SQL COMMANDS

➢ What is SQL?

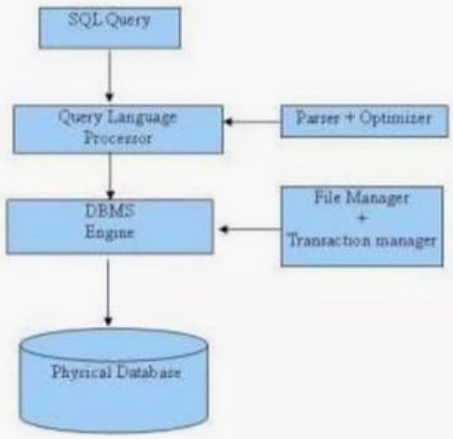
- **Structured Query Language** and it helps to make practice on SQL commands which provides immediate results.
- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.
- SQL is the standard language for Relation Database System.
- All relational database management systems like MySQL, MS Access, and Oracle, Sybase, Informix, and SQL Server use SQL as standard database language.

➢ Why SQL?

- Allows users to create and drop databases and tables.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows users to access data in relational database management systems.
- Allows embedding within other languages using SQL modules, libraries & pre-compilers.
- Allows users to set permissions on tables, procedures, and views

➢ SQL Architecture:

- When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.
- There are various components included in the process.
- These components are:
  - Query Dispatcher
  - Optimization Engines
  - Classic Query Engine
  - SQL Query Engine, etc.
- Classic query engine handles all non-SQL queries but SQL query engine won't handle logical files.
- Simple diagram showing SQL Architecture:



➢ SQL Commands:

- The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP.
- These commands can be classified into groups based on their nature:

➢ DDL - Data Definition Language:

- ***DDL defines the conceptual schema providing a link between the logical and the physical structure of the database.***
- The functions of the Data Definition Language (DDL) are:

1. DDL defines the physical characteristics of each record, filed in the record, field"s data type, field"s length, field"s logical name and also specify relationship among those records.
2. DDL describes the schema and subschema.
3. DDL indicate the keys of records.
4. DDL provides data security measures.
5. DDL provides for the logical and physical data independence.

- Few of the basic commands for DDL are:

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

➢ DML - Data Manipulation Language:

- ***DML provides the data manipulation techniques like selection, insertion, deletion, updation, modification, replacement, retrieval, sorting and display of data or records.***
- DML facilitates use of relationship between the records.
- DML provides for independence of programming languages by supporting several high-level programming languages like COBOL, PL/1 and C++.
- Few of the basic commands for DML are:

Command	Description
SELECT	Retrieves certain records from one or more tables
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

➢ DCL - Data Control Language:

- These SQL commands are used for providing security to database objects.
- The different DCL commands are:

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

➢ TCL – Transaction Control Language:

- It includes commands to control the transactions in a database system.
- The commonly used commands are:

Command	Description
COMMIT	Make all the changes made by the statements issued permanent.
ROLLBACK	Undoes all changes since the beginning of transaction or since a save point.

➢ Data Types in SQL:

The following are the most common data types of SQL:

SL No	DATA TYPE	DESCRIPTION
1	NUMBER	A variable-length column. Allowed values are zero, positive and negative numbers
2	CHAR	A variable length field up to 255 character in length
3	VARCHAR/VARCHAR2	A variable length field up to 2000 character in length
4	DATE/TIME	A fixed length field. The time is stored as a part of the date. The default format is DD/MON/YY
5	LONG	A variable length filed up to 2 GB in length
6	RAW	A variable length filed used for binary data up to 2000 in length
7	LONG RAW	A variable length filed used for binary data up to 2GB in length

1. NUMBER:

- Used to store a numeric value in a field column.
- It may be decimal, integer or real value.
- **General syntax**: **NUMBER(n, d)**
  Where **n** specifies the number of digits and **d** specifies the number of digits to right of the decimal point.
  Example: marks NUMBER(3), average NUMBER(2, 3)

2. CHAR:

- Used to store a character type data in a column.
- **General syntax:** **CHAR(size)** o Where size represents the maximum (255 Characters) number of characters in a column.
- **Example**: name CHAR(15)

