**Class 01**

**Types of Data**

**==============**

**We have two types of data.**

**1) Unstructured Data**

**2) Structured Data**

**1) Unstructured Data**

**---------------------**

**A data which is not in readable format is called unstructured data.**

**In general, meaningless data is called unstructured data.**

**ex:**

**201 Lakemba SYD NSW AUS**

**2) Structured Data**

**------------------**

**A data which is in readable format is called structured data.**

**In general, meaning full data is called structured data.**

**ex:**

**Unit Locality City State Country**

**---- -------- ----- ----- --------**

**201 Lakemba SYD NSW AUS**

**Management system**

**=================**

**Management system is a software which is used to manage the database.**

**Using management system we can perform following activities very easily.**

**1) Adding the new data**

**2) Modifying the existing data**

**3) Dropping the unnecessary data**

**4) Selecting the required data**

**Q) What is the difference between DBMS and RDBMS?**

**DBMS RDBMS**

**-------- -----------**

**It stands for Database Management System. It stands for Relational Database Management System.**

**It stores the data in files. It stores the data in tables.**

**It is not designed to store large amount of It is designed to store large amount of data. Of data.**

**It provides support for single user at a time. It provides support for multiple users at a time.**

**There is no data security. There is high data security.**

**It does not support normalization. It supports normalization.**

**(It reduce code redundency)**

**Oracle**

**=======**

**It is one of the database which is used to store structured data.**

**It is a RDBMS database.**

**It is product of Oracle Corporation.**

**It is classified into two types.**

**Oracle**

**|**

**|--------------------------------------------------------|**

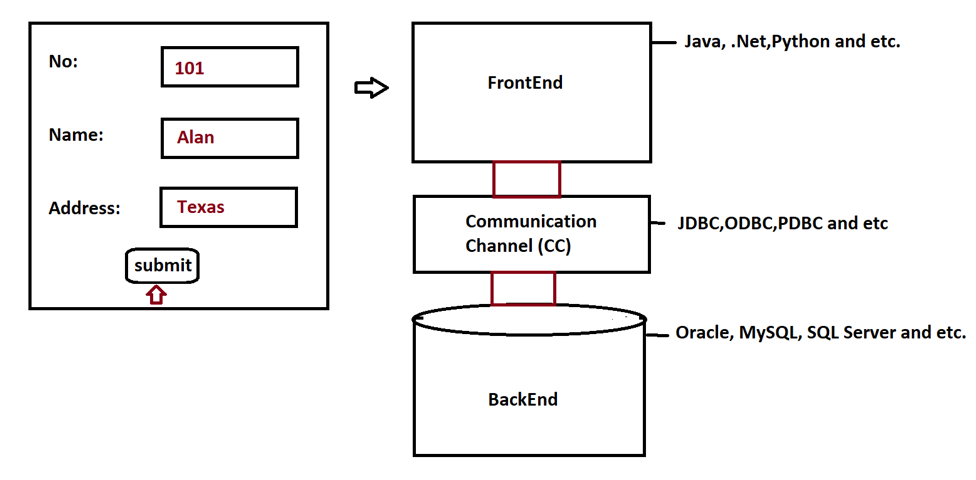
**SQL PL/SQL**

**(Structured Query Language) (Procedural / Structured Query Language)**

**Client-Server Architecture**

**==========================**

**Diagram: oracle1.1**

****

**Above architecture describes how our frontend data goes to backend.**

**FrontEnd**

**------------**

**The one which is visible to the enduser to perform some operations is called frontend.**

**ex:**

**Java,.Net,Python,Perl,D2K and etc.**

**Communication channel**

**---------------------------------**

**It acts like a bridge between frontend and backend.**

**ex:**

**JDBC- Java Database Connectivity**

**ODBC- Open Database Connectivity**

**PDBC- Python Database Connectivity**

**BackEnd**

**--------**

**The one which is not visible to the enduser but it performs operations based on the instructions given by frontend is called backend.**

