**Day 1 : 10/16/20**

emp.txt

Id, name, salary

store employee details

pipe command

ls | mkdir Abc

command | command | command | command

**Day 2 : 10/19/20**

\* Unix Command line

Given a file, say employee.csv, containing data in the following format

empid,initials,department,leave\_available

1000,KK,CTO\_OFFICE,12

2001,MA,HEXAVARSIT,8

1001,IK,CTO\_OFFICE,9

2002,UM,HEXAVARSIT,7

1002,SK,CTO\_OFFICE,13

2003,VR,HEXAVARSIT,6

2004,JK,HEXAVARSIT,6

\* Give an unix command which returns the record with the maximum leave\_available

\* In the above case, the output should be

1002,SK,CTO\_OFFICE,13

$ --- > bash shell

% ---🡪 C shell

echo $SHELL

/bin/csh

/bin/bash

cat /etc/shells

\* Give an unix command which returns one row per department - department, sum of leave\_available of all employees in that department

\* In the above case, the output should be

CTO\_OFFICE,34

HEXAVARSIT,2­­7

**Day 3 :**

simple if

case (switch statement)

1: create the file : customer.txt

If file present don’t create else create it.

2: add customer info cusid,custname,age

3: display all customer details.

4: display specific customer details using custId

5:

**3 days (HTML/CSS/JavaScript)**

https://[www.google.com](http://www.google.com) URL : Uniform resource locator

http : hyper text transfer protocol (secure)

www: world wide web

google : domain

com : commercial

------http/https(req)--------------------🡪

Client Server

🡨------http/https(res)------------------

Html/html5

Css/Css3

JavaScript (JS)

Validation using

JavaScript

HTML5 Features

HTML : hypertext markup Language which help to create web pages.

Web pages help to display the contents in different format like bold, italic, clips etc

HTML is use to create static as well as dynamic web page.

HTML provided lot of pre-defined tags (elements)

Syntax

<tagName> opening tag

</tagName> closing tag

<tagName/> self closing tag

Html tags

1. Html
2. Head : meta – data (data about data)
3. Body : actual contents to display on web page.
4. P (paragraph tag)
5. Br (break tag) : it doesn’t contains closing tag
6. Heading tag (6 types start from h1(largest) to h6(smallest))
7. b (bold)
8. i (italics)
9. u (underline)

Attribute : Attribute is known as properties of a tags.

Syntax

<tagName name1=”value1” name2=’value2’ name3=value3></tagName>

P and heading tag attribute

<p align=’left/right/center/justify’></p>

<h1 align=’left/right/center/justify’></h1>

Font tag : This tag is use change color, face(style) and size.

Image tag :

<img src=”imageName.jpeg/gif/” />

Hyperlink tag : This tag is use to connect external web page as well as point

to specific para(contents) within a same web page.

2 types

1. External hyperlink
2. Internal hyperlink (bookmark)

External hyperlink

<a href=”pathOfPage”>Text</a>

a – anchor

href – hyper reference

Add the images

**List tags:**

It is use to display the items

1. UnOrder List
2. Order List
3. Definition list

**Table Tag :**

Employee Details

|  |  |  |
| --- | --- | --- |
| **Id** | **Name** | **Salary** |
| 100 | Ravi | 12000 |
| 101 | Seeta | 14000 |
| 102 | Meeta | 16000 |

Form Tag :

Syntax

<input type=”text/password/radio/checkbox/button/submit/reset/file” />

Query Param :

url?key=value&key=value&key=value

Get : By default every form method.

Data will send through url using query param concept.

We can send maximum 256 character data.

Performance fast.

Post : we have write method=”post” in form tag

Data send through body part of http protocol.

Data secure and we can send huge data.

Performance slow compare to get methods.

Get/Post/Put/Delete : But html /html5 doesn’t support put and delete method.

Get : Get the Resources ie employee, customer, product

Post : Create the Resource ie employee, customer, product

Put : Update the Resources ie employee, customer, product

Delete : Delete the Resources ie employee, customer, product

**Day 4 :**

Html 4

.xhtml (document type definition )

<!doctype public url=”…………………………..dtd”>

Html 5

<!DOCTYPE html>

VSCode (Visual Studio) : UI training (angular / react )

.net

Web Page

3 parts

1. Contents : HTML/HTML5 (Basic Action using submit and reset)
2. Presentation : Look and Feel : CSS/CSS3
3. Action : JavaScript

CSS : Cascading Style sheet :

With the help of CSS we can apply good look and feel (formatting style for the contents).

We can achieve **separation of concern** : Contents separate and formatting style.

CSS 3 types

1. Inline CSS
2. Internal CSS or embedded CSS
3. External CSS

Inline CSS

Syntax

<tagName style=”property:value;property:value;”></tagName>

Internal CSS or embedded CSS

Syntax

In head

<style>

Selector {property:value;property:value;}

</style>

Types of Selectors

1. \* universal selector

\*{color:green;}

1. Specific selector

tagName {property:value}

1. Multi selector

tagName, tagName {property:value}

1. Class selector (Local class selector)

tagName.className {property:value}

1. Global class selector

.className {property:value}

1. Id selector

#idName{property:value}

Class selector Vs Id selector

Class : groups of the tag

Id : unique for that tag

DOM : Document Object Model

**JavaScript:**

JavaScript was object based interpreter scripting language. Which help to do client side validation as well as to create dynamic effect for the web page.

JavaScript using ES5 Features

ECMAScript : It is a concept. And JavaScript is a implementation of ES5.

Object Based Vs Object Oriented

Class, object, Encapsulation, Inheritance, Polymorphism and abstraction.

JavaScript is object oriented interpreter scripting language from ES6 onwards.

All browser doesn’t support ES6 all features.

Interpreter

Compiler

Both are translator : converting one format to another format.

Compiler convert all code at time where interpreter convert line by line.

Syntax (script tags)

<script type=”text/JavaScript”> opening tag

</script> closing tag

This tag must be in between head or body or within any html tags.

Variable : variable is a name which hold some value.

Value will change during the execution of program.

JavaScript is loosely data types.

int a=10; //C or C++ or Java

To declare the variable in JS we have keyword as **var**

Syntax

var variableName;

DataTypes

number

string

boolean

object reference

undefined

etc

primitive data types : it is use to store only value.

Syntax to create the memory of pre-defined or user-defined objects

var referenceName = new ObjectName();

var obj = new Date()

var obj = new String();

var obj = new Number();

Operators:

Arithmetic Operator :

+, -, \*, /, %(remainder)

Relation Operator

>, >=, <, <= , ==, !=

Logical Operator

&& || !

