ORACLE

Oracle Content (2 Months): -



Topic-1: -DBMS

Topic-2: - ORACLE

Topic-3: - SQL

->Introduction to SQL

->Sub-Languages of SQL

->Datatypes in oracle SQL

->Operators in oracle SQL

->Functions in oracle SQL

->Clauses in oracle SQL

->Joins

-.>Constraints

->Subqueries

->Views

->Sequences

->Indexes

Topic-4: - Normalization

->What is Normalization?

->Where do we want to use Normalization?

->Why do we need Normalization?

->Types of Normalization

>First normal form

>Second normal form

>Third normal form

>BCNF (Boyce-Codd normal form)

>Fourth normal form

>Fifth normal form

Topic-5: - PL/SQL

->Introduction to PL/SQL

->Difference between SQL and PL/SQL

->Conditional & Looping statements

->Cursors

->Exception Handling

->Stored procedures

->Stored Functions

->Triggers

**ORACLE**

->In IT field users are interacting with two types of applications. Those are

1.Front End Applications

2.Back End Applications

1) Front-End Application: -

->FEA is an application where the end-users interact to an application directly.

Ex: - Register from, Login Form, View Profile forms, Home page, etc.

->UI technologies are HTML, CSS, JavaScript, Angular JS, React JS, etc.

2) Back-End-Application: -

->BEA is an application where we store the end-user’s data/information.

Ex: - Database.

->DB technologies are ORACLE, SQL SERVER, MYSQL, POSTGRESQL, DB2, etc.

=> Server- Side Technologies: -

-> These technologies are used to establish a connection between front end application and back- end application.

-> Server- Side Technologies are Java, .Net, Python, PHB, etc.

SERVER

Java

JDBC

FEA

.Net

Database

ADO.NET

UI Form

**Topic – 1(DBMS)**

=> What is Data?

->It is a raw fact. (i.e. Characters, Numbers, Special characters and Symbols)

->Data never gives meaningful statements to users.

Ex: - SMITH is data 10021 is data

MILLER is data 10022 is data

ADAMS is data 10023 is data

=>What is information?

->Processing data is called information.

->Information always provides meaningful statements to users

Ex: - Employee Name Employee Id

SMITH 10021

MILLER 10022

ADAMS 10023

=>What is Database?

->It is a memory which is used to store inter-related information of a particular organization.

=>What is inter-related information?

->Depends on each other.

Ex: - SBI Organization

-group of branches ----------------🡪Group of Customers

>group of departments

>group of employees.

Ex: - no department = no employees

no employees = no department

no customers = no products

no products = no customers.

=>Types of Databases?

->There are two types of databases in the real world. Those are

1.OLTP ( Online Transaction Processing )

2.OLAP ( Online Analytical Processing )

1.OLTP: -

->These databases are used for saving/storing “day-to-day” transactional information.

Ex: - Oracle, SQL SERVER, MySQL, PostgreSQL, Db2, etc.

2.OLAP: -

->These databases are used for storing “historical data/information “. (i.e. Bigdata)

Ex: - Data Warehouse (DWH)

OLAP db (dw)

Day-to-day

OLTP DB

App1

Day-to-day

ETL

App2

E-Extract

T-Transfer

L-Loading

User

Day-to-day

App3

Ex: - Informatica, SSLS (msbi), power bi, Apache airflow.

=> What is DBMS?

->It is software which is used to manage and maintain data/information within the database memory.

->By using DBMS s/w we will perform the following operations are:

>Creating database

>Creating tables

>Inserting data

>Updating data

>Selecting data

>Deleting data

->Here DBMS s/w will act as an interface between user and database.

USER

DBMS s/w (Interface)

Database

=> DBMS models?

->There are three types of DBMS models are there. Those are

1.Hierarchial Database Management System (HDBMS)

Ex: - IMS s/w (Information Management System).

2.Network Database Management System (NDBMS)

Ex: - IDBMS s/w (Integrated database management system)

NOTE: - HDBMS, NDBMS models are outdated in real time.

3. Relational Database Management System (RDBMS):

->There are two modules in RDBMS, those are

i. Object Relational Database Management System (ORDBMS)

ii. Object Oriented Database Management System (OODBMS)

i. ORDBMS: -

->Data can be stored/organized in the form of tables.

->A table is a collection of rows and columns

>Rows can be called as “record/tuples”.

>Columns can be called ad “attributes/fields”.

->A row is nothing but a group of columns in a table.

-> These databases are depending on “SQL”. So that these are called as “SQLDATABASES” in real-time.

Ex: - Oracle, SQL Server, MySQL, Db2, etc.

ii. OODBMS: -

->Data can be stored/organized in the form of “objects”.

->These databases are depending on “OOPS Concept “but not SQL. So that these are called as “NOSQL DATABASES” in real time.

**TOPIC-2(ORACLE)**

=> Introduction to Oracle: -

-> Oracle is an RDBMS product/ORDBMS module which was introduced by “Oracle Corporation” in 1979. Oracle is used to store data/information permanently and security.

->Oracle can be deployed in any platform like Windows, Linux, Unix, Solaris, MAC,. Etc.

->Oracle is a platform independent an RDBMS product.

=>What is platform?

->It is combination of operating systems and Micro Processor.

->There are two types of platforms. Those are

i. Platform dependent

ii. Platform independent

i. Platform dependent: -

->It supports only one operating system with the combination of any micro-processor.

Ex: - Cobal, Pascal, C++, C.

ii. Platform independent: -

->It supports any operating system with the combination of any micro-processor.

Ex: - Oracle, Java, .net core, Python, Etc.

=>Types of oracle software editions: -

There are two types of editions. Those are

i. Oracle Express Edition: -

->Supporting partial features of oracle.

Ex: - recycle bin, flash back, purge, partition, tables, are not allowed.

ii. Oracle Enterprise Edition: -

->Supporting all features of oracle.

Ex: - recycle bin, flashback, purge, partition, tables, are allowed.

=>Versions of oracle software’s: -

->The first version of oracle software is “Oracle 1.0”.

- Oracle 1.0

- Oracle 2.0

- Oracle 3.0

- Oracle 4.0

- Oracle 5.0

- Oracle 6.0

- Oracle 7.0

- Oracle 8.0

- Oracle 8i (Internet)

- Oracle 9i

- Oracle 10g (Grid technologies)

- Oracle 11g

- Oracle 12c (Cloud technologies)

- Oracle 18c

- Oracle 19c

- Oracle 20c

- Oracle 21c

- Oracle 23c (Beta Version)

=>How to download oracle 19c enterprise edition software: -

->Supports on WOS-10 or WOS-11. WOS [Windows Operating System]

->Supports on Hard Disk (HD)-500gb, SSD – 60, RAM – 1gb+.

->It’s only valid for 90 days (3 months).

->For downloading oracle s/w we should create an account in oracle website.

->Download URL= ([https://www.oracle.com/in/database/technologies/oracle 19c-windows-download.html](https://www.oracle.com/in/database/technologies/oracle%2019c-windows-download.html)).

=>How to install oracle 19c/21c enterprise edition software: -

->Follow the provided video in google classroom.

NOTE: -

->Once we installed oracle software there are two components installed in the system.

i. Client Component

ii. Server Component

i. Client Component: -

->By using client components, we will perform the following three steps:

Step-1: - User can connect to oracle server: -

Enter username : System (default username)

Enter password : LION (Created at installation level)

CONNECTED>>

Here, Username is not case-sensitive, but password is case-sensitive.

Step-2: - User can send requests to an oracle server: -

Request: SQL query/SQL Command.

Step-3: - User can get response from oracle server: -

Response: Result/Output.

Ex: - Client tools = {SQL PLUS [CUI (Character user interface) Environment, Manual Coding, using this s/w for whole oracle course], SQL Developer, Toad [These two are GUI (Graphical user interface) Environment]

ii. Server Components: -

->There are two more sub-components in Server.

a. Instance

b. Database

a. Instance: -

->It is a temporary memory which will be allocated from RAM (Random Access Memory).