**ex:**

**Oracle, MySQL, SQL Server, MongoDB , NoSQL and etc.**

**SQL**

**====**

**SQL stands for Structured Query Language which is pronounce as SEQUEL.**

**This language is used to interact with oracle database.**

**It is a command based language.**

**It is a case insensitive language.**

**Every command must starts with verb.**

**Every command must ends with semicolon.**

**It is developed in the year 1972 by Mr.Codd (by IBM).**

**Sub languages of SQL**

**====================**

**We have five sub languages of SQL.**

**1) DDL (Data Definition Language)**

**2) DML (Data Manipulation Language)**

**3) DRL/DQL (Data Retrieve/Query Language)**

**4) TCL (Transaction Control Language)**

**5) DCL (Data Control Language)**

**1) DDL (Data Definition Language)**

**-------------------------------**

**This language is used to maintain the objects in database.**

**It is a collection of five commands.**

**ex: create, alter, drop , truncate and rename.**

**This commands mainly used for tables. The changes made by these commands are permanent.**

**2) DML (Data Manipulation Language)**

**-----------------------------------**

**This language is used to manipulate the data which is present in database.**

**It is a collection of four commands.**

**ex:**

**insert, update, delete and merge**

**3) DRL/DQL (Data Retrieve/Query Language)**

**------------------------------------**

**This language is used to retrieve the data from database.**

**It is a collection of one command.**

**ex:**

**select**

**4) TCL (Transaction Control Language)**

**---------------------------------------**

**This language is used to maintain the transaction of database.**

**It is a collection of three commands.**

**ex:**

**commit, rollback and savepoint**

**5) DCL (Data Control Language)**

**------------------------------**

**This language is used to control the access of data to the user.**

**It is a collection of two commands.**

**ex:**

**grant and revoke**

**Table**

**=====**

**Table is an object which is used to represent the data.**

**Table is a collection of rows and columns.**

**Data which is present in a table is a case sensitive.**

**Rows in table is also known as records or tuples.**

**Columns in table is also known as attributes or fields**

**ex:**

**NO NAME ADD**

**---------------------------------------**

**101 | Alan | Florida**

**---------------------------------------**

**102 | Jose | Texas**

**---------------------------------------**

**103 | Lisa | Chicago**

**---------------------------------------**

**Here table contains 3 rows and 3 columns.**

**Oracle**

**=======**

**Version : 10g**

**Vendor : Oracle Corporation**

**Website : www.oracle.com/in/products**

**Software : Expression Edition**

**Port No : 1521**

**Username : system (default)**

**Password : admin**

**Download link :**

[**https://drive.google.com/file/d/0B9rC21sL6v0td1NDZXpkUy1oMm8/view?usp=drive\_link&resourcekey=0-aKooR3NmAh\_eLo\_qGw\_inA**](https://drive.google.com/file/d/0B9rC21sL6v0td1NDZXpkUy1oMm8/view?usp=drive_link&resourcekey=0-aKooR3NmAh_eLo_qGw_inA)