Assignment Operator

=

Increment and decrement operator

++ , --

++ increment the value by 1

--decrement the value by 1

Pre-increment and post increment

Pre-decrement and post decrement

++n; increment and store or assign

n++; assign or store and then increment

Ternary operator

Condition ? true : false;

===

if statement

1. Simple if

if(condition){

true block

}

1. If else

if(conditions) {

true block

}else {

false block

}

1. Nested if

if(condition) {

if(conditions) {

true block

}else {

False block

}

}else {

if(conditions) {

}

}

1. If ladder or if else if

if(conditions) {

}else if(conditions) {

}else if(conditions) {

}else {

}

switch statement

syntax

switch(variableName) {

case value:block1;

break;

case value:block2;

break;

case value:block3;

break;

default: wrongblock;

break;

}

switch, case, break and default are keywords.

Looping : it is use to execute the statement again and again till the conditions becomes false.

While loop

Do while loop

For loop

**functions :**

set of instruction to perform a specific task.

1. Pre-defined function
2. alert(“Msg”); This function is use to display alert or popup message on web page.
3. prompt(); This function is use to receive the value through keywords.
4. eval() : It is use to convert string to number
5. confirm : This popup contains two button ok and cancel

If user click ok then it return true else return false.

1. User-defined function

Do

1: Add , 2 :Sub : alert

: prompt

Switch statement

1 : Addition of two number

Alert to receive the value of a and b

eval : convert string to number

2: Subtraction of two number

Alert to receive the value of a and b

Confirm (to do the another operation)

While

**HTML5 : Few forms**

**CSS**

**JavaScript**

**Day 4 :**

user-defined function

syntax

function functionName(parameterList) {

}

1. function no parameter and no return type

2. function passing parameter and no return type

3. function no passing parameter but return type

4. function passing parameter and return type

ES6 React JS

Events : Events is interaction between user and html components.

It is delegation model.

Types of Events

In JavaScript all events start with on\*

like

onClick

onDblClick

onMouseOver

onMouseOut

onKeyUp

onKeyDown

onSubmit

onBlur

onFocus

onChange

etc

Event is provide bridge between html and JavaScript

DOM : document object model

DOM is a api(application programming interface)

which provide set of method and properties which help to read, write and

update html contents dynamically.

JavaScript objects :

In JavaScript object are divided into two types

pre-defined objects.

object : any real world entity.

JavaScript internally follow object hierarchy.

object ---> property (state) ---> have - varialbes

behaviour --------->do/does - functions

object

property (variables)

behaviour (functions)

object

property

behaviour

object

BOM : Browser Object Model

DOM : Document Object Model

user-defined objects.

Interval Functions

setInterval()

setTimeout()

clearInterval()

Day 5 :

**Box Model :**

div{

    border-style: groove;

    border-width: 1;

    border-color: red;

    width: 200px;

    height: 300px;

}

div{

    border: 2px solid red;

    width: 200px;

    height: 300px;

}

Position : Relative ; we can use top, left, right and bottom properties for that tag

Position : Absolute : We can’t top, left, right, bottom this property

But user can decided where you want to display that tag base upon the event.

Position : Fixed

**Database**

Copy OJDBC jar from here: H:\common\Trainee\_2020

<< OLE Object: Picture (Device Independent Bitmap) >>

The userids are train1-20  the password for **train1**is ‘**pitchers1**’  for

train2 the password is ‘pitchers2’, etc.

   The tnsnames entry you’ll need is below.

INDIATRN =

  (DESCRIPTION =

    (ADDRESS = (PROTOCOL = TCP)(HOST = [indiaoracletraining-dev-oracle-rds.c2zuiscfni9y.us-east-1.rds.amazonaws.com](http://indiaoracletraining-dev-oracle-rds.c2zuiscfni9y.us-east-1.rds.amazonaws.com/))(PORT = 1521))

    (CONNECT\_DATA = (SID = INDIATRN)))

Username: train1

**Database : (Oracle ) -- > SQL / PLSQL Using Oracle 5 days**

**SQL -- > 3 days**

**PlSQL 🡪 2 SQL**

**Data : Raw facts**

**Information : Processed Data or Meaningful data.**

**Database : Storing the data and information in proper or common (tables) format.**

**Input : keyboard, initialization , file, database or through network**

**Process : business rules.**

**Output : console, storing in file, database, through send to network.**

1. **File**
2. **Database**

**File base system**

1. **We can store data permanently.**
2. **We can store huge data.**
3. **We can apply security for data (read, write or execute)**

**Limitation of file base system**

1. **Data redundancy (means storing same data again and again) (Duplicate records)**
2. **Data inconsistency (format of the file).(txt, doc, xsl, pdf, etc)**

**.txt (delimiter for each records as next line or / or , or tab space)**

**.txt(delimiter for each data**

**Emp.txt**

**Id,name,salary**

**IdNameSalary**

1. **Security (Read or Write mode (Execute shell scripting or batch file))**

**Database : Tables Format (Table format)**

**DBMS : Database Management System : It is a software which help to store the Data in the table format like columns and rows.**

**Excel Sheet. DBMS**

**VB (Visual Basic ) with Excel**

**Database Model :**

**HDBMS : Hierarchal Model**

**NDBMS :**

**RDBMS : Relation DBMS (Database management system)**

**1970**

**Dr. EF Codd’s Rules : 12 Rules – 0 to 11**

**The Database which support all 12 rules which provided by EF Codd’s rules**

**RDBMS : Relational Database Management System**

**RDBMS Databases**

**Database Name Company**

**MySQL Sun MicroSystem (Oracle)**

**Oracle Oracle**

**SQL Server Microsoft**

**2019, 2020**

**Db2 IBM**

**SQL : Structure Query Language**

**SQL Developer tools**

**Commands (CLI )Command line interface.**

**80% to 90%**

**SQL : Five types (Sub types )**

1. **DQL(Data Query Language) or DRL(Data Retrieval Language) (View Records)**
   1. **Select Clause** 
      1. **Select all records**
      2. **Select specific records**
      3. **Select with where clause**
      4. **Select with group by clause**
      5. **Select with having clause**
      6. **Select with order by**
      7. **Select with join**
      8. **Select with sub query**
      9. **Select with union operator**
2. **DDL (Data Definition Language) (table structure )**
   1. **Create**
   2. **Drop**
   3. **Alter**
   4. **Rename**
   5. **Truncate**
3. **DML (Data Manipulation Language) (work on data )**
   1. **Insert**
   2. **Delete**
   3. **Update**
4. **TCL (Transactional Control Language)**
   1. **Commit**
   2. **Rollback**
   3. **Savepoint**
5. **DCL (Data Control Language)** 
   1. **Create the user**
   2. **Grant the permission to the user**
   3. **Revoke the permission from the user**

