Introduction to SQL (Structured Query Language):-

. Structured Query Language (SQL) is a language that provides an interface to relational database systems. SQL was developed by IBM in the 1970s for use in system R and as well as in ISO and ANSI standard. SQL is often pronounced **SEQUEL.**

In common usage SQL also encompasses DML(Data Manipulation Language), DDL(Data Definition Language),DCL(Data Control Language and DQL(Data Query Language)

Advantages of SQL

1. SQL can be used by a range of users, including those with little or no programming experience.
2. It is a non procedural language.
3. It reduces the amount of time required for creating and maintaining systems.
4. It is an English-like language.

Disadvantages of SQL

1. No programming facility to handle errors.
2. One SQL statement can be passing to the oracle engine at a time.

It is an object so it occupies space

Components of SQL:

(1) DDL (Data Definition Language)

It is a set of SQL commands used to create, Modify and Delete database Structures but no data.

Examples: Create, Alter, Drop, Truncate, Grant, Revoke, Comment

(2) DML (Data Manipulation Language)

It is an area of SQL that allows changing data within the database.

Examples: Insert, Update, Delete, Lock

(3) DCL (Data Control Language)

and TCL (Transaction Control Language)

It is the components of SQL statements that control access to data and to the database. DCL statements are grouped with DML statements.

Examples: Commit, Rollback, Savepoint, Set Transaction

(4) DQL (Data Query Language)

It is the components of SQL statements that allows getting data from the database.

Examples: Select

Oracle Data Types:-

Basic Data types:

Data type	Description
CHAR(size)	▪ This data type is used to store character strings value of **fixed length.** ▪ The size in brackets determines the number of characters the cell can hold. ▪ The maximum no. of characters this data type can hold up to **255** characters. ▪ This data type the data held in right padded with spaces to whether length specifies.
VARCHAR(size)/ VARCHAR2(size)	▪ This data type is used to store **variable length** alphanumeric data. ▪ It is more flexible form of character data type. ▪ The maximum this data type can hold up to **4000 characters.** ▪ One difference between char and varchar data type values using non-padded semantic. i.e. The inserted values will not be padded with spaces.
DATE	▪ This data type is used to represent date and time. ▪ The standard format is **dd-mon-yy** as in 12-may-83. ▪ Date time stores date in the format 24 hour formats.
NUMBER(P,S)	▪ The Number data type is used to **store numbers.** ▪ Numbers of any magnitude may be stored up to 38 digit of precession. ▪ Valid values are Zero, Positive and Negative. ▪ The Precession (P) determines the maximum length of data. ▪ The Scale(S) determines the number of places to the right of the decimal. If the scale is omitted the default is zero.
LONG	▪ This data type is used to store variable length character strings up to **2 GB.** ▪ Long data can be used to store array of binary data in **ASCII format.** ▪ Only one LONG value can be defined per table.
RAW/ LONG RAW	▪ This data type is used to store binary data, such as digitized picture or image. ▪ RAW data type can have maximum length of 255 **bytes**. ▪ LONG RAW data type can contain up to **2 GB.**

Create Table Command :

- The CREATE TABLE command defined each column of the table uniquely.
- Each column has a minimum of three attributes, a name, data type and size (i.e column width)
- Each table column definition is a single clause in the create table syntax.
- Each table column definition is separated from the other by comma.
- Finally, the SQL statement is terminated with a semi colon (;).

Rules for Creating Tables :-

i. A name can have maximum up to 30 characters.
ii. Alphabets from A-Z, a-z and number from 0-9 are allowed.
iii. A name should begin with an alphabet.
iv. The use of special character like _ (Underscore) is allowed and also is recommended. (Special characters like $, # are allowed in only Oracle)
v. SQL reserved words not allowed. For example create, select, insert and so on.

Syntax :	**Example:**
CREATE TABLE <TableName> **(** <ColumnName1> <datatype> (<size>), <ColumnName2> <datatype> (<size>), -- <ColumnNameN> <datatype> (<size>) **);**	**CREATE TABLE** emp **(** Empno number(5), Ename char(20), Sal number(7,2) **);** SQL> Table Created.

Each Column must have data type. The column should either be defined as null or not null and if the value is left blank the database assumes "NULL" as default.

Creating a Table from a Table :-

Syntax:	**Example:-**
Create table <TableName> (<ColumnName1>,<ColumnName2>) **AS** **Select** <ColumnName1>,<ColumnName2> form <TableName>;	**Create table** test1 (eno,empnm) **AS** **Select** Empno,Ename from emp;

i.e.: Create a new table test1 having two fields like eno, empnm from the source table named emp and rename the field Empno to eno and Ename to empnm.