->Here, data can be stored temporarily

b. Database: -

->It is a permanent memory which will be allocated from Hard Disk.

->Here, data can be stored permanently.

Ex: - ATM (Automatic Teller Machine) is the real time example.

SERVER

🡺**Client-Server Architecture: -**

CILENT

2.Database

Request to

1.Instance

Client Tools:

SQL PLUS, TOAD,SQL DEVELOPER,SQL NAVIGATOR,…etc.

Temporary Storage

Permanent Storage

Instance Memory

(allocate from RAM)

USER

Response from

DATABASE MEMORY

(Allocates from HD)

NOTE: -

->When we want to work with oracle database server. We need to follow the following two procedures

Step-1: Connect

Step-2: Communicate

Step-1: - Connect:

->When we want to connect to oracle, we need a client tool is known as “SQLPLUS”

Step-2: - Communicate:

->When we want to communicate with database we need a database language is known as “SQL”.

🡺SQL PLUS vs SQL: -

|  |  |
| --- | --- |
| SQL PULS | SQL |
| 1.It is a client tool which was introduced by “Oracle corporation”. | 1.It is a database language which was introduced by “IBM”. |
| 2.It is used to connect to an oracle server. | 2.It is used to communicate with database. |
| 3.It will act as an editor for writing & executing SQL queries and PL/SQL programs | 3. SQL contains the following five sub-languages are (DDL, DML, DQL/DRL, TCL, DCL) used to perform some operations over database. |

=> How to connect to oracle server: -

> Go to all programs

> Open oracle – oraDB19home | folder

> Click on SQLPLUS icon

Enter username: SYSTEM

Enter password: LION

Connected.

=> How to create a new username and password in oracle:

Syntax: -

Create user <username> identified by <password>;

Ex: -

SQL> CONN

Enter username: SYSTEM/LION

Connected.

SQL> CREATE USER MYDB9AM IDENTIFIED BY 123.

User created.

SQL> CONN

Enter username: MYDB9AM/123

Error:

ORA-01045: User MYDB(AM lacks CREATE SESSION privileges;

Logon denied.

NOTE: -

->In oracle database server every new user is called as “dummy user” (i.e., no permission)

If the user want to connect to oracle server then the user need a permission from DBA (System).

=> How to give all permissions to user:

Syntax:

Grant <privilege name> to <username>;

Ex:

SQL> CONN

Enter username: SYSTEM/LION

Connected.

SQL> GRANT DBA TO MYDB9AM ;

Grant succeeded.

SQL> CONN

Enter username: MYDB9AM

Enter password: 123

Connected.

=> How to change password for user:

Syntax:

Password;

Ex:-

SQL> CONN

Enter username: MYDB9AM/123

Connected.

SQL> PASSWORD;

Changing password for MYDB9AM

Old password: 123

New password: ABC

Retype new password: ABC

Password changed.

SQL> CONN

Enter username; MYDB9AM/ABC

Connected.

=> How to re-create a new password if we forget it:

Syntax:

Alter user <username> identified by <new password>;

Ex:-

SQL> CONN

Enter username: SYSTEM/LION

Connected.

SQL>ALTER USER MYDB9AM IDENTIFIED BY MYDB9AM;

User altered.

SQL> CONN

Enter username: MYDB9AM/MYDB9AM

Connected.

=> How to view usernames in oracle if we forget it:

Syntax:

Select username from all\_users;

Ex: -

SQL> CONN

Enter username: SYSTEM/LION

Connected.

SQL> SELECT USERNAME FROM ALL\_USERS;

=> How to drop a user:

Syntax:

Drop user <username> cascade;

Ex: -

SQL> CONN

Enter username: SYSTEM/LION

Connected.

SQL> DROP USER MYDB9AM CASCADE;

User dropped.

=> How to clear the screen of SQL plus editor:

Syntax:

Cl scr;

Ex: -

SQL> CL SCR;

=> How to disconnect from oracle server:

Syntax:

Exit;

Ex: -

SQL> EXIT;

**TOPIC-3(SQL)**

🡺Introduction to SQL: -

->SQL [Structured Query Language] is database language which was introduced by “IBM”.

->SQL is used to communicate with any database in real-time.

Ex:- Oracle, MySQL, SQL server, db2, Sybase, PostgreSQL, etc.

-> Initially SQL is called as “SEQUEL” language and later renamed as “SQL”.

-> SQL is not a case-sensitive language. i.e. user can write SQL queries in either upper/lower/combination of upper and lower case characters.

Ex: -

SELECT \* FROM EMP; -----------🡪executed

Select \* from emp; ------------------🡪executed

Select \* From Emp; -----------------🡪executed

-> In oracle storage of data is a case-sensitive.

-> Every SQL query should ends with “;” (Semi-colon).

🡺Sub – Languages of SQL :

1. Data Definition Language (DDL) :

-> CREATE

-> ALTER

> ALTER – MODIFY

> ALTER – ADD

> ALTER – RENAME

> ALTER – DROP

-> RENAME

-> TRUNCATE

-> DROP

==>New features in oracle-10g enterprise edition are :

-> RECYCLEBIN

-> FLASHBACK

->PURGE

2. Data Manipulation Language (DML) :

-> INSERT

-> UPDATE

-> DELETE

3. Data Query/Retrieval Language (DQL/DRL) :

-> SELECT (read only)

4. Transaction Control Language (TCL) :

-> COMMIT

-> ROLLBACK

-> SAVEPOINT

5. Data Control Language (DCL) :

-> GRANT

-> REVOKE

=========================================================

1. (DDL) :

=> CREATE: -

-> It is used to create a new database object such as Table, View sequence, Index, Procedure, Function, Triggers, etc.

\* How to create a new table in oracle :

Syntax:

Create table <table name>(<Column name 1)<data type>[size], <Column name 2)<data type>[size], <Column name 3)<data type>[size],…);

\* Data types in oracle:

-> Data type is an attribute which is used to store “What type data” into a column.

-> Oracle supports the following datatypes are,

>Number datatype

>Character/String datatype

>Long datatype

>Date datatype

> Raw & long raw datatype

> LOB datatype

i. Number datatype:

a. Number (p)

b. Number (p,s)

a. Number (p):

-> To store integer format data only.

b. Number (p,s):

-> To store integer and also float values.

Precision (p) :

-> Counting all digits including left and right sides of a decimal point in the expression.

-> The maximum size of precision is 38 digit.

Ex:- 126 🡺 p=3

756323 🡺 p=6

45.23 🡺 p=4

9585.35 🡺 p=6

Scale (s):

-> Counting the right-side digits of a decimal point in the expression.

-> There is no maximum size of scale because it is a part of precision value.

Ex: - 45.23 🡺 p=4, s=2

9585.351 🡺 p=7, s=3

NOTE: - Here the scale should be less than the precision value.

Ex: -

Sno number (3) PRICE number (8,2)

0 0.0

1 25.18

2 85.17

. .

. .

. .

. .

999 999999.99

1000 – error 1000000(1000000.00) – error

ii. Character/String datatype: -

-> Storing string format data only.

-> In database string can be represents with ‘<string>’.

Ex:- ENAME Char (10)

Smith -> Error

‘Smith’ -> Smith (Accepted)

1234 -> Error

‘1234’ -> 1234

34.12 -> Error (Accepted)

’34.12’ -> 34.12 (Accepted)

String Format

||

Characters only String Alphanumeric String Format

Format ||

|| [A-Z, a-z, 0-9, @,$,#,.etc]

[A-Z, a-z] Ex:-‘Smith123@gmail.com’

Ex:- ‘smith’, ‘SMITH’, .etc ,Pan Card, HTNO,.etc..

Types Of String Datatypes: -

-> These datatypes again classified into two types.

1. Non-unicode datatype: -

-> Supplying to store “localized data”.(i.e., English Language only)

i. Char (size)

ii. Varchar2 (size)

i. Char (size): -

->It is a fixed length datatype(i.e., Static memory)

-> It will store non-unicode character in the form of 1 char = 1 byte.