**View**

**Index**

**Sequence**

**SQL :**

**PlSQL :**

**MySQL/ SQL Server / Db2 --🡪**

**username : root**

**password : \*\*\*\*\*\***

**Oracle :**

**Username : scott**

**Password : tiger**

**Sql server**

**Username : sys**

**Password : \*\*\*\***

**NoSQL**

**MongoDB**

**MySQL/ SQL Server / Db2 --🡪**

**show databases; This command is use do display the database.**

**use databaseName; This command is use to move the specific database.**

**Database : database is a collection of tables or views etc.**

**show tables; This command is use to display all tables present in existing database.**

**create database databaseName; This command is use to create the database.**

**Above four command are invalid in oracle database.**

**username itself is a database in oracle database.**

**system**

**scott**

**admin**

**oracle**

**hr**

**select \* from dual;**

**select \* from tab; : tab is a pre-defined table provided by oracle databases which help to display all pre-defined as well as user-defined tables table in that account.**

**Oracle eXpress Edition : small database. It support basic feature of oracle database.**

**Oracle enterprise edition : it support all features of oracles.**

**ORDBMS**

**Day 7 : 27-10-20**

**DBMS : Database Management system**

**RDBMS : Relational Database Management System**

**DBMS 🡪 Table, Columns , Rows**

**RDBMS -🡪 Relation, Attribute , Records (Tuple)**

**Trainer\_Student**

**TId TName Tech Sid SName Age**

**1 Raj Java 101 Seeta 21**

**1 Raj Java 102 Reeta 22**

**1 Raj Java 103 Meeta 23**

**2 Ravi C/C++ 104 Keeta 24**

**2 Ravi C/C++ 104 Teeta 25**

**Trainer**

**PK(Primary Key): doesn’t allow duplicate as well as null value.**

**TId TName Tech**

**1 Raj Java**

**2 Raj Java**

**3 Ravi C/C++**

**Students**

**PK FK(Foreign Key)**

**Sid SName Age TS\_Id**

**100 Seeta 21 1**

**101 Meeta 22 1**

**102 Keeta 23 2**

**103 Teeta 24 3**

**104 Leeta 26 null**

**FK is use to refer the primary of same table or different table. FK allow only those records(values) which present in primary key. It allow null value.**

**SQL : Structure Query Language**

1. **DRL or DQL (Data Query or Retrieval Language)**

**MySQL/SQL Server / Db2**

**Create database mydb;**

**Use mydb;**

**Show tables;**

**Scott : is database**

**Select \* from tab;**

**Select \* from tableName;**

**This command is use to display all records from that table**

**Table structure**

**desc tableName;**

**Filter the columns**

**Select columnName, columnName, columnName from tableName;**

**Columns alias**

**Select columnName as columnAlias, columnName columnAlias from tableName**

**select employee\_id as id,first\_name||' '||last\_name Full\_Name,salary,salary+5000 as Gross\_Salary from employees;**

**Employee\_Id,Full\_Name, BasicSalary, HRA, DA,PF,Gross\_Salary**

**Salary = 10%**

**DA = 5%**

**PF = 7%**

**GrossSalary = Salary + HRA + DA – PF**

**Filter records using Where clause**

1. **Relational Operator**

**>, >=, <, <=, = (equal), != or <>**

**Syntax**

**Select \* from tableName where columName RO value;**

**Select columName, columnName from tableName where columName RO value**

**Select \* from employees where salary > 15000;**

**Select employee\_id,first\_name from employees where salary > 15000;**

**Select employee\_id,salary from employees where first\_name =’Steven’;**

**select first\_name,job\_id,department\_id from employees where department\_id <> 90;**

**MySQL default date format is yyyy-mm-dd**

**Oracle default date format is dd-mon-yy**

**select first\_name,salary from employees where hire\_date ='14-Jun-97';**

**select first\_name,salary from employees where hire\_date > '31-Mar-95';**

1. **Between operator (Numerical and Date value)**

**Filter the range of records**

**Select \* from tableName where columnName between min and max**

**select first\_name from employees where employee\_id between 100 and 150;**

**select first\_name,salary from employees where salary between 10000 and 17000;**

1. **In operator (more then on specific)**

**Select \* from tableName where columName in(v1,v2,v3)**

**Value number then write directly if value are alphanumeric or date then write in single quote.**

**select first\_name from employees where employee\_id in(100,110,105,150,1000);**

**select first\_name,job\_id from employees where job\_id in('ST\_MAN','IT\_PROG');**

1. **Like operator (Specific only one records)**

**Select \* from tableName where columnName like value;**

**select employee\_id,salary from employees where first\_name like 'Steven';**

**select employee\_id,salary from employees where first\_name = 'Steven';**

**select employee\_id,first\_name,salary from employees where first\_name like 'S%';**

**Start with S Character**

**select employee\_id,first\_name,salary from employees where first\_name like '%n';**

**end with s character**

**select employee\_id,first\_name,salary from employees where first\_name like '%r%';**

**contains r character**

**% zero or one or many**

**select employee\_id,first\_name,salary from employees where first\_name like 'S\_e%';**

**Start with S, then 2nd character may be anything, 3rd may be e then one or more any character.**

**\_ any one single character**

1. **is null :**

**select first\_name,salary,commission\_pct from employees where commission\_pct is null;**

1. **logical and, or, not**
2. **select \* from tableName where columnName op value and columnName op value (both the conditions must be true)**
3. **select \* from tableName where columnName op value or columnName op value (any condition must be true then we will get the records)**

**select first\_name,salary from employees where salary > 10000 and department\_id=90;**

**select first\_name,salary from employees where salary > 10000 or department\_id=90;**

1. **not**
2. **not between min and max**
3. **not in (v1,v2,v3)**
4. **not like value**
5. **is not null**

**Order by clause ( it is use to display the records in ascending or descending order)**

**select first\_name, salary from employees order by salary asc;**

**select first\_name, salary from employees order by salary;**

**select first\_name, salary from employees order by salary desc;**

**Multi sort (Order by clause )**

**select first\_name,department\_id,salary from employees order by department\_id asc,salary asc;**

**Functions**

**Function is use to write the set of instruction to perform a specific task.**

**Oracle**

**Dual**

**Select \* from dual;**

**2 types**

**All function takes one or more than one parameter and return the value. ( it must return the values).**

1. **Pre-defined function**

**Two types**

1. **Single row function : The output or return type apply for every records individually depends upon the function.**
2. **Character function**