If the source table emp was populated with records then the target table test1 will also be populated with the same.

Inserting Data into Tables:-

Insert Statement :-

Once table is created the most natural thing to do is load this table with data to be manipulated later. When inserting a single row of data into the table, the insert operation

- Create a new row in the database table
- Loads the value passed into the column specifies.

Syntax 1:	Example :

INSERT INTO <TableName>
VALUES
(<expression1>,<expression2)**;**
	INSERT INTO emp
	VALUES
	(22,'Ashish',20000)**;**

Syntax 2: (Using Substitute Variable)	**Example : (Using Substitute Variable)**

INSERT INTO <TableName>
VALUES
(<&ColumnName1>,<&ColumnName2>)**;**
	INSERT INTO emp
	VALUES
	(&eno,'&ename',&sal)**;**

Syntax 3 :	**Example :**

INSERT INTO <TableName>
(<columnName1>,<ColumnName2>)
VALUES
(<expression1>,<expression2);
	INSERT INTO emp
	(Empno,Ename,Sal)
	VALUES
	(11,'BHavesh',10000);

Description:

- Character expression placed within the insert into statement must be enclosed in a single quotes (').
- In the INSER INTO sql sentence table columns and value have a one to one relationship.
- In an INSERT INTO statement, if there are exactly the same numbers of values as there are columns.
- The values are sequenced in exactly in accordance with the data type of the table column.

Viewing Data in the Tables.:-

Que: - Explain Select statement with syntax and proper example.
(How to viewing or filtering a data in oracle? Explain in brief.

Select Statement:-

Once data has been inserted into tables, the next most logical operation would be to view what has been inserted.

The Select command is used to retrieve rows selected from one or more tables.

➢ All Rows and All Columns.

Syntax:-	**Example:**
Select <ColumnName1> to ColumnNameN> From <TableName>; i.e. **Select** * from **<TableName>;**	**Select** * from **emp;**

ColumnName1 to ColumnNameN represent all table column Names.

❖ FILTERING THE DATA:-

The ways of filtering table data are:

- Selected Columns and All Rows
- Selected Rows and All Columns.
- Selected Columns and Selected Rows.

➢ Selected Columns and All Rows :-

The retrieval of specific columns from a table can be done as shown below.

Syntax:-	**Example:-**
Select <ColumnName1>,<ColumnName2> **From** <TableName>;	**Select** Empno,Ename **from** emp;

➢ Selected Rows and All Columns :-

Oracle Provides the option of using a WHERE clause in an SQL query to apply a filter on the rows retrieved and it must be based on specific condition.

When Specifying a condition in the WHERE clause all standard operator such as Logical, Arithmetic and Predicates can be used.

Syntax:-	**Example:-**
Select * From <TableName> **WHERE** <Condition>;	**Select** * from emp **Where** empno=11;

➢ Selected Columns and Selected Rows :-

To view specific set of rows and columns from a table.

Syntax:-	**Example:-**
Select <ColumnName1>,<ColumnName2> From <TableName> **WHERE** <Condition>;	**Select** Empno,Ename from emp **Where** Sal>=10000;

Eliminating Duplicate Rows Using a Select Statement:-

Distinct Clause:-

- A table could hold duplicate rows. In such case to view only unique rows the **DICTINCT** clause can be used.
- The DISTICT clause allows removing duplicates from the result set. The DISTINCT clause can only be used with select statement.
- The DISTINCT clause scans through the values of the column/s specified and display only unique values from amongst them.

Syntax:	**Example:-**
Select **DISTINCT** <ColumnName1>, <ColumnName2> from <TableName>;	Select **DISTINCT** Empno from emp;

In example, removes duplicate values of employee number from employee table

.

Sorting Data in a Table:-

Order By Clause:-

- Oracle allows data from a table to be viewed in a sorted order. The rows retrieved from the table will be sorted in either ascending or descending order depending on the condition specified in the Select statement.
- The Order By clause sorts the result set based on the column specified.
- The Order By clause can only be used in SELECT statement.

Syntax:-	**Example:-**

Select * from <TableName>
Order By <ColumnName1>,<ColumnName2>
< [sort order]> ;
	(i) **Select** * form emp **Order by** ename;
	(ii) Select * from emp
	order by ename **desc;**

- For Viewing the data in descending sorted order the word **DESC** must be mentioned after column name and before the semi colon (;) in the order by clause.

- There is no mention of the sort order, the oracle engine sorts in ascending order by default.

- In the ORDER By clause The ColumnName2 is followed by ColumnName1.

- In second example views all the data in the descending order by ename.

Delete Statement : -

- Delete commands deletes rows from the table that satisfies the condition provided by its WHERE clause and returns the number of records deleted.
- If a delete statement without WHERE clause is issued then, all rows are deleted.

