SQL

Introduction to SQL:

- SQL is stands for Structured Query Language used to define and manipulate the data in Relational Database Management System.
- In a database environment, the interactions between the Client and the Server are only through SQL
- It is a non-Procedural language.
- It is a Fourth Generation Language.
- SQL is a simple English-like Language. End-users with little or no experience in data processing can learn SQL features very quickly.
- SQL was first introduced by IBM Research and was introduced into the commercial market first by Oracle Corporation in 1979.
- It is a unified language i.e. common for many databases.
- SQL provides the following functionalities:
  a. Creation of tables.
  b. Querying the exact data.
  c. Change the data structure and the data.
  d. Combine and calculate the data to get the required information.

SQL Command Classifications:

Based on the type of action that each command performs, SQL commands can be broadly classified as follows:

Classifications	Description	Commands
DDL (Data Definition Language)	DDL commands are used to define the structure of a table, or modify the structure. It deals with Metadata	CREATE,ALTER,DROP, TRUNCATE,DESC
DML (Data Manipulation language)	DML commands are used to access a database including inserting, updating and inserting. It deals with the data.	INSERT,UPDATE, DELETE,SELECT
DCL (Data Control Language)	DCL commands are used to restrict or grant access to tables including privileges and the saving and restoring data. It deals with accessibility and transaction changes.	GRANT,REVOKE
TCL (Transaction Control Language)	Is used to complete fully or undo the transactions	COMMIT,SAVEPOINT, ROLLBACK

Data types: the most commonly used data types in Oracle SQL are:

1. **Char(n):** Char data type is a fixed length character data of length n bytes. Default size is 1 byte and it can hold a maximum of 2000 bytes. Character data types pad blank spaces to the fixed length if the user enters a value lesser than the specified length.

Example:

Sname char(10) stores up to 10 characters of data in the column Sname.

2. **Varchar2(size):** Varchar2 data type is a variable length character string. The maximum length of varchar2 data type is 4000 bytes. Unlike char datatype, blank spaces are not padded to the length of the string. So, this is more preferred than character datatypes since it does not store the maximum length.

   Example:

   City varchar2(10) stores up to 10 characters of data in the column city.

3. **Number(p, s):** The number data types can store numeric values where p stands for the precision and s stands for the scale.

   Example:

   Sal number – Here the scale is 0 and the precision is 38

   Sal number(7) – Here the scale is 0 and the number is a fixed point number of 7 digits

   Sal number(7,2) – Stores 5 digits followed by 2 decimal points.

4. **DATE:** Date data type is used to store date and time values. The default format is 'DD-MON-YY'. The valid data for a date data type ranges from January 1, 4712 BC to December 31, 9999 AD. Date data type stores 7 bytes one each for century, year, month, day, hour, minute and second.

5. **Other Data Types:** In addition to the above data types there are many other data types are available. They are:

   - **RAW(n):** RAW data type stores binary data of length n bytes. The maximum size is 255 bytes.
   - **LONG:** Stores character data of variable length up to 2 Gigabytes(GB)
   - **LONG RAW:** Stores up to 2 Gigabytes(GB) of raw binary data. A Table cannot contain more than one LONG column. This data type cannot support provisions like other data types.
   - **INTEGER(P):** Stores a Signed Integer, decimal or binary. P digits wide.
   - **FLOAT(P):** Stores a Floating point number.
   - **LOB Data types:** LOB data types stores up to 4 GB of data. This data type is used for storing video clippings, large images, history documents etc. LOB data types can be
     - CLOB Character Large Objects (Internal LOB)
     - BLOB Binary Large Objects (Internal LOB)
     - BFILE Binary File (External LOB)

SQL Constraints: SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table. These are used to ensure proper data entry in tables. These are added while creating, modifying or altering a Table structure.