**select upper(first\_name) from employees;**

**select upper('raj') from dual;**

**select upper('raj'),lower('RAJ'),initcap('raj') from dual;**

**select concat(first\_name,' ',last\_name) from employees; (it takes only two parameter)**

**select concat(first\_name,' ',last\_name) from employees;(In MySQL database it takes n parameter)**

**nested function**

**select concat(first\_name,concat(' ',last\_name)) from employees;**

**(nested function, first inner function execute, inner function output is input for outer functions).**

**or**

**select first\_name|| ‘ ‘ ||last\_name from employees**

**( it doesn’t support in MySQL)**

**Select substr(content,start,numberOfChar) from dual**

**select substr('rajdeepkumar',4) from dual;**

**select substr('rajdeepkumar',4,4) from dual;**

**select length('Raj Deep') from dual;**

**select trim(‘ Raj Deep ‘) from dual; help to remove the space before and after**

**select length(trim(' Raj Deep ')) from dual;**

**Number function**

**round()**

**trunc()**

**Date function :**

**Sysdate**

**Months\_between(newDate,oldDate)**

**select months\_between(sysdate,'01-Jan-20') from dual;**

**add\_months(sysdate,3)**

**select add\_months(sysdate,4) from dual;**

**select next\_day(sysdate,'Sun') from dual;**

**select last\_day(sysdate) from dual;**

1. **Display all employees first\_name and numberOfYearOfExp from employees tables.**

**Hire\_date (**

**20 to 30 year**

**Remove decimal number but trunc**

1. **YearOfExp more than 25 years**

1. **Multi row function or aggregate function : The output or return type apply for group of records depending upon the group by clause. (if we doesn’t use group by clause we will get only one output for multi row function if the table contains one or more than one records).**

**Day 8 : 10/28/2020**

**Conversion functions**

**Oracle Database Data types**

**number(10) -🡪 maximum number of digits is 10**

**number(10,2)-🡪 total number is 10, before decimal 8 and after decimal 2**

**int ---🡪**

**float -🡪**

**char 🡪 it is use to store single character**

**varchar(10)-🡪 number of character**

**varchar2(10) --🡪number of character**

**date -🡪 it is use to store date and time information**

**timestamp -🡪 to store time**

**etc**

**to\_char : converting date to varchar/varchar2**

**to\_char(dateDetails,’Date Format’)**

**select to\_char(sysdate,'YYYY') from dual;**

**select to\_char(sysdate,'DD-MM-YYYY') from dual;**

**select to\_char(sysdate,'DD-MON-YYYY') from dual;**

**select to\_char(sysdate,'DD-MON-YYYY HH:MI:SS') from dual;**

**select to\_char(sysdate,'DAY DD MONTH YEAR') from dual;**

**to\_char : converting number to char**

**to\_date :**

**to\_number**

**nvl : null value**

**nvl(null,1) : If value is null replace by second parameter**

**nvl(‘Raj’,1) : if value not null contains same value.**

**select first\_name,salary,nvl(commission\_pct,0) from employees;**

**nvl2(first\_parameter,second\_parameter,third\_parameter)**

**if first\_parameter is a not null then we will get second parameter**

**if first\_parameter is equal to null then we will get third parameter.**

**Case (This function is equal to switch statement in programming language)**

**Syntax**

**Case expr when comparision\_expression then return\_value**

**when comparision\_expression then return\_value**

**when comparision\_expression then return\_value**

**else**

**return\_value end**

**Job\_ID :**

**select first\_name,salary,Job\_id,case job\_id when 'AC\_MGR' then salary\*0.10 when 'SH\_CLERK' then salary\*0.05 else salary\*0.01 end as Bonus from employees;**

**Decode : decode is a short cut of case function**

**Select first\_name,salary,Job\_id,decode(job\_id,’AC\_MGR’,salary\*0.10,’SH\_CLERK’,salary\*0.05,salary\*0.01) as bonus from employees**

**Rather display 0 u have to display the message as no commission. (hint you have to take conversion function help).**

**nvl2**

**Multi – row functions or aggregate functions**

1. **sum()**
2. **max()**
3. **min()**
4. **avg()**
5. **count()**

**select sum(salary) from employees;**

**select sum(salary) as Total\_Salary from employees;**

**select max(salary) from employees;**

**select min(salary) from employees;**

**select sum(salary) as total,min(salary) as min\_sal,max(salary) as max\_sal, avg(salary) as avg\_salary from employees;**

**select count(commission\_pct) from employees;**

**select count(manager\_id) from employees;**

**select count(employee\_id) from employees;**

**select count(\*) from employees;**

**select count(nvl(commission\_pct,0)) from employees**

**Sub group**

**Group by Clause**

**Employees table -🡪 FK Columns (Job\_Id, Department\_id, Manager\_Id)**

**Employee\_Id 🡪 PK**

**Group by Department Id**

**select sum(salary) from employees group by department\_id;**

**select department\_id,sum(salary) from employees group by department\_id;**

**aggregate function with where clause and group by**

**select department\_id,sum(salary) from employees where department\_id is not null group by department\_id;**

**Having clause**

**Having clause must be after group by where clause must be before group by**

**Having clause is use to check the condition on group of records where clause is use to check the condition on single records**

**Where clause (Single row function)**

**select department\_id,sum(salary) from employees where first\_name like initcap('steven') group by department\_id;**

**Having clause (Multi row function)**

**select department\_id,sum(salary) from employees where department\_id is not null group by department\_id having sum(salary) > 50000;**

**select department\_id,sum(salary) from employees where department\_id is not null group by department\_id having sum(salary) > 50000 order by department\_id desc;**