3. VARCHAR/VARCHAR2:

- It is used to store variable length alphanumeric data.
- **General syntax:** **VARCHAR(size) / VARCHAR2(size)**

- Where size represents the maximum (2000 Characters) number of characters in a column.
- **Example**: address VARCHAR2(50)

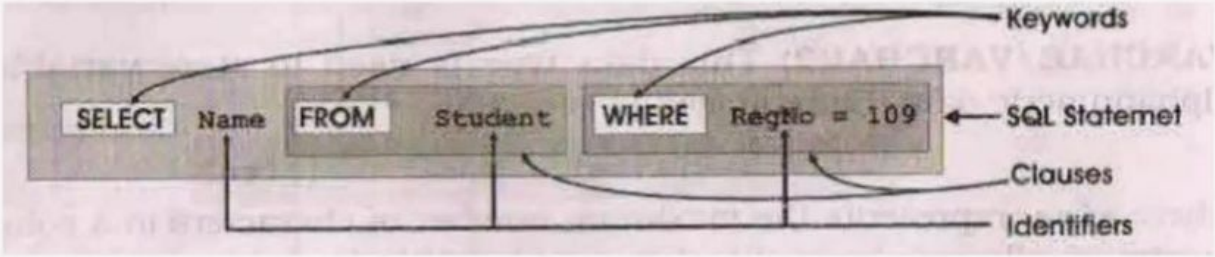
4. **DATE:**
   - It is used to store date in columns.
   - SQL supports the various date formats other than the standard D-MON-YY.
   - **Example**: dob DATE

5. **TIME:**
   - It is used to store time in columns.
   - SQL supports the various time formats other than the standard hh-mm-ss.
   - Every DATE and TIME can be added, subtracted or compared as it can be done with other data types.

6. **LONG:**
   1. It is used to store variable length strings of up to 2GB size.
   2. **Example**: description LONG

➢ Structure of SQL command:

- Any SQL command is a combination of keywords, identifiers and clauses.
- Every SQL command begins with a keyword (CREATE, SELECT, DELETE and so on) which as a specific meaning to the language.



- SELECT, FROM and WHERE are keywords.
- The clauses are "FROM student" and "WHERE RegNo=109".
- Here SELECT and FROM are mandatory, but WHERE is optional.
- Name, Student, RegNo, are identifier that refers to objects in the database.
- Name and RegNo are column names, while Student is a table name.
- The equal sign is an operator and 109 is a numeric constant.

➢ What is an Operator in SQL?

- An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations.
- Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement. ○ Arithmetic operators (+, -, *, / %) ○ Comparison operators (>, <, >=, <=, =, !=, <>, !<, !>)
- Logical operators (AND, OR, NOT, IN, BETWEEN, EXISTS, ALL, ANY, LIKE, UNIQUE)

➢ SQL Logical Operators:

Here is a list of all the logical operators available in SQL.

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.**
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS NULL	The NULL operator is used to compare a value with a NULL value.
UNIQUE	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

➢ Implementation of SQL Commands

✓ CREATE TABLE

- The SQL **CREATE TABLE** statement is used to create a new table.
- Creating a basic table involves naming the table and defining its columns and each column's data type.
- **Syntax**: Basic syntax of **CREATE TABLE** statement is as follows:

```
CREATE TABLE Table_name
(
   column1 datatype,
column2 datatype,    column3
datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);
```

- Here CREATE TABLE is the keyword followed by the Table_name, followed by an open parenthesis, followed by the column names and data types for that column, and followed by a closed parenthesis.
- For each column, a name and a data type must be specified and the column name must be a unique within the table definition.
- Column definitions are separated by commas (,).
- Uppercase and lowercase letters makes no difference in column names.
- Each table must have at least one column. ○ SQL commands should end with a semicolon (;).
- Example: Create a table "STUDENT" that contains five columns: RegNo, Name, Combination, DOB and Fees.

```
CREATE TABLE STUDENT
(
        RegNo           NUMBER (6),
        Name            VARCHAR2 (15),
        Combination     CHAR (4),
        DOB             DATE,
        Fees            NUMBER (9, 2),
        PRIMARY KEY ( RegNo )
);
```

