Task 7

DDL

- The data definition language is used to create an object, alter the structure of an object and also drop already created object.
- DDL commands:

1. **Create** table command
2. **Alter** table command
3. **Truncate** table command
4. **Drop table** command
5. Rename

CREATION OF TABLES

- Table is a primary object of database, used to store data in form of rows and columns. It is created using following command:
- **Syntax:** CREATE TABLE <tablename>

```
(colname1 DATATYPE,
 colname2  DATATYPE ,
  :
  :
 colnameN DATATYPE ) ;
```

SQL DATATYPES

Character or string DATATYPES:

1. **CHAR**
2. **VARCHAR | VARCHAR2**
3. **NCHAR**
4. **NVARCHAR2**

Numeric datatypes

1. **Number**

Date values

Date

SQL DATATYPES

Characer or string DATATYPES:

- **CHAR(size)-**

  FIXED LENGTH CHARACTER DATATYPE

  Size is optional

  Default size is 1 byte

  MAX SIZE 2000 bytes

  Allows 0-9, a-z,A-Z, SPECIAL CHARACTERS.

- **EXAMPLE : ename CHAR**

  ename CHAR(20)

SQL DATATYPES

Character or string DATATYPES:

- **VARCHAR2(size)**

  Variable length character datatype

  Size is mandatory

  Max size is 4000 bytes.

- **Example : ename VARCHAR(10)**

  ename VARCHAR2(10)

SQL DATATYPES

Character or string DATATYPES:

- **NCHAR(SIZE)**

  Fixed-length character data.

  Size is optional

  National character

  Stores multi byte characters.

  Max size is 1000 bytes.

SQL DATATYPES

Character or string DATATYPES:

- **NVARCHAR2(SIZE)**

  Variable-length character dataType.

  Size is Mandatory

  National character

  Stores multi byte characters.

  Max size is 2000 bytes.

SQL DATATYPES

Numeric datatypes

1. Number(P,S)   P- PRECISION, S- SCALE

- Stores numeric values that can be negative or positive.
- Size is optional
- Precision is the total number of digits in the Integral part+ decimal part. Ranges from 1 to 38.
- Scale is the number of digits to the right of the decimal point in the number. Scale ranges from -84 to 127.
- For example, the number 1234.56 has a precision of 6 and a scale of 2. So to store this number, you need NUMBER(6,2).

SQL DATATYPES

Date datatype

Date : Supports date Values.

Default format is ‘DD-MON-YY’ OR ‘DD-MON-YYYY’

EXAMPLES : dob DATE

joiningdate DATE

SQL DATATYPES

BOOLEAN

- Accepts values TRUE, FALSE, NULL

CREATION OF TABLES

- **EXAMPLE1**
- SQL> CREATE TABLE SAILORS

```
SQL>  CREATE TABLE SAILORS
      (SID NUMBER(5) PRIMARY KEY,
       SNAME VARCHAR(10),
       RATING  NUMBER(10),
        AGE NUMBER(3));
```

- **Table Created.**

CREATION OF TABLES

Desc command

- DESCRIBE command is used to view the structure of a table .

- SQL>DESC SAILORS;

CREATION OF TABLES

- **Example 2:** Create a BOATS table with Fields (BID,BNAME,COLOR)and display using DESCRIBE command.
- SQL> CREATE TABLE boats

```
                    (bid NUMBER(4),
                     bname VARCHAR(20),
                       );
```

- SQL>DESC boats;

CREATION OF TABLES

- **Example 3:** Create an RESERVES table with fields (SID , BID ,DAY) and display using DESCRIBE command.
- SQL> CREATE TABLE reserves (

```
                              bid NUMBER(5),
                               sid Number(5),
                               bookingday  DATE);
```

- SQL>   DESC reserves;

CREATION OF TABLES

- **Example4:** Create employee table with fields(ENO, ENAME, MGR, DEPTNO,SALARY, HIRING DATE)
- SQL> CREATE TABLE emp(

```
eno NUMBER(5),
ename  varchar(20),
mgr  VARCHAR(10),
deptno NUMBER(5),
sal NUMBER(7,2),
 hiringdate DATE);
```

CREATION OF TABLES

- **EXAMPLE5:**

```
SQL> CREATE TABLE dept(
                         dno NUMBER(5),
                        dname  VARCHAR(20),
                        dlocation VARCHAR(20)
                        );
```

CREATION OF TABLES

Create table using existing table

```
SQL> CREATE TABLE <newtablename>
         as
         SELECT *  FROM    <oldtablename>;
SQL> CREATE TABLE emp2
         as
         SELECT *  FROM    emp;
```

Insert values into a table

- The INSERT INTO statement is used to insert new records in a table.
- Syntax:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

- INSERT INTO table_name(column1,column2,..)