-> The maximum size of char datatype is 2000 bytes.

=> Disadvantage: -

-> Memory wasted because of it is static.

ii. Varchar2 (size): -

-> It is a variable length datatype (i.e dynamic memory)

-> It will store non-unicode characters in the form of 1 char = 1 byte

-> The maximum size of varchar2 datatype is 4000 bytes.

=>Advantage: -

-> Memory saved not wasted.

Fixed Length Datatype

NAME CHAR (10)

10 Bytes

‘HELLO’

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | E | L | L | O |  |  |  |  |  |

HELLO

HEL

HE

‘HEL’

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | E | L |  |  |  |  |  |  |  |

HERE, memory wasted.

‘HE’

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | E |  |  |  |  |  |  |  |  |

Variable Length Datatype

NAME VARCHAR2 (10)

10 Bytes

‘HELLO’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| H | E | L | L | O |

HELLO

HEL

HE

‘HEL’

|  |  |  |
| --- | --- | --- |
| H | E | L |

HERE, No memory wasted.

‘HE’

|  |  |
| --- | --- |
| H | E |

2. Unicode datatypes: -

-> Supporting to store “globalized data” (i.e. All National Language)

i. NChar (size)

ii. NVarchar (size)

Here, N stands for ‘National Language’-[196 languages]

i. NChar (size) :

-> It is a fixed length datatype (i.e, Static memory)

-> It will store Unicode characters in the form of 1 Char = 1 byte.

-> The maximum size of NChar datatype is 2000 bytes.

Disadvantages: -

->Memory wasted.

ii. NVarchar2 (size) :

-> It is a variable length datatype (i.e, dynamic memory)

-> It will store Unicode characters in the form of 1 char = 1 byte.

-> The maximum size of Nvarchar2 datatype is 4000 bytes.

Advantages: -

-> Memory saved.

\* Long datatype: -

-> It is a variable length datatype(i.e, dynamic memory)

-> It will store both non-unicode & Unicode characters in the form of 1 char = 1 byte.

-> The maximum size of long datatypes is 2gb.

\* Date datatype: -

-> To store date and time information of a particular day/transaction.

-> The range of date datatype in oracle is ’01-JAN-4712’BC to ’31-DEC-9999’ AD.

-> There are two types of date datatypes

a. Date

b. Timestamp

a. Date :-

-> It will store date and time information but time is optional.

-> When user did not insert time in oracle will take time ’00:00:00 am’ by default.

-> By default format of oracle is ‘DD:MON:YY/YYYY HH:MI:SS’

-> The maximum size is 7 bytes and it is a fixed memory.

Ex: - ‘DD-MON-YY/YYYY HH:MI:SS’

’04-JAN-25/2025 00:00:00

1 1 2 1 1 1 🡪7 bytes(fixed)

-> Even if we not provide time value it will take 7 bytes.

b. Timestamp: -

-> It will store date and time information along with milliseconds.

-> The default format of timestamp is ‘DD:MON-YY/YYYY HH:MI:SS:MS’

-> The maximum size is 11 bytes and it is a fixed memory.

Ex: - ‘DD-MON-YY/YYYY HH:MI:SS:MS’

’01-JAN-25/2025 00:00:00:00’

1 1 2 1 1 1 4 🡪11 bytes

\* Raw and Long Raw Datatypes: -

-> These datatypes are used for storing image file/audio file/video files in the form of 01010101010 binary format.

>Raw - Static datatype - 2000 bytes

>Long Raw - Dynamic datatype - 2gb

\* LOB datatype: -

-> LOB stands for large objects datatype

i. BLOB

ii. CLOB

iii. NCLOB

i. BLOB :-

-> It stands for binary large object.

-> It will store image/audio/video files in the form of 010101010001 binary format.

-> It is dynamic datatype.

-> The maximum size is 4gb.

ii. CLOB :-

-> It stands for Character Large Object.

-> It will store non-unicode characters in the form of 1 char = 1 byte

-> It is a dynamic datatype.

-> The maximum size is 4gb.

iii. NCLOB :-

-> It stands for National Character Large Object.

-> It will store Unicode characters in the form of 1 char = 1 byte.

-> It is a dynamic datatype.

->The maximum size is 4gb.

🡺Non-Unicode Characters:-

* Char (size) - 2000 bytes
* Varchar (size) - 4000 bytes
* Long - 2gb
* CLOB - 4gb

🡺Unicode Characters:-

* NChar (size) - 2000 bytes
* NVarchar (size) - 4000 bytes
* Long - 2gb
* NCLOB - 4gb

🡺Binary data:-

* Raw - 2000 bytes
* Long Raw - 2gb
* BLOB - 4gb

\* How to create a new table in oracle :-

Syntax:-

Create table <table name>(<column name1><datatype>[size],<column name2><datatype>[size],…);

Ex:-

SQL>CONN

Enter user-name: MYDB9AM/MYDB9AM

Connected

SQL> CREATE TABLE STUDENT(STID NUMBER(4),SNAME CHAR(8),SFEE(8,2));

Table created.

\* How to view the structure of a table in oracle:-

Syntax:-

Desc <table name>; (describe command)

Ex:-

SQL> DESC STUDENT;

\* How to view list of table in oracle window:-

Syntax:-

Select \* from tab; (TAB is pre-defined table)

Ex:-

SQL> SELECT \* FROM TAB;

==> ALTER:

-> It is used to modify/change the structure of a table.

-> There are four sub-commands of alter.

i. ALTER-MODIFY

ii. ALTER-ADD

iii. ALTER-RENAME

iv. ALTER-DROP

i. ALTER-MODIFY:

-> To change datatype from one datatype to another datatype and also the size of the datatype of a specific datatype.

Syntax:-

ALTER TABLE<TABLE NAME>MODIFY<COLUMN NAME><NEW DATATYPE>[NEW SIZE];

Ex:

SQL> ALTER TABLE STUDENT MODIFY SNAME VARCHAR2(20);

ii. ALTER-ADD:

->To add new column to an existing table

Syntax:

ALTER TABLE <TABLE NAME>ADD NEW<NEW COLUMN NAME><DATATYPE>[SIZE];

Ex:

ALTER TABLE STUDENT ADD SADDRESS VARCHAR2(50);

iii. ALTER-RENAME:

->To rename a column name in the table.

Syntax:

ALTER TABLE<TABLE NAME>RENAME COLUMN<OLD COLUMN NAME>TO<NEW COLUMN NAME>;

Ex:

ALTER TABLE STUDENT RENAME COLUMN SNAME TO STUDENT NAMES;

iv. ALTER-DROP:

->To drop/delete a column from an existing table.

Syntax:

ALTER TABLE<TABLE NAME>DROP COLUMN<COLUMN NAME>;

Ex:

ALTER TABLE STUDENT DROP COLUMN SFEE;

==> RENAME:

->To change a table name.

Syntax:

RENAME<OLD TABLE NAME>TO<NEW TABLE NAME>;

Ex:

SQL> RENAME STUDENT TO STUDENT\_DETAILS;

SQL> RENAME STUDENT\_DETAILS TO STUDENT;

==> TRUNCATE:

->Deleting all rows but not columns of table.

->We cannot delete a specific row from a table by using TRUNCATE because it does not allow “WHERE” clause condition.

->Deleting all rows from a table permanently

Syntax:

TRUNCATE TABLE<TABLE NAME>;

Ex:

SQL> TRUNCATE TABLE STUDENT WHERE STID=1001; --NOT ALLOWED.

SQL> TRUNCATE TABLE STUDENT; --ALLOWED.

==> DROP:

-> To drop the entire table. (i.e., collection of rows & columns)

Syntax:

DROP TABLE<TABLE NAME>;

Ex:

SQL> DROP TABLE STUDENT;

-> Before oracle-10g enterprise edition once we drop a table from a database then it was permanently deleted. Whereas from oracle-10g enterprise once we drop a table from a database then it was temporarily deleted.

\* New features in the oracle-10g enterprise edition:

- RECYCLEBIN

- FLASHBACK

- PURGE

==> RECYCLEBIN:

-> It is a system defined table which is used to store the information about deleted tables from database.

