###  Introduction To Constraints ====>
======================================

* One of the core functions of any Database Management System is to ensure the integrity of data during its life cycle.

* Data Integrity, in simple terms, means that data should remain ‘consistent’ and ‘accurate’ as time goes by.

* In Oracle, “constraints” are a facility to enforce rules to make sure that only allowable data values are stored in the database.

* A constraint, as the name suggests, puts restrictions/checks on the type or value of data that can be stored in the database table.

##  Types Of Constraints ===>
=============================

* There are five types of Integrity Constraints available in Oracle and they are :
- Not Null
- Check
- Unique
- Primary Key
- Foreign Key

# Not Null ==>
==============

* By default all columns in a table can contain NULL values.

* If we want to ensure that a column must always have a value, i.e. it should not be left blank, then we have to define a NOT NULL constraint on it.

* The database will throw an error if NULL values are entered in the column which has NOT NULL applied on it.

# Check ==>
===========

* Use the CHECK CONSTRAINT to validate values entered into a column.

* For example , in EMP table we might not want the SALARY to be NEGATIVE.

* For such situations we define a CHECK constraint

# Unique ==>
============

* A UNIQUE integrity constraint requires that every value in a column or set of columns (key) be unique—that is, no two rows of a table have duplicate values in a specified column or set of columns.

* For example in a table called STUDENTS , the column called PROJECT_TITLE must be created with UNIQUE constraint as we don’t want two students to have the same Project Title.

* However , NULL value is still allowed.

# Primary Key ==>
=================

* PRIMARY KEY constraint is a combination of NOT NULL and UNIQUE constraints.

* The column or the set of columns on which PRIMARY KEY is defined will allow only UNIQUE and NOT NULL values.

* However PRIMARY KEY itself has a constraint that in a table there can be only one PRIMARY KEY constraint.

# Foreign Key ==> (Referencing table)
=====================================

* It is frequently required that data in one table should be validated by comparing it to data in another table.

* For example , if we add a new order in our ORDERS table, we must crosscheck that a valid product corresponding to this order is present in our PRODUCTS table.

* To achieve this kind of data integrity, FOREIGN KEY constraint is used. This type of validation is also known as REFERENTIAL INTEGRITY.

* A FOREIGN KEY constraint always makes reference to a PRIMARY KEY or a UNIQUE constraint of other tables.

* The table that has a FOREIGN KEY defined is called child table or referencing table.

* The table that has a PRIMARY KEY or UNIQUE constraint defined is called parent table or referenced table.

##  Techniques Of Applying Constraints ===>
===========================================

* Oracle allows us to apply constraints either at the COLUMN LEVEL or at the TABLE LEVEL :-

* Column-level constraints :- Are declared as part of a column definition and apply only to that column. They are also called INLINE CONSTRAINTS.

* Table-level constraints :- Are declared independently from any column definitions (traditionally, at the end of a CREATE TABLE statement) and may apply to one or more columns in the table. A table constraint is required when we wish to define a constraint that applies to more than one column. They are also called OUT OF LINE CONSTRAINTS.

##  Column Level Constraints ====>
==================================

# Syntax Of Applying Column Level Constraints ==>

- CREATE TABLE table_name (

```
Column1 datatype (size) constraint <constraint_name>
<constraint_type>,
Column2 datatype (size) constraint <constraint_name>
<constraint_type>,
Column3 datatype (size) constraint <constraint_name>
<constraint_type>,
………. );
```

# Create a table called STUDENTS with the following columns and constraints:

```
Column Name    Data Type   Constraint
===========    =========   ==========
Roll_No          Number    Should not accept repeating values
Name            Varchar2   hould not accept null values
Percentage       Number
```

```
- Create Table STUDENTS(
Roll_No NUMBER(3) constraint ST_RN_UN UNIQUE,
Name VARCHAR2(15) constraint ST_NM_NN NOT NULL,
Per NUMBER(5,2)
);
```

```
- desc students;
Name                    Null?    Type
--------------------- -------- ----------------------------
ROLL_NO                         NUMBER(3)
NAME                  NOT NULL  VARCHAR2(10)
PER                             NUMBER(5,2)
```

# Create a table called VENDOR_MASTER with the following columns and constraints:

```
Column Name     Data Type     Constraint
===========    ===========   ============
Vendor_Id        Varchar2    Should be PRIMARY KEY of the table
Product_Id       Varchar2    Should not accept null values
City             Varchar2    Should only accept Bhopal & Indore
```

```
-desc vendor_master;
Name                     Null?    Type
---------------------- -------- ----------------------------
VENDOR_ID              NOT NULL VARCHAR2(10)
PRODUCT_ID             NOT NULL VARCHAR2(15)
CITY                            VARCHAR2(6)
```

## Table Level Constraints ====>
=================================

* A TABLE-LEVEL CONSTRAINT references one or multiple columns and is defined separately, after the definition of all the columns.

* Points To Remember:
- All constraints exceptfor the NOT NULL constraint can be defined at the TABLE LEVEL.

- We must use a TABLE-LEVEL CONSTRAINT if we are constraining more than one constraint on the same column.

- If we are referring to the column of the SAME TABLE in a CHECK CONSTRAINT , then also it should be a TABLE LEVEL CONSTRAINT.

- The syntax for adding REFERENTIAL CONSTRAINT requires additional clause called FOREIGN KEY.

# Syntax Of Applying Table Level Constraints ===>

```
-CREATE TABLE table_name(
Column1 datatype (size) ,
Column2 datatype (size) ,
Column3 datatype (size) ,
constraint <constraint_name>
<constraint_type>(<col_name>),
………. );
```

# Create a table called BOOKS with the following columns and constraints:

| Column Name | Data Type | Constraint |
|---|---|---|
| Book_Id | Number | Should be PRIMARY KEY |
| Book_Name | Varchar2 | Should not accept null values |
| Book_Price | Number | Should allow values between 400 and 700 only |
| Book_Author_Id | Number | Should be foreign KEY |

```
- create table books(
  2  book_id number(3),
  3  book_title varchar2(30) constraint bk_bt_nn not null,
  4  book_price number(3),
  5  book_author_id number(3),
  6  constraint bk_id_pk primary key (book_id),
  7  constraint bk_pr_ch check(book_price between 400 and 700),
  8  constraint bk_aid_fk foreign key (book_author_id) references authors)
```

```
- desc books;
Name                            Null?    Type
----------------------------- -------- -------------
BOOK_ID                         NOT NULL NUMBER(3)
BOOK_TITLE                      NOT NULL VARCHAR2(30)
BOOK_PRICE                               NUMBER(3)
BOOK_AUTHOR_ID                           NUMBER(3)
```

# Create a table called ORDER_DETAILS with the following columns and constraints:

| Column Name | Data Type | Constraint |
|---|---|---|
| Order_Id | Number | Should be PRIMARY KEY |
| Prod_Id | Number | Should not accept null values |
| Ord_Date | Date | |
| Del_Date | Date | Should be greater than Ord_Date |

```
- Create Table ORDER_DETAILS(
Order_Id NUMBER(3),
Prod_Id NUMBER(3) constraint ORD_DET_PID_NN NOT NULL,
Ord_Date DATE,
Del_Date DATE,
Constraint ORD_DET_OID_PK PRIMARY KEY(Order_Id),
Constraint ORD_DET_DD_CH CHECK(Del_Date > Ord_Date));
```