===

1. 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.

SYNTAX:-

CREATE TABLE tableName(col1 number() NOT NULL, col2 varchar2(50) NOT NULL, col3 number2(50), PRIMARY KEY (col1));

Applying / Defining a PRIMARY KEY constraint on Single column:

CREATE TABLE Persons(ID number(3) NOT NULL PRIMARY KEY, LastName varchar2(50) NOT NULL, FirstName varchar2(50), Age number(3));

drop table Persons;

Applying / Defining a PRIMARY KEY constraint on multiple columns:

CREATE TABLE Persons(ID number(3) NOT NULL, LastName varchar2(50) NOT NULL, FirstName varchar2(50), Age number(3), CONSTRAINT PK_Person PRIMARY KEY (ID, LastName));

drop table Persons;

In the example above there is only ONE PRIMARY KEY (PK_Person). However, the VALUE of the primary key is made up of TWO COLUMNS (ID + LastName).

===

SQL PRIMARY KEY on ALTER TABLE:-

```
CREATE TABLE Persons(ID number(3) NOT NULL, LastName varchar2(50) NOT
NULL, FirstName varchar2(50), Age number(3));

ALTER TABLE Persons ADD PRIMARY KEY (ID);

drop table Persons;

CREATE TABLE Persons(ID number(3) NOT NULL, LastName varchar2(50) NOT
NULL, FirstName varchar2(50), Age number(3));

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

drop table Persons;
```

===

DROP a PRIMARY KEY Constraint:-

```
ALTER TABLE Persons DROP CONSTRAINT PK_Person;

drop table Persons;
```

==

2. SQL FOREIGN KEY Constraint:-

==

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

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.

CREATE TABLE childTable(col1 number() NOT NULL, col2 number() NOT NULL, col3 number(), PRIMARY KEY (col1), FOREIGN KEY (col3) REFERENCES parentTable(parent_Primary_key));

==

Foreign Key Constraint in Oracle:-

==

CREATE TABLE Department(Id number(3) PRIMARY KEY, Name VARCHAR2(20), Location VARCHAR2(20));

INSERT INTO Department (Id, Name, Location) VALUES (10, 'IT', 'Hyderabad');
INSERT INTO Department (Id, Name, Location) VALUES (20, 'HR', 'Delhi');
INSERT INTO Department (Id, Name, Location) VALUES (30, 'Finance', 'Mumbai');

select * from department;

INSERT INTO Department (Id, Name, Location) VALUES (10, 'IT', 'Hyderabad');

ERROR at line 1:

```
ORA-00001: unique constraint (SYSTEM.SYS_C007003) violated
```

```
CREATE TABLE Employee(ID number(3) PRIMARY KEY, Name VARCHAR2(20),
Salary number(6), DepartmentId number(2) REFERENCES Department(Id));
```

```
INSERT into Employee VALUES (101, 'Anurag', 25000, 10);
INSERT into Employee VALUES (102, 'Pranaya', 32000, 20);
INSERT into Employee VALUES (103, 'Hina', 35000, 30);
```

Now, try to execute the following INSERT Statement. Here, we are passing the Department Id value as 40 which does not exist in the Department table.

Rule 1:-

We cannot insert a value into the foreign key column if that value is not existing in the reference key column of the parent (master) table. For example,

```
INSERT into Employee VALUES (104, 'Sambit', 52000, 40);
```

```
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.SYS_C007007) violated - parent key not
found
```

Rule 2:-

We cannot update the reference key value of a parent table (Department) if that value has a corresponding child record in the child table (Employee) without addressing what to do with the child records. For example,

```
UPDATE Department SET Id = 100 WHERE Id = 10;
```

```
ERROR at line 1:
ORA-02292: integrity constraint (SYSTEM.SYS_C007007) violated - child record
found
```

Rule 3:-

We cannot delete a record from the parent table (Departmet) provided that the records reference key value has a child record in the child table (Employee) without addressing what to do with the child record. For example,

```
DELETE FROM Department WHERE Id = 10;
```

```
ERROR at line 1:
ORA-02292: integrity constraint (SYSTEM.SYS_C007007) violated - child record
found
```

We cannot delete the parent table also when there is a foreign key primary key linking:-

```
drop table department;
```

```
ERROR at line 1:
ORA-02449: unique/primary keys in table referenced by foreign keys
```

==

Features of Primary Key and Foreign Key:-

==

Primary Key in Oracle:-

1. The Primary Key Constraint in Oracle is uniquely identifying a record in a table.

2. Primary Key constraint neither accepts null values nor duplicate values on the column on which it is applied.

3. We can create only one Primary Key on a table in Oracle and that primary key constraint can be created either on a single column or multiple columns.

Foreign Key in Oracle:-

1. The Foreign Key in Oracle is a column in a table that is a unique key (either primary or unique key) in another table.

2. A Foreign Key can accept both null and duplicate values.

```
insert into employee values(105, 'Aditi Kadu', 40000, 10);

select * from employee;

select * from department;
```

3. We can create more than one Foreign key on a table in Oracle.

===
3. Referential Integrity Constraints in Oracle:-
===
The Referential Integrity Constraints in Oracle are nothing but the foreign key constraints that tell Oracle Database to perform certain actions whenever a user attempts to delete or update a primary key for which existing foreign keys point.