-> It is a similar to recyclebin in a computer.

\*. How to view deleted tables in recyclebin;

Syntax:

SELECT OBJECT NAME, ORIGINAL\_NAME FROM RECYCLEBIN;

OBJECT\_NAME ORIGINAL\_NAME

BIN$LMtpjn+7TWOIFBg2W+UBIQ==$O STUDENT

==> FLASHBACK:

-> It is a DDL command which is used to restore a table from recyclebin to database.

Syntax;

FLASHBACK TABLE<TABLE NAME>TO BEFORE DROP;

Ex:

SQL> FLASHBACK TABLE STUDENT TO BEFORE DROP;

==> PURGE:

->It is DDL command which is used to drop a table permanently.

Syntax:

DROP TABLE<TABLE NAME>PURGE;

Ex:

SQL> DROP TABLE STUDENT PURGE;

2. DATA MANIPULATE LANGUAGE(DML):

==> INSERT:

-> To insert a new row.(i.e., data) into a table.

-> There are two methods to insert data into a table.

METHOD-1:

-> To insert values for all columns.

Syntax:

INSERT INTO<TABLE NAME>VALUES(VALUE 1,VALUE 2,…);

Ex:

SQL> INSERT INTO STUDENT VALUES(1021,’SMITH’,25000);

METHOD-2:

-> To insert values for required columns.

Syntax:

INSERT INTO<TABLE NAME>(REQUIRED COLUMN NAMES)VALUES(VALUE 1,VALUE 2,……);

Ex:

SQL> INSERT INTO STUDENT(STID) VALUES (1022);

SQL> INSERT INTO STUDENT (STID, SFEE) VALUES (1023,32000);

SQL> INSERT INTO STUDENT (STID, SNAME, SFEE) VALUES(‘1024’,’ALLEN’,45000);

SQL> INSERT INTO STUDENT (SNAME, SFEE, STID) VALUES(MILLER’,18000,1025);

\*How to insert values into a table dynamically:

METHOD-1:

Syntax:

INSERT INTO<TABLE NAME>VALUES<&<COLUMN NAME1>,&<COLUMN NAME2>,……);

Ex:

SQL>INSERT INTO STUDENT VALUES(&STID,’&SNAME’,&SFEE);

Enter value for stid:1026

Enter value for sname:JAMES

Enter value for sfee: 58000

SQL>/

Enter value for stid: 1027

Enter value for sname:SCOTT

Enter value for sfee: 38000

METHOD-2:

Syntax:

INSERT INTO<TABLE NAME>(REQUIRED COLUMN NAMES)VALUES(&<COLUMN NAME1>,&<COLUMN NAME2.,…..);

Ex:

SQL>INSERT INTO STUDENT(STID)VALUES(&STID);

Enter value for stid: 1029

SQL>/

Enter value for stid: 1030

SQL>/

----------------------------------

----------------------------------

----------------------------------

==> UPDATE:

-> To update all rows data in a table at a time.

(or)

-> To update a specific row data in a table by using “WHERE” clause condition.

Syntax:

UPDATE<TABLE NAME>SET<COLUMN NAME1>=<VALUE1>,<COLUMN NAME2>=<VALUE2>,……..[WHERE <CONDITION>];

Ex:

SQL>UPDATE STUDENT SET SNAME=’SMITH’ WHERE STID=1023;

SQL>UPDATE STUDENT SET SNAME=’FORD’,SFEE=15000 WHERE STID=1021;

SQL>UPDATE STUDENT SET SFEE=25000;

==> DELETE:

->To delete all rows from a table at a time.

(or)

-> To delete a specific row from a table by using “WHERE” conditions.

Syntax:

DELETE FROM<TABLE NAME>[WHERE<CONDITION>];

Ex:

SQL>DELETE FROM STUDENT WHERE SNAME=’SMITH’;

SQL>DELETE FROM STUDENT;

NOTE: -Using DELETE we can delete all rows from table but not deleting the structure of the table.

\*Difference between DELETE vs TRUNCATE:

|  |  |
| --- | --- |
| DELETE | TRUNCATE |
| 1.It is a DML operation. | 1. It is a DDL operation. |
| 2.Deleting specific row. | 2.Cannot delete a specific row. |
| 3.Supporting “WHERE” clause | 3.Does not support “WHERE” clause. |
| 4.Delete data temporarily. | 4.Deleting data permanently. |
| 5.We can restore deleted data by using “ROLLBACK”. | 5.We cannot restore deleted data by using “ROLLBACK’. |
| 6.Deleting rows in one-by-one manner. | 6.Deleting rows as a page wise. |
| 7.Execution speed is slow. | 7.Execution speed is fast. |

3.(DQL/DRL)Data Query/Retrieval Language:

==>SELECT:

-> To retrieve all rows from a table at a time.

(or)

-> To retrieve a specific row from a table by using “WHERE” condition.

Syntax:

SELECT\*/<LIST OF COLUMN NAMES>FROM<TABLE NAME>[WHERE<CONDITION>];

Ex:

SQL>SELECT \* FROM EMP WHERE JOB=’MANAGER’;

SQL>SELECT \* FROM DEPT WHERE DNAME=’SALES’;

SQL>SELECT \* FROM DEPT;

SQL>SELECT \* FROM EMP;

\*Operators:

==>CONCATENATION OPERATOR(||):

-> To add two or more than two expressions.

Syntax:

<EXPRESSION 1>||<EXPRESSION 2>||<EXPRESSION 3>||,….

Ex:

SQL>SELECT ‘THE EMPLOYEE’||’ ‘||ENAME||’ ‘||’IS WORKING AS A’||’ ‘||JOB FROM EMP;

==>DISTINCT keyword:

->To eliminate duplicate values from a column temporarily.

Syntax:

DISTINCT<COLUMN NAME>

Ex:

SQL>SELECT DISTINCT JOB FROM EMP;

SQL>SELECT DISTINCT DEPTNO FROM EMP;

==> ALIAS NAME:

->It is a temporary name/alternate name for column and table.

->Alias names are created at two levels.

i.Column level alias names:

->In this level we are creating alias name for columns.

ii. Table level alias names:

->In this level we are creating alias name for table.

Syntax:

SELECT<COLUMN NAME>[AS]<COLUMN ALIAS NAME1>,<COLUMN NAME2>[AS]<COLUMN ALIAS NAME2>,…. FROM <TABLE NAME>[AS]<TABLE ALIAS NAME>;

Ex:

SQL>SELECT DEPTNO AS DEPARTMENT\_ID,DNAME AS DEPARTMENT\_NAME,LOC AS DEPARTMENT\_LOCATION FROM DEPT DEPARTMENT;

(or)

SQL>SELECT DEPTNO DEPARTMENT\_ID,DNAME AS DEPARTMENT\_NAME,LOC DEPARTMENT\_LOCATION FROM DEPT DEPARTMENT;

=>NOTE:

->When we want to display the large scale data tables in proper systematically in SQLPLUS editor then we must set the following two properties are,

1. PAGESIZE n

2. LINES n

1. PAGESIZE n:

->By default each page contains 14 rows in SQLPLUS editor.

->If we want to display more than 14 rows in a single page then we set pagesize property.

->Here ‘n’ is represent no of rows in a page and the maximum size of pagesize property is 5000 rows.

Ex:SQL>SET PAGESIZE 100;

2. LINES n:

->By default each line contains 80 bytes(i.e 80 chars) in SQLPLUS editor.

->If we want to display more than 80 characters in a single line then we set line property.

->Here ‘n’ is nothing but no of bytes/characters in a line.

-.Maximum bytes/characters of a line is 32767 bytes/characters.

Ex:SQL>SET LINES 160;

\*OPERATORS IN ORACLE SQL:

->To perform some operations on the given operand value.

->Oracle supports the following operators those are,

🡺Assignment operators -> =

🡺Arithmetic operators -> +,-,\*,/

🡺Relational operators -> <,>,<=,>=,!=,(or)<>

🡺Logical operator s -> AND,OR,NOT

🡺Set operators -> UNION,UNION ALL,

INTESECT,MINUS.

