SQL COMMANDS

➢ What is SQL?

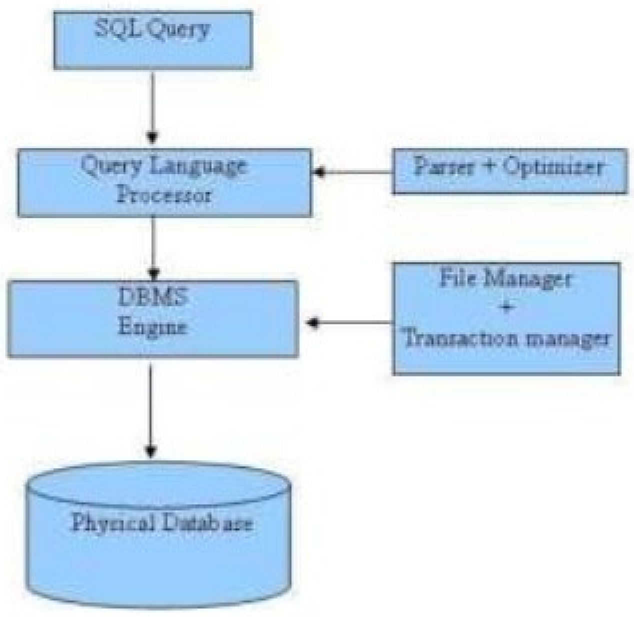
- **Structured Query Language** and it helps to make practice on SQL commands which provides immediate results.
- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.
- SQL is the standard language for Relation Database 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)** ○ 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:

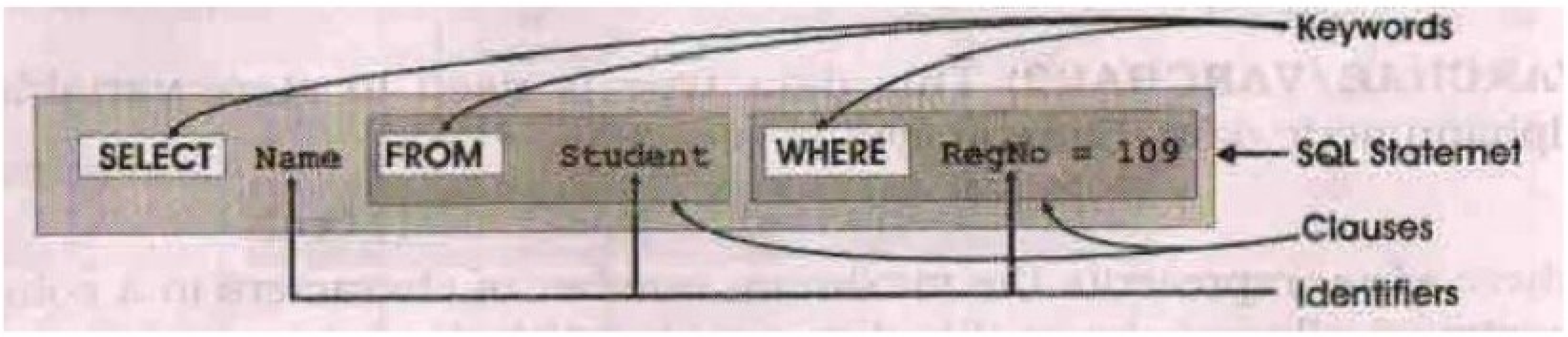
- It is used to store time in columns.
- SQL supports the various time formats other than the standard hh-mm-ss.
- Every DATE and TIME can be added, subtracted or compared as it can be done with other data types.

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. o Arithmetic operators (+, -, *, / %) o 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. o SQL commands should end with a semicolon (;).
- Example: Create a table "STUDENT" that contains five columns: RegNo, Name, Combination, DOB and Fees.

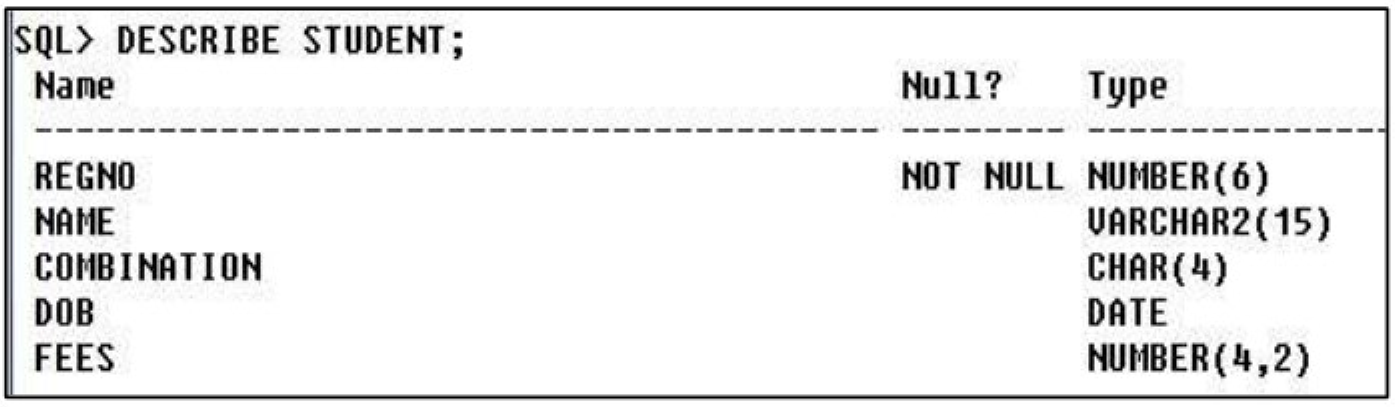
```
CREATE TABLE STUDENT
(
        RegNo           NUMBER (6),
        Name            VARCHAR2 (15),
        Combination     CHAR (4),
        DOB             DATE,
        Fees            NUMBER (9, 2),
        PRIMARY KEY ( RegNo )
);
```

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

RegNo	Name	Combination	DOB	Fees

▯ Viewing the table information:

o The **DESCRIBE or DESC** command displays name of the columns, their data type and size along with the constraints.