VALUES(value1, value2) //numeric type

If character value ‘value’

Ex: value(id,’string’)

EX:

```
        INSERT INTO emp(eid,ename)
        VALUES(12,’EMP1’);      or
        VALUES(‘12’,’EMP1’);
```

##For viewing purpose

→ SELECT * FROM table_name;

ALTER TABLE

- ALTER TABLE statement used to add, delete or drop , modify columns in a table.
- It is also used to rename a table column(s).

ADD a column:

- **SYNTAX:** ALTER TABLE <table name>
  ADD colname DATATYPE (SIZE);
- SQL>

ALTER TABLE boats ADD bcolor VARCHAR2(10)

ALTER TABLE

ADD MORE columns:

- **SYNTAX:** ALTER TABLE <table name>
  ADD (colname1 DATATYPE, colname2 DATATYPE…);

- Example : SQL> ALTER TABLE boats
  ADD(blocation VARCHAR2(10),
  bprice number);

ALTER TABLE

To DROP a column:

- **SYNTAX:**
```
ALTER TABLE <table name>
DROP COLUMN <column name>;
```

- Example:
```
SQL>ALTER TABLE boats
DROP COLUMN bprice;
```

ALTER TABLE

- **Modify column:**
- **SYNTAX:** ALTER TABLE <table name>
  MODIFY <colname> **<NEW DATATYPE>(<NEW SIZE>)** ;
- Example : SQL> ALTER TABLE boats
  MODIFY bid number(7);

ALTER TABLE

- **Rename column:**
- **SYNTAX:** ALTER TABLE <table name>
  RENAME COLUMN <oldname> TO <newname>;
- **Example:**
- SQL> ALTER TABLE emp
  RENAME COLUMN eno TO eid;

RENAME

RENAME A TABLE

- Rename command is used to give new names for existing tables.
- SQL> **RENAME <oldtablename> TO <newtablename>;**

- Ex: SQL>RENAME boats TO boats2;

TRUNCATE A TABLE

- Truncate command is used to delete all records from a table.
- SQL> TRUNCATE TABLE <tablename>;
- **Example**: SQL>TRUNCATE TABLE boats1;

DROP A TABLE

DROP A TABLE

- Drop command is used to remove an existing table permanently from database.

- **SQL> DROP TABLE <tablename>;**

- **Example: SQL>DROP TABLE sailors;**

summary

- CREATE TABLE talbe_name(attributes data_type,……);

- INSERT INTO table_name(attributes,…)
- VALUES(values,…);

- OR

- INSERT ALL
- INTO table_name(attributes,…) VALUES(values,…)
- .
- .
- .
- INTO table_name(attributes,…) VALUES(values,…)
- SELECT * FROM DUAL;

- ALTER TABLE table_name
- {
-   ADD (col_name data_type,…);
-   DROP COLUMN col_name;
-   MODIFY col_name newdata_type;
-   RENAME COLUMN old_name TO new_name;
- }

- RENAME oldtable_name TO newtab_name;

- TRUNCATE TABLE table_name;

- DROP TABLE table_name;

CONSTRAINTS

- Constraints are used to specify rules for the data in a table.
- If there is any violation between the constraint and the data action, the action is aborted by the constraint(Maintain data integrity).
- It can be specified when the table is created (using CREATE TABLE statement)
- Or after the table is created (using ALTER TABLE statement).

CONSTRAINTS

1. **Domain Integrity constraints**
   - DEFAULT
   - NOT NULL
   - CHECK
2. **Entity integrity constraints**
   - UNIQUE
   - PRIMARY KEY
3. **Referential Integrity constraints**
   - FOREIGN KEY

NOT NULL

- Does not allow NULL values.
- Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.
- Can be defined at column level only.

NOT NULL

Syntax:

- CREATE TABLE <Table_Name>

  (colname datatype (*size*) **NOT NULL**, ……. *);*

***Example*:**

```
CREATE TABLE student
                    (sno NUMBER(3) NOT NULL,
                     sname CHAR(10)
                     );
```

NOT NULL

- SQL> CREATE TABLE emp2(
  eno NUMBER(5) **CONSTRAINT CT1 NOT NULL**,
  ename varchar(20),
  sal NUMBER(7,2),
  hiringdate DATE);

SQL> DESC emp2;

NOT NULL(for MULTIPLE COLUMNS)

```
SQL> CREATE TABLE EMP3(
         eno NUMBER(5)  NOT NULL,
        ename  varchar(20)  NOT NULL,
        sal NUMBER(7,2),
        hiringdate DATE);

SQL> DESC EMP3;
```

DEFAULT

- This constraint sets a default value for the column when no value is specified at record insertion.
- Defined at column level.
- ***Syntax:*** **CREATE TABLE <Table_Name>**

  (colname datatype DEFAULT <defaultvalue>);
- **Example:**

CREATE TABLE student

(sno NUMBER(3),

sname VARCHAR(20) **DEFAULT ‘ABC’);**

DEFAULT

Example2 : Create table emp3

```
(eno  number,
 ename  varchar2(20),
 dno  number  default 10);
```

CHECK

- When we insert the value in to a column, then the value will be first checked for certain conditions before inserting the value into that column.
- Allows valid range of values into a column.
- Can be defined at column level or table level.