- It creates an empty STUDENT table which looks like this:

RegNo	Name	Combination	DOB	Fees

- Viewing the table information:
  - The **DESCRIBE or DESC** command displays name of the columns, their data type and size along with the constraints.

```
SQL> DESCRIBE STUDENT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 REGNO                                     NOT NULL NUMBER(6)
 NAME                                               VARCHAR2(15)
 COMBINATION                                        CHAR(4)
 DOB                                                DATE
 FEES                                               NUMBER(4,2)
```

✓ ALTER Statement:

- The table can be modified or changed by using the ALTER command.
- Syntax: Basic syntax of ALTER TABLE statement is as follows:

1. ALTER TABLE Table_name
   ADD (column_name1 DataType, Cloumn_name2 DataType…);
2. ALTER TABLE Table_name
   MODIFY (column_name1 DataType, Cloumn_name2 DataType…);
3. ALTER TABLE Table_name
   DROP (column_name1 DataType, Cloumn_name2 DataType…);

- Example:

```
SQL> ALTER TABLE STUDENT ADD (Address VARCHAR2 (30));

Table altered.

SQL> ALTER TABLE STUDENT MODIFY (Address VARCHAR2 (40));

Table altered.

SQL> ALTER TABLE STUDENT DROP (ADDRESS);

Table altered.
```

- Using the ALTER TABLE command the following tasks cannot be performed
  - Changing a table name.
  - Changing the column name.
  - Decreasing the size of a column if table data exists.
  - Changing a column's data type.

✓ DROP TABLE:

- The SQL **DROP TABLE** statement is used to remove a table definition and all data, indexes, triggers, constraints, and permission specifications for that table. □ Syntax: Basic syntax of

- **DROP TABLE** statement is as follows:

```
DROP TABLE Table_name;
```

- Example:

```
SQL> DROP TABLE STUDENT;

Table dropped.
```

✓ INSERT:

- The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.
- Syntax:
- There are two basic syntaxes of INSERT INTO statement as follows:

```
INSERT INTO TABLE_NAME [(column1, column2, column3,...columnN)]
VALUES (value1, value2, value3,...valueN);
```

- Here, column1, column2,...columnN are the names of the columns in the table into which you want to insert data.
- You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.
- **METHOD 1:** The SQL INSERT INTO syntax would be as follows:

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

- Example: Following statements would create six records in STUDENT table:

```
SQL> INSERT INTO STUDENT VALUES(1401,'RAMESH','PCMC','07-AUG-99',14000);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1402,'JOHN','PCMB','15-SEP-99',13500);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1403,'GANESH','PCME','19-AUG-99',16000);
```

```
1 row created.
SQL> INSERT INTO STUDENT VALUES(1404,'MAHESH','PCMC','14-JAN-98',17650);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1405,'SURESH','PCMB','03-MAR-98',11500);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1410,'ARUN','PCMC','01-APR-04',13000);
```

- **METHOD 2:** The SQL INSERT INTO syntax would be as follows:

```
SQL> INSERT INTO STUDENT (REGNO, NAME, FEES) VALUES (1411, 'SHREYA',24000);
1 row created.
SQL> INSERT INTO STUDENT (REGNO, COMBINATION,FEES) VALUES(1412,
'PCMB',21000);
1 row created.
```

- All the above statements would produce the following records in STUDENT table:

```
SQL> SELECT * FROM STUDENT;

     REGNO NAME            COMB DOB              FEES
---------- --------------- ---- --------- ----------
      1401 RAMESH          PCMC 07-AUG-99      14000
      1402 JOHN            PCMB 15-SEP-99      13500
      1403 GANESH          PCME 19-AUG-99      16000
      1404 MAHESH          PCMC 14-JAN-98      17650
      1405 SURESH          PCMB 03-MAR-98      11500
      1410 ARUN            PCMC 01-APR-04      13000
      1411 SHREYA                              24000
      1412                 PCMB                21000

8 rows selected.
```

