Exp.No: 1 # DDL, DML, CONSTRAINTS

Date :

AIM: To study and work with DDL and DML Commands in Oracle.

DESCRIPTION:

DDL commands in SQL:

- CREATE Command.
- DROP Command.
- ALTER Command.
- TRUNCATE Command.
- RENAME Command.

CREATE TABLE: The CREATE TABLE statement is used to create a new table in a database.

Syntax: CREATE TABLE *table_name* (*column1 datatype*, *column2 datatype*, *column3 datatype*, ...);

The column parameters specify the names of the columns of the table. The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

Create Table Using Another Table

A copy of an existing table can also be created using CREATE TABLE. The new table gets the same column definitions. All columns or specific columns can be selected. If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

Syntax

CREATE TABLE *new_table_name* AS
SELECT *column1*,*column2*,... FROM *existing_table_name* WHERE ....;

ALTER: To alter the definition of a table in one of these ways:

- to add a column
- to add an integrity constraint
- to redefine a column (data type, size, default value)
- to explicitly allocate an extentSyntax

ALTER TABLE tablename MODIFY (columnname [datatype] [constraint]);

ALTER TABLE tablename ADD (columnname datatype [constraint]);

DROP: This command is used to drop the table.

Syntax: DROP TABLE tablename;

RENAME : RENAME tablename to new-tablename;

TRUNCATE: Delete all the records in the table retaining its structure.
Syntax: Truncate table table name>;

Example SQL> truncate table employee;

DML commands in SQL:

- SELECT Command.
- INSERT Command.
- UPDATE Command.
- DELETE Command.

SQL SELECT Statement

The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

Syntax

SELECT *column1, column2, ...*
FROM *table_name*;

INSERT INTO Statement

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

Syntax

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

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

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

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

UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

Syntax

UPDATE *table_name*
SET *column1 = value1, column2 = value2, ...*
WHERE *condition*;

SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

Syntax

DELETE FROM *table_name* WHERE *condition*;

CREATE: To create a table, the basic structure to hold user data, specifying this information:

- column definitions
- integrity constraints

- the table's tablespace
- storage characteristics
- an optional cluster
- data from an arbitrary querySyntax

CREATE TABLE tablename (columnname datatype [constraint [constraint name]], [columnname .....] [, constraints]);

Constraints:

- NOT NULL every record must have a value for the column
- UNIQUE value for each record must be distinct
- CHECK checks to make sure the condition is satisfied
- PRIMARY KEY defines the primary key of the table

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL PRIMARY KEY on CREATE TABLE

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

Syntax

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int);
```

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int, CONSTRAINT PK_Person PRIMARY KEY (ID,LastName));
```

SQL PRIMARY KEY on ALTER TABLE

To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following SQL:

Syntax

```
ALTER TABLE Persons ADD PRIMARY KEY (ID);
```

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

Note: If you use ALTER TABLE to add a primary key, the primary key column(s) must have been

declared to not contain NULL values (when the table was first created).

DROP a PRIMARY KEY Constraint

To drop a PRIMARY KEY constraint, use the following SQL:

Syntax

```
ALTER TABLE Persons DROP PRIMARY KEY;

ALTER TABLE Persons DROP CONSTRAINT PK_Person;
```

SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different. Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns. A PRIMARY KEY constraint automatically has a UNIQUE constraint. However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

SQL UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

Syntax

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int);
```

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE (ID,LastName));
```

SQL UNIQUE Constraint on ALTER TABLE

To create a UNIQUE constraint on the "ID" column when the table is already created, use the following SQL:

Syntax

```
ALTER TABLE Persons ADD UNIQUE (ID);
```

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

DROP a UNIQUE Constraint

To drop a UNIQUE constraint, use the following SQL:

Syntax

```
ALTER TABLE Persons DROP CONSTRAINT UC_Person;
```

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column. If you define a CHECK constraint on a column it will allow only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

SQL CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

Syntax

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18));
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes'));
```

SQL CHECK on ALTER TABLE

To create a CHECK constraint on the "Age" column when the table is already created, use the following SQL:

Syntax

```
ALTER TABLE Persons ADD CHECK (Age>=18);
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons
    ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');
```

DROP a CHECK Constraint

To drop a CHECK constraint, use the following SQL:

Syntax