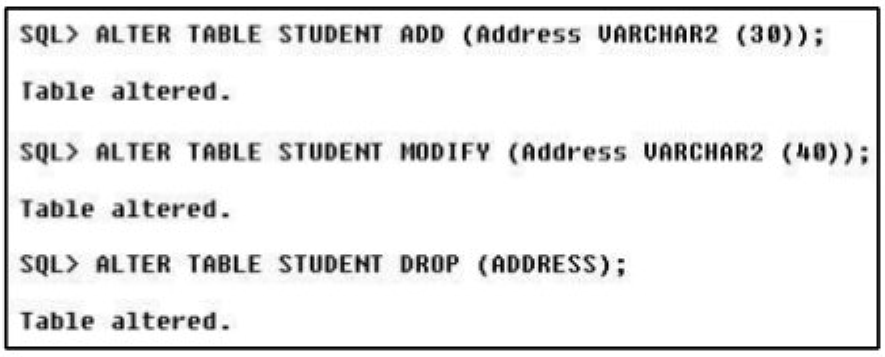
```
SQL> DESCRIBE STUDENT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------
 REGNO                                     NOT NULL NUMBER(6)
 NAME                                               VARCHAR2(15)
 COMBINATION                                        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.
```

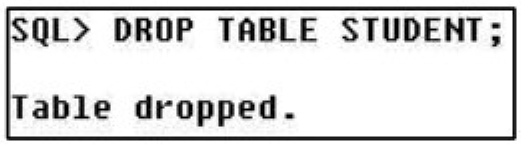
- Using the ALTER TABLE command the following tasks cannot be performed
    - Changing a table name.
    - Changing the column name.
    - Decreasing the size of a column if table data exists.
    - Changing a column's data type.

✓ 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);
```

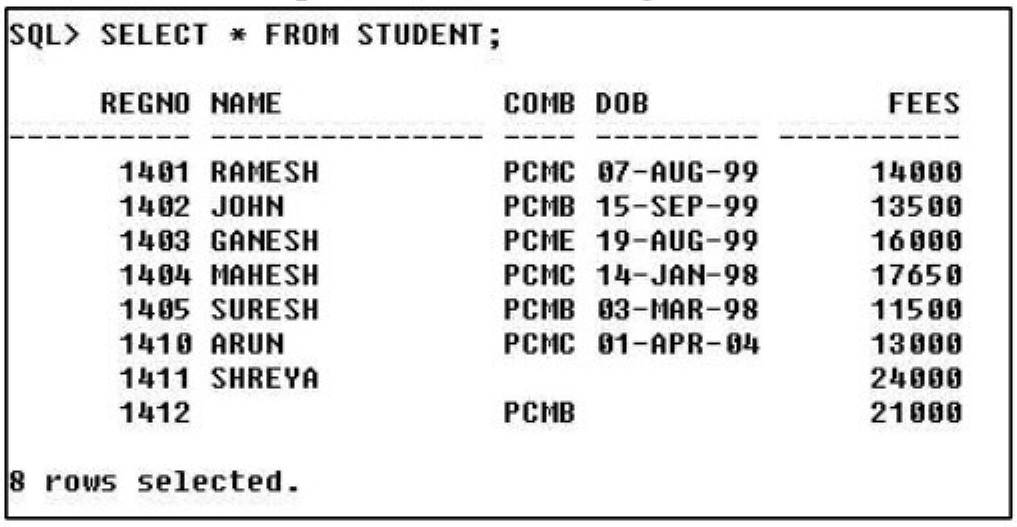
```
1 row created.
SQL> INSERT INTO STUDENT VALUES(1404,'MAHESH','PCMC','14-JAN-98',17650);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1405,'SURESH','PCMB','03-MAR-98',11500);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1410,'ARUN','PCMC','01-APR-04',13000);
```

- **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:

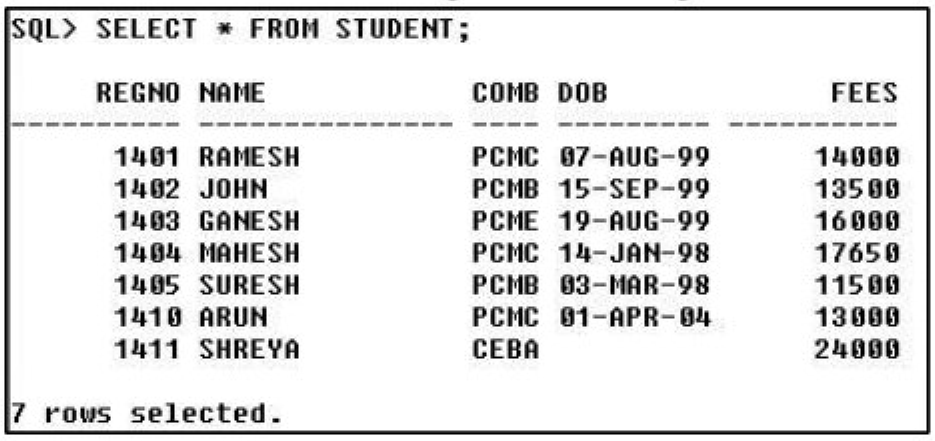
SELECT column1, column2, columnN
FROM Table_name;
	Compulsory Part

[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:

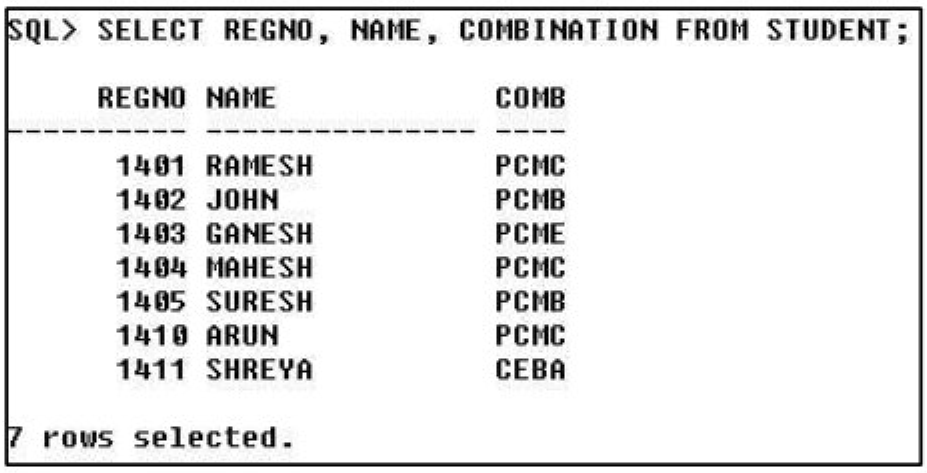
```
SELECT * FROM table_name;
```

- 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:



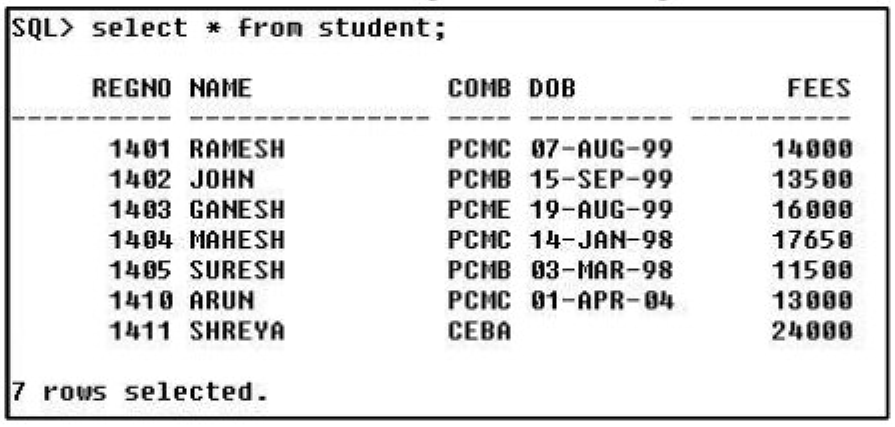
```
SQL> SELECT REGNO, NAME, COMBINATION FROM STUDENT;