**Class 02:**

**Establish the connection with database**

**======================================**

**To execute any comment in database or to perform any operation on database we need to**

**establish the connection with database software.**

**Once work with database is completed we need to close the connection with database.**

**ex:**

**---**

**SQL> connect**

**username : system**

**password : admin**

**SQL> disconnect**

**ex:**

**----**

**SQL> conn**

**username : system**

**password : admin**

**SQL> disc**

**ex:**

**---**

**SQL> conn system/admin**

**SQL> disc**

**create command**

**==============**

**It is used to create a table in a database.**

**syntax:**

**------**

**create table <table\_name>(col1 datatype(size),col2 datatype(size),......,**

**colN datatype(size));**

**ex:**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12));**

**create table dept(deptno number(3),dname varchar2(10),dloc varchar2(10));**

**create table emp(eid number(3),ename varchar2(10),esal number(10,2),**

**deptno number(3),job varchar2(10),comm number(8));**

**Describe command**

**================**

**It is used to display the structure of a table.**

**syntax:**

**desc <table\_name>;**

**ex:**

**desc emp;**

**desc dept;**

**desc student;**

**Insert command**

**===============**

**Insert command is used to insert row/record in a table.**

**syntax:**

**---------**

**insert into <table\_name> values (value1,value2,....,valueN);**

**ex:**

**insert into student values(101,'raja','hyd');**

**insert into student values('ravi',102,'delhi'); //invalid**

**insert into student values(102,'ravi'); // invalid**

**insert into student values(102,'ravi',null);**

**Note:**

**-----**

**null represent undefined or unavailable.**

**approach2**

**---------**

**insert into student(sno,sname,sadd) values(103,'ramana','vizag');**

**insert into student(sno,sname) values(104,'ramulu');**

**approach3**

**---------**

**Using '&' symbol we can insert dynamic inputs.**

**ex:**

**insert into student values(&sno,'&sname','&sadd');**

**commit command**

**===============**

**It is used to make the changes permanent to database.**

**syntax:**

**commit;**

**dept table**

**===========**

**create table dept(deptno number(3),dname varchar2(10),dloc varchar2(10));**

**insert into dept values(10,'CSE','HYD');**

**insert into dept values(20,'ECE','PUNE');**

**insert into dept values(30,'MEC','VIZAG');**

**insert into dept values(40,'EEE','DELHI');**

**commit;**

**emp table**

**==========**

**create table emp(eid number(3),ename varchar2(10),esal number(10,2),**

**deptno number(3),job varchar2(10),comm number(8));**

**insert into emp values(201,'Alan',9000,10,'Clerk',null);**

**insert into emp values(202,'Jose',19000,10,'Clerk',500);**

**insert into emp values(203,'Kelvin',45000,20,'HR',300);**

**insert into emp values(204,'Nelson',23000,20,'HR',900);**

**insert into emp values(205,'Lisa',21000,30,'Manager',500);**

**insert into emp values(206,'Jesicca',37000,30,'Manager',800);**

**commit;**

**select command**

**===============**

**It is used to retrieve the records from database table.**

**syntax:-**

**select \* from <table\_name>;**

**Here '\*' means all rows and columns.**

**ex:**

**select \* from student;**

**select \* from dept;**

**select \* from emp;**

**Projection**

**----------**

**Selecting specific columns from database table is called projection.**

**ex:**

**select sno,sname,sadd from student;**

**select sno,sname from student;**

**select sname from student;**

**Arithmetic operations**

**-----------------------------**

**In select command we can perform arithmetic operations.**

**ex:**

**select sno,sname,sadd from student;**

**select sno-100,sname,sadd from student;**

**select sno+100,sname,sadd from student;**

**Column alias**

**---------------**

**A userdefined heading given to a column is called column alias.**

**Column alias is temperory.**

**We can create column alias for any column.**

**ex:**

**select sno-100,sname,sadd from student;**

**select sno-100 as SNO,**

**sname,sadd from student;**

**select sno as roll\_no,**

**sname as Name,**

**sadd as City from student;**

**Interview Queries**

**------------------**

**Q) Write a query to display all employees information from employee table?**

**select \* from emp;**

**Q) Write a query to display employee id, employee name and employee salary from emp table?**

**select eid,ename,esal from emp;**

**Q) Write a query to display list of tables present in database?**

**select \* from tab;**

**Q) Write a query to display logical database name?**

**select \* from global\_name;**

**Q) Write a query to display employee id,employee name, employee salary and annual salary from**

**employee table?**

**select eid,ename,esal,esal\*12 from emp;**

**Q) Write a query to display employee id,employee name, employee salary and annual salary as**

**ANNUAL\_SAL from employee table?**

**select eid,ename,esal,esal\*12 as ANNUAL\_SAL from emp;**

**where clause**

**============**

**It is used to select specific records from database table.