1. **NOT NULL:** The column must contain a value. The column cannot be empty.
2. **UNIQUE:** The values must be unique. It does not allow duplicate values, but it allows null values.
3. **PRIMARY KEY:**The values must be unique. It does not allow duplicate values and null values.
4. **DEFAULT:** It provides a default value for the column when value is not supplied for that.
5. **CHECK:** It is used for checking the values of the column against the conditions specified.
6. **FOREIGN KEY:** It allows only the values matching to the values of another primary key column of another table or the same table.

DDL COMMANDS: DDL stands for Data Definition Language. These commands are used to create, alter, modify, and delete a structure of a database table. They deal with metadata.

1. **CREATE Command:** This command is used to create Oracle database objects like tables, indexes, views, etc.,

   To create a table, use CREATE TABLE command

   Syntax: CREATE TABLE <table name> (column1 data type (size),[CONSTRAINT constraint name], column1 data type (size),[CONSTRAINT constraint name]);

   Example: CREATE TABLE STUDENT (ADMN NUMBER(3) PRIMARY KEY, ENAME VARCHAR2(20) NOT NULL, CLASS VARCHAR2(10) DEFAULT 'BCOM');
2. **DESC Command:** After creating the table, viewing the structure can be done using DESCRIBE followed by the name of the table.

   Syntax: DESC <table name>;

   Example: DESC student;
3. **DROP Command:** This command is used to delete Oracle database objects like tables, indexes, views, etc., This command drops the data and the structure is permanently removed from the database.

   Syntax: DROP <Object type> <Object name>

   Example: DROP TABLE STUDENT;
4. **RENAME Command:** this command is used to rename a database object.

   Syntax: RENAME <Old object Name> TO <New object Name>;

   Example: RENAME STUDENT TO STU;
5. **ALTER Command:** This command is used to alter the structure of a database object. A table's structure can be altered using the ALTER TABLE Command. There are different Options available with ALTER TABLE Command; They are:

   i. <u>To add a new column</u> to an existing table.

   Syntax: ALTER TABLE <table name> ADD (column1 data type, column2 data type ...);

   Example: ALTER TABLE STU ADD(MARKS NUMBER(3), AVERAGE NUMBER(5,2));

ii. <u>To modify the length of an existing column</u> of a table.

Syntax: ALTER TABLE <table name> MODIFY (column1 data type, column2 data type ...);

Example: ALTER TABLE STU MODIFY(ADMN NUMBER(4));

iii. <u>To Add Constraint to an existing column</u> of a table.

Syntax: ALTER TABLE <table name> ADD CONSTRAINT <column Name> <Constraint>;

Example: ALTER TABLE STU ADD CONSTRAINT NOT NULL(MARKS);

iv. <u>To Drop(Delete) an existing column</u> of a table.

Syntax: ALTER TABLE <Table name> DROP COLUMN <Column1>, <Column2>;

Example: ALTER TABLE STU DROP COLUMN AVERAGE;

v. <u>To Enable/Disable/Drop Constraint of an existing column</u> of a table.

Syntax: ALTER TABLE <Table-name> ENABLE/DISABLE <Column-Name> <Constraint>;

Example: ALTER TABLE STU ENABLE CONSTRAINT ADMN;
ALTER TABLE STU DISABLE CONSTRAINT CLASS;
ALTER TABLE STU DISABLE CONSTRAINT CLASS;

- **DML Commands:** DML Stands for Data Manipulation Language. DML commands are used to access a database including inserting, updating and inserting. It deals with the data.

1. **SELECT Command:** This command is used to retrieve data from a table. It allows filtering of records and columns based on conditions. A basic SELECT statement contains two clauses or parts:

   Syntax: SELECT <column name(s)>/* FROM <table name(s)> WHERE <conditions>

i. **To display all the records of a table** or entire information of a table

   Syntax: SELECT * FROM <Table Name>;

   Example: SELECT * FROM STU;

ii. **To display filtered or a few records** from a table based on condition.

   Syntax: SELECT * FROM <Table Name> WHERE <Condition >;

   Example: SELECT * FROM STU WHERE MARKS >=600;

iii. **To display selected columns** from a table.

   Syntax: SELECT <column1, column2, ...> FROM <Table Name>;

   Example: SELECT ADMN, SNAME, MARKS FROM STU;

iv. **To display selected columns those satisfy a condition** from a table.

   Syntax: SELECT <Column1, Column2, ...> FROM <Table Name> WHERE <Condition>;

   Example: SELECT ADMN, SNAME, MARKS FROM STU WHERE MARKS>=600;

2. **INSERT Command:** Insert command is used to add one or more rows to a table. The values are separated by commas. There are various options available with INSERT Command. They are:

i. **To insert a new record** into a table. The values are entered in the same ORDER specified by the structure of the table.

Syntax: INSERT INTO <table name> VALUES (value1, value2, value3...);

Example: INSERT INTO STU VALUES (890, 'RAJU', 'BCOM(BI)', 660, 92.5);

i. **To insert a few column values** into a table.

Syntax: INSERT INTO <table name> (column list) VALUES (value list);

Example: INSERT INTO STU (ADMN, SNAME, MARKS) VALUES (890, 'Venkat', 678);

ii. **To insert more records at a time into a table or through parameter subscription.**

Syntax: INSERT INTO <Table Name>(<columns list>) VALUES (<Parameters list.);

Example: INSERT INTO STU(ADMN, SNAME, COURSE, MARKS, AVERAGE) VALUES (&ADMN, '&SNAME', '&COURSE', &MARKS, &AVERAGE);

When you execute the Query it displays like below:

```
Enter value for ADMN :
Enter value for SNAME :
Enter value for COURSE :
Enter value for MARKS :
Enter value for AVERAGE:
```

Supply values every time and press enter, finally the record is inserted into the table. To enter more record as above type '/' at SQL prompt it repeats the above procedure. In this way we may enter any number of records.

3. **UPDATE Command:** UPDATE Command is used to update the values of attributes (Fill columns with values, text based of conditions)

Syntax: UPDATE <Table Name> SET <Column Name>= <Value>, <Column Name>= <Value> WHERE <Condition>;

Example: i) To fill course with 'bcom'.
UPDATE Student SET course='bcom';

ii) To fill Total with sum of three subjects.
UPDATE Student SET total=tel+hin+eng;

ii) To fill the res with 'pass';
UPDATE student SET res = 'pass' WHERE tel>=35;

4. **DELETE Command:** To delete all rows or a few rows in a r table using a condition.

Syntax: DELETE from <Table Name> WHERE <Condition>;

Example: i) To delete all the rows of a table
DELETE FROM Student; or DELETE student;

ii) To delete all the records with course = 'bcom'.
DELETE FROM Student WHERE course = 'bcom';

TRUNCATE Command (DDL Command): this command is used to delete all rows or the entire information from a table.

Syntax: TRUNCATE TABLE <Table Name> **Example:** TRUNCATE TABLE student;

DCL Commands: DCL Stands for Data Control Language. DCL commands are used to restrict or grant access to tables including privileges and the saving and restoring data. It deals with accessibility and transaction changes. There are mainly two commands in this.

GRANT Command: This command is used to give / provide / grant privileges on dataset objects to other users. The following types of privileges can be granted:

- Delete or Insert data/records from a specific table.
- Alter the structure and data of a specific table
- Select data from a table, view, or a subset of columns in a table.
- Create a trigger or index on a table.
- Update data in a table or in a subset of columns in a table.
- Run a specified function or procedure.

Syntax: GRANT < privilege_list / role > ON < Object_name> TO <User_List / Public> WITH GRANT OPTION;

Example: GRANT ALL ON STUDENT TO PUBLIC;

To grant all permissions on STUDENT Table to Everyboday.

GRANT SELECT, INSERT ON STUDENT TO REDDY;

To grant select and insert permissions on STUDENT tables to User REDDY.

GRANT ALL ON STUDENT TO RAO WITH GRANT OPTION;

To grant all permissions to the user RAO on the table STUDENT and RAO can give further grants on STUDENT Table to other users.

REVOKE Command: This command is used to cancel privileges on database objects to the users.

Syntax: REVOKE < privilege_list / role > ON < Object_name> FROM <User_List / Public> ;

Example: REVOKE INSERT ON STUDENT FORM REDDY;

To revoke/ cancel INSERT privilege on STUDENT table from REDDY;

REVOKE ALL ON STUDENT FROM PUBLIC;

To revoke/cancel ALL privileges on STUDENT table from all the users.

TCL Commands: TCL Stands for Transaction Control Language. Is used to complete fully or undo the transactions.

1. **COMMIT Command:** This command is used to make the changes permanent. Generally this is used with delete command..

   Example: To delete a specified group of records permanently and save the changes on the specified table.

   DELETE FROM STUDENT WHERE COURSE = 'BCOM';
   COMMIT;

   In the above example the DELETE command deletes the records and COMMIT command makes the changes permanent.

2. **ROLLBACK Command:** This command is used to cancel the changes or restore the data of a Table. This command works only before using COMMIT command.

   DELETE FROM STUDENT WHERE COURSE = 'BCOM';
   ROLLBACK;

The above command restores the records deleted using DELETE Command.

3. **SAVEPOINT:** This command is used to set a margin for the above commands COMMIT or ROLLBACK. It is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

   Example:
   ```
   DELETE FROM STUDENT WHERE CLASS = 'BSC';
   SAVEPOINT SP1;
   DELETE FROM STUDENT WHERE CLASS = 'BCOM';
   ROLLBACK TO SP1;
   COMMIT;
   ```

   Here we may restore the records of BCOM back but not BSC records, because we have cancelled all the changes up to SP1 only. SP1 is save point that is set after deleting BSC records.

Date functions: The Date functions available in SQL are:

1. **SYSDATE:** This function returns the system date.
   Ex: SELECT SYSDATE FROM DUAL;
2. **ADD_MONTHS:** This function adds given number of months the given date.
   Syntax: SELECT ADD_MONTHS (< Date >, <No. of Months add);
   Ex: i. SELECT ADD_MONTHS(SYSDATE, 3) FROM DUAL;
   It adds three months to the system date.
   ii. SELECT ADD_MONTHS('2-JAN-2016', 4) FROM DUAL;
   It adds months to the given date 2-jan-2016.
   Iii. SELECT ADD_MONTHS ('2-JAN-2016', -2) FROM DUAL;
   It adds -2 (i.e. subtracts) months to the given date 2-jan-2016.
3. **LAST-DAY:** It returns the last day of the given month.
   Syntax: SELECT LAST_DAY(< Date >) FROM DUAL;
   Ex: i. SELECT LAST_DAY(SYSDATE) FROM DUAL;
   Return the last day of current month.
   ii. SELECT LAST-DAY('2-FEB-2001') FROM DUAL returns '28-feb-01'.
   iii. SELECT LAST-DAY('2-FEB-2012') FROM DUAL returns '29-feb-12'.
4. **NEXT-DAY:** It returns the date of given weekday in next week.
   Syntax: SELECT NEXT_DAY(< Date >, <Week day name>) FROM DUAL;
   Example: SELECT NEXT_DAY('14-JUL-2017, 'MON') FROM DUAL;
   It returns 17-jul-2017 it is date of next Monday after given date.
5. **MONTHS-BETWEEN:** It returns the number of months between two dates.
   Syntax: SELECT MONTHS_BETWEEN (<ending date>, <starting date>) FROM DUAL;
   Example: SELECT MONTHS_BWTWEEN('1-DEC-2016', '1-JAN-2016') FROM DUAL;

NUMERIC Functions: The Date functions available SQL are:

1. **ABS:** This function returns the Absolute value of the given Number.
   Example: SELECT ABC(-100), ABS(30) FROM DUAL;
   Output: 100 30