     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:

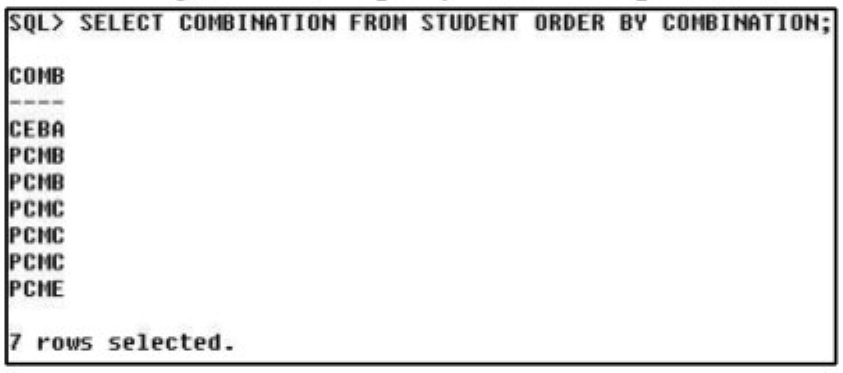
SELECT DISTINCT	column1, column2,.....columnN
FROM	Table_name
WHERE	[condition]

- 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:

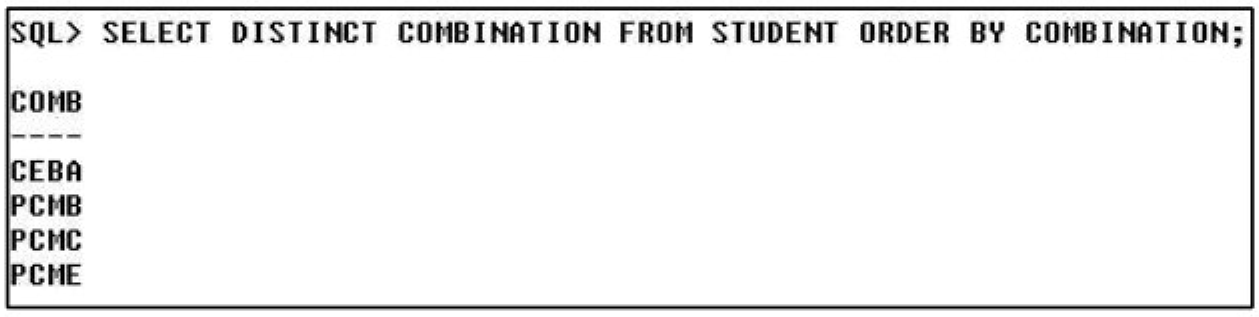


```
SQL> SELECT COMBINATION FROM STUDENT ORDER BY COMBINATION;

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:



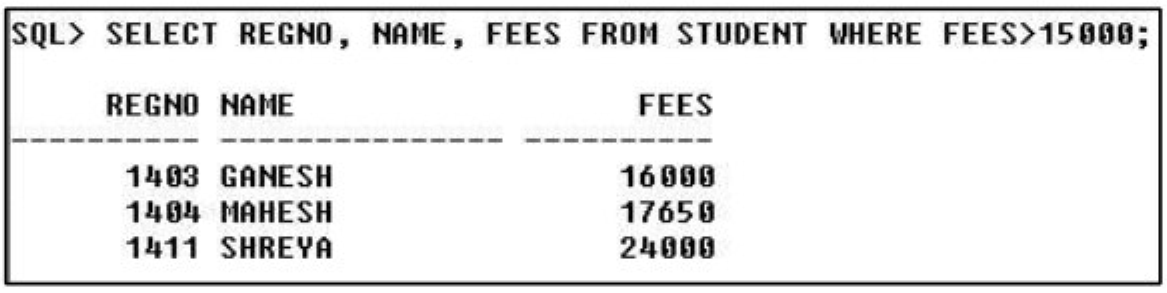
```
SQL> SELECT DISTINCT COMBINATION FROM STUDENT ORDER BY COMBINATION;