In order to tell what actions to perform whenever a user trying to delete or update a primary key value for which existing foreign key values point, Oracle provided the following Referential Integrity Constraints which we need to set while creating the foreign key constraints.

SET NULL: If a user tries to execute delete statement(s) that will affect rows in the foreign key table, then those values will be set to NULL when the primary key record is deleted in the Primary key table. The important thing that we need to keep in mind is that the foreign key columns affected must allow NULL values.

CASCADE: If a user tries to execute delete the statement(s) which will affect the rows in the foreign key table, then those rows will also be deleted when the primary key record is deleted.

Example:-
Department Table (Primary Key Table):-
CREATE TABLE Department(Id number(3) PRIMARY KEY, Name VARCHAR2(10));

Insert into Department values (10, 'IT');
Insert into Department values (20, 'HR');
Insert into Department values (30, 'INFRA');

select * from department;

Employees Table (Foreign Key Table):-

```
CREATE TABLE Employees(Id number(3) PRIMARY KEY, Name VARCHAR2(20) NOT
NULL, DepartmentID number REFERENCES Department(Id));

INSERT into Employees VALUES (101, 'Anurag', 10);
INSERT into Employees VALUES (102, 'Pranaya', 20);
INSERT into Employees VALUES (103, 'Hina', 30);

select *from employees;
```

Now, consider the above 2 tables (Department and Employees). If we delete the row with ID = 10 from the Department table (parent table) then the row with ID = 101 in the Employees table (child table) becomes an orphan record. That means we will not be able to tell the Department name for that employee. So, the referential integrity constraint in Oracle is basically used to define the actions that the Oracle server should take.

DELETE Rules in Oracle:-

If you want to delete a record in the parent table (in our case the Department table) when they have a corresponding child record in the child table (in our case the Employees table), Oracle is provided with a set of rules to perform delete operations known as DELETE rules.

If we want to delete a record from the parent table when they have corresponding child records in the child table then we provide some set of rules to perform delete operations on the parent table. Those rules are called “CASCADE RULES”.

i) **ON DELETE CASCADE**

ii) **ON DELETE SET NULL**

Note: DELETE rules were not imposed on the master table, they are imposed on the child table, and that too on the foreign key column.

==

CASCADE Referential Integrity Constraints in Oracle:-

==

DROP TABLE Employees;

CREATE TABLE Employees(Id number(3) PRIMARY KEY, Name VARCHAR2(20) NOT NULL, DepartmentID number(3)REFERENCES Department(Id) ON DELETE CASCADE);

We have set the ON DELETE rules as CASCADE. This means if we delete a record from the Department table (Primary Key Table) for which if there are some records exist in the Employees table (Foreign Key Table), then those records will also be deleted.

INSERT into Employees VALUES (101, 'Anurag', 10);
INSERT into Employees VALUES (102, 'Pranaya', 20);
INSERT into Employees VALUES (103, 'Hina', 30);

Now, delete the Department whose Id is 10 from the Department table by executing the below SQL Statement.

DELETE FROM Department WHERE Id = 10;

Now, you can see the above DELETE statement executed successfully, and further if you notice the employees whose DepartmentId is 10 are also deleted from the Employees table automatically. You can verify the same by executing the below SELECT statement.

SELECT * FROM Employees;

==

SET NULL Referential Integrity Constraints in Oracle:-

==

Let’s delete the existing Employees table and again create the Employees table. As you can see , we have set the ON DELETE rules as SET NULL. This means if we delete a record from the Department table for which if there are some records exist in the Employees table, then those records will also be set as NULL values. First, delete the Employees table by executing the below SQL Statement.

```
DROP TABLE Employees;
```

Then truncate the Department table and add the three records by executing the below SQL Statement.

```
TRUNCATE TABLE Department;

Insert into Department values (10, 'IT');
Insert into Department values (20, 'HR');
Insert into Department values (30, 'INFRA');
```

Now, create the Employees table by executing the below SQL Statement.

```
CREATE TABLE Employees(Id number(3) PRIMARY KEY, Name VARCHAR2(20) NOT
NULL, DepartmentID number(3) REFERENCES Department(Id) ON DELETE SET
NULL);
```

Now, insert the following records into the Employees table by executing the below INSERT Statements.

```
INSERT into Employees VALUES (101, 'Anurag', 10);
INSERT into Employees VALUES (102, 'Pranaya', 20);
INSERT into Employees VALUES (103, 'Hina', 30);
```

Now, delete the Department whose Id is 10 from the Department table by executing the below SQL Statement.

```
DELETE FROM Department WHERE Id = 10;
```

Now, you can see the above DELETE statement executed successfully, and further if you notice the Employees table, those employees whose DepartmentId is 10 are set to NULL automatically. You can verify the same by executing the below SELECT statement.

```
SELECT * FROM Employees;
```

===

Points to Remember:-

===

If we want to delete a record from the parent table when they have corresponding child records in the child table then we provide some set of rules to perform delete operations on the parent table. those rules are called "cascade rules".

ON DELETE CASCADE

ON DELETE SET NULL

ON DELETE CASCADE: Whenever we are deleting a record from the parent table then that associated child records are deleted from the child table automatically.

ON DELETE SET NULL: Whenever we are deleting a record from the parent table then that associated child records are set to null in the child table automatically.

===

NOTE: Same rules apply for:

ON UPDATE CASCADE and

ON UPDATE SET NULL.

NOTE: Please do the same thing for update query and check it for yourself.

==