**

**syntax:**

**-------**

**select \* from <table\_name> where condition;**

**ex:**

**select \* from student where sno=101;**

**select \* from student where sname='raja';**

**select \* from student where sadd='pune';**

**Interview Queries**

**------------------**

**Q) Write a query to display student information whose is living in hyderabad?**

**select \* from student where sadd='hyd';**

**Q) Write a query to display employees information those who are working in 10 department?**

**select \* from emp where deptno=10;**

**Q) Write a query to display employee information whose commission is null?**

**select \* from emp where comm is null;**

**class 3**

**update command**

**===============**

**update command is used to update the values in a row.**

**syntax:**

**------**

**update <table\_name> set <col\_name>=value where condition;**

**ex:**

**update student set sname='rani' where sno=101;**

**update student set sname='Alan',sadd='USA' where sno=103;**

**commit;**

**rollback;**

**Note:**

**-----**

**If we are not using any where clause then all rows will be udpated.**

**ex:**

**update student set sname='raja';**

**update student set sno=101;**

**update student set sadd='hyd';**

**delete command**

**================**

**A delete command is used to delete the rows from database table.**

**syntax:**

**------**

**delete from <table\_name> where condition;**

**ex:**

**delete from student where sno=101;**

**delete from student where sname='ravi';**

**delete from student where sadd='pune';**

**commit;**

**Note:**

**-----**

**If we won't use where clause then all rows will be deleted.**

**ex:**

**delete from student;**

**delete from emp;**

**delete from dept;**

**Interview Questions**

**====================**

**Q) Write a query to terminate all the employees those who are working as a Clerk?**

**delete from emp where job='Clerk';**

**Q) Write a query to display employees information whose commision is null?**

**select \* from emp where comm is null;**

**Q) Write a query to increment salary by 1000 whose employee id is 201?**

**update emp set esal=esal+1000 where eid=201;**

**Logical Operators**

**=================**

**Logical operators are used to declare multiple conditions in a query.**

**We have three logical opreators.**

**1) AND**

**2) OR**

**3) NOT**

**1) AND**

**-------**

**It returns the records only if our condition is true.**

**All conditions must be from same row only.**

**ex:**

**---**

**select \* from emp where eid=201 AND ename='Alan';**

**select \* from emp where eid=201 AND ename='Ana'; //no rows selected**

**select \* from emp where eid=201 AND ename='Jose';//no rows selected**

**2) OR**

**---------**

**It returns the records only if one condition is true.**

**Here conditions can be from any row.**

**ex:**

**--**

**select \* from emp where eid=201 OR ename='Alan';**

**select \* from emp where eid=201 OR ename='Ana';**

**select \* from emp where eid=201 OR ename='Jose';**

**3) NOT**

**--------**

**It will return the records except the condition.**

**A <> symbol denoted as not operator.**

**ex:**

**select \* from emp where NOT eid=202;**

**select \* from emp where eid<>202;**

**select \* from emp where job<>'Manager';**

**Interview Queries**

**----------------**

**Q) Write a query to display employees information whose salary is greater then**

**20000 and less then 50000?**

**select \* from emp where esal>20000 AND esal<50000;**

**Q) Write a query to display employees information those who are not working in 10**

**department?**

**select \* from emp where deptno<>10;**

**Q) Write a query to display employee information whose employee id is 201,202 and 203?**

**select \* from emp where eid=201 OR eid=202 OR eid=203;**

**Between operator**

**===============**

**Between operator returns the records those who are in the range of values.**

**In between operator we will take lower limit then higher limit.**

**ex:**

**select \* from student where sno between 101 and 105;**

**select \* from emp where esal between 5000 AND 20000;**

**select \* from emp where deptno between 10 AND 30;**

**IN operator**

**==============**

**IN operator is a replacement of OR operator.**

**In operator returns the records those who are matching in the list of values.**

**ex:**

**select \* from student where sno IN(101,102,103);**

**select \* from student where sname IN('raja','ravi','Alan');**

**Interview Queries**

**------------------**

**Q) Write a query to delete employees information whose employee id 201,202 and 203?**

**delete from emp where eid IN (201,202,203);**

**Q) Write a query to display employees information whose deptno number between 10 to 30?**

**select \* from emp where deptno between 10 and 30;**

**Pattern Matching operators**

**===========================**

**Pattern matching operators are used to select the letters from table.**

**Pattern matching operators take the support of like keyword.**

**We have two type of pattern matching operators.**

**1) Percentage (%)**

**2) Underscore (\_)**

**1) Percentage (%)**

**-----------------**

**Q) Write a query to display employees information whose emplyoee name starts with 'A' letter?