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

✓ WHERE clause – (Extracting specific rows)

- The SQL **WHERE** clause is used to specify a condition while fetching the data from single table or joining with multiple tables.
- If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.
  The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc., which we would examine in subsequent chapters.
- Syntax: The basic syntax of SELECT statement with WHERE clause is as follows:

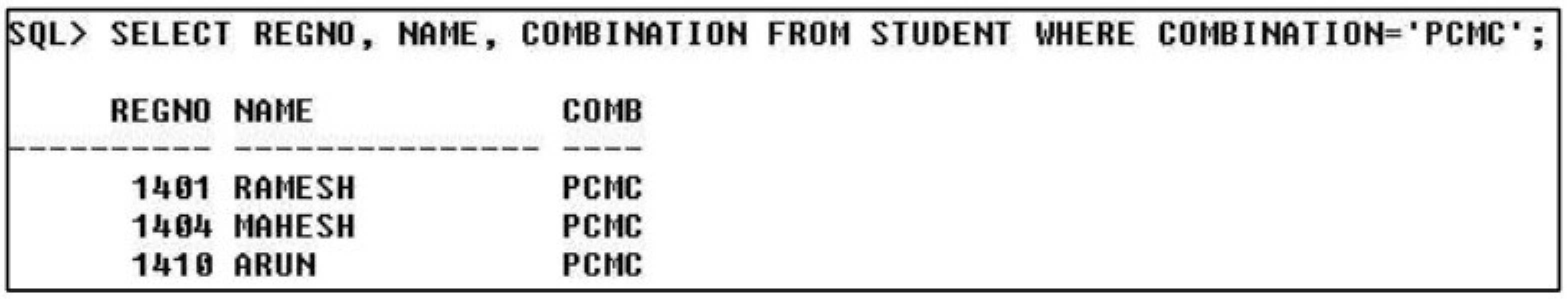
SELECT	column1, column2, columnN
FROM	Table_name
WHERE	[condition]

- You can specify a condition using comparison or logical operators like >, <, =, LIKE, NOT, etc.
- Following is an example which would fetch REGNO, NAME and FEES fields from the STUDENT table where FEES is greater than 15000:



```
SQL> SELECT REGNO, NAME, FEES FROM STUDENT WHERE FEES>15000;

     REGNO NAME                  FEES
---------- --------------- ----------
      1403 GANESH               16000
      1404 MAHESH               17650
      1411 SHREYA               24000
```

- Following is an example, which would fetch REGNO, NAME and COMBINATION fields from the STUDENT table for a COMBINATION is „PCMC".
- Here, it is important to note that all the strings should be given inside single quotes ('') where as numeric values should be given without any quote as in above example:



```
SQL> SELECT REGNO, NAME, COMBINATION FROM STUDENT WHERE COMBINATION='PCMC';

     REGNO NAME            COMB
---------- --------------- ----
      1401 RAMESH          PCMC
      1404 MAHESH          PCMC
      1410 ARUN            PCMC
```

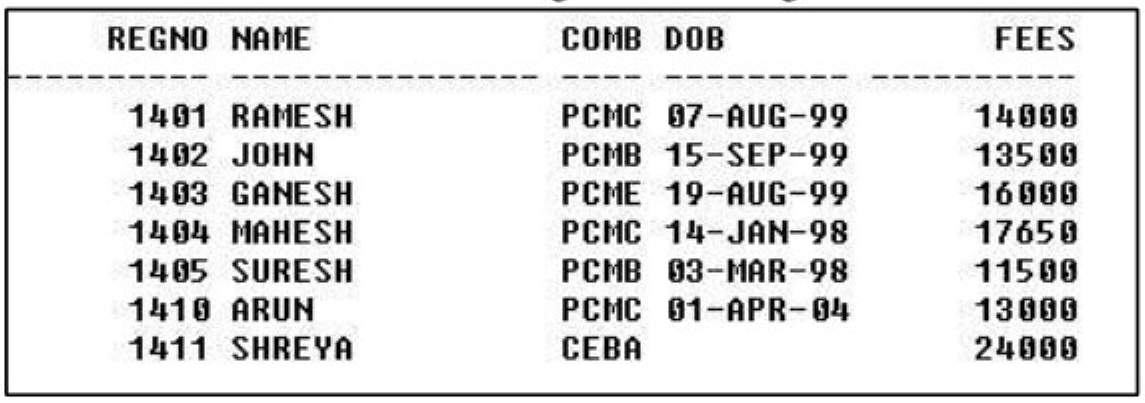
- The SQL **AND** and **OR** operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called ***conjunctive operators***.
- These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

✓ The AND Operator:

- The **AND** operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
- Syntax: The basic syntax of AND operator with WHERE clause is as follows:

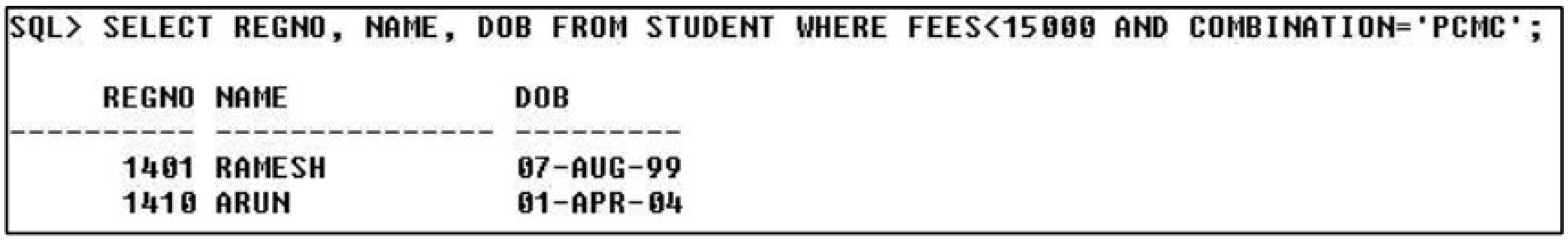
SELECT	column1, column2, columnN
FROM	Table_name
WHERE	[condition1] AND [condition2]...AND [conditionN];