✓ UPDATE:

- SQL provides the ability to change data through UPDATE command.
- The UPDATE command used to modify or update an already existing row or rows of a table.
- The basic syntax of UPDATE command is given below.

```
UPDATE    Table_name
SET       column_name = value
          [, column_name =value …….]
[WHERE    condition];
```

Example:

```
SQL> UPDATE STUDENT SET COMBINATION='CEBA' WHERE REGNO=1411;
1 row updated.
```

```
SQL> UPDATE STUDENT SET NAME='AKASH' WHERE REGNO=1412;
1 row updated.
```

✓ DELETE command:

- In SQL, an already existing row or rows are removed from tables through the use of DELETE command.
- The basic syntax of DELETE command is given below.

```
DELETE   Table_name
 [WHERE  condition];
```

Example:

```
SQL> DELETE STUDENT WHERE REGNO=1412;
1 row deleted.
```

✓ SELECT:

- SQL **SELECT** statement is used to fetch the data from a database table which returns data in the form of result table. These result tables are called result-sets.
- Syntax: The basic syntax of SELECT statement is as follows:

SELECT column1, column2, columnN FROM Table_name;	**Compulsory Part**
[WHERE condition(s)] [GROUPBY column-list] [HAVING condition(s)] [ORDER BY column-name(s)];	Optional Part

- Here, column1, column2...are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax:

```
SELECT * FROM table_name;
```

- Example: Consider the STUDENT table having the following records:

```
SQL> SELECT * FROM STUDENT;

     REGNO NAME            COMB DOB              FEES
---------- --------------- ---- --------- ----------
      1401 RAMESH          PCMC 07-AUG-99      14000
      1402 JOHN            PCMB 15-SEP-99      13500
      1403 GANESH          PCME 19-AUG-99      16000
      1404 MAHESH          PCMC 14-JAN-98      17650
      1405 SURESH          PCMB 03-MAR-98      11500
      1410 ARUN            PCMC 01-APR-04      13000
      1411 SHREYA          CEBA                24000

7 rows selected.
```

- Following is an example, which would fetch REGNO, NAME and COMBINATION fields of the customers available in STUDENT table:

```
SQL> SELECT REGNO, NAME, COMBINATION FROM STUDENT;

     REGNO NAME            COMB
---------- --------------- ----
      1401 RAMESH          PCMC
      1402 JOHN            PCMB
      1403 GANESH          PCME
      1404 MAHESH          PCMC
      1405 SURESH          PCMB
      1410 ARUN            PCMC
      1411 SHREYA          CEBA

7 rows selected.
```

✓ DISTINCT:

- The SQL **DISTINCT** keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.

  There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.

- Syntax: The basic syntax of DISTINCT keyword to eliminate duplicate records is as follows:

```
SELECT DISTINCT    column1, column2,.....columnN
FROM               Table_name
WHERE              [condition]
```

- Example: Consider the STUDENT table having the following records:

```
SQL> select * from student;

     REGNO NAME            COMB DOB              FEES
---------- --------------- ---- --------- ----------
      1401 RAMESH          PCMC 07-AUG-99      14000
      1402 JOHN            PCMB 15-SEP-99      13500
      1403 GANESH          PCME 19-AUG-99      16000
      1404 MAHESH          PCMC 14-JAN-98      17650
      1405 SURESH          PCMB 03-MAR-98      11500
      1410 ARUN            PCMC 01-APR-04      13000
      1411 SHREYA          CEBA                24000

7 rows selected.
```

- First, let us see how the following SELECT query returns duplicate combination records:

```
SQL> SELECT COMBINATION FROM STUDENT ORDER BY COMBINATION;

COMB
----
CEBA
PCMB
PCMB
PCMC
PCMC
PCMC
PCME

7 rows selected.
```

- Now, let us use DISTINCT keyword with the above SELECT query and see the result:

```
SQL> SELECT DISTINCT COMBINATION FROM
STUDENT     ORDER BY COMBINATION;
```

- This would produce the following result where we do not have any duplicate entry: