UNIT 4:
SQL ESSENTIALS

1. WHAT IS SQL?

- SQL stands for Structure Query Language.
- IBM developed it in 1970.
- It is a standard language used to interact with modern RDBMS.
- SQL is a cross-platform language.
- It is known as database language and the DBMS are supporting SQL.
- The programmer may submit query to DBMS using SQL to get the result from stored data and manage the database.
- SQL is English language, it has simple command set.
- It is a statement & syntax oriented language that is very easy to use.
- SQL is non-procedural language.
- SQL is used to Creating / maintaining, manipulating data & controlling database.

2. LIST THE RULES TO WORK WITH SQL.

- Any SQL statement must **start with a verb** like select, create, alter, etc.
- Each **verb followed by adjectives** like from, where, having, etc.
- There must **put an extra space between verb and adjective**.
- A comma is useful to separate the field name.
- A semicolon **(;)** is useful to complete the SQL statements.
- We can split our statement into multiple lines.
- Keywords **cannot be abbreviated** or split across lines.
- Identifiers, operators are separated by one or more spaces
- Variable name and length must be less than or equal to **30 characters.**
- **Character & date** values must be **enclosed within single quotes (‘’)**.

3. WHAT IS SQL *PLUS?

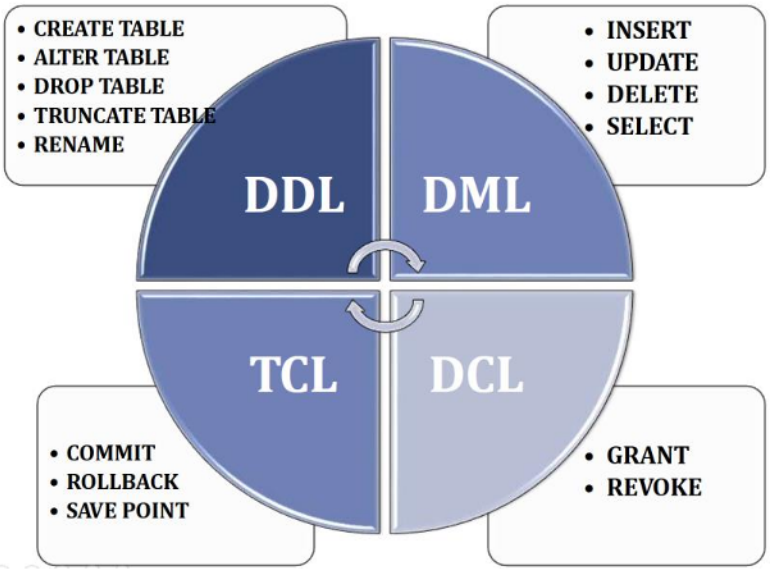
- SQL*Plus is an Oracle tool that recognizes and submits SQL statements to the Oracle9i Server for execution.
- It provide I/O interface for SQL that communicate with DBMS.
- It contains its own command language.
- It provides a line editor for modify SQL statements.
- It access local & remote database.
- It executes SQL statement to retrieve, modify, insert and removed data from statements for reparative use in future.
- It performs calculations & print query result in the form of report.
- It is an interactive user interface to the ORACLE DBMS.
- Typically SQL *PLUS is used to issue quires & to view the query results on the screen.

4. DIFFERENCE: SQL AND SQL * PLUS

SQL	SQL * PLUS
SQL is structure Query language.	SQL * Plus is an oracle's property software
SQL provides commands to manipulate database.	SQL*Plus cant manipulate database
SQL statement oriented	Possible to write down group of statements
Can't Write any command as a short name	Short name for SQL *plus (desc , clear screen)
SQL command must be ended with ;	No needs to terminate command with ;
It's a Language	It's a Software /environment
DDL, DCL,TCL,DML, DQL(SELECT)	Format related, comments, environment related command, report related
It follow ANSI standard	Follow Oracle standard
Can't provide formatting	Provide Format related commands

5. EXPLAIN CATEGORIES OF SQL COMMANDS.

- SQL commands can be roughly divide into three categories with regard to their functionality.
  - Firstly, there are those used to create and maintain the database structure.
  - The second category includes those commands that manipulate the data in such structures.
  - Third there are those that control the use of the database



- There are 4 types of SQL commands.
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
  - DCL (Data Control Language)
  - TCL (Transaction Control Language)

DDL (DATA DEFINITION LANGUAGE)

- The DDL is a set of SQL commands used to create, modify and delete the structure of database table but not data.

DDL (DATA DEFINITION LANGUAGE)	
CREATE	Used to creates table or any object
ALTER	Used to alters existing database or tables
DROP	Used to drops existing tables or objects
TRUNCATE	Used to removes whole data at a time
RENAME	Used to renames an objects or tables

DML (DATA MANIPULATION LANGUAGE)

- It is used to work and manage data in tables.
- By using these commands, we can insert, update or delete table rows of the database.

DML (DATA MANIPULATION LANGUAGE)	
INSERT	Used to insert data in a table
UPDATE	Used to updates existing data in a table
SELECT	It gives view of databases or tables
DELETE	Used to deletes particular records in a database or tables

DCL (DATA CONTROL LANGUAGE)

- It is used for the control of data.

DCL (DATA CONTROL LANGUAGE)	
GRANT	It provides rights for user on database
REVOKE	It revokes user rights

TCL (TRANSACTION CONTROL LANGUAGE)

- It is used to manage the changes that made DML statements.
- It allows statements to be grouped together into logical transactions.

TCL (TRANSACTION CONTROL LANGUAGE)	
COMMIT	Used to save buffer data to storage device
ROLLBACK	Used to restore database to original since last commit
SAVEPOINT	It keeps break in save action

6. EXPLAIN DATA TYPES OF SQL.

- There are 5 types of data types of in Oracle like
  1. **Character**
  2. **Number**
  3. **Date**
  4. **LOB**
  5. **Other**

Data Type	Usage	Size/ Storage Capacity	Example
Number			
Number <p>,<s>	This data type used to store negative and positive integers, fixed-point value	Default scale (S)=0 Precision (P)=38	roll number, mobile number, age , product price, quantity , unit cost

Character			
Char (<Size>)	It is used to store fixed length character data.	Default : 1 Maximum: 255	Name, city, result, grade
Varchar (<Size>)	This data type is used to store variable length alphabetical data.	2000 characters.	Name , last name, grade, product name

Varchar2 (<Size>)	This data type is used to store variable length alphanumerical data.	2000 characters	Address, Subject code
Long (<size>)	This data type is used to store variable length character strings up to 2GB. It is generally used to store arrays of binary data in ASCII format. It cannot indexed and you cannot apply generally character functions such as SUBSTR. It is similar to MEMO data type of MS-ACCESS	2 GB	Binary data
Date			
Date	This data type is us used to represent date and time. This data type stores information such as century, year, month, day, hour, minute and second Date format: **DD-MON-YY**	6 bytes	Birthdate, join date, birth time, arrival time, expiry date
LOB			
BLOB	It is used to store Binary large object. It is typically used to store graphics, image, charts, sound, or video data.	4 GB	Stud image, Product image
CLOB	It stores character large objects, up to 4GB in length.	4 GB	Stud image, Product image
Other data type			

Row	This data type is useful when the data is a binary data that is not interpreted by other database	255 bytes	Currency
Long Row	This data type is useful when the data is a binary data that is not interpreted by other database.	2 GB	Currency

7. EXPLAIN CREATE COMMAND WITH EXAMPLE

- CREATE TABLE is a DDL command.
- This command is used to create table.
- The CREATE table command is useful a new table in your present user.
- When we create a table we have to specify fieldname, data type and size.

Rules Table Names and Column Names:

- Must begin with a letter
- Must be 1 to 30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle Server reserved word.

SYNTAX:

CREATE TABLE <table name>

(

<Field name>><Data type> (<Size>) [CONSTRAINT],

<Field name><Data Type> (<Size>) [CONSTRAINT],

……..

);

In CREATE TABLE Statement:

<Table name>	Specify name of the table
<Column name>	Specify fieldname
<data type>	Specify the type of data that field hold
<size>	Specify length for field
[CONSTRAINT]	Specify name of constraint if any

EXAMPLE
CREATE TABLE student (rollno number(3), first_name varchar(20), last_name varchar(20), address varchar2(30),city varchar(10),pincode number(10));

8. DISCUSS: DESCRIBE COMMAND.

- This command is used to view the structure of the table.
- Syntax:

  DESC<TABLE_NAME>;

  DESCRIBE <TABLE_NAME>;
- Example:

  DESC EMP;

  DESCRIBE EMP;

9. EXPLAIN DDL STATEMENTS WITH EXAMPLE.

- DDL stands for Data Definition Language.
- It is used to create, modify or delete the database structure but not data.
- Following are the list of DDL commands:
    - CREATE TABLE
    - ALTER TABLE
    - DROP TABLE
    - TRUNCATE TABLE
    - RENAME

CREATE TABLE:

- This command is used to create table.

SYNTAX:

CREATE TABLE <table name>
(
<Field name>><Data type> (<Size>) [CONSTRAINT],
<Field name><Data Type> (<Size>) [CONSTRAINT],
……..
);

EXAMPLE:

CREATE TABLE STUD
(
Rlno number (3),
Sname char (8),
Bdate date
);

ALTER TABLE:

- By using, alter table command we can modify the structure of the database.
- Using ALTER TABLE statement to:
    - Add a new column
    - Modify an existing column
    - Define a default value for the new column
    - Drop a column
    - Change column name

SYNTAX:

- **ADDING NEW COLUMNS**

  ALTER TABLE <table name> ADD (new<column name> data type (size));
- **MODIFYING EXISTING COLUMNS**

  ALTER TABLE <table name> MODIFY (<column name> new data type (new size));

- **DROPING EXISTING COLUMNS**

  ALTER TABLE <table name> DROP (<column name>);

- In the syntax:

<Table name>	Specify name of the table
<Column name>	Specify fieldname
<data type>	Specify the type of data that field hold
<size>	Specify length for field

EXAMPLE:

- **ADDING NEW COLUMNS**

  ALTER TABLE student ADD (country VARCHAR (10));
- **MODIFYING EXISTING COLUMNS**

  ALTER TABLE student MODIFY (last_name VARCHAR2 (30));
- **DROPING EXISTING COLUMNS**

  ALTER TABLE dept DROP (country);
- **RENAME COLUMNS**

  ALTER TABLE dept RENAME COLUMN grno TO ROLLNO;

DROP TABLE:

- This command is removes the definition of a table.
- When you drop a table, the database loses all the data in the table.
- When a table is no longer needed, it can be dropped.
- Both the table definition and the rows in the table are dropped, and the space allocated for the table is made available for other database objects.

SYNTAX:

DROP TABLE <table name>;

In syntax

<Table name>	Specify name of the table

EXAMPLE:

DROP TABLE student;

RENAME:

- This command is used to rename a table, view, sequence, or synonym.

SYNTAX:

RENAME <old table name > TO <new table name>;

In the syntax:

<old name>	Specify the old name of the table
<new name>	Specify the new name of the table

EXAMPLE:

RENAME student TO stud;

TRUNCATE TABLE:

- This command is removes all the row of table to release the storage space used by that table.
- The TRUNCATE statement is similar to the DELETE statement. Both of the statements will delete rows from a table.
- The main difference is that the DELETE can be more selective (in other words, using a WHERE clause).
- The TRUNCATE statement simply removes all rows from a table.
- The TRUNCATE statement will also appear to run faster than a DELETE in most cases.

SYNTAX:

TRUNCATE TABLE <table_name>;

- In syntax

<Table name>	Specify name of the table

EXAMPLE:

TRUNCATE TABLE student;

10. EXPLAIN DML STATEMENTS WITH EXAMPLE.

- DML stands for Data Manipulation Language.
- The DML commands are the set of SQL commands that can be used to insert, modify or delete the records of the table.
- Following are the list of DML commands:
  - INSERT
  - UPDATE
  - DELETE
  - SELECT

INSERT

- This command is used to insert a row in table.
- With the help of Insert into command, we can enter one record at a time.

SYNTAX:

INSERT INTO <table name> [(<column name>, <column name>)]
VALUES (<value1>, <value2>…);

In syntax

<Table name>	Specify name of the table
<Column name>	Specify column or field name
Values	Specify the data or value that you want to enter in a field

EXAMPLE:
INSERT INTO student VALUES (1, ‘avni’,’bscit’ 1);

UPDATE:

- Update command used to update the records of a table.
- We can modify existing data with the help of Update.

SYNTAX:

UPDATE TABLE SET <column_name> = value [, <column_name> = value....]
[WHERE <condition>];

In syntax:

<Table name>	Specify name of the table
<Column name>	Specify fieldname for update value
WHERE<condition>	Specify Condition for updation

EXAMPLE:

UPDATE dept SET deptno = 20 WHERE empid=110;

DELETE

- This command is used to remove one or all records from table.

SYNTAX:

DELETE FROM <table_name> [WHERE <condition>];

In syntax

<Table name>	Specify name of the table
WHERE<condition>	Specify Condition for deletion

EXAMPLE:

```
DELETE From stud;
DELETE From stud where rollno=1;
```

SELECT

- This command is also known as DQL-data query language command.
- It is used to display all or specific records from database.
- We can also display list of tables available in current user.

SYNTAX:

SELECT */ <FIELDS> FROM <TABLE_NAME> |TAB
[WHERE <CONDITION>;

*	Display all fields & records
<Fields>	To select some specific fields
TAB	Display list of tables
<Table name>	Specify name of the table
WHERE<condition>	Specify Condition for display records

11.

EXAMPLE:

```
SELECT * FROM   TAB;
SELECT * FROM STUD;
SELECT * FROM STUD WHERE RLNO>5;
```

12. WRITE DOWN THE DIFFERENCE BETWEEN DROP TABLE & TRUNCATE TABLE.

Truncate table	Drop table
TRUCATE TABLE delete all records from table/database.	DROP TABLE delete all records along with structure.
TRUNCATE TABLE will not affect the schema of table	DROP TABLE will change the schema of table
Remove only records/data.	Remove records/data & structure.
SYNTAX: TRUNCATE TABLE <table name>;	**SYNTAX:** DROP TABLE <table name>;
EXAMPLE: TRUNCATE TABLE STUD;	**EXAMPLE:** DROP TABLE STUD;

13. EXPLAIN DCL COMMAND WITH EXAMPLE.

- DCL is used to control the data access to the database. DCL concerns with the Recovery and Security issue of database.
- This category of command allows us to grant and revoke privileges and permissions.
- Grant and revoke are DCL commands.

GRANT

- Grant means permission.
- The GRANT statement to give privileges to a specific user or role or to all users
- The following types of privileges can be granted:
  - Assign role: connect, resource & DBA.
  - Insert data into a specific table.
  - Update data into a specific table.
  - Display data of any table.
  - Delete data from a specific table.

SYNTAX:

GRANT <PRIVILEGE> | ROLE ON [<OBJECT NAME>] TO <USER NAME>;

EXAMPLE:

GRANT CONNECT TO USER1;

GRANT INSERT, UPDATE ON CUSTOMER TO USER1;

REVOKE

- This command use to remove privileges from a specific user or role from all users.
- The following types of privileges can be revoked:

- Role: Connect resource & DBA.
- To insert data into a specific table.
- To update data into a specific table.
- To Display data of any table.
- To delete data from a specific table.

SYNTAX:

REVOKE <privilege>| ROLE ON [<object name>] FROM <user name>;

EXAMPLE:

REVOKE CONNECT FROM USER1;

REVOKE INSERT, UPDATE, ON CUSTOMER FROM USER1;

14. EXPLAIN TCL WITH SUITABLE EXAMPLE:

- Transaction is a sequence of SQL statements that Oracle treats as a single block of unit.
- To save a set of changes made to a table using DML commands and to get back to the original state before changes have been made we use TCL.
- Three TCL commands are :
  - Commit
  - Rollback
  - Save point

Commit

- A set of changes made to a table using DML (INSERT, UPDATE, DELETE) commands are temporary.
- To make the changes permanent, COMMIT command must be needed after DML commands.

Syntax: -

```
COMMIT;
```

Example: -

```
DELETE FROM EMP WHERE NO=500;
COMMIT;
```

Rollback

- The ROLLBACK statement allows you to change your mind about a transaction.
- It brings back the original state of the tables to the state as of the last COMMIT statement or the beginning of the current transaction.

Syntax: -

ROLLBACK [TO SAVEPOINT <Savepoint_Name>];

Example: -

DELETE FROM EMP WHERE NO=500; ROLLBACK;

NOTE: - You cannot ROLLBACK the transaction after the COMMIT. So, After the COMMIT you cannot use ROLLBCAK command.

Savepoint

- Additionally, you can use SAVEPOINT to further subdivide the DML statements within a transaction before the final COMMIT of all DML statements within the transaction.
- SAVEPOINT essentially allows partial rollbacks within a transaction.

Syntax: -

SAVEPOINT <Savepoint_Name>;

ROLLBACK TO SAVEPOINT <Savepoint_Name>;

Example: -

DELETE FROM EMP WHERE NO=500; SAVEPOINT A1;

ROLLBACK TO SAVEPOINT A1;

15. EXPLAIN DQL WITH SUITABLE EXAMPLE.

- **SELECT** is **Data Query Language** command.
- In its most basic form, the SELECT statement has a list of columns to select from a table, using the SELECT ... FROM syntax.
- The * means "all columns."
- The SELECT statement is used to select data stored in your database.

SYNTAX

SELECT */<fieldname> |[DISTINCT <fieldname>] FROM <table_name>
[GROUP BY <fieldname> [HAVING <condition>]]
[ORDER BY <fieldname> [ORDER]];

- In syntax

*	Use to display all fields & records
<field name>	Specify fieldname to display some specific fields
DISTINCT	Used to remove repeated values from fields
<table name>	Specify name of the table to display
WHERE <condition>	Used to retrieve records that match some specific condition
GROUP BY	Used to perform function on group value
HAVING	Used to match conditional records with group function
ORDER BY	Arrange records in some specific order

EXAMPLE

SELECT ALL ROWS/COLUMNS FORM TABLE

SELECT * FROM DEPT;

SELECT SPECIFIC COLUMNS /ROWS FORM TABLE

SELECT ENAME,SAL,JOB FROM EMP;

SELECT ALL ROWS/COLUMNS FORM TABLE

SELECT * FROM DEPT;

REMOVE DUPLICATE ROWS

```
SELECT DISTINCT * FROM DEPT;
```

USE ARITHMETIC OPERATOR(+,_,*,/)

```
SELECT ENAME,SAL,SAL + 1000 FROM EMP;
```

USE RELATIONAL OPERATPR(>,<,>=,<=,=,<>)

```
SELECT * FROM EMP WHERE DEPTNO>10;
```

USE LOGICAL OPERATOR(AND,OR,NOT)

```
SELECT ENAME,SAL,JOB FORM EMP WHERE DEPTNO>10 OR DEPTNO <30;
```

SORTING DATA WITH SELECT

```
SELECT * FROM EMP ORDER BY ENAME DESC;
```

16. EXPLAIN OPERATORS WITH EXAMPLE

- An operator is a symbol, which represents particular operations on some data.
- Oracle / SQL provides following operators:

Arithmetic	Logical	Relational	Like	Miscellaneous
+	AND	>	LIKE	IS NULL
-	OR	<		IN
*	NOT	>=		BETWEEN
/		<=		
		=		
		!=		
		< >		

Arithmetic Operators

- It is used to perform arithmetic operations on a field that contain number data type.
- There are 4 types of operators like : **+ , - , * , /**

SYNTAX:

SELECT <field name > **AOP** <value> FROM <table name>;

EXAMPLE:	
+	SELECT name, sal + 1000 FROM EMP;
-	SELECT empno, name, comm, comm-100 FROM EMP;
*	SELECT name, sal, sal*10 FROM EMP;
/	SELECT sal / 12 FROM stud;

Relational Operator

- It is used to check condition.
- There are seven comparison operators used with oracle. : **> , < , >=, <=, !=, <> ,=**

SYNTAX:

SELECT <column name> FROM <table name> WHERE <column name> COMPARISIO OPERATOR <value>;

EXAMPLE:

```
SELECT * FROM EMP WHERE salary>5000;
SELECT * FROM EMP WHERE salary<5000;
SELECT * FROM EMP WHERE salary>=5000;
SELECT * FROM EMP WHERE salary<=5000;
SELECT * FROM EMP WHERE empname='Astha';
```

LOGICAL OPERATOR

- Logical operators are used to bind 2 or more conditions.
- There are 3 types of logical operators like
  - AND OPERATOR
  - OR OPERATOR
  - NOT OPERATOR

AND	Return true when all conditions are true
OR	Return true when any one or all conditions are true
NOT	Return true when all conditions are false

SYNTAX:

SELECT <column name> FROM <table name> WHERE <condition> **AND/OR** <condition>;

SELECT <column name> FROM <table name> WHERE NOT <condition> **AND/OR** <condition>;

EXAMPLE:

SELECT name, salary FROM EMP WHERE salary>5000 AND salary<10000;
SELECT name, salary FROM EMP WHERE salary>5000 OR salary<10000;
SELECT name, salary FROM EMP WHERE NOT salary>5000 OR salary<10000;

LIKE OPERATOR

- It is called pattern-matching operator.
- It uses wild card characters to match pattern.
- There are two wild card characters are mostly used **% AND _**.
- Pattern must be enclosed in single quote.
- This operator is also used with WHERE condition.

SYNTAX:

SELECT <column name> FROM <table name> WHERE <column name> LIKE 'PATTERN';

EXAMPLE:

SELECT * FROM EMP WHERE last name LIKE '_A%';

MISCELLANEOUS OPERATOR

- Oracle provides some special purpose operator that is called miscellaneous operators.

IN OPERATOR

- IN operator used to select the records that match in the set of values.

SYNTAX:

SELECT <column name> FROM <table name> WHERE <column name> IN (matching pattern);

EXAMPLE:

SELECT no, name, sal, FROM EMP WHERE deptno **IN** (10, 20, 30);

BETWEEN OPERATOR

- This operator will be used to specify the range of values.
- The **BETWEEN** operator display rows on the based on a range of values.

SYNTAX:

SELECT <column name> FROM <table name> WHERE <column name> **BETWEEN <**search criteria> **AND** <search criteria>;

EXAMPLE:

SELECT * FROM EMP WHERE sal BETWEEN 2500 AND 5000;

IS NULL OPERATOR

- The NULL condition is used with IS operator and IS NOT condition.
- A NULL values means the value is unavailable, unassigned, unknown or inapplicable.

SYNTAX:

SELECT * / <Fields> FROM <table_name> WHERE <field name> IS NULL;

EXAMPLE:

SELECT * FROM EMP WHERE sal IS NULL;

17. EXPLAIN CONSTRAINTS.

- Constraints enforce rules at the table level.
- We can define rules on data in a table when a row is inserted, updated or deleted from table.
- Constraint may be defined either at the same time as the table is created Or After the table has been created.