  You can combine N number of conditions using AND operator. For an action to be taken by the SQL statement, whether it be a transaction or query, all conditions separated by the AND must be TRUE.
- Example: Consider the STUDENT table having the following records:



```
     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
```

- Following is an example, which would fetch REGNO, NAME and DOB fields from the STUDENT table where fees is less than 1500 AND combination is „PCMC:



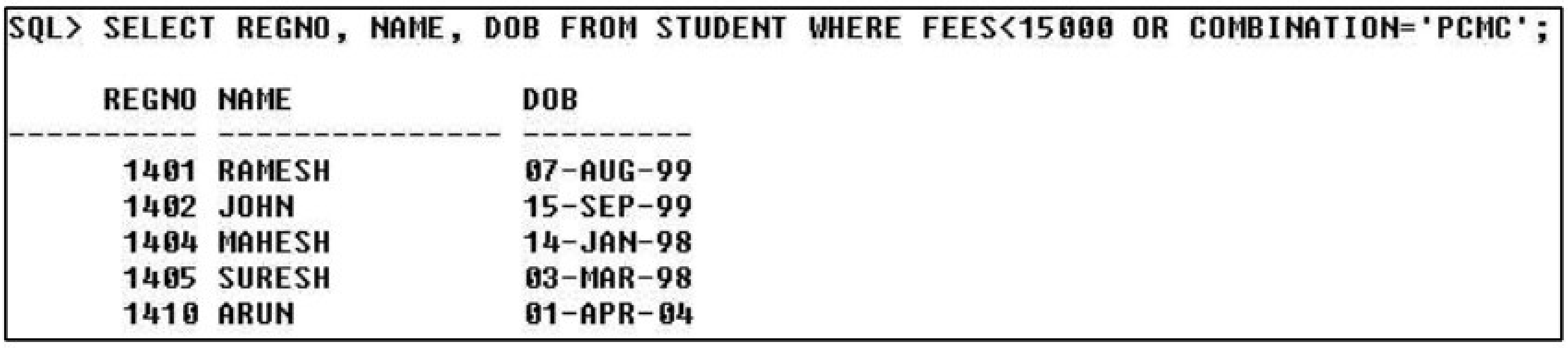
```
SQL> SELECT REGNO, NAME, DOB FROM STUDENT WHERE FEES<15000 AND COMBINATION='PCMC';

     REGNO NAME            DOB
---------- --------------- ---------
      1401 RAMESH          07-AUG-99
      1410 ARUN            01-APR-04
```

✓ The OR Operator:

- The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
- Syntax: The basic syntax of OR operator with WHERE clause is as follows:

SELECT	column1, column2, columnN
FROM	Table_name
WHERE	[condition1] OR [condition2]...OR [conditionN];

- You can combine N number of conditions using OR operator. For an action to be taken by the SQL statement, whether it be a transaction or query, only any ONE of the conditions separated by the OR must be TRUE.
- Following is an example, which would fetch REGNO, NAME and DOB fields from the STUDENT table where fees is less than 1500 OR combination is „PCMC:



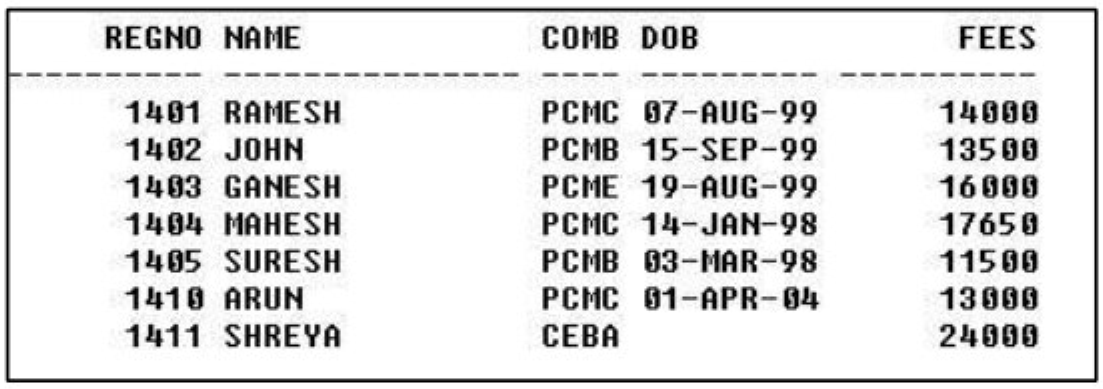
```
SQL> SELECT REGNO, NAME, DOB FROM STUDENT WHERE FEES<15000 OR COMBINATION='PCMC';

     REGNO NAME            DOB
---------- --------------- ---------
      1401 RAMESH          07-AUG-99
      1402 JOHN            15-SEP-99
      1404 MAHESH          14-JAN-98
      1405 SURESH          03-MAR-98
      1410 ARUN            01-APR-04
```

✓ ORDER BY – (Sorting the data)

- The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sorts query results in ascending order by default.
- Syntax: The basic syntax of ORDER BY clause is as follows:

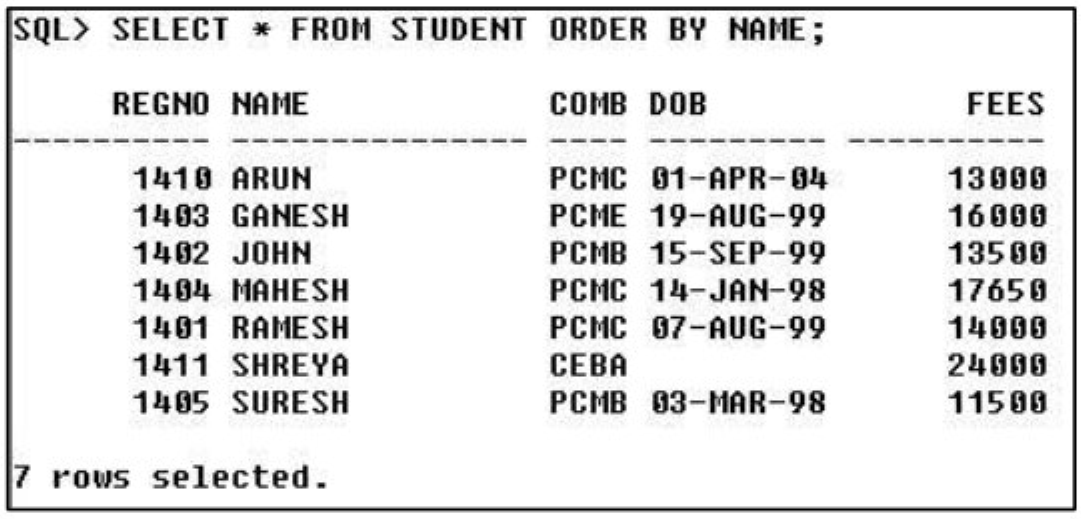
```
SELECT        column-list
FROM          Table_name
[WHERE        condition]
[ORDER BY     column1, column2, .. columnN] [ASC | DESC];
```

- You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort, that column should be in column-list.
- Example: Consider the STUDENT table having the following records:



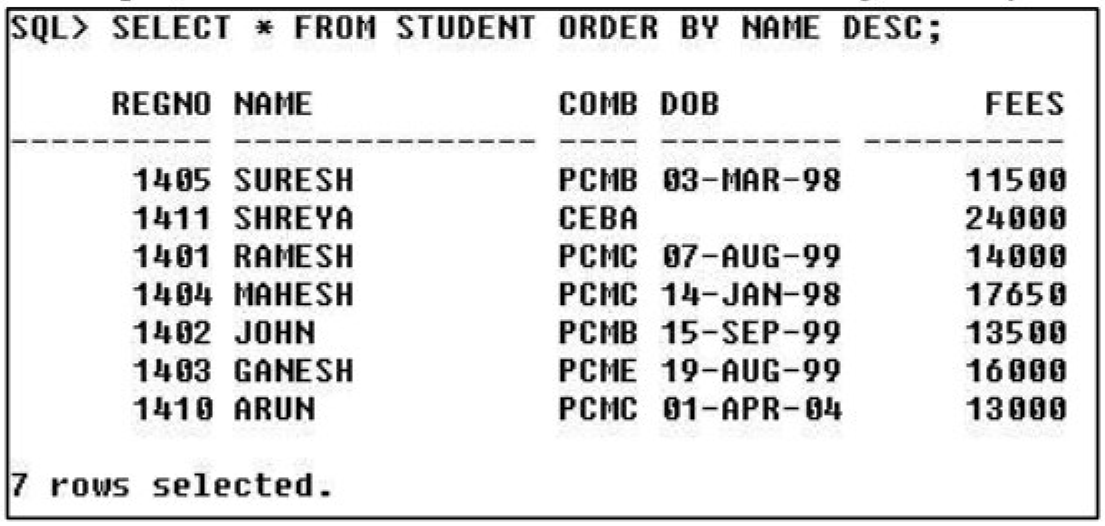
```
     REGNO NAME            COMB DOB             FEES
---------- --------------- ---- --------- ----------
      1401 RAMESH          PCMC 07-AUG-99      14000
      1402 JOHN            PCMB 15-SEP-99      13500
      1403 GANESH          PCME 19-AUG-99      16000
      1404 MAHESH          PCMC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

- Following is an example, which would sort the result in ascending order by NAME:



```
SQL> SELECT * FROM STUDENT ORDER BY NAME;

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

7 rows selected.
```

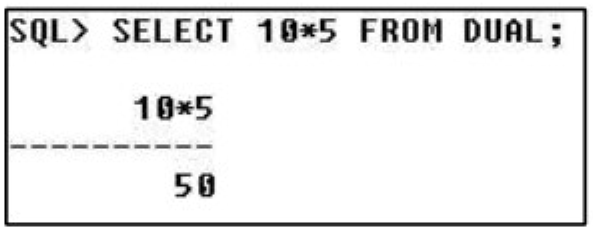
- Following is an example, which would sort the result in descending order by NAME:



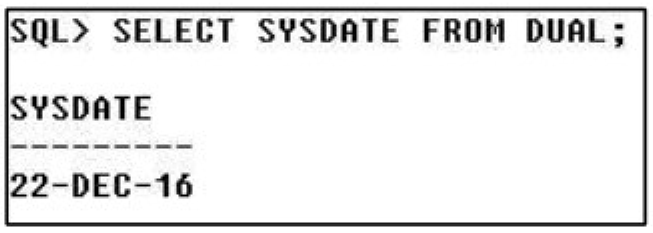
```
SQL> SELECT * FROM STUDENT ORDER BY NAME DESC;

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

7 rows selected.
```

✓ Working out simple calculations.

- Whenever we want to perform simple calculations such as 10 / 5, we can perform using SELECT statement which causes an output on monitor.
- But SELECT requires table name to operate.
- One can make use of the dummy table provided by SQL called DUAL which is a single row and single column table.
- It is used when data from table is not required.
- For example, when a calculation is to be performed such as 10*3, 10/2 etc. and to display the current system date, we could use the following queries.



