Ex.No: 03	CREATION OF VIEWS, SYNONYMS, SEQUENCE, INDEXES, SAVEPOINT

AIM:

To Study the Creation of Views, Synonyms, Sequence, Indexes and Savepoint.

CREATING A VIEW:

DESCRIPTION:

A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed.

SYNTAX:

Create or replace view <view name> [Column alias name] as <query> [with <options> conditions];

INDEX:

DESCRIPTION:

Index is a database object, provides a fast access path to Column that are indexed

SIMPLE INDEX:

A simple index on Column

SYNTAX:

Create index <name> on <table_name> [conditions];

COMPOSITE INDEX:

An index created on more than one Column of table:

SYNTAX:

Create index <name> on <table-name> [column_name, Column_name];

SAVEPOINT

Description:

It is used to save the changes in table

Syntax:

```
Savepoint <tablename>;
```

SYNONYMS

Description:

A synonym is an alternative name for objects such as tables, views etc

SYNTAX:

```
Create synonym <name> for <table-name>;
```

SEQUENCE

DESCRIPTION:

Sequence is a DB object that can generate unique sequential value

SEQUENCE WITH CYCLE:

specify the sequence will continue to generate value after reaching either maximum or minimum value

SYNTAX:

```
Create sequence <name> start with [value operator] by [value]
maxvalue[value] no cycle cache [value];
```

SEQUENCE WITH NO Cycle:

Specify sequence can't generate value after reaching max or min value

SYNTAX:

```
Create sequence <name> start with [value operator] by
[value] maxvalue [value] nocycle;
```

Ex. No : 04	SET VARIOUS CONSTRAINTS LIKE NOT NULL, PRIMARY KEY, FOREIGN KEY and CHECK CONSTRAINTS

AIM:

To study set various constraints like Not Null, Primary Key, Foreign Key and Check constraints.

DOMAIN INTEGRITY CONSTRAINTS

(a) NOT NULL CONSTRAINT

DESCRIPTION:

This constraint is used to specify that a column may never contain a NULL value.

SYNTAX:

Create table <tablename> (col1 datatype NOT null, col2 datatype);

USING ALTER STATEMENT

SYNTAX:

alter table <table name> modify <column name> not null

(b) DEFAULT CONSTRAINT

DESCRIPTION:

Default constraint provides a default value to a column when the insert into statement does not provide a specific value

USING CREATE STATEMENT:

SYNTAX:

Create table <tablename> (col1 datatype default value, col2 datatype)

USING ALTER STATEMENT:

alter tabble <table name> modify (column default value);

(c) CHECK CONSTRAINT

DESCRIPTION:

Constraint validates incoming columns at row insertion time

SYNTAX:

Create table <table name> (col1 datatype [constraint constraint name] check (condition));

ENTITY INTEGRITY CONSTRAINTS

a) UNIQUE CONSTRAINT

DESCRIPTION:

The Unique constraint ensures that all values in a column are distinct.

SYNTAX:

Create table <tablename> (col1 datatype [constraint name] unique);

(b) PRIMARY KEY

DESCRIPTION:

A primary key is used to uniquely identify each row in a table.

SYNTAX:

Create table <table name> (col1 datatype [constraint cname] primary key);

REFERENTIAL INTEGRITY CONSTRAINT

DESCRIPTION:

A foreign key is a field that points to the primary key of another table.

SYNTAX:

Create table <table name> (col1 datatype [constraint name] primary key, col..i [constraint fk_cname references <parent table name> (col1));

Ex.No: 5	CREATING RELATIONSHIP BETWEEN THE DATABASES AND RETRIEVE RECORDS USING JOINS FOR THE BELOW RELATIONS

AIM:

To study the database Creating relationship between the databases and retrieve records using joins for the below relations.

TYPES OF JOINS:

1. simple Join
2. Self Join
3. Outer Join

SIMPLE JOIN:

It retrieves from 2 tables having a common column

(a) Equi-join

A join, which is based on equalities, is called equi-join

Select * from <table1>, <table2> where <table1>.[column] = <table2>.[column]

(b) Non - Equi -join

It specifies the relationship between columns belonging to different tables

Select * from <table1>, <table2> where <table1>.[column] [condition] <table2>.[column]

SELF JOIN:

Joining of a table to itself is known as self join.

Select * from <table1> <table2> where [condition]

INNER JOIN:

Select * from <tablename1>, <table name2> where [condition]

NATURAL JOIN:

Select * from <tablename1> Natural join <table name2>;

CROSS JOIN:

Select * from <tablename1> Cross join <tablename2>;

OUTER JOIN:

It extends the results of a simple join. The symbol (+) represents outer join

LEFT OUTER JOIN:

Select * from <tablename1>, <tablename2> where <table name1>.<column name>(+) = <tablename2>.<column>;

RIGHT OUTER JOIN:

Select * from <tablename1>, <tablename2> where <tablename1>.<columnname>= <tablename2>.<columname>(+);

FULL OUTER JOIN:

Select * from <tablename1> full join <table name 2> on <tablename1>.<column> = <tablename2>.<column>;