CATEGARIES OF CONSTRAINTS: -

1) DOMAIN CONSTRAINT (Associate with Column)

a. NOT NULL

b. CHECK

c. DEFAULT

2) ENTITY CONSTRAINT (Associate with ROW)

a. PRIMARY KEY

b. UNIQUE

3) REFERENTIAL INTEGRITY CONSTRAINT (Link between two table)

a. FOREIGN KEY

PRIMARY KEY

- A primary key constraint creates a primary key for table.
- Primary key cannot contain NULL values as well as cannot contain duplicate entry.
- A table can have only one primary key constraint.
- It can apply at column level as well as table level.

SYNTAX:

CREATE TABLE <table name>
(<column name> <Data Type><Size> PRIMARY KEY);
EXAMPLE:

CREATE TABLE stud (stud_no number (10) PRIMARY KEY, city varchar (10), mobile varchar (10));

FOREIGN KEY

- A FOREIGN key is a column or set of column that refers to a primary key in the same table or another table.
- A foreign key value must match an existing primary key value or unique key value or else is Null.
- Foreign key must reference either a primary key or unique key column.
- Foreign key is a column that defines how tables related to each other.
- It is a single or group of column whose values are derived from the Primary Key of some other table.
- The table in which foreign key is defined that is called a Foreign Table or Detail Table.

SYNTAX:

CREATE TABLE <table name> (<column name> <Data Type><Size>
REFERENCES <table name> (<column name>));

EXAMPLE:
CREATE TABLE mark (studno number (10) REFERENCES student (student_no), result char (10));

UNIQUE KEY

- Unique key constraints require that every value in a column or set of columns be unique.
- That means no two rows of a table can have duplicate values in specified column or set of columns.
- The column included in the definition of unique key constraints is called unique key.

SYNTAX:

CREATE TABLE <table name> (<column name> <Data Type><Size> UNIQUE);

EXAMPLE:

CREATE TABLE stud (studno number (10) UNIQUE, address varchar2 (20), city varchar (10), mobile varchar (10));

CHECK

- Check constraints defines a condition that each row must satisfied.
- The condition can use the same as we specified in query condition.
- It checks the column in a table.
- It returns an output if the column is true otherwise it returns an error.
- It allows duplicate values but not allow NULL values.
- It can apply at column level as well as table level.

SYNTAX:

CREATE TABLE <table name> (<column name> <Data Type><Size> **CHECK (condition)**);

EXAMPLE:

CREATE TABLE stud (empno number (3) CHECK (empno BETWEEN 101 AND 300), ename varchar (20), mobile varchar (10));

NOT NULL CONSTRAINT

- NOT NULL constraint protects one or more columns in a table.
- It allows duplicate values.
- It does not allow NULL values.
- It can be apply only at a column level.

Principles of NULL value: -

- Setting a NULL value is appropriate when the actual value is unknown.
- A NULL value is not equivalent to "zero" for the "number" data-type or "space" for "Character" data-type.
- A NULL value will evaluated to null in any expression. Example: - NULL * 10 = NULL
- NULL value can be inserted into the columns of any data-type.
- If a column has NULL values the oracle ignores the 'UNIQUE', 'FOREIGN KEY', and 'CHECK' constraints.

SYNTAX:

CREATE TABLE <table name> **(**<column name><Data Type><Size> **NOT NULL);**

EXAMPLE:

CREATE TABLE address (name varchar (20) NOT NULL, address varchar2 (30), city varchar (15) NOT NULL);

DEFAULT CONSTRAINT

- DEFAULT constraint allows you to set default value in a particular field.

- When we are creating any table at that time we can assign a default value to a column.
- When user is inserting value in the columns and by mistake leaves the column empty to which we have assign “Default” keyword, then the oracle engine will automatically insert the default value in that column.
- It is use with CREATE TABLE and ALTER TABLE command.

SYNTAX:

CREATE TABLE <table name> **(**<column name><Data Type><Size> **DEFAULT ‘value’**);

EXAMPLE:

CREATE TABLE address (name varchar (20) NOT NULL, address varchar2 (30), city varchar (15) **DEFAULT ‘RAJKOT’**);

18. EXPLAIN GROUP BY AND HAVING WITH EXAMPLE.

- The GROUP BY clause is another section of the select statement.
- This optional clause tells Oracle to group rows based on distinct values that exist for specified columns.
- The GROUP BY clause creates a data set, containing several sets of records grouped together based on a condition.

P_NO	QTY_ORDER	QTY_DISP
P1	10	10
P4	3	3
P6	7	7
P2	4	4
P5	10	10
P3	2	2
P1	6	6
P6	4	4
P4	1	1
P6	8	8

- SELECT P-NO, SUM (QTY_ORDERD) “TOTAL QTY” FROM SALES_DETAIL GROUP BY P_NO;

OUTPUT: -

P_NO	TOTAL QTY
P1	16
P2	4
P3	2
P4	4
P5	10
P6	19

HAVING CLAUSE:

- The HAVING clause can be used in combination with the GROUP BY clause.
- HAVING imposes a condition on the GROUP BY clause, which further filters the groups created by GROUP BY clause.

SELECT P-NO, SUM (QTY_ORDERD) “TOTAL QTY” FROM SALES_DETAIL GROUP BY P_NO HAVING P_NO=’P1’ OR P_NO=’P4’;

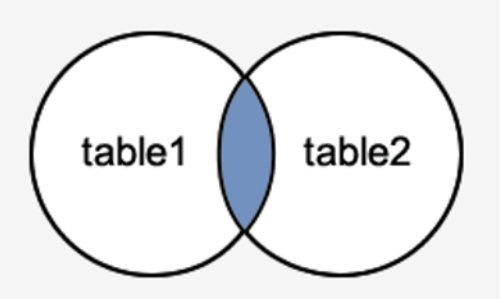
OUTPUT: -

P_NO	TOTAL QTY
P1	16
P4	4

19. WHAT IS JOIN? EXPLAIN VARIOUS TYPES OF JOIN.

- Join is use to get data from the multiple tables.
- To joining table, we required a common field in both tables.
- Join occurs in WHERE clause of SELECT statement.
- There are 4 types of join:
    - Inner Join/ equi join
    - Outer join
    - Self-Join
    - Cross Join
- With the help of SELECT statement, we can join up to 15 tables.