**Single row and Multi row assignment Question**

Single row function12:24

1. Display first name, salary, and round the salary to thousands.12:24

2. Display the first word in job title.12:24

3. Display the length of first name for employees where last name contain character ‘b’ after 3rd position.12:24

4. Display the number of days between system date and 1st January 1995.12:25

5. Display how many employees joined in each month of the current year.12:25

Multi row functions12:26

1. Display average salary of employees in each department who have commission percentage.12:26

2. Display job ID for jobs with average salary more than 10000.12:26

3. Display years in which more than 10 employees joined.12:27

4. Display departments where any manager is managing more than 5 employees.12:27

5. Display department id and number of employees in the department.

**DDL and DML**

**Data Definition Language: Working Table Structure (Create, Drop and Modify).**

**Create**

**Drop**

**Truncate**

**Alter**

**Syntax to create the Table**

**Create table tableName(columnName dataTyp1, columnName dataType2…..columnName dataTypen);**

**Table 🡪EmpDetails**

**Id Name Salary**

**Number(10), varchar2(10) number(10,2)**

**create table empdetails(id number(10), name varchar2(10), salary number(10,2));**

**desc empdetails**

**DML: : Data Manipulation Language (Insert, Delete and Update)**

**Insert query**

**Insert into tableName(c1,c2,c3) values(v1,v2,v3);**

**insert into empdetails(id,name,salary) values(1,'Raj',12000);**

**insert into empdetails(id,name) values(2,'Ravi');**

**insert into empdetails values(5,'Ajay',null);**

**insert into empdetails(id) values(3);**

**insert into empdetails values(6,null,null);**

**insert into empdetails values(4,'Ramesh',14000);**

**insert into empdetails(name,salary,id) values(‘Dinesh’,14000,7);**

**Update Query**

**Syntax**

**Update tableName set columnName = value;**

**Update empdetails set salary = 25000;**

**Update query with where clause**

**Update tableName set columnName = value where columnName = value;**

**update empdetails set salary = 15000 where id=2;**

**update empdetails set name = 'Balaji' where id=3;**

**update empdetails set salary = 18000 where id in(3,5,6);**

**update empdetails set name = 'Ramesh' where id =7;**

**update one value base upon more than one condition satisfies**

**update one value base upon any one condition satisfies**

**update empdetails set salary=20000 where name like '%e%' and id=4;**

**update empdetails set salary=22000 where name like '%e%' or id=2;**

**update more than one value depending upon one conditions.**

**update empdetails set name='Raj Deep',salary=salary+2000 where id=1;**

**Delete Query**

**Syntax**

**Delete from tableName;**

**All records delete from table**

**Delete with where clause**

**Delete from tableName where columnName = value;**

**delete from empdetails where id=7;**

**delete from empdetails where salary > 20000;**

**delete from empdetails where name like 'Raj';**

**Drop table**

**Delete from tableName (without where clause)**

**All records are deleted but table structure present in database.**

**Drop table tableName; (All records as well as table structure remove from database).**

**drop table empdetails;**

**Truncate table tableName; All records are deleted but table structure available in database.**

|  |  |  |
| --- | --- | --- |
| **Truncate** | **Delete** | **Drop** |
| **DDL** | **DML** | **DDL** |
| **All records deleted** | **All records deleted** | **All records deleted** |
| **Table structure available** | **Table structure available** | **Table structure removed** |
| **We can’t use where clause** | **We can use where clause** | **We can’t use where clause** |
| **We can’t use rollback (undo)** | **We can use rollback (Undo)** | **--** |

**Alter command**

**Add column**

**Modify column**

**Drop column**

**alter table empdetails add desg varchar2(2);**

**alter table empdetails modify desg varchar2(10);**

**alter table empdetails drop column salary;**

**select \* from empdetails;**

**Rename tableName**

**alter table empdetails rename to employeedetails;**

**alter table employeedetails rename column desg to designation;**

**Join :**

**De-Normalization**

**Join is use to retrieve more than one columns from more than one table with or without conditions.**

**Create table table1(srno int, name varchar2(10), salary float)**

**Insert into table1 values(1,’Raj’,12000);**

**Insert into table1 values(2,’Reeta’,14000);**

**Create table table2(accno int, name varchar2(10), amount float)**

**Insert into table2 values(111,’Raj’,500);**

**Insert into table2 values(222,’Seeta’,1000);**

**Cartesian Product : m \* n**

**select srno,salary,accno,amount from table1, table2;**

**2 \* 2 = 4**

**select srno,table1.name,salary,accno,table2.name,amount from table1, table2;**

**but with common columns**

**Table alias**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2;**

**select \* from table1, table2;**

**Equi – Join**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2 where t1.name=t2.name;**

**Outer Join**

**Left outer join**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2 where t1.name=t2.name(+);**

**Right outer join**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2 where t1.name(+)=t2.name;**

**Inner Join**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 inner join table2 t2 on t1.name=t2.name;**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 join table2 t2 on t1.name=t2.name;**

**Outer Join**

**Left outer join**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 left outer join table2 t2 on t1.name=t2.name;**

**Right outer join**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 right outer join table2 t2 on t1.name=t2.name;**

**Full Outer join**

**select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 full outer join table2 t2 on t1.name=t2.name;**

**Employee**

**10**

**5 working on project**

**5 on bench**

**Project**

**10**

**4 active**

**6 not started yet**

**Equi – Join Or Inner Join**

**N table it require n-1 min conditions**

**2 table it require 1 condition**

**3 table it require 2 condition (1st – 2nd table and 1-3rd table or 2 – 3rd table)**

**Using where clause – Equi – Join**

**select emp.first\_name,emp.salary,dept.department\_name from employees emp, departments dept where emp.department\_id=dept.department\_id;**

**Using on clause – Inner join**

**select emp.first\_name,emp.salary,dept.department\_name from employees emp join departments dept on emp.department\_id=dept.department\_id;**

**Self – Join : Joining the same table itself is known as self join.**

**When doing join it not mandatory column name must be same but the value as well as their data types must be same.**

**TCL :**

1. **User-defined function (Pl SQL )(Stored Procedure and functions)**

**Day 9 : 10/29/20**

**Sub Query**

**Query within another query is known as sub query**

**Syntax**

**Outer Query (Inner Query)**

**First Inner Query will execute and Inner query output is input for Outer Query.**

**Outer Query (Inner Query (Inner Inner Query ))**

**Inner Query must be return only one column (means not more than one or \* ).**

**Inner Query**

**Some query retrieve only one record or result (If inner query using Aggregate function (Without Group By Clause) or where clause with primary key).**

**Some Query retrieve more than one records**

**Sub Query mainly divided into two types**

1. **Single row sub query**
2. **Multi row sub query**

**Syntax of Single row sub query**

**Outer query RO/Like (Inner Query)**

**Syntax of Multi row sub query**

**Outer query in / any ro / all ro (Inner Query)**

**Find person name or details whose salary is > avg salary of all employees.**

**select avg(salary) from employees;**

**select first\_name from employees where salary > 6461;**

**select first\_name from employees where salary > (select avg(salary) from employees);**

**select first\_name from employees where salary > (select max(salary) from employees where department\_id=90);**

**Multi – Row functions**

**In Operator**

**Find the person details whose job\_id is equal to those employee working in a department 90.**

**Find the manager\_name(first\_name) who manages more than 4 employees**

**select count(\*) from employees group by manager\_id;**

**select first\_name from employees where employee\_id in (select manager\_id from employees group by manager\_id having count(\*) > 4);**