🡺Special operators -> (+ve) (-ve)

IN NOT IN

BETWEEN NOT BETWEEN

IS NULL IS NOT NULL

LIKE NOT LIKE

=> Assignment operators:

->To assign a values to an attribute/to a variable.

Syntax: <column name><assignment operator><value>

Ex:

SQL>UPDATE EMP SET SAL=34000;

SQL>UPDATE EMP SET JOB=’HR’ WHERE EMPNO=7788;

=>Arithmetic operators:

->To perform addition, subtraction, multiple and division.

Syntax:<column name><arithmetic operator><value>

Ex: Write a query to display all employees salaries after adding 1000/-?

SQL>SELECT ENAME, SAL AS OLD\_SALARY, SAL+1000 AS NEW\_SALARY FROM EMP;

Ex: Write a query to display EMPNO, ENAMES, DEPTNO, BASIC\_SALARY and ANNUAL\_SALARY of the employees who are working under deptno is 10?

SQL>SELECT EMPNO, ENAME, DEPTNO, SAL AS BASIC\_SALARY,

2 SAL\*12 AS ANNUAL\_SALARY FROM EMP WHERE DEPTNO=10;

Ex: Write a query to display employee’s salaries after increment of 5%?

SQL>SELECT ENAME, SAL AS BEFORE\_INCREMENT,

2 SAL+SAL\*5/100 AS AFTER\_INCREMENT FROM EMP;

Ex: Write a query to display EMPNO, ENAME, BASIC\_SALARY, 5% of HRA, 10% of DA, 5% of PF, GROSS\_SALARY and NET\_SALARY of employees who are working as a MANAGER?

SQL>SELECT EMPNO, ENAME, SAL AS BASIC\_SALARY,

2 SAL\*0.05 AS HRA, SAL\*0.1 AS DA, SAL\*0.05 AS PF,

3 SAL+SAL\*0.05+SAL\*0.1+SAL\*0.05 AS GROSS\_SALARY,

4 SAL+SAL\*0.05+SAL\*0.1-SAL\*0.05 AS NET\_SALARY,

5 FROM EMP WHERE JOB=’MANAGER’;

Ex: Write a query to display employee’s salaries after decrement of 5%?

SQL> SELECT ENAME, SAL AS BEFORE\_DECREMENT,

2 SAL-SAL\*0.05 AS AFTER\_DECREMENT FROM EMP;

=>Relational operators:

->Comparing a specific column value with user defined condition in the query.

Syntax:

Where<column name><relational operator><value>

Ex: Write a query to display the list of employees who are joined after 1981?

SQL>SELECT \* FROM EMP WHERE HIREDATE>’31-DEC-1918’;

Ex: Write a query to display the list of employees who are joined before 1981?

SQL>SELECT \* FROM EMP WHERE HIREDATE<’01-JAN-1981’;

=>Logical operator:

->To check more than one condition in the query.

->AND, OR, NOT.

=>> AND operator:

->It return a value if both conditions are TRUE in the query.

Condition-1 Condition-2

T T => T

T F => F

F T => F

F F => F

Syntax: where<condition 1> and <condition 2>

Ex: Write a query to display employees whose name is ‘SMITH’ and working as ‘CLERK’?

SQL>SELECT \* FROM EMP WHERE ENAME=’SMITH’ AND JOB=’CLERK’;

=>>OR operator:

-> It return a value if any one condition is TRUE in the query.

Condition-1 Condition-2

T T => T

T F => T

F T => T

F F => F

Syntax: where<condition 1>or<condition 2>

Ex: Write a query to display employees details who are working under the job is ‘ANALYST’,’PRESIDENT’?

SQL> SELECT \* FROM EMP WHERE JOB=’ANALYST’ OR JOB=’PRESIDENT’;

=>>NOT operator:

->It return all values from a table except the given conditional values in the query.

Syntax: not<condition 1>and not<condition 2>

Ex: Write a query to display employees who are not working under deptno 10 and 30?

SQL>SELECT \* FROM EMP WHERE NOT DEPTNO=10 AND NOT DEPTNO=30;

=> SET operators:

->Set operators are used to combined the results of two select quries.

Syntax: <select query 1><set operator><select query 2>

Ex: A={10,20,30}, B={30,40,50}

-UNION:

->To combined two sets values without duplicates.

A U B={10,20,30,40,50}

-UNION ALL:

->To combined two sets values with duplicates.

A UL B={10,20,30,30,40,50}

-INTERSECT:

->It returns common values from both sets.

A | B={30}

-MINUS:

->It returns uncommon values from the left side but not the right side set.

A – B={10,20}

B – A={40,50}

DEMO TABLES:

SQL> SELECT \* FROM EMP\_HYD;

EID ENAME SAL

1021 SMITH 85000

1022 ALLEN 65000

1023 WARD 38000

SQL> SELECT \* FROM EMP\_MUMBAI;

EID ENAME SAL

1021 SMITH 85000

1024 MILLER 69000

Ex: Write a query to display employees who are working in both branches?

SQL>SELECT \* FROM EMP\_HYD INTERSECT SELECT \* FROM EMP\_MUMBAI;

Ex: Write a query to display employees who are working in HYD but not in MUMBAI?

SQL> SELECT \* FROM EMP\_HYD MINUS SELECT \* FROM EMP\_MUMBAI.

Ex: Write a query to display all employees details who are working in the organization?

SQL> SELECT \* FROM EMP\_HYD UNION ALL SELECT \* FROM EMP\_MUMBAI; (including duplicates).

SQL> SELECT \* FROM EMP\_HYD UNION SELECT \* FROM EMP\_MUMBAI; (excluding duplicates)

=> Special operators:

\*IN operator:

->Comparing group of values with a single condition.

Syntax:

Where<column name> in (<list of values>);🡺(+ve)

Where<column name>not in (<list of values>);🡺(-ve)

Ex: Write a query to display the list of employees who are working as “CLERK”,”SALESMAN”,”MANAGER”? 🡺(+VE)

SQL>SELECT \* FROM EMP WHERE JON IN

(‘CLERK’,’SALESMAN’,’MANAGER’);

Ex: Write a query to display the list of employees who are not working as “CLERK”,”SALESMAN”,”MANAGER”?

SQL>SELECT \* FROM EMP WHERE JON IN

(‘CLERK’,’SALESMAN’,’MANAGER’);

\*BETWEEN Operator:

->Comparing a particular range value.

Syntax:

Where<column name>between<low value>and<high value>; 🡺(+ve)

Where<column name>not between<low value>and<high value>; 🡺(-ve)

NOTE:

->Between operator will return values including source and destination value from the given range value.

Ex: Write a query to display list of employees who are joined in 1981?

SQL>SELECT \* FROM EMP WHERE HIREDATE BETWEEN ’01-JAN-1981’ AND ’31-DEC-1981’;🡺(+ve)

Ex: Write a query to display list of employees who are not joined in 1981?

SQL>SELECT \* FROM EMP WHERE HIREDATE NOT BETWEEN ’01-JAN-1981’ AND ’31-DEC-1981’;🡺(-ve)

\*IS Operator:

-> Comparing nulls in a table.

Syntax:

Where<column name>is null; 🡺(+ve)

Where<column name>is not null; 🡺(-ve)

Ex: Write a query to display the list of employees whose commission is undefined/unknown/null? 🡺(+ve)

SQL>SELECT \* FROM EMP WHERE COMM IS NULL;

Ex: Write a query to display the list of employees whose commission is not undefined/unknown/null? 🡺(-ve)

SQL>SELECT \* FROM EMP WHERE COMM IS NOT NULL;

\*\* What is NULL?

->It is a unknown/a undefined value/a empty.

->Null! = 0 and Null! = space.

->When we perform arithmetic operations with null then it returns null only.