INNER JOIN/ EQUI JOIN



- Inner join known as equi **join.**
- Inner join operator is used equal (=) sign in the query.
- WHERE clause is use in this type of join.
- Condition used in this type of join.

SYNTAX:

```
SELECT columns FROM table1 INNER JOIN table2
ON
table1.column = table2.column;
```

EXAMPLE

EMP		
3EMPNO	**ENAME**	**DEPTNO**
110	JAY	10
111	RAJ	20
112	RAM	10
113	MEET	30
114	NIL	10

DEPT	
DEPTNO	**DNAME**
10	ADMIN
20	SALES
30	MARKETING

```
SELECT E.ENAME, E.DEPTNO, D.DNAME FROM EMP E INNER JOIN , DEPT D
ON
E.DEPTNO=D.DEPTNO;
```

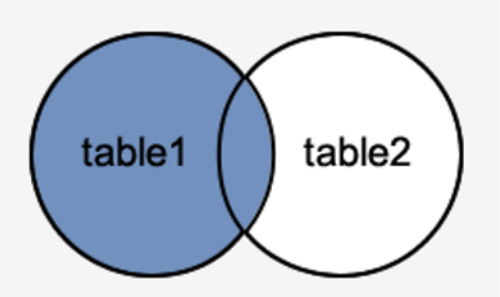
OUTPUT:

OUTPUT		
ENAME	**DEPTNO**	**DNAME**
JAY	10	ADMIN
RAJ	20	SALES
RAM	10	ADMIN
MEET	30	MARKETING
NIL	10	ADMIN

OUTER JOIN

- There are two types of outer join:
  - Left outer join
  - Right outer join
  - Full outer join

LEFT OUTER JOIN:



- It will return all rows from right table with matching rows of left table
- If left side not found than put null
- It is possible by using (+) = operator in older versions of oracle.

SYNTAX:

```
SELECT columns FROM table1 LEFT OUTER JOIN table2
ON
table1.column = table2.column;
```

EXAMPLE:

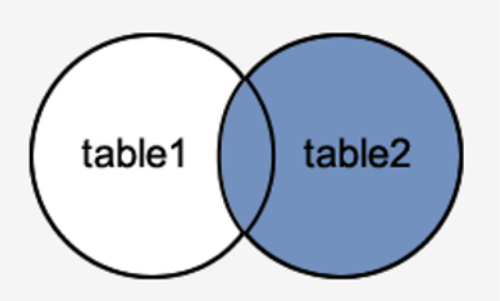
STUD	
RLNO	NAME
1	JAY
2	RAJ
3	RAM
4	MEET
5	NIL

MARK	
RLNO	PER
1	55
3	65
4	85
6	45
8	65

```
SELECT S.RLNO, S.NAME, M.PER FROM STUD S LEFT OUTER JOIN MARK M
ON
S.RLNO =M.RLNO;
```

OUTPUT		
RLNO	NAME	PER
1	JAY	55
		65
3	RAM	85
4	MEET	45
		65

RIGHT OUTER JOIN:



- It will return all rows from left table with matching rows of right table
- If right side not found than put null
- It is possible by using (+) = operator in older versions of oracle.

SYNTAX:

SELECT columns FROM table1 RIGHT OUTER JOIN table2
ON
table1.column = table2.column;

EXAMPLE:

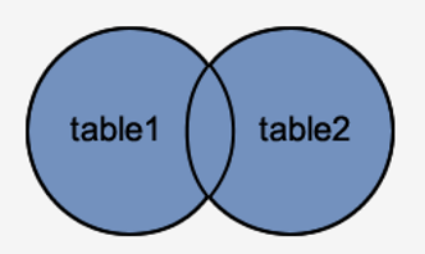
STUD	
RLNO	NAME
1	JAY
2	RAJ
3	RAM
4	MEET
5	NIL
9	AVNI

MARK	
RLNO	PER
1	55
3	65
4	85
6	45
8	65

```
SELECT S.RLNO, S.NAME, M.PER FROM STUD S RIGHT OUTER JOIN MARK M
ON
S.RLNO=M.RLNO ;
```

OUTPUT		
RLNO	**NAME**	**PER**
1	JAY	55
2	RAJ	
3	RAM	85
4	MEET	45
5	NIL	
9	AVNI	

FULL OUTER JOIN



- This type of join returns all rows from the LEFT-hand table and RIGHT-hand table with nulls in place where the join condition is not met.

```
SYNTAX:

SELECT columns FROM table1 FULL OUTER JOIN table2
ON
table1.column = table2.column;
```

EXAMPLE:

STUD	
RLNO	NAME
1	JAY
2	RAJ
3	RAM
4	MEET
5	NIL
9	AVNI

MARK	
RLNO	PER
1	55
3	65
4	85
6	45
8	65

```
SELECT  S.NAME, M.PER FROM STUD S FULL OUTER JOIN MARK M
ON
S.RLNO=M.RLNO ;
```

NAME	PER
JAY	55
RAJ	NULL
RAM	85
MEET	45
NULL	65
NIL	NULL
AVNI	NULL

SELF JOIN

- A self-join can joins the table by itself.
- WHERE clause is use in this type of join.

EXAMPLE:

STUD		
RLNO	NAME	ID
1	JAY	3
2	RAJ	5
3	RAM	1
4	MEET	4
5	NIL	2

```
SELECT S1.RLNO, S1.NAME, S2.NAME FROM STUD S1, STUD S2 WHERE
S1.RLNO=S2.ID;
```

OUTPUT		
RLNO	**NAME**	**NAME**
1	JAY	RAM
2	RAJ	NIL
3	RAM	JAY
4	MEET	MEET
5	NIL	RAJ

CROSS-JOIN

- A cross join is a join that is no join between the table.
- WHERE clause is not use in this type of join.