**Find first\_name whose salary is greater than any of the employee working department 60**

**Inner Query**

**select salary from employees where department\_id=60**

**min\_salary = 4200**

**max\_salary = 9000**

**any**

**>any ( > min\_salary of inner query) ie > 4200**

**select first\_name from employees where salary >any (select salary from employees where department\_id=60);**

**select first\_name,salary from employees where salary >any (select salary from employees where department\_id=60);**

**select first\_name,salary from employees where salary > (select min(salary) from employees where department\_id=60) ;**

**Find first\_name whose salary is greater than all of the employee working department 60**

**all**

**>all ( > max\_salary of inner query) ie > 9000**

**select first\_name,salary from employees where salary >all (select salary from employees where department\_id=60);**

**select first\_name,salary from employees where salary > (select max(salary) from employees where department\_id=60);**

**exists**

**not exists**

**outer Query where exists (inner query)**

**If Inner query return true then outer query execute**

**If inner query return false then outer query doesn’t execute.**

**select first\_name from employees where exists (select \* from employees where department\_id=100);**

**Keys :**

**EmpId,EmpName,Age,Salary,PhNumber,EmailId, jobId,JobDesc,accnoNumber,typeOfAccount,custId,customername,addId,city,state,pincodde,projectId,projectName,startDate,endDate etc**

**Etc**

**Super key : all set combination are known as Super set**

**EmpId**

**AccNumber**

**custId**

**AddId**

**porjectId**

**phNumber**

**EmailId**

**JobId**

**EmpId,EmpName**

**JobId,JobDesc**

**projectId,projectName**

**EmpId,phNumber**

**EmpId,phnumber,FirstName**

**EmpId,Phnmber,FirstName,jobId**

**Candidate keys :**

**EmpId,Phnmber,FirstName,jobId : Super Key**

**EmpId,Phnmber,FirstName :Super key**

**EmpId,Phnmber : Super key**

**EmpId :Super key**

**PhNumber :Super**

**FirstName : not a super key**

**JobId :Super key**

**EmpId,PhNumber, JobId are Candidate keys.**

**Primary key**

**Primary key :**

**EmpId PK**

**AccNumber CK**

**custId CK**

**AddId CK**

**porjectId CK**

**phNumber CK**

**EmailId CK**

**JobId CK**

**In single table only one column as a primary key**

**It doesn’t allow duplicate as well as null value.**

**Unique Key :**

**EmpId PK**

**AccNumber UniqueKey**

**custId UniqueKey**

**AddId UniqueKey**

**porjectId UniqueKey**

**phNumber UniqueKey**

**EmailId UniqueKey**

**JobId UniqueKey**

**In Single table we can create more than one column as unique key**

**It doesn’t allow duplicate but it can allow null value.**

**Foreign key :**

**It is use to refer the primary key of same table or different table**

**It allow duplicate records which present in primary key column. It allow null value.**

**Composite Key: If we create primary key(combination of more than one column). Both column independently allow duplicate but combination must be unique.**

**Alternate key : They are type of candidate or unique key**

**Secondary key :**

**Surrogate key : it help to generate auto increment number.**

**Constraints :Constraints is use to restrict the user to insert invalid data like duplicate records, -ve value. Etc**

1. **Not null**
2. **Default**
3. **Check**
4. **Unique**
5. **Primary key**
6. **Foreign key**
7. **Composite primary key**

**EmployeeDetails**

**PK not null null >21 male|female, 250000 and 50000, Default GWG**

**EmpId ,FName,LName,Age ,Gender,Salary,PhNumber,DOB, Company**

**create table employeedetails(**

**empid int primary key,**

**fname varchar2(10) not null,**

**lname varchar2(10),**

**age int check(age>21),**

**gender varchar2(10) check(gender in('male','female')),**

**salary number(10) check(salary between 25000 and 50000),**

**phnumber number(10) unique,**

**dob date not null,**

**company varchar(10) default 'GWG');**

**insert into employeedetails values(100,'Raj','Deep',24,'male',34000,9900,'30-Jan-85','Wipro');**

**insert into employeedetails values(101,'Raj','Deep',24,'male',34000,9901,'30-Jan-85','Wipro');**

**insert into employeedetails values(102,'Raj','Deep',24,'male',50000,9902,'30-Jan-85','Wipro');**

**insert into employeedetails values(103,'Seeta',null,22,'female',45000,9903,'10-Feb-90','IBM');**

**insert into employeedetails values(104,'Reeta',null,22,'female',45000,9904,'10-Feb-92','TCS');**

**insert into employeedetails(empid,fname,lname,age,gender,salary,phnumber,dob) values(105,'Leeta',null,24,'female',30000,9906,'10-Mar-92');**

**Relationship :**

**ER-Model : Entity Relationship Diagram**

**4 types**

1. **One – To – Many (PK-🡪FK) : Trainer – Student**
2. **Many – To – One (FK🡨-PK) : Employee – Department /Project**
3. **One – To – One (PK-🡪 or 🡨 PF) Shared Primary key Person -🡪PanCard/Passport**
4. **Many – To – Many (PK----🡪FK and FK🡨--PK)**