Ex: If X = 100;

1. X+NULL = 100+NULL = NULL
2. X-NULL = 100-NULL = NULL
3. X\*NULL = 100\*NULL = NULL
4. X/NULL = 100/NULL = NULL

Ex: Write a query to display EMPNO, ENAME, SAL, COMM and SAL+COMM of the employees whose name is “SMITH”?

SQL>SELECT EMPNO, ENAME, SAL, COMM, SAL+COMM FROM EMP WHERE ENAME=’SMITH’;

Output: -

EMPNO ENAME SAL COMM SAL+COMM

7369 SMITH 800

->In the above example the employee “SMITH” is not taking any commission. So that commission + salary is 800 only but it returns NULL.

->To overcome the above problem then we must use a pre-defined function in oracle is “NVL ()”.

\*\*What is NVL (Exp1, Exp2):

->NVL stands for NULL VALUE.

->It is used to replace a user defined value in place of NULL.

->This function is having two arguments those are expression 1 and expression 2.

>If Exp1 is NULL then it return Exp2 value (user defined value).

>If Exp1 is NOT NULL then it return Exp1 value.

Ex:

SQL>SELECT NVL(NULL,0) FROM DUAL; -----------------🡪0

SQL>SELECT NVL(NULL,500) FROM DUAL; --------------🡪500

SQL> SELECT NVL (0,500) FROM DUAL; -------------------🡪0

SQL> SELECT NVL (100,500) FROM DUAL; ----------------🡪100

SOLUTION:

SQL> SELECT EMPNO, ENAME, SAL, SAL + NVL (COMM,0) FROM EMP WHERE ENAME = “SMITH”;

Output: -

EMPNO ENAME SAL SAL+COMM

7369 SMITH 800 800

\*\*NVL2(Exp 1, Exp 2, Exp 3):

-> It is an extension of NVL ().

-> This function is having three arguments are expression 1, expression 2 and expression 3.

>If Exp1 is NULL then it returns Exp3 (user defined value).

>If Exp1 is NOT NULL then it returns Exp2 value (user defined value).

Ex:

SQL>SELECT NVL2 (NULL,0,100) FROM DUAL; --------------🡪100

SQL>SELECT NVL2 (500,0,100) FROM DUAL; -----------------🡪0

Ex: Write a query to update employees commission in a table based on the following conditions:

Cond-1: - If employee commission is null then update those employees commission as 800.

Cond-2: - If employee commission is not null then update those employees commission as COMM+400.

SQL>UPDATE EMP SET COMM = NVL2(COMM, COMM+400, 800);

\*LIKE Operator:

->Comparing a specific string character pattern.

->When we use LIKE operator, we must use the following two wildcard operators are,

i) % ->It represents the remaining group of characters after selected character.

ii) \_ ->Counting a single character from the expression.

Syntax:

(+ve) -> where<column name>like’[<wildcard operator>]<specific character pattern>[<wildcard operator>]’;

(-ve) -> where<column name>not like’[<wildcard operator>]<specific character pattern>[<wildcard operator>]’;

Ex: Write a query to fetch employees whose name start with “S” character?

SQL>SELECT \* FROM EMP WHERE ENAME LIKE ‘S%’;

S %

= =

S MITH

S URESH

S UMAN

S OCTT

Ex: Write a query to fetch employees whose name is having 4 characters?

SQL>SELECT \* FROM EMP WHERE ENAME LIKE ‘\_\_\_\_’;

ENAME

======

SMITH

WARD

SURESH

KING

SUMAN

FORD

SCOTT

Ex: Write a query to fetch employees whose name having “I” character?

SQL> SELECT \* FROM EMP WHERE ENAME LIKE ‘%I%’;

Ex: Write a query to fetch employees whose name having the second character is “O”?

SQL>SELECT \* FROM EMP WHERE ENAME LIKE ‘\_O%’;

Ex: Write a query to fetch employees who are joined in 1981?

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%81’;

Ex: Write a query to fetch employees who are joined in the month of DECEMBER?

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%DEC%’;

Ex: Write a query to fetch employees who are joined in the month of JUNE,DECEMBER?

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%JUN%’ OR HIREDATE LIKE ‘%DEC%’;

Ex: Write a query to fetch employees who are joined in the month of DECEMBER in 1981?

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%DEC%’ AND HIREDATE LIKE ‘%81’;

(OR)

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%DEC-81’;

(OR)

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%DEC\_81’;

(OR)

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%DEC%81’;

Ex: Write a query to fetch employees whose name not started with “S” character?

SQL>SELECT \* FROM EMP WHERE ENAME NOT LIKE ‘S%’;

🡺Functions in Oracle SQL:

->To perform some operation on the given operand values.

->Oracle supports two types of functions.

1. Pre-defined functions

> Use in SQL & PL/SQL

2. User-defined functions

>Use in PL/SQL

1. Pre-defined functions:

->These functions are also called as “built-in-functions” in oracle.

->It again divided into two types those are,

i. Single row functions (Scalar function)

ii. Multiple row functions (grouping function)

i. Single row functions (Scalar function) :

->These functions always return a single value.

>Numeric functions

>String/Character functions

>Date functions

>Null functions

>Conversion functions

>Analytical functions

\* How to call a function in oracle:

Syntax: select <fname>(value(s)) from dual;

\*What is dual?

->It is a pre-defined table in oracle.

->It is used to test functionalities of a function.

->It contains a single row and a single column.

->It is also called as “dummy table” in oracle.

\*To view the structure of dual table:

Syntax: - desc<table name>

Ex: SQL>DESC DUAL;

\*TO view data in dual table:

Syntax: select \* from <table name>;

Ex: SQL>SELECT \* FROM DUAL;

>Numeric function:

=>ABS() :

->To convert (-ve) sign values into (+ve) sign values.

Syntax: abs(n)

Ex:

SQL>SELECT ABS(-34) FROM DUAL; --------------🡪34

SQL>SELECT EMPNO,ENAME,SAL,COMM,ABS(COMM-SAL) AS TOTAL FROM EMP;

=>CEIL() :

->It return a value which is equal to (or) greater than to the given expression.

Syntax: ceil(n)

Ex:

SQL>SELECT CEIL(9.0) FROM DUAL; ------------------🡪9

SQL>SELECT CEIL(9.1) FROM DUAL; ------------------🡪10

Ex:

SQL>SELECT ENAME, SAL, CEIL(SAL\*0.05) AS INCREMENT\_AMOUNT FROM EMP;

=>FLOOR() :

->It return a value which is equals to (or) less than to the given expression.

Syntax: floor(n)

Ex:

SQL>SELECT FLOOR(9.0) FROM DUAL; -------------🡪9

SQL>SELECT FLOOR(9.9) FROM DUAL; -------------🡪9

Ex:

SQL>SELECT ENAME,SAL,FLOOR(SAL\*0.05) AS INCREMENT\_AMOUNT FROM EMP;

=>MOD() :

->It return the remainder value of the given expression.

Syntax: mod(m,n)

Ex:

SQL>SELECT MOD(10,2) FROM DUAL;

SQL>SELECT \* FROM EMP WHERE MOD(EMPNO,2)=0; ->(even empno)

SQL>SELECT \* FROM EMP WHERE MOD(EMPNO,2)=1; ->(odd empno)

=>POWER() :

->It return the power of the given expression.

Syntax: power(m,n)

Ex:

SQL>SELECT POWER(2,3) FROM DUAL; ----------------🡪8

SQL>SELECT ENAME,JOB,SAL,POWER(SAL,2) AS RESULT FROM EMP WHERE JOB=’CLERK’;

=>ROUND() :

->It return the nearest value to the given expression.

->It return nearest value based on 0.5 expression.