EXAMPLE

UNI	
ID	**C_NAME**
1	BBA
2	BCA
3	BSCIT

COLLEGE	
NO	**NAME**
1	JJK
2	MVM

EXAMPLE:

```
SELECT C.NAME, U.C_NAME FROM COLLEGE C, UNI U;
```

20. EXPLAIN DUAL TABLE

- DUEL is a small oracle inbuilt table.
- The table owned by SYS and it is available for all the users.
- The DUAL table consists only one column called D and a row containing value X.
- It supports arithmetic calculation and data formatting.
- It is also known as DUMMY table

EXAMPLE:

```
SELECT SYSDATE FROM DUAL;
```

OUTPUT:

```
D
---------
X
```

21. EXPLAIN MATHS/ NUMERIC FUNCTIONS:

ABS ()

Syntax: Abs (<number>)

Description:

This function Returns positive value of negative numbers

Example:

SELECT ABS(-55) FROM DUAL;

Output:

55

SIGN ()

Syntax: Sign (<number>)

Description:

This function Return sign of given number. If number is positive then return 1, negative then -1 & zero then return 0

Example:

SELECT Sign(-55) FROM DUAL;

Output:

-1

COS ()

Syntax: Sign (<number>)

Description:

This function Returns cos angles value.

Example:

SELECT COS(60) FROM DUAL;

Output:

-9.52413

SIN ()

Syntax: Sin (<number>)

Description:

This function Returns Sin angles value.

Example:

SELECT SIN(60) FROM DUAL;

Output:

-.3048106

TAN ()

Syntax: Tan (<number>)

Description:

This function Returns tan angles value.

Example:

SELECT TAN (60) FROM DUAL;

Output:

.32004039

CEIL ()

Syntax: ceil (<number>)

Description:

This function Returns the nearest highest integer values of an expression, which is greater than or equal to given number.

Example:

SELECT CEIL(3.544) FROM DUAL;

Output:

3

FLOOR ()

Syntax: Floor(<number>)

Description:

This function Returns the nearest lowest integer values of an expression, which is less than or equal to given number.

Example:

SELECT FLOOR(3.545) FROM DUAL;

Output:

4

TRUNC ()

Syntax: Trunc (<number>)
Description:

This function Return integer portion from given number

Example:

SELECT TRUNC(5.60) FROM DUAL;

Output:

5

ROUND ()

Syntax: Round (<number>, <significant>)
Description:

This function Returns the round of values of given expression
According to significant.

Example:

SELECT ROUND(5.458,2) FROM DUAL;

Output:

4.46

SQRT ()

Syntax: Sqrt (<number>, <significant>)
Description:

This function Returns the square root of specified number

Example:

SELECT SQRT(25) FROM DUAL;

Output:

5

MOD ()

Syntax: MODE (<n>, <M>)
Description:

This function Returns reminder by dividing **n** with **m.**

Example:

SELECT MOD (25, 3) FROM DUAL;

Output:

1

POWER ()

Syntax: Power (<n>, <m>)

Description:

This function Returns the powered value of **n** according to **m.**

Example:

SELECT POWER (5, 3) FROM DUAL;

Output:

125

LEAST ()

Syntax: Least (<n1>, <n2>...)

Description:

This function Returns maximum value from the list of values

Example:

SELECT LEAST (5, 55, -8, 11, 25, 3) FROM DUAL;

Output:

-8

GREATEST ()

Syntax: Greatest (<n1>, <n2> …)

Description:

This function Returns maximum value from the list of values

Example:

SELECT GREATEST (5, 3, 15, 22, 75, 55) FROM DUAL;

Output:

75

22. EXPLAIN AGGREGATE FUNCTIONS

- Aggregate functions known as GROUP functions.
- It produce a single result by performing calculations of field's value.
- There are five aggregate functions :
    - Sum()
    - Avg()
    - Min()
    - Max()
    - Count()

FUNCTION NAME	Syntax	DESCRIPTION	EXAMPLE
SUM()	Sum(<field name>)	Returns sum of expression	SELECT SUM(SAL) FROM EMP;
AVG()	Avg(<field name>)	Returns average of expression	SELECT AVG(SAL) FROM EMP;
MAX()	Max (<field name>)	Returns maximum value of expression	SELECT MAX(SAL) FROM EMP;
MIN()	Min (<field name>)	Returns minimum value of expression	SELECT MIN(SAL) FROM EMP;
COUNT()	Count (<field name>)	Return the count of values	SELECT COUNT(SAL) FROM EMP;

23. EXPLAIN CONVERSION FUNCTIONS:

To_Char ()

Syntax: TO_CHAR(value [, format_mask])

Description:

The Oracle/PLSQL TO_CHAR function converts a number or date to a string.

Example:

```
SELECT   TO_char (123) FROM DUAL;
SELECT To_Char(sysdate, 'yyyy/mm/dd') from dual;
```

Output:

123

2023/jan-12

To_number ()

Syntax: TO_number(value [, format_mask])

Description:

The Oracle/PLSQL TO_NUMBER function converts a string to a number.

Example:

```
SELECT   TO_number ('123') FROM DUAL;
```

Output:

123

To_date ()

Syntax: TO_date (value [, format_mask])

Description:

The Oracle/PLSQL TO_CHAR function converts a number or date to a string

Example:

```
SELECT TO_DATE('070903', 'MMDDYY') FROM DUAL;
```

Output:

09-JUL-03

24. EXPLAIN DATE FUNCTIONS

- Date functions used to perform operations on date.
- SQL provides following data functions:

SYSDATE ()

Syntax: sysdate

Description:

This function return current system date.

Example:

SELECT sysdate FROM DUAL;

Output:

01-mar-2022

SYSTIMESTAMP ()

Syntax: systimestamp

Description:

This function return current system time stamp along with date, time stamp, zone & am/pm.

Example:

SELECT systimestamp FROM DUAL;

Output:

14-MAR-22 09.26.25.831000 AM +05:30

LAST_DAY ()

Syntax: Last_day(date)

Description:

This function return a date that shows last day of the month.