check

Syntax:

- **CREATE TABLE <tablename>**

  (colname datatype(*size*) *CHECK(logical expression), ….);*
- ***Example:***

```
        CREATE TABLE student (sno NUMBER (3),
                              sname CHAR(10),
         deptno  NUMBER  CHECK(deptno  IN(10,20,30)));
```

CHECK

- **Example2:**

```
CREATE TABLE  emp_check (eno NUMBER,
                         ename CHAR(10),
                         deptno  number,
                         manager  varchar(10),
              CHECK(deptno  IN(10,20,30)) );
```

UNIQUE Constraint

- The UNIQUE constraint ensures that all data values in a column are different i.e. data values must be unique
- A unique constraint is an integrity constraint that ensures the data stored in a column, or a group of columns, is unique among the rows in a table.

UNIQUE Constraint

- Does not allow duplicate values.
- Allows NULL values.
- Can be defined using one column or using a group of columns(**composite key**).
- Can be defined at column level or table level.

UNIQUE Constraint

Syntax:

CREATE TABLE <tablename>
(colname datatype(*size*) ***UNIQUE*** , *colname2, ….);*

- ***Example:***

CREATE TABLE student (rollno NUMBER **UNIQUE** ,
name CHAR(10));

UNIQUE Constraint

- ***Example:***

  CREATE TABLE stud_1 (rollno NUMBER **UNIQUE** ,
  name CHAR(10));

  (or)

- CREATE TABLE stud_1 (rollno NUMBER ,
  name CHAR(10),
  UNIQUE(rollno));

UNIQUE Constraint

- **Using a group of columns(composite key)**

Example : Create table emp_unique

```
( eno number,
 ename  varchar2(20),
  dno  number default 10,
 unique(eno, ename) );
```

PRIMARY KEY

- The PRIMARY KEY constraint is used to uniquely identify each record in a table.
- A **column or combination of columns** can be created as primary key, to identify a record uniquely in a table .
- A table can have only ONE primary key.
- It does not allow duplicate values and NULL values.
- It can be defined at column level or table level.

PRIMARY KEY

Syntax:

- CREATE TABLE <tablename>

  (colname datatype(*size*) ***PRIMARY KEY***, *col2, ….);*

Example:

- **CREATE TABLE emp**

```
                    (eid  NUMBER(3) PRIMARY KEY,
                     ename VARCHAR(20),
                     salary NUMBER);
```

PRIMARY KEY

- CREATE TABLE emp
  (eid **NUMBER(3) CONSTRAINT pk PRIMARY KEY**,
  ename VARCHAR(20),
  salary NUMBER);
- CREATE TABLE emp (eid NUMBER(3),
  ename VARCHAR(20),
  salary NUMBER,
  CONSTRAINT pk PRIMARY KEY);

PRIMARY KEY

- **Primary key using a group of columns:**

- CREATE TABLE emp

```
(eid  NUMBER(3),
 ename VARCHAR(20),
 salary NUMBER,
PRIMARY  KEY(eid, ename) );
```

FOREIGN KEY

- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.
- A foreign key is used to establish a link between two tables.
- FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

FOREIGN KEY

cid	Cname	Age
1	Peter	35
2	henry	23
3	smith	45

orderid	ordernumber	cid
1	77895	2
2	44678	3
3	22354846	3

Sql>

- create table customers(cid number primary key,cname varchar(16),age number);
- insert all
- into customers values(1,'peter',35)
- into customers values(2,'henry',23)
- into customers values(3,'smith',45)
- select * from dual;
- select * from customers;

- create table Orders(orderid number,ordernumber number,cid number references customers(cid));
- insert all
- into Orders values(1,77895,2)
- into Orders values(2,44678,3)
- into Orders values(3,22354846,3)
- select * from dual;
- select * from Orders;

FOREIGN KEY

- CREATE TABLE customers(cid number PRIMARY KEY,
                                              cname VARCHAR(20) ,
                                              age NUMBER);

FOREIGN KEY

- CREATE TABLE Orders (orderid NUMBER PRIMARY KEY,

  ordernumber NUMBER ,

  cid NUMBER **REFERENCES customers(cid)**

  );
- EX: CREATE TABLE new_name(col1 datatype,...,

  Id NUMBER REFERENCES existing_table(Id));

FOREIGN KEY

- CREATE TABLE orders (orderid NUMBER PRIMARY KEY,

```
                                        ordernumber  NUMBER ,
      cid number CONSTRAINT FK  REFERENCES customers(cid)
                                              );
```

FOREIGN KEY

- CREATE TABLE Orders (orderid NUMBER ,

```
CREATE TABLE  Orders (orderid NUMBER ,
                      OrderNumber  NUMBER ,
                      cid NUMBER ,
                      PRIMARY KEY (orderid),
CONSTRAINT **FK** FOREIGN KEY(cid) REFERENCES customers(cid)
                      );
```