**

**select \* from emp where ename like 'A%';**

**Q) Write a query to display employees information whose employee name ends with 'n' letter?**

**select \* from emp where ename like '%n';**

**Q) Write a query to display employees information whose employee name having middle letter**

**as 'l' letter?**

**select \* from emp where ename like '%l%';**

**2) Underscore (\_)**

**-----------------**

**Q) Write a query to display employee information whose employee name having second letter as 'l'?**

**ex:**

**select \* from emp where ename like '\_l%';**

**Q) Write a query to display employee information whose employee name having second last letter as 's'?**

**select \* from emp where ename like '%s\_';**

**Q) WRite a query to display employee information whose employee name having third letter as 'l' ?**

**select \* from emp where ename like '\_\_l%';**

**DDL commands**

**=============**

**1) create (tables)**

**2) alter (columns)**

**3) drop (tables)**

**4) truncate (records)**

**5) rename (tables)**

**drop command**

**==========**

**It is used to drop /delete the table from database.**

**syntax:**

**drop table <table\_name>;**

**ex:**

**drop table student;**

**drop table dept;**

**drop table emp;**

**rename command**

**================**

**It is used to rename the table name.**

**syntax:**

**rename <old\_name> to <new\_name>;**

**ex:**

**rename student to students;**

**rename emp to employees;**

**rename dept to departments;**

**truncate command**

**===============**

**It is used to delete the records permanently.**

**syntax:**

**truncate table <table\_name>;**

**ex:**

**truncate table student;**

**truncate table emp;**

**truncate table dept;**

**Q) What is the difference between delete and truncate command?**

**delete truncate**

**-------- ----------**

**It deletes the rocords temperory. It delete the records permanently.**

**We can rollback the data. We can't rollback the data.**

**Where clause can be used. Where clause can't be used.**

**Class 04:**

**Alter command**

**=============**

**Using alter command we can perform following activities very easily.**

**i) Adding new columns**

**ii) Modifying the columns**

**iii) Renaming the columns**

**iv) Dropping the columns**

**i) Adding new columns**

**---------------------**

**Using alter command we can add new columns in a existing table.**

**syntax:**

**-----**

**alter table <table\_name> add (col\_name datatype(size));**

**ex:**

**alter table student ADD (state varchar2(10));**

**alter table student ADD (pincode number(8));**

**alter table student ADD (state varchar2(10),pincode number(8));**

**update student set state='Telangana' where sno=101;**

**ii) Modifying the columns**

**------------------------**

**Using alter command we can modify the columns.**

**We can increase or decrease the size of a column only when existing values are fit into new size.**

**syntax:**

**----**

**alter table <table\_name> MODIFY (col datatype(size));**

**ex:**

**desc student;**

**alter table student MODIFY (state varchar2(15));**

**desc student;**

**We can change the datatype of a column only when that column is empty.**

**ex:**

**desc student;**

**alter table student MODIFY (pincode varchar2(8));**

**desc student;**

**iii) Renaming the columns**

**----------------------**

**Using alter command we can rename the column name.**

**syntax:**

**-------**

**alter table <table\_name> rename column <old\_name> to <new\_name>;**

**ex:**

**alter table student rename column state to district;**

**alter table emp rename column esal to dailywages;**

**alter table emp rename column job to designation;**

**iv) Dropping the columns**

**----------------------------**

**Using alter command we can drop the columns.**

**syntax:**

**alter table <table\_name> drop (col);**

**ex:**

**alter table student drop (state,pincode);**

**alter table student drop (district,pincode);**

**Copy of a table or duplicate table**

**==================================**

**Using create and select command we can create copy of a table or duplicate table.**

**ex:**

**create table employees as select \* from emp;**

**create table employees as select \* from emp where deptno=10;**

**create table employee as select eid,ename,esal from emp;**

**create table employee as select \* from emp where eid IN (201,202,203);**

**create table employee as select \* from emp where comm is null;**

**create table employee as select \* from emp where ename like 'A%';**

**create table employee as select \* from emp where eid=201 and ename='Alan';**

**create table employee as select \* fro emp where eid<>202;**

**cl scr**

**======**

**It is used to clear the output screen of SQL command prompt.**

**syntax:**

**------ cl scr**

**Functions**

**==========**

**Functions are used to manipulate the data and give the result.**

**We have two types of functions.**

**1) Group Functions / Multiple row Functions**

**2) Scalar Functions / Single row Functions**

**1) Group Functions**