- The verb DELETE in SQL is used to remove either:
  - **All rows from a table.**

  OR
  - **A Set of rows from a table.**

➢ **Removal of All Rows :-**

Syntax1:-	**Example:-**
Delete from <TableName>;	**Delete** from emp; i.e. Delete all rows from employee table.

➢ **Removal of Specific Row(s).**

Syntax2:-	**Example:-**
Delete from <TableName> **Where** <Condition>;	**Delete** from emp **where** eno=22; i.e. Delete all rows from employee table whose empno is 22..

Updating (Changing) the content of a Table :-

Que: - How to change the content of a table? Explain with syntax and proper example.

Update Statement:-

- The Update command is used to change or modify data values in a table.
- The Verb Update in SQL is used to either UPDATE :
  - All the rows from a table.

  OR
  - A select set of rows from a table.

➢ **Updating All Rows:-**

Syntax:	**Example:-**

UPDATE <TableName> **SET**
<ColumnName1>=<Expression1>,
<ColumnName2>=<Expression2>;
	UPDATE emp **SET**
	Sal=15000;
	i.e. Update all rows from emp table with salary 15000.

➢ **Updating Records Conditionally:-**

Syntax:	**Example:-**

UPDATE <TableName> **SET**
<ColumnName1>=<Expression1>,
<ColumnName2>=<Expression2>
WHERE <Condition>;
	UPDATE emp **SET**
	Sal=20000
	WHERE eno=11;
	i.e. Update a rows of eno is 11 with sal 20000.

- The Update Statement updates columns in the existing table's rows with new values.
- The **SET** clause indicates which columns data should be modified and new values that they should hold.
- The **WHERE** clause, if given specifies which rows should be updated, otherwise all table rows are updated.

Modifying (Changing) the Structures of Tables:-

Que: - Explain Alter Statement with syntax and suitable example.
(How to changing the structure of a table? Explain with example.)

Alter Command:-

- The structure of a table can be modified using the ALTER TABLE command.
- ALTER TABLE allows changing the structure of an existing table. With ALTER TABLE it is possible to **add** or **delete** columns**,** change the data type of existing columns or rename columns or table itself.

➢ **Adding a New Columns:-**

Syntax:-	**Example:-**
ALTER TABLE <TableName> **ADD** (<NewColumnName1> <Datatype> (<Size>), (<NewColumnName2> <Datatype> (<Size>));	**Alter Table** emp **ADD** (Job char(15)); i.e. Add a new column Job in the emp table.

➢ **Dropping a column from a table:-**

Syntax:-	**Example:-**
ALTER TABLE <TableName> **DROP** column <ColumnName>;	**Alter Table** emp **DROP** column Job; i.e. Drop a column Job from the emp table.

➢ **Modifying existing columns:-**

Syntax:-	**Example:-**
ALTER TABLE <TableName> **MODIFY** (<ColumnName> <Datatype> (<size>) **);**	**Alter Table** emp **MODIFY** **(** eno number(10) **);** i.e. Change the size of the column eno with size 10 of the emp table.

Restriction on the Alter Table:-

The following tasks can not be performed when using the Alter Table Clause.

Restrictions:-

- Change the name of the table.
- Change the Name of the columns.
- Decrease the size of a column if table data exists.

Displaying the Table Structure:-

DESC Statement:-

To display information about the columns defined in a table use the following syntax.

Syntax:-	**Example:-**
DESC[RIBE] <TableName>;	**Desc** emp; Output: Name Null? Type Empno Number(5) Ename Char(20)

This command display the column names, data type and the special attributes connected to the table.

Destroying Tables:-

Drop Table Statement:-

- Drop table statement with the table name can destroy a specific table.
- If a table is dropped all records held within it are lost and can not be recovered.

Syntax:-	**Example:-**
DROP TABLE <TableName>;	**DROP TABLE** test1; i.e Removes the table test1 along with the data held.

Basic Commands:

✓ SPOOL :-

- The spool command can be used to save a SQL*PLUS session into a file.
- To start the spool session, simple type spool filename.lst after the prompt SQL>.
- To end the spool session, simple type spool off after the prompt SQL>.
- Any session information such as both SQL statement entered from the keyboard and its execution result between the command line spool filename.lst and the command line
- Spool off will be saved in the file filename.lst. If the file extension .lst is omitted, the system will automatically append it for you.

```
Example:  SQL>  spool emp.lst

SQL>  select  *   from  emp;

SQL>  spool off
```

- In this example, the SQL statement and its execution result will be spooled or saved into the file with the name emp.lst.

✓ Save:-

Que: - Explain in brief: Save Command.