Example:

SELECT last_day('01-jan-23') FROM DUAL;

Output:

31-Jan-23

NEXT_DAY ()

Syntax: Next_day (date,'weekday')

Description:

It returns the date of first weekday named by character day.

Example:

SELECT NEXT_DAY ('14-mar-23', 'Sunday') FROM DUAL;

Output:

19-MAR-23

ADD_MONTHS ()

Syntax: ADD_MONTHS (Date, N)

Description:

It returns the numbers of months between Date1 and Date2. **Example:**

SELECT ADD_MONTHS ('01-DEC-22', 2) FROM DUAL;

Output:

02-FEB-23

MONTHS_BETWEEN ()

Syntax: MONTHS_BETWEEN (Date-1, Date-2)

Description:

This function adds number of months to a specific date and it returns the date after adding number of months specified with the function.

Example:

SELECT MONTHS_BETWEEN ('2-FEB-22', '2- MAR-22') FROM DUAL; **Output:**

1

25. EXPLAIN STRING FUNCTIONS WITH EXAMPLE.

ASCII ()

Syntax: ASCII (<char>)
Description:

Returns an ASCII code value of a character.

Example:

SELECT ASCII ('A') FROM DUAL;

Output:

65

CHR ()

Syntax: CHR (<asci no>)
Description:

Converts a numeric value to its corresponding ASCII character.

Example:

SELECT CHR (65) FROM DUAL;

Output:

A

UPPER ()

Syntax: Upper (<String>/<field name>)
Description:

This function convert string into uppercase.

Example:

SELECT UPPER ('oracle') FROM DUAL;

Output:

ORACLE

INITCAP ()

Syntax: INITCAP (<String>/<field name>)
Description:

This function converts the first character in each word in a specified string to uppercase and the rest to lowercase.

Example:

SELECT INITCAP ('DATABASE MANAGEMENT SYSTEM') FROM DUAL;

Output:

Database Management System

LOWER ()

Syntax: LOWER (<String>/<field name>)

Description:

This function convert string into lowercase.

Example:

SELECT LOWER ('DATABASE') FROM DUAL;

Output:

database

LENGTH ()

Syntax: LENGTH (<String>/<field name>)

Description:

This function Returns length of given string.

Example:

SELECT LENGTH ('ORACLE') FROM DUAL;

Output:

6

LTRIM ()

Syntax: LTRIM (<String>)

Description:

This function remove extra space from leading place.

Example:

SELECT LTRIM(' ORACLE') FROM DUAL;

Output:

ORACLE

RTRIM ()

Syntax: RTRIM **(<String>/<field name>)**

Description:

The RTRIM () function removes extra spaces from right side of the string.

Example:

SELECT RTRIM ('ORACLE ') FROM DUAL;

Output:

ORACLE

TRIM ()

Syntax: TRIM (<String>)

Description:

The TRIM () function removes extra spaces from left & right side of the string.

Example:

SELECT TRIM (' ORACLE ') FROM DUAL;

Output:

ORACLE

CONCAT ()

Syntax: CONCAT (<String1>, <String 2>)

Description:

The CONCAT () function adds two or more strings together. This function is identical to || operator.

Example:

SELECT CONCAT ('SQL' , 'SERVER') FROM DUAL;

Output:

SQLSERVER

SUBSTR ()

Syntax: SUBSTR (<String>,<start no>,<length>)

Description:

This function Returns substring from a string. It contain three arguments:

- <String>: specify a string from which you want to retrieve substring
- <start no>: specify the position of the first character
- <Length>: specify total number of characters to be retrieve.

Example:

SELECT SUBSTR ('DATA QUERY LANGUAGE',6,5) FROM DUAL;

Output:

QUERY

REPLACE ()

Syntax: SUBSTR (<String>,<search string>,<replace string>)

Description:

This function Returns a string by replacing other string. It contain 3 arguments.

- <String>: specify a string from which you want to replace string
- <search string>: specify the string that you want to search for replacement
- <replace string >: specify new string that you want to replace with search string.

Example:

SELECT REPLACE ('DATA QUERY LANGUAGE','QUERY', 'DEFINITION') FROM DUAL;

Output:

DATA DEFINITION LANGUAGE

LPAD ()

Syntax: LPAD (<String1, X , <String2>)

Description:

It used to pad or add a string to the left side of the original string. This function accepts three parameter.

- <String1>: The actual string, which is to be padded.
- X: specify length of a final string after the left padding.
- <String 2 >: String that to be added to the left side of the Original Str.

Example:

SELECT LPAD ('ORACLE', 10 ,'*') FROM DUAL;

Output:

****ORACLE

RPAD ()

Syntax: RPAD(<String1, X , <String2>)

Description:

It used to pad or add a string to the right side of the original string. This function accepts three parameter.

- <String1>: The actual string, which is to be padded.
- X: specify length of a final string after the left padding.
- <String 2 >: String that to be added to the left side of the Original Str.

Example:

SELECT RPAD ('ORACLE',10,'*') FROM DUAL;

Output:

ORACLE****

USER ()

Syntax: USER

Description:

This function return current login user name.

Example:

SELECT USER FROM DUAL;

Output:

SYS

TRANSLATE ()

Syntax: TRANSLATE (String, charactersToChange, charactersTranslated)

Description: The TRANSLATE function converts characters in a string from one value to another value.

- string - this is the string of characters to be changed.
- charactersToChange - this is the list of characters to change.
- charactersTranslated - this is the list of characters that will replace the characters to be changed

Example:

SELECT TRANSLATE('ORACLE IS DBMS','DBMS','1234') FROM DUAL;

Output:

ORACLE IS 1234

26. WRITE DOWN COMPLETE NAME.

SQL	Structure query language
IBM	International business machine
ANSI	American National Standards Institute
DDL	Data definition language
DML	Data manipulation language
DCL	Data control language
DQL	Data query language
TCL	Transaction control language
DBA	Database administrator
LOB	Large Object
BLOB	Binary large object
CLOB	Character large object