>If expression is <0.5 then add 0 to the main expression

>If expression is =0.5 then add 1 to the main expression

>If expression is >0.5 then add 1 to the main expression

Syntax: round(<main expression>,[decimal places])

Ex:

SQL>SELECT ROUND(45.2) FROM DUAL; -----------🡪45

Solution:

45.2 =>0.2<0.5 then add 0 to main expression

+0

===

45

===

SQL>SELECT ROUND(45.5) FROM DUAL; ------------🡪46

Solution:

45.5 =>0.5=0.5 then add 1 to main expression

+1

===

46

===

SQL>SELECT ROUND(45.8) FROM DUAL; ------------🡪46

Solution:

45.8 =>0.8>0.5 then add 1 to main expression

+1

===

46

===

SQL>SELECT ROUND(45.245,2) FROM DUAL; ------------🡪45.25

Solution:

45.245 =>45.24=>0.5=0.5 then add 1 to main expression

+1

===

45.25

===

SQL>SELECT ROUND(45.240,2) FROM DUAL; ------------🡪45.24

Solution:

45.240=>45.24=>0.0<0.5 then add 1 to main expression

+0

===

45.24

===

SQL>SELECT ROUND(45.249,2) FROM DUAL; ------------🡪45.25

Solution:

45.249 =>45.24=>0.9>0.5 then add 1 to main expression

+1

====

45.25

====

Ex: SQL>SELECT ENAME,DEPTNO,SAL,ROUND(SAL\*0.03) AS RESULT FROM EMP WHERE DEPTNO=30;

=>TRUNC () :

->It return an exact value from the given expression.

->It does not consider 0.5 value in the expression.

Syntax: trunc (<main expression>[,decimal places])

Ex:

SQL>SELECT TRUNC (45.3) FROM DUAL; -----------🡪45

SQL>SELECT TRUNC (45.8) FROM DUAL; -----------🡪45

SQL>SELECT TRUNC (45.82,1) FROM DUAL; -------🡪45.8

SQL>SELECT TRUNC (45.86,1) FROM DUAL; -------🡪45.8

>Character/String functions:

=>LENGTH () :

->It return the length of the given string.

Syntax : length(string)

Ex:

SQL>SELECT LENGTH(‘HELLO’) FROM DUAL; ------------🡪5

SQL>SELECT LENGTH(‘WEL COME’) FROM DUAL; ------🡪8 (count spaces also)

Ex:

SQL>SELECT ENAME,LENGTH(ENAME) AS NO\_OF\_CHARACTERS FROM EMP;

SQL>SELECT \* FROM EMP WHERE LENGTH(ENAME)<5;

SQL>SELECT \* FROM EMP WHERE LENGTH(ENAME)=5;

SQL>SELECT \* FROM EMP WHERE LENGTH(ENAME)>5;

=>LOWER ();

->It converts upper case characters into lower case characters.

Syntax: lower(string)

Ex: SQL>SELECT LOWER (‘HELLO’) FROM DUAL; -------🡪hello

Ex:

SQL> UPDATE EMP SET ENAME=LOWER(ENAME) WHERE JOB=’MANAGER’;

SQL>UPDATE EMP SET ENAME-LOWER(ENAME);

=>UPPER ():

->It converts lower case characters into upper case characters.

Syntax: upper(string)

Ex: SQL>SELECT EMP SET ENAME=UPPER(ENAME);

=>INITCAP ():

->To convert the initial character is capital from the given string.

Syntax: initcap(string)

Ex:

SQL>SELECT INITCAP(‘hello’) FROM DUAL; --------🡪Hello

SQL>SELECT INITCAP(‘kamal kumar’) FROM DUAL; ----🡪Kamal Kumar

Ex:

SQL>UPDATE EMP SET ENAME=INITCAP(ENAME);

=>CONCAT ();

->To add two string expressions.

Syntax: concat(string1,string2)

Ex: SQL>SELECT CONCAT (‘GOOD’,’MORNING’) FROM DUAL; 🡪GOOD MORNING

Ex:

SQL>SELECT ENAME, CONCAT(‘Mr./Miss.’,ENAME) FROM DUAL;

SQL>UPDATE EMP SET ENAME=CONCAT(‘Mr.’,ENAME);

=>LTRIM ():

->To remove unwanted characters from the left side of the given string.

Syntax: ltrim(string, ‘trimming characters’)

Ex:

SQL>SELECT LTRIM(‘XXXSMITH’,’X’) FROM DUAL;

🡪SMITH

SQL>SELECT LTRIM(‘XYZSMITH’,’XYZ’) FROM DUAL;

🡪SMITH

=>RTRIM ():

->To remove unwanted characters from the right side of the given string.

Syntax: rtrim(string, ‘trimming characters’)

Ex:

SQL>SELECT RTRIM(‘SMITHXXX’,’X’) FROM DUAL;

🡪SMITH

SQL>SELECT RTRIM(‘SMITHXYZ’,’XYZ’) FROM DUAL;

🡪SMITH

=>TRIM ():

->To remove unwanted characters from both sides of the given string.

Syntax: trim(‘trimming character’ from string)

Ex: SQL>SELECT TRIM(‘X’ FROM ‘XXSMITHXXX’) FROM DUAL;

🡪SMITH

Ex:

SQL>SELECT TRIM(‘XY’ FROM ‘XYSMITHXY’) FROM DUAL;

ERROR at line 1:

ORA-30001:trim set should have only one character.

=>LPAD ():

->Filling the required character on left side of the given string when string length is less than to user defined length.

Syntax: lpad(string,<user defined length>,’<required character>’)

Ex:

SQL>SELECT LPAD(‘SMITH’,1) FROM DUAL;

🡪S

SQL>SELECT LPAD(‘SMITH’,1,’X’) FROM DUAL;

🡪S

SQL>SELECT LPAD(‘SMITH’,8,’X’) FROM DUAL;

🡪XXXSMITH

=>RPAD ():

->Filling the required characters on right side of the given string when string length is less than to user defined length.

Syntax: rpad(string,<user defined length>,’<required character>’)

Ex: SQL>SELECT RPAD(‘SMITH’,8,’X’) FROM DUAL;

🡪SMITHXXX

=>REPLACE ():

->To replace string to string/ string to character/ character to string.

Syntax: replace(string,’<old characters>’,’<new characters>’)

Ex:

SQL>SELECT REPLACE(‘HELLO’,’ELLO’,’OD’) FROM DUAL;

🡪HOD

SQL>SELECT REPLACE(‘HELLO’,’ELLO’,’D’) FROM DUAL;

🡪HD

SQL>SELECT REPLACE(‘HELLO’,’L’,’OD’) FROM DUAL;

🡪HEODODO

SQL>SELECT REPLACE(‘HELLO’,’L’,’D’) FROM DUAL;

🡪HEDDO

=>TRANSLATE ():

->To translate character by character in the given string expression.

Syntax: translate(string,’<old characters>’,’<new characters?’)

Ex:

SQL>SELECT TRANSLATE(‘HELLO’,’HLO’,’XYZ’) FROM DUAL;

🡪XEYYZ

Here, H=X, L=Y, O=Z

SQL>SELECT TRANSLATE(‘HELLO’,’HLO’,’XY’) FROM DUAL;

🡪XEYY

Here, H=X, L=Y, O=Removed.

=>SUBSTR ():

->fetching the required sub string from the given expression.

Syntax: substr(string,<starting position of character>,<length of the character>)

| (+ve)-length

Expression:

-7 -6 -5 -4 -3 -2 -1

W E L C O M E

1 2 3 4 5 6 7

Ex:

SQL>SELECT SUBSTR(‘WELCOME’,1,1) FROM DUAL; --------🡪W

SQL>SELECT SUBSTR(‘WELCOME’,4,2) FROM DUAL; --------🡪CO

SQL>SELECT SUBSTR(‘WELCOME’,5,4) FROM DUAL; --------🡪OME

SQL>SELECT SUBSTR(‘WELCOME’,-4,4) FROM DUAL; --------🡪COME

SQL>SELECT SUBSTR(‘WELCOME’,-6,2) FROM DUAL; --------🡪EL

SQL>SELECT SUBSTR(‘WELCOME’,-6,-2) FROM DUAL; --------🡪

=>INSTR ():

->It remove the occurrence the position of character from the given string expression.

