar”,98);
* INSERT INTO Student\_marks(Reg\_No,Name,Marks) VALUES (“ab53”,”kavya”,86);
* INSERT INTO Student\_marks(Reg\_No,Name,Marks) VALUES (“ab03”,”madhan”,70);



**UPDATE**

UPDATE Student\_marks SET Marks= '90' WHERE Name= 'nisha';

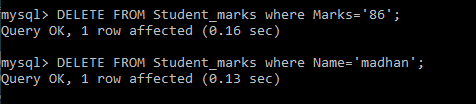




**DELETE**

● DELETE FROM Student\_marks where Marks=’86’;

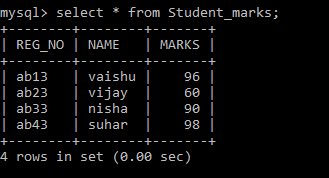
● DELETE FROM Student\_marks WHERE name= 'madhan';



**DRL COMMAND**

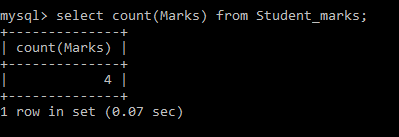
**SELECT**

● SELECT \* FROM student;

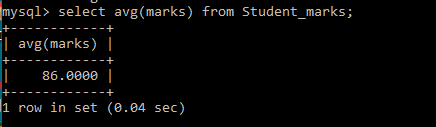


**AGGREGATE FUNCTIONS**

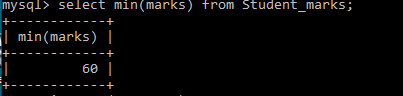
● SELECT COUNT (Marks) FROM Student\_marks;



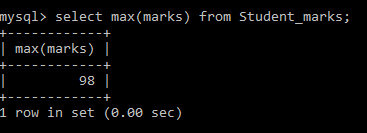
● SELECT AVG (Marks) FROM student;



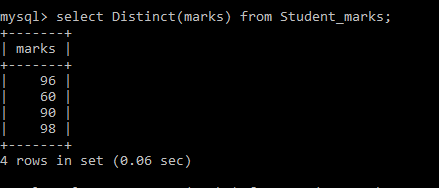
● SELECT MIN (Marks) FROM student;



● SELECT MAX (Marks) FROM student;



● SELECT DISTINCT (Marks) FROM student;



● SELECT COUNT\_BIG (Marks) FROM student;

**TCL COMMAND**

**COMMIT**

COMMIT TRANSACTION;**ROLLBACK**

ROLLBACK;



**SAVEPOINTS**

● BEGIN TRANSACTION;

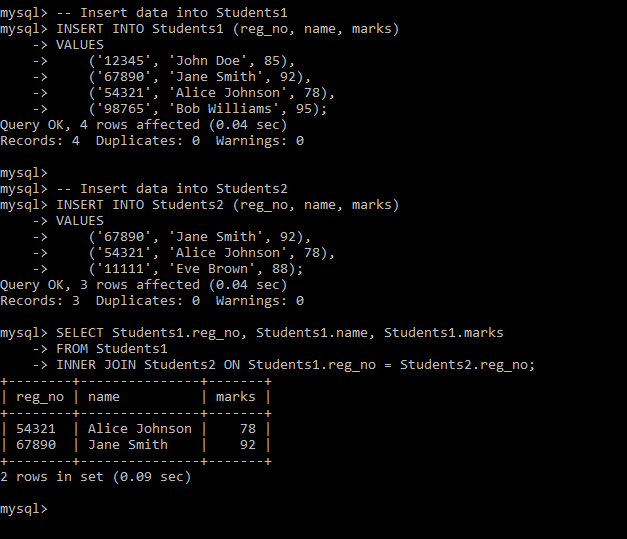
● SAVE TRANSACTION A;

● ROLLBACK TRANSACTION A;

**JOINS**

**INNER JOIN**

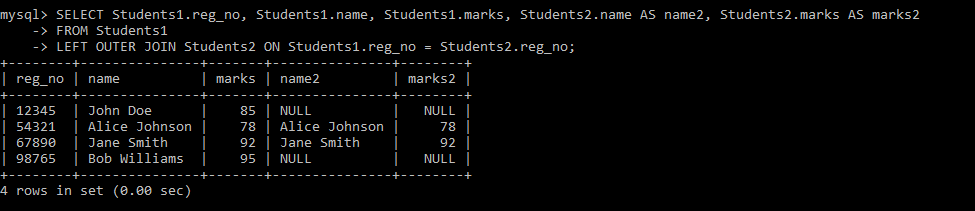
SELECT Students1.reg\_no, Students1.name, Students1.marks FROM Students1 INNER JOIN Students2 ON Students1.reg\_no = Students2.reg\_no;



**FULL OUTER JOIN**

;

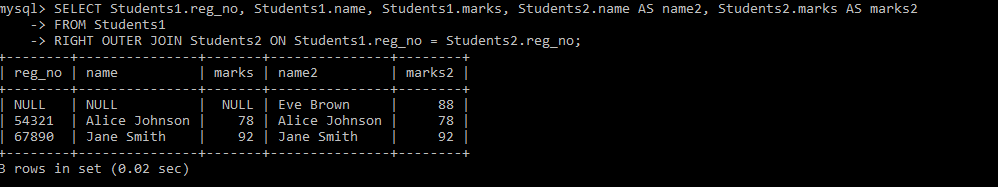
**LEFT JOIN**



**SELECT Students1.reg\_no, Students1.name, Students1.marks, Students2.name AS name2, Students2.marks AS marks2 FROM Students1 LEFT OUTER JOIN Students2 ON Students1.reg\_no = Students2.reg\_no;**

**RIGHT JOIN**

**SELECT Students1.reg\_no, Students1.name, Students1.marks, Students2.name AS name2, Students2.marks AS marks2 FROM Students1 RIGHT OUTER JOIN Students2 ON Students1.reg\_no = Students2.reg\_no;**



**SELF JOIN**

SELECT customer.customer\_id, customer.name, customer.order\_id,

order\_table.product

FROM customer, order\_table

WHERE customer.order\_id = order\_table.order\_id;

**CROSS JOIN**

SELECT \* from customer cross join order\_table;

DBMS VS RDBMS

|  |  |
| --- | --- |
| DBMS | RDBMS |
| Stores data as files | Stores data in the form as table |
| It support only one user | It support multi user |
| It does not support client server architecture | It support client-server architecture |
| Low software and hardware requirement | High software and hardware requirement |
| Data redundancy is common | Key and indices are not allow data redundancy |