The save command will save the current contents of the SQL*PLUS buffer into the file filename in your current working directory.

```
SQL> SAVE filename
```

Example: SQL> SAVE z:\oracle\emp.sql;

```
SQL> SAVE filename  replace
```

The save command with the keyword replace at the end of the command line will overwrite the current content in the file filename with the current content in the buffer.

```
SQL> SAVE filename  append
```

The save command with the keyword append at the end of the command line will appended the current content in the file filename with the current content in the buffer.

When you save a SQL statement in the SQL*PLUS buffer into a file, it is a good practice to save the file with the extension .sql.

✓ **SET LINESIZE**:

It will set the number of character per line.

SYNTAX:

SET LINE[SIZE] <value>

Example: Set line[size] 100;

✓ **SET PAGESIZE**:

The PAGESIZE setting tells SQL*Plus the number of printed lines that will fit on one page of output.

SYNTAX: SET PAGE[SIZE> <value>

Example: Set page[size] 50;

✓ **GET:**

Loads a SQL statement or PL/SQL block from a host operating system file into the SQL buffer.

SYNTAX: GET file_name[.ext]

Example: Get E:\Tempfile.SQL

✓ **EDIT**:

- Invokes a host operating system text editor on the contents of the specified file or on the contents of the SQL buffer.
- The SQL*Plus EDIT command allows you to invoke the text editor of to use in editing SQL statements. The specific editor invoked depends on the operating system, The default editor under Windows is Notepad.

SYNTAX: EDIT [file_name[.ext]]

Example: ed E:\Tempfile.SQL

Pseudo Columns:

✓ **ROWID:-**

It is fixed-length binary data. Every record in the database has a physical address of rowid. The format of rowed is: BBBBBBB.RRRR.FFFFF.

Where BBBBBBB is the block in the database file.

RRRR is the row in the block.

FFFFF is the database file.

SQL> select rowid,empno from emp;

```
ROWID                        EMPNO
------------------ -----------------------
AAADVLAABAAAKVCAAG      7369
AAADVLAABAAAKVCAAH      7499
AAADVLAABAAAKVCAAI      7521
```

✓ **ROWNUM:**

For each row returned by a query, the ROWNUM pseudocolumn returns a number indicating the order in which Oracle selects the row from a table or set of joined rows. The first row selected has a ROWNUM of 1, the second has 2, and so on.

```
SQL> select rownum,ename from emp;
ROWNUM    ENAME
---------- ----------
       1          KING
       2          BLAKE
       3          CLARK
```

You can use ROWNUM to limit the number of rows returned by a query, as in this example:

```
SQL> SELECT * FROM employees WHERE ROWNUM > 6
```

✓ **USER:**

USER returns the name of the session user (the user who logged on).

```
SQL> select user from dual;
      USER
   ------------------------------
      SYSTEM
```

✓ **UID:**

UID returns an integer that uniquely identifies the session user (the user who logged on).

```
SQL> select uid from dual;
      UID
----------
       5
```

✓ **SYSDATE**:

SYSDATE returns the current date and time set for the operating system on which the database server resides.

```
SQL> select sysdate from dual;
           SYSDATE
              ---------
            21-Jun-24
```

Transaction control language statements – COMMIT, ROLLBACK and SAVEPOINT

COMMIT	**Use**	Using the commit statement ends the current transaction and makes permanent any changes made during the transaction.
	Syntax	COMMIT:
	Example:-	SQL> update emp set sal=1000 where eno=1; 2 row updated SQL> commit; Commit complete That means update two rows permanently in emp table.
ROLLBACK	**Use:-**	A rollback does exactly the opposite of commit; It ends the transaction but undoes any changes made during the transaction.
	Syntax:-	ROLLBACK [TO [SAVEPOINT] <SavePointName>];
		Where, **SAVEPOINT**: is optional and is used to rollback a transaction partially, as far as the specified savepoint **SAVEPOINTNAME:** is a savepoint created during the current transaction.
	Example:-	SQL> update emp set sal=1000 where eno=1; 2 row updated SQL> rollaback; That means undo the data of emp table without updation.
	1.	
SAVEPOINT	**Use**	Savepoint marks and saves the current point in the processing of a transaction. When a SAVEPOINT is used with a ROLLBACK statements, parts of a transaction can be undone.
	Syntax:-	SAVEPOINT <SavePointName>;
	Example:-	SQL> SAVEPONT s1;
	A ROLLBACK operation performed **with** the SAVEPOINT clause amounts to the following: 1. A predetermined portion of the transaction is rolled back. 2. Retains the save point rolled back to, but loses those created after the named savepoint. 3. Releases all transactional locks that were acquired since the savepoint was taken.	