Syntax: instr(string,<specific character>,<string position of character>,<occurrence position of character>)

Expression:

-13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

H E L L O W E L C O M E

1 2 3 4 5 6 7 8 9 10 11 12 13

Note:

->Counting characters from left to right (or) from right to left but the position of characters are fixed. (always positive positions)

Ex:

SQL>SELECT INSTR(‘HELLO WELCOME’,’L’,1,2) FROM DUAL; -🡪4

SQL>SELECT INSTR(‘HELLO WELCOME’,’L’,3,1) FROM DUAL; -🡪3

SQL>SELECT INSTR(‘HELLO WELCOME’,’E’,4,3) FROM DUAL; -🡪0

SQL>SELECT INSTR(‘HELLO WELCOME’,’E’,-1,1) FROM DUAL; -🡪13

SQL>SELECT INSTR(‘HELLO WELCOME’,’O’,-7,1) FROM DUAL; -🡪5

SQL>SELECT INSTR(‘HELLO WELCOME’,’,’,-2,1) FROM DUAL; -🡪12

SQL>SELECT INSTR(‘HELLO WELCOME’,’M’,-2,-1) FROM DUAL;

ERROR at line:1

ORA-01428: argument ‘-1’ is out of range.

>DATE FUNCTIONS:

=>SYSDATE:

->It return the current date information of the system.

Syntax: sysdate

Ex:

SQL>SELECT SYSDATE FROM DUAL; -------------🡪24-JAN-25

SQL>SELECT SYSDATE+5 FROM DUAL; -------------🡪29-JAN-25

SQL>SELECT SYSDATE-5 FROM DUAL; -------------🡪19-JAN-25

=>ADD\_MONTHS ():

->To add/ subtract the number months to/ from the given date information.

Syntax: add\_months(date,<no.of months>)

Ex:

SQL>SELECT ADD\_MONTHS(SYSDATE,3) FROM DUAL; 🡪24-APR-25

SQL>SELECT ADD\_MONTHS(SYSDATE,-3) FROM DUAL; 🡪24-OCT-24

Ex:

SQL>CREATE TABLE PRODUCT(PNAME VARCHAR2(10), MFG\_DATE DATE, EXP\_DATE DATE);

SQL>INSERT INTO PRODUCT (PNAME,MFG\_DATE) VALUES(‘P1’,’09-SEP-2022’);

SQL>INSERT INTO PRODUCT (PNAME,MFG\_DATE) VALUES(‘P2’,’25-DEC-2023’);

SQL>COMMIT;

SQL>SELECT \* FROM PRODUCT; (before update)

SQL>UPDATE PRODUCT SET EXP\_DATE=ADD\_MONTHS(MFG\_DATE,24);

SQL>SELECT \* FROM PRODUCT; (after update)

=>LAST\_DAY ():

->It return the last day from the given month in the date expression.

Syntax: last\_day(date)

Ex:

SQL>SELECT LAST\_DAY(SYSDATE) FROM DUAL; 🡪31-JAN-25

SQL>SELECT LAST\_DAY(’12-FEB-25’) FROM DUAL; 🡪28-FEB-25

=>MONTHS\_BETWEEN ():

->It return the number of months between the given two dates.

->Here, date 1 is always greater than to date 2 otherwise it returns (-ve) sign values.

Syntax: months\_between(date1,date2)

Ex:

SQL>SELECT MONTHS\_BETWEEN(’05-JUN-2024’,’05-JUN-2025’) FROM DUAL; 🡪-12

SQL>SELECT MONTHS\_BETWEEN(’05-JUN-2025’,’05-JUN-2024’) FROM DUAL; 🡪12

>Conversion functions:

=>i) TO\_CHAR () :

->To convert date type into character type and also display date in different formats.

Syntax: to\_char(sysdate,<interval>)

Year formats:

YYYY ->Year in four digits format

YY ->Last two digits from year

YEAR ->TWENTY TWENTY-FIVE

CC ->Centaury 21

AD/BC ->AD year/BC year

Ex:

SQL>SELECT TO\_CHAR(SYSDATE,’YYYY YY YEAR CC BC’) FROM DUAL;

Output: 2025 25 TWENTY TWENTY-FIVE 21 AD

Month functions:

MM ->Month in Number format

MON ->First three characters from month string

MONTH ->Full name of month.

Ex:

SQL>SELECT TO\_CHAR(SYSDATE,’MM MON MONTH’) FROM DUAL;

Output: 01 JAN JANUARY

Day formats:

DDD ->Day of the year

DD ->Day of the month

D ->Day of the week

Sun - 1

Mon - 2

Tue - 3

Wed - 4

Thu - 5

Fri - 6

Sat - 7

DAY ->Full name of the day

DY ->First three characters of the day string

Ex:

SQL>SELECT TO\_CHAR(SYSDATE,’D DD DDD DY DAY’) FROM DUAL;

Output: 7 25 025 SAT SATURDAY

Quarter format:

Q ->One digit quarter of the year

1🡪JAN-MAR

2🡪APR-JUN

3🡪JUL-SEP

4🡪OCT-DEC

EX:

SQL>SELECT TO\_CHAR(SYSDATE,’Q’) FROM DUAL;

Output: 1

Week format:

WW ->Week of the year

W ->Week of the month

Ex:

SQL>SELECT TO\_CHAR(SYSDATE,’W WW’) FROM DUAL;

Output: 4 04

Time format:

HH ->Hour part in 12 hours format

HH24 ->Hour part in 24 hours format

MI ->Minute part

SS ->Seconds part

AM/PM ->AM Time (or) PM Time

Ex:

SQL>SELECT TO\_CHAR(SYSDATE,’HH HH24 MI SS PM’) FROM DUAL;

Output: 11 11 45 29 PM

Ex: Write a query to display the list of employees details who are joined in 1981 by using TO\_CHAR ()?

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’YYYY’)=’1981’;

(OR)

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’YY’)=’81’;

(OR)

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’YEAR’)=’NINETEEN EIGHT-ONE’;

Ex: Write a query to display the list of employees details who are joined in 1980,1982,1983 by using TO\_CHAR ()?

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’YYYY’) IN(‘1980’,’1982’,’1983’);

(OR)

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’YYYY’)=’1980’ OR TO\_CHAR(HIREDATE,’YYYY’)=’1982’ OR TO\_CHAR(HIREDATE,’YYYY’)=’1983’;

Ex: Write a query to display the list of employees who are joined in the month of DECEMBER?

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’MON’)=’DEC’;

(OR)

SQL>SELECT \* FROM EMP WHERE HIREDATE LIKE ‘%81’;

Ex: Write a query to display the list of employees details who are joined in the month of DECEMBER in 1982?

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’MON’)=’DEC’ AND TO\_CHAR(HIREDATE,’YYYY’)=’1982’;

(OR)

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’MMYYYY’)=’121982’;

Ex: Write a query to display employees and their joined day from emp table?

SQL>SELECT ENAME, HIREDATE, TO\_CHAR(HIREDATE,’DAY’) AS DAYS FROM EMP;

Ex: Write a query to display the list of employees who are joined on weekends?

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’DAY’) IN (‘SATURDAY’,’SUNDAY’);

Ex: Write a query to display employees and their joined week from emp table?

SQL>SELECT ENAME, HIREDATE, TO\_CHAR(HIREDATE,’WW’) AS WEEKS FROM EMP; (YEAR)

(OR)

SQL>SELECT ENAME, HIREDATE, TO\_CHAR(HIREDATE,’W’) AS WEEKS FROM EMP; (MONTH)

Ex: Write a query to display employees details who are joined in 1st week of DECEMBER in 1981?

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’WMONYY’)=’1DEC81’;

(OR)

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’WMMYY’)=’11281’;

(OR)

SQL>SELECT \* FROM EMP WHERE TO\_CHAR(HIREDATE,’MONYYW’) IN (DEC811’);

(OR)

SQL>SELECT \* FROM EMP WHERE

2 TO\_CHAR(HIREDATE,’W’)=’1’ AND

3 TO\_CHAR(HIREDATE,’MON’)=’DEC’ AND

4 TO\_CHAR(HIREDATE,’YYYY’)=’1981’;