```
SQL> SELECT 10*5 FROM DUAL;

      10*5
----------
        50
```



```
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
---------
22-DEC-16
```

✓ SQL Functions:

- The SQL functions serve the purpose of manipulating data items and returning a result.
- There are many built in functions included in SQL and can be classified as **Group Functions and Scalar Functions**.
- **Group Functions:** o Functions that act on set of values are called group functions.
    - o A group functions can takes entire column of data as its arguments and produces a single data item that summarizes the column.
    - o Following are the SQL group functions.

Function	Description
AVG	Returns average value of „N‟, ignoring NULL values
COUNT(expr)	Returns the number of rows where „expr‟ is not NULL
COUNT(*)	Returns the number of rows in the table including duplicates and those with NULL values
MIN	Returns minimum value of „expr‟
MAX	Returns maximum value of „expr‟
SUM	Returns sum of values „N‟

- **Scalar Functions:**
  - o Functions that act on only one value at a time are called scalar functions. o We can further classify the functions using the type of data with they are designed to work.

Function	Description
Numeric Functions	Work with numbers. Examples: ABS, POWER, ROUND, SQRT
String Functions	Work with character based data. Examples: LOWER, INITCAP, UPPER, SUBSTR, LENGTH, LTRIM, RTRIM
Date Functions	Work with Date data types. Example: ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN, NEXT_DAY
Conversion Functions	These functions are used to convert one type of data to another. Example: TO_NUMBER, TO_CHAR, TO_DATE

Consider the EXAMINATION table:

RegNo	Name	CC	Phy	Che	Mat	Cs	Total	City
101	Ajay	C1	98	100	97	99	394	Hassan
102	Banu	C2	38	50	37	49	174	Belur
103	Chandan	C2	100	100	97	99	396	Mysuru
104	John	C3	78	80	67	79	304	Alur
105	Kaleem	C1	88	80	91	79	338	Hassan
106	Raheem	C2	100	98	97	79	374	Hassan
107	Sanjay	C3	47	60	56	78	241	Alur
108	Tarun	C3	33	34	77	28	172	Arasikere
109	Uday	C2	100	98	97	79	374	Hassan
110	Venki	C3	47	60	56	78	241	Belur

✓ COUNT ( ) Function:

- This function is used to count the number of values in a column.
- COUNT (*) is used to count the number of rows in the table including duplicates and those with NULL values.
- Example 1:
  - SELECT COUNT (*) FROM EXAMINATION;
  - The above query returns 10.
- Example 2:
  - SELECT COUNT (RegNo) FROM EXAMINATION WHERE CC = „C3‟ ;
  - The above query returns 4.

✓ AVG ( ) Function:

- This function is used to find the average of the values in a numeric column.
- Example 1:
    - SELECT AVG (Cs) FROM EXAMINATION;
    - The above query returns 74.7

✓ SUM ( ) Function:

- This function is used to find the sum of the values in a numeric column.
- Example:
    - SELECT SUM (Phy) FROM EXAMINATION;
    - The above query returns 729 ✓ **MAX ( ) Function:**
- This function is used to find the maximum values in a column.
- Example:
    - SELECT MAX (Phy) FROM EXAMINATION;
    - The above query returns 100

✓ MIN ( ) Function:

- This function is used to find the minimum values in a column.
- Example:
    - SELECT MIN (Phy) FROM EXAMINATION;
    - The above query returns 33

✓ GROUP BY (Grouping Result)

- The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups.
- The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
- **Syntax**: The basic syntax of GROUP BY clause is given below.

```
SELECT      column1, column2
FROM        Table_name
WHERE       [ conditions ]
GROUP BY    column1, column2
ORDER BY    column1, column2
```

- Example 1: To find the number of students in each college.

```
SELECT          CC, COUNT (CC)
FROM            EXAMINATION
GROUP BY        CC;
```

- Example 2: To find the number of students, sum, average, maximum, minimum marks in computer science from each city.

```
SELECT          City, COUNT (City), SUM (Cs), AVG (Cs), MAX (Cs), MIN (Cs)
FROM            EXAMINATION GROUP
BY       City;
```

➢ SQL CONSTRAINTS:

- Constraints are the rules enforced on data columns on table.
- These are limiting the type of data that can go into a table.
- This ensures the accuracy and reliability of the data into the database.
- SQL allows two types of constraints.
    - **Column level constraints**: These constraints are defined along with the column definition when creating or altering a table structure. These constraints apply only to individual columns.
    - **Table level constraints**: These constraints are defined after all the table columns when creating or altering a table structure. These constraints apply to groups of one or more columns.
- Following are the commonly used constraints available in SQL.

Constraints	Description
NOT NULL	Ensures that a column cannot have NULL value
UNIQUE	Ensures that all values in column are different
PRIMARY KEY	Uniquely identified eac row in a database table.
FOREIGN KEY	Uniquely identified each rown in any other database table
DEFAULT	Provides a default value for a column when none is specified
CHECK	Ensures that all values in a column satisfy certain condition.

✓ NOT NULL Constraint:

- By default column can hold NULL values.
- When a column is defined as NOT NULL then the column becomes a mandatory column.
- It implies that a value must be entered into the column if the row is to be inserted for storage in the table.
- Example: Consider the following CREATE TABLE command creates a new table called PRODUCT and add six columns, two which PID and Description specify not to accept NULLs.

```
CREATE TABLE PRODUCT
(
          PID            CHAR (4)          NOT NULL,
          Description    VARCHAR2 (25), NOT NULL
          CompanyId      CHAR (10),
          DOM            DATE,
          Type           CHAR (10),
          Price          NUMBER (10,2)
);
```

✓ UNIQUE Constraints:

- This constraint ensures that no rows have the same value in the specified column(s). A table must have many unique keys.
- Example: UNIQUE constraint applied on PID of PRODUCT table ensures that no rows have the same PID value, as shown below

```
CREATE TABLE PRODUCT
(
          PID               CHAR (4)          NOT NULL UNIQUE,
          Description       VARCHAR2 (25), NOT NULL
          CompanyId         CHAR (10),
          DOM               DATE,
          Type              CHAR (10),
          Price             NUMBER (10,2)
);
```

✓ PRIMARY KEY Constraints:

- A primary key is a field which uniquely identifies each row in a database table. A primary key in a table has special attributes:
- By default this column is NOT NULL. It defines the column as a mandatory column i.e. the column cannot be left blank.
- The data held in this column must be unique.
- Example:

```
CREATE TABLE PRODUCT
(
          PID               CHAR (4)          PRIMARY KEY,
          Description       VARCHAR2 (25), NOT NULL
          CompanyId         CHAR (10),
          DOM               DATE,
          Type              CHAR (10),
          Price             NUMBER (10,2)
);
```

✓ FOREIGN KEY Constraint:

- A FOREIGN KEY is used to link two tables together.
- A foreign key is a column whose values are derived from the PRIMARY KEY of some other table.
- Example:

```
CREATE TABLE PRODUCT
(
          PID               CHAR (4)          PRIMARY KEY,
          Description       VARCHAR2 (25), NOT NULL
          CompanyId         CHAR (10) REFERENCES COMPANY (CID)
          DOM               DATE,
          Type              CHAR (10),
          Price             NUMBER (10,2)
);
CREATE TABLE COMPANY
(
          CID               CHAR (10) PRIMARY KEY,
          CProfile          VARCHAR2 (200),
          Noofproducts      NUMBER (20),
    DOE            DATE
);
```

✓ DEFAULT Constraints:

- A default value can be specified for a column using the DEFAULT clause.
- The DEFAULT constraint provides a default value to a column when the INSERT INTO command does not provide a specific value.
- Example:

```
CREATE TABLE PRODUCT
(
          PID             CHAR (4)          PRIMARY KEY,
          Description     VARCHAR2 (25), NOT NULL
          CompanyId       CHAR (10),
          DOM             DATE,
          Type            CHAR (10),
          Price           NUMBER (10, 2) DEFALUT 1000.00
);
```

✓ CHECK Constraints:

- The CHECK Constraint is used to establish a TRUE/FALSE condition that is applied to the data placed in a column.
- If a value does not meet the condition, it cannot be placed in the column.
- Example:

```
CREATE TABLE PRODUCT
(
          PID             CHAR (4)          CHECK (PID LIKE ‘P%’),
          Description     VARCHAR2 (25),
          CompanyId       CHAR (10),
          DOM             DATE,
          Type            CHAR (10),
          Price           NUMBER (10, 2) CHECK (Price>0)
);
```

✓ TABLE Constraints:

- When a constraint is applied to a group of columns of the table, it is called a table constraint.
- Column constraint is defined along with the end of the column.
- Table constraints are defined at the end of the table.
- Example:

```
CREATE TABLE PRODUCT
(
          PID             CHAR (4)          NOT NULL,
          Description     VARCHAR2 (25) NOT NULL,
          CompanyId       CHAR (10),
          DOM             DATE,
          Type            CHAR (10),
          Price           NUMBER (10, 2),
          PRIMARY KEY (PID, Description)
);
```

➢ Joins

- The SQL **Joins** clause are used to fetch/retrieve data from two or more tables based on the join condition which is specified in the WHERE condition.
- Basically data tables are related to each other with keys. We can used these keys relationship in SQL joins.

✓ SQL Join Types:

There are different types of joins available in SQL:

- o **INNER JOIN**: returns rows when there is a match in both tables.
- o **LEFT JOIN**: returns all rows from the left table, even if there are no matches in the right table.
- o **RIGHT JOIN**: returns all rows from the right table, even if there are no matches in the left table.
- o **FULL JOIN**: returns rows when there is a match in one of the tables.
- o **SELF JOIN**: is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- o **CARTESIAN JOIN**: returns the Cartesian product of the sets of records from the two or more joined tables.

➢ Creating VIEWs:

- Database Views are created using the CREATE VIEW statement.
- Views can be created from a single table, multiple tables or another view.
- To create a view, a user must have the appropriate system privilege according to the specific implementation.
- Syntax: The basic CREATE VIEW syntax is as follows:

```
SQL> CREATE VIEW   view_name AS
       SELECT      column1, column2…..
      FROM         table_name
      WHERE        [ condition ];
```

➢ Privileges and Roles:

- Privileges: Privileges defines the access rights provided to a user on a database object.
- There are two types of privileges:
  - o **System Privileges**: This allows the user to CREATE, ALTER, or DROP database objects.
  - o **Object privileges**: This allows the user to EXECUTE, SELECT, INSERT, UPDATE or DELETE data from database objects to which privileges apply.

CHAPTER 14 – STRUCTURED QUERY LANGUAGE BLUE PRINT				
VSA (1 marks)	SA (2 marks)	LA (3 Marks)	Essay (5 Marks)	Total
-	01 Question	-	01 Question	02 Question
-	Question no 17	-	Question no 36	06 Marks

Important Questions Two

Marks Questions:

1. Give the syntax and example for DELETE command in SQL. [March 2015]
2. List the data types supported in SQL. [June 2015]
3. Write the syntax for DELETE and INSERT commands in SQL. [March 2016]
4. Mention the logical operators used in SQL. [June 2016]
5. Give the syntax and example of UPDATE command in SQL. [March 2017]
6. What is the difference between ORDER BY and GROUP BY clause used in SQL? Give example for each. [June 2017]
7. List the relational operators supported by SQL.
8. Why the DROP command used? Write its syntax.
9. What are the difference between DROP and DELETE command?
10. Give the syntax and example for CREATE VIEW command in SQL.
11. Classify the built-in functions in SQL.

Five Marks Question:

1. Explain various group functions in SQL. [March 2015, March 2017, June 2017]
2. What is data definition language? Explain SELECT and UPDATE command. [June 2015]
3. Describe any five logical operators available in SQL. [March 2016]
4. What is SQL? Explain the different SQL commands.
5. What is the purpose of CREATE command? Write the syntax and example of CREATE command.
6. Explain SELECT statement with syntax and with minimum 3 examples.
7. Explain 5 variations of SELECT command.
8. What are SQL constraints? Explain any two constraints.

****