```
ALTER TABLE Persons DROP CONSTRAINT CHK_PersonAge;
```

SQL NOT NULL Constraint

By default, a column can hold NULL values. The NOT NULL constraint enforces a column to NOT accept NULL values. This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

SQL NOT NULL on CREATE TABLE

The following SQL ensures that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int);
```

SQL NOT NULL on ALTER TABLE

To create a NOT NULL constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

Syntax

```
ALTER TABLE Persons ALTER COLUMN Age int NOT NULL;
ALTER TABLE Persons MODIFY COLUMN Age int NOT NULL;
ALTER TABLE Persons MODIFY Age int NOT NULL;
```

QUESTIONS

1. **Write a query to create a table student with the following list of attributes Sno, Name,Register number, Marks.**

SQL> create table stud(sno number,name varchar2(15),regno number,marks number);

2. Write a query to rename the table name.

SQL> alter table stud rename to student;

3. Write a query to describe the table.

SQL> desc student;

4. Write a query to change the type of an attribute Name as Number.

SQL> alter table student modify name number;

5. Write a query to create a table BankAccount and assign Primary key Constraint for the attribute Accno and assign NOTNULL Constraint for the field CustomerName.

SQL> create table BankAccount(sno number,customername varchar2(10) not null,acno number Primary key,bal number);

6. Write a query to Add a Primary key for the attribute Regno of the table Student.

SQL> alter table student add(primary key(regno));

7. Write a query to Alter “Student” table to add “Gender” and “City” columns and verify it.

SQL> alter table student add(Gender varchar2(5), City varchar2(15));

8. Write a query to Alter “student” table to modify the column “Name” as “Sname” andadd Unique Constraint on it and verify it.

SQL> alter table student modify(Name varchar2(20) constraint Sname unique);

9. Write a query to Alter “student” table to drop the “City” column and verify it.

SQL> alter table student drop column City;

10. Write a query to alter “student” table to set the “Gender” column as unused and verify it.

SQL> alter table student set unused(Gender);

11. Write a query to view the constraint names and its types of the “student” table from the User_Constraints table.

SQL> select constraint_name,constraint_type from user_constraints where table_name=’student’;

12. Write a query to view the column associated with the constraints names of theBankAccount table from the user_cons_columns view.

SQL> select constraint_name,column_name from user_cons_columns where table_name=’BankAccount’;

13.Write a query to create a FOREIGN KEY constraint on the "Accno" column when the "Loan” table which is already created.

SQL> alter table Loan add foreign key(Accno) references BankAccount(Accno);

14. Write a query to create a table stud with attributes rno,name,sal and add checkconstraint to attribute sal.

SQL> create table stud(rno number(5),name varchar2(10),sal number(10) constraint no_ck check(sal between 10000 and 30000));

SQL> insert into stud values(&rno,'&name',&sal);
Enter value for rno: 567
Enter value for name: sachin
Enter value for sal: 29000

old 1: insert into stud values(&rno,'&name',&sal)
new 1: insert into stud values(567,'sachin',29000)

15. Write a query to disable the Unique key (Sname) Constraint on the student table without Dropping it or Recreating it and verify it.

SQL> alter table student disable constraint Sname;

16. Write a query to enable or activate the Unique key Constraint(Sname) on "student" table and verify it.

SQL> alter table student enable constraint Sname;

17. Write a query to remove the Unique key(Sname) Constraint from the student table and verify it.

SQL> alter table student drop constraint Sname;

18. Write a query to remove the Primary key Constraint on the "BankAccount" table andDrop the associated Foreign key Constraint on the Loan.Accno column and verify it.

SQL> alter table student drop primary key cascade;

19. Write a query to Rename the "student" table to "Student" and verify it.

SQL> rename student to Student;

20. Write a query to Truncate the table.

SQL> truncate table student;Table truncated.

21. Write a query to delete the table permanently.

SQL> drop table student;

VIVA Questions

1) What is DBMS?
2) What is a database?
3) What is a database system?
4) What are the advantages of DBMS?

5) What is a checkpoint in DBMS?

RESULT

Thus the database was created and the SQL queries are written to retrieve information from the database.