**One – to – Many bi-directional**

**Students <---🡪SkillSet**

**Many – to – Many**

**Trainer**

**PK**

**TId TName Tech (Primary table)**

**1 Raj Java**

**2 Raju .net**

**3 Ravi C**

**PK FK**

**Student**

**Sid SName Age TSId (Secondary Table )**

**101 Seeta 21 1**

**102 Meeta 22 1**

**103 Keeta 23 2**

**104 Leeta 24 2**

**105 Veeta 25 null**

**Trainer**

**create table trainer(tid int primary key,tname varchar2(10) not null, tech varchar(10) not null);**

* **Column level constraints**

**create table trainer(tid int,tname varchar2(10) not null, tech varchar(10) not null,**

**constraints t\_pk primary key(tid));**

* **Table level constraints**

**insert into trainer values(1,'Raj','Java');**

**create table student(stid int primary key,sname varchar2(10) not null, age int, tsid int references trainer(tid));**

* **Column level constraints**

**create table student(sid int,sname varchar2(10) not null, age int, tsid int,**

**constraints s\_pk primary key(sid),**

**constraints ts\_fk foreign key(tsid) references trainer(tid));**

* **Table level constraints**

**insert into student values(100,'Seeta',21,1);**

**We can’t drop primary table if table link with foreign key of another secondary table.**

**Many – to – Many**

**Students**

**Sid PK**

**SName**

**create table students(sid int, sname varchar(10) not null, constraints s1\_pk primary key(sid));**

1. **Raj**
2. **­­Ravi**

**SkillSet**

**SSId PK**

**SSName**

**create table skillset(ssid int, ssname varchar(10) not null, constraints ss1\_pk primary key(ssid));**

**100 C**

**101 Java**

**102 .net**

**Student\_SkillSet ­­**

**SSS\_Id pk**

**Sid FK for Student**

**SSId FK for SkillSet**

**111 1 100**

**112 1 101**

**113 1 102**

**114 2 100**

**115 2 101**

**create table students\_skillset(sss\_id int,**

**sid int,**

**ssid int,   
constraints sss\_pk primary key(sss\_id),**

**constraints sid\_fk foreign key(sid) references students(sid),**

**constraints ssid\_fk foreign key(ssid) references skillset(ssid));**

**select \* from students\_skillset;**

**insert into students\_skillset values(115,2,101);**

**StudentName Technology**

**Use Join**

**Using where as well as on clause**

**One – To – One (Using Shared Primary key)**

**Person (Primary )**

**PK**

**PId PName**

**create table person(pid int, pname varchar(10), constraints pp\_pk primary key(pid));**

**PK**

**PanCard (Secondary)**

**PCId ValidDate**

**create table pancard(pcid int primary key references person(pid), validdateinfo date);**

**or**

**create table pancard(pcid int, validdateinfo date,constraints pc\_pk primary key(pcid), constraints pc\_fk foreign key(pcid) references person(pid));**

**DDL Assignments with Constraints**

**Table ---> Customer**

**custId, firstName,lastName,age,city, mobileNumber, dob**

**Add the Constraints**

**custId is Primary Key**

**firstName not null**

**age must be greater than 21**

**mobile must be unique**

**Table ----> Branch**

**branchId, branchName, city**

**Add the Constraints**

**branchId is Primary Key**

**branchName not null**

**city not null**

**Table -----> Account**

**accountNumber, openingBalance, typeOfAccount, status,BankId,CustId**

**Add the Constraints**

**accountNumber is primary key**

**openingBalance must be greater than 5000**

**typeOfAccount must be saving/current**

**BankId is foreign key refer to BranchId(Primary key) Branch table**

**CustId is foreign key refer to Customer(Primary key) Customer table**

**Table ----> Transaction**

**transactionId, transactionDate, MediumOfTransaction, TransactionAmount**

**Add the Constraints**

**transactionId is primary key**

**Table ----> Loan**

**LoanId, loanAmount, customerId and bankdId**

**Add the Constraints**

**loadId is primary key**

**loanAmount must be +ve**

**BankId is foreign key refer to BranchId(Primary key) Branch table**

**Using alter command add / drop constraints**

**Update and delete cascade**

**TCL : Transactional Control Language**

**Commit**

**Rollback**

**Savepoint**

**DML Operation**

**Insert / Delete and Update**

**Update account set amount = amoun-500 where account =123;**

**Update account set amount = amount+500 where account =567;**

**ACID Properties**

**Savepoint savepointName;**

**Example**

**Savepoiont s1;**

**Sequence : sequence is pre-defined object which help to create auto increment number in oracle database.**

**Create sequence sequenceName;**

**Create sequence pro\_seq**

**pro\_seq.nextval : It help to increment the number start from 1 and increment by 1**

**create sequence pro\_seq1 start with 100 increment by 2;**

**View : View is use to create the virtual table it always point to base table.**

**Create view viewName**

**as**

**Select clause**

**If we are created view on single with all columns through view we can do DML Operation.**

**create view prod\_view1 as select \* from product;**

**create view prod\_view2 as select \* from product where price > 100000;**

**create view prod\_view3 as select \* from product where price > 50000 with check option;**

**(We can insert only those records where price must be > 50000)**

**insert into prod\_view3 values(7,'Mobile',42000); (Error price must be > 50000)**

**insert into prod\_view3 values(7,'Mobile',52000);**

**create view prod\_view4 as select \* from product with read only;**

**WE can’t do any DML Operation through this view**

**insert into prod\_view4 values(8,'Pant',2500); (Error )**

**create the view**

**as**

**select query with join with group by with sub query**

**select \* from viewName;**

**create view myview**

**as**

**select t.tid,t.tname,s.sname,s.age from trainer t, student s where t.tid = s.tsid;**

**select \* from myview**

**DCL :**

**PL SQL**

**Day 10 : 11/2/20**

**PL SQL :** Programming or Procedure Language on SQL (Structure Query Language).

SQL :

DRL, DDL, DML (Insert/Delete/Update):

Limit of SQL.

In SQL we can execute only one query at time(The query can effect more than one records).

If We wan to execute more than one query with term and conditions( programming features).

PL SQL syntax

declare

variable declaration (optional)

begin

coding or coding

SQL and PL SQL

exception (optional)

exception handling block

end; (it must be end with semicolon)

Welcome program

begin

dbms\_output.put\_line('Welcome to Pl SQL');

end;

by default plsql console is disable

to enable pl sql console

set serveroutput on;

Variables : variable is a name which hold the value.

Scalar variable (The value can hold only one value).

Syntax to declare the variable

variableName datatype;

variable initialization

variableName datatype := value;

Different type of scalar variable

variablename datatype;

varaibleName datatype:=value

variableName dateTye not null:=value; // it must hold value

variableName constant datatype:=value; //here must be assign the value as well as we can’t change the value

PL SQL Select clause with into clause

Select columName1, columnNme2,….. columnNamen into variableName1, variableName2…. variableNamen from tableName where clause with primary key column

## **Variables anchors :** When pl sql engine or tools execute the select statement it check the size and datatypes of variable. If any mismatch between data type as well as size of data types. We will get the error.

Variables anchors makes Pl SQL flexible even if the table structure change in future. It automatically load the data type as well data type size.

Syntax

variableName tableName.columnName%type;

if statements

if else

if else if

Looping :

1. Simple loop
2. While loop
3. For loop

Scalar data types : This type of data types can hold only value at time.

Composite Data types : This type of data type can hold more than one value.

1. Collection : It is a type of composite data type which help to store more than one value of same type like array in C or Java.
2. Records : It is a type of composite data type which help to store more than one value of different types like Structure in C or Class in Java

Collection :

VArray : The VArray is one type of collection data types.

Syntax

type arrayVariableName is varray(size) of datatype;

variableName arrayVariableName;

variableName number(10);

Nested table : size increase dynamically.

Syntax

Type tableVariableName is table of datatype;

variableName tableVariableName;

**Records :** It is a type of composite data type which help to store more than one value of same or different types like Structure in C language.

Syntax

type recordTypeName is record(variableName datatype, variableName datetype…..varaibleName datype);

recordVariable recordTypeName;

**RowType : Row type is a type of attribute which help to create the records of that specific table with all column as well as type of that columns (with number character with precision).**

**Syntax**

Record\_VariableName tableName%rowtype;

The table columnName itself is behave as record member.

Record\_variable.record\_member (tableColumnName)

PL SQL Assignment

1. Write a program to interchange the salaries of employee 120 and 122.
2. Increase the salary of employee 115 based on the following conditions: If experience is more than 10 years, increase salary by 20% If experience is greater than 5 years, increase salary by 10% Otherwise 5%
3. Change commission percentage as follows for employee with ID = 150. If salary is more than 10000 then commission is 0.4%, if Salary is less than 10000 but experience is more than 10 years then 0.35%, if salary is less than 3000 then commission is 0.25%. In the remaining cases commission is 0.15%.
4. Find out the name of the employee and name of the department for the employee who is managing for employee 103.
5. Change salary of employee 130 to the salary of the employee with first name ‘Joe’. If Joe is not found then take average salary of all employees.

If more than one employee with first name ‘Joe’ is found then take the least salary of the employees with first name Joe.

**Day 11 : 11/3/20**

**Exception Handling :**

**Exception is a type of error which occurs in PLSQL block when PLSQL engine execute the PL SQL code (No SQL and SQL Statement).**

**Because of exception your application will get terminate abnormally.**

**Block**

**1**

**4 line**

**6 line**

**10 code**

**In PLSQL the exception are divided into three types.**

1. **Pre-defined exception (pre-defined name and pre-defined error code as well as message).**
2. **Pre-defined exception (pre-defined error but not pre-defined name).**
3. **User-defined exception (user-defined name and user-defined error code with some range provided by oracle).**

**Syntax of exception**

**declare**

**begin**

**1**

**To**

**20**

**exception**

**when exception\_name then**

**do task**

**when exception\_name then**

**do task**

**when exception\_name then**

**do task**

**when others then**

**other task**

**end;**

**-- First Exception example**

**set serveroutput on;**

**declare**

**a number(2):=10;**

**result number(2);**

**begin**

**result := a/0;**

**dbms\_output.put\_line('Result is '||result);**

**exception**

**when others then**

**dbms\_output.put\_line('Exception generated');**

**end;**

**set serveroutput on;**

**declare**

**a number(2):=10;**

**result number(2);**

**v\_name varchar2(10);**

**begin**

**result := a/1;**

**dbms\_output.put\_line('Result is '||result);**

**select first\_name into v\_name from employees;**

**--insert into product values(100,'Abc',10000,21);**

**--dbms\_output.put\_line('Record inserted');**

**exception**

**when zero\_divide then**

**dbms\_output.put\_line('Divided by zero');**

**dbms\_output.put\_line(SQLCODE);**

**dbms\_output.put\_line(SQLERRM);**

**when too\_many\_rows then**

**dbms\_output.put\_line('to many recods found');**

**--dbms\_output.put\_line(SQLCODE);**

**--dbms\_output.put\_line(SQLERRM);**

**when others then**

**dbms\_output.put\_line('to many recods found');**

**dbms\_output.put\_line(SQLCODE);**

**dbms\_output.put\_line(SQLERRM);**

**end;**

**user-defined exception name with pre-defined error code**

**foreign\_key\_exception**

**set serveroutput on;**

**declare**

**dup\_rec exception; -- created the variable of type exception**

**foreign\_key\_exception exception;**

**pragma exception\_init(dup\_rec,-1); -- register that exception using pre-defined procedure**

**pragma exception\_init(foreign\_key\_exception,-02291);**

**begin**

**--insert into trainer values(1,'Ravi','Java');**

**--dbms\_output.put\_line('REcord inserted');**

**insert into student values(1,'Seeta',21,2);**

**dbms\_output.put\_line('Record inserted successfully');**

**exception**

**when dup\_rec then**

**dbms\_output.put\_line('Primay key exception - record already present');**

**when foreign\_key\_exception then**

**dbms\_output.put\_line('foreign key exception - record must present in primary - trainer table');**

**when others then**

**dbms\_output.put\_line('Generic Exception');**

**dbms\_output.put\_line(SQLCODE);**

**end;**

**user-defined name as well as user-defined error code**

**Error code must be between the range of -20000 to -20999**

**Cursor : A Cursor is a temporary memory created by PL SQL engine when we execute SQL (DML,DDL,DRL) query in PL - SQL Block.**

**That temporary memory hold the details about that query which executed in PL SQL block**

**Types of cursor**

1. **Implicit cursor : DML Operation**

**All implicit cursor start with sql% followed by cursor name**

1. **Sql%found : Boolean**
2. **Sql%notfound : Boolean**
3. **Sql%rowcount : number**
4. **Sql%isopen : Boolean**
5. **Explicit cursor : DRL or DQL Operation**

**Explicit cursor can use with DRL or DQL which help to retrieve or fetch one record at a time.**

**steps to create explicit cursor**

1. **create cursor**
2. **open cursor**
3. **fetch record**
4. **close cursor**

**Cursor creation syntax**

**cursor cursorName is select clause**

**syntax to open the cursor**

**open cursorName;**

**fetch the records from cursor**

**fetch cursorName into recordsName**

**Pl SQL Block**

1. **Anonymous block (No name for this block so we can’t do reusability)**
2. **Names block : we can do reusability** 
   1. **Stored Procedure or Procedure**
   2. **Function**

**Syntax**

**Create procedure procedurename**

**as**

**begin**

**end procedureName;**

**Procedure Parameters**

**types of parameters**

1. **In parameter (by default) (read the value)**
2. **Out parameter (write)**
3. **In Out parameter (read and write)**

**Function is a type of named block which return mandatory one value using return keyword.**

**Syntax**

**Create or replace function functionName**

**Return datatype**

**As**

**Begin**

**return value;**

**End functionName;**

**Trigger : Trigger is a type of special stored procedure which get fire or automatically get call when we do DML Operation on specific table.**

**Trigger is a constructor in OOPs Language.**

**Syntax**

**Create or replace trigger triggerNam**

**Before/After**

**Insert/Delete/Update**

**On tableName**

**For each row**

**Begin**

**end triggerName;**

**On single table we can create 6 trigger**

**Trigger task always execute on background of application.**

**Emp Table**

**EmpId,EmpName,EmpSalary : Insert the records**

**Emp\_Track\_Info**

**EmpId,EmpName,dateandtime**

**Package : package is a collection of procedure, functions, cursor etc.**

**Package divided into two parts**

**Package header part**

**Package body part**

**P1 and p2 inside a package**

**mypack**

**p1, p2**

**Product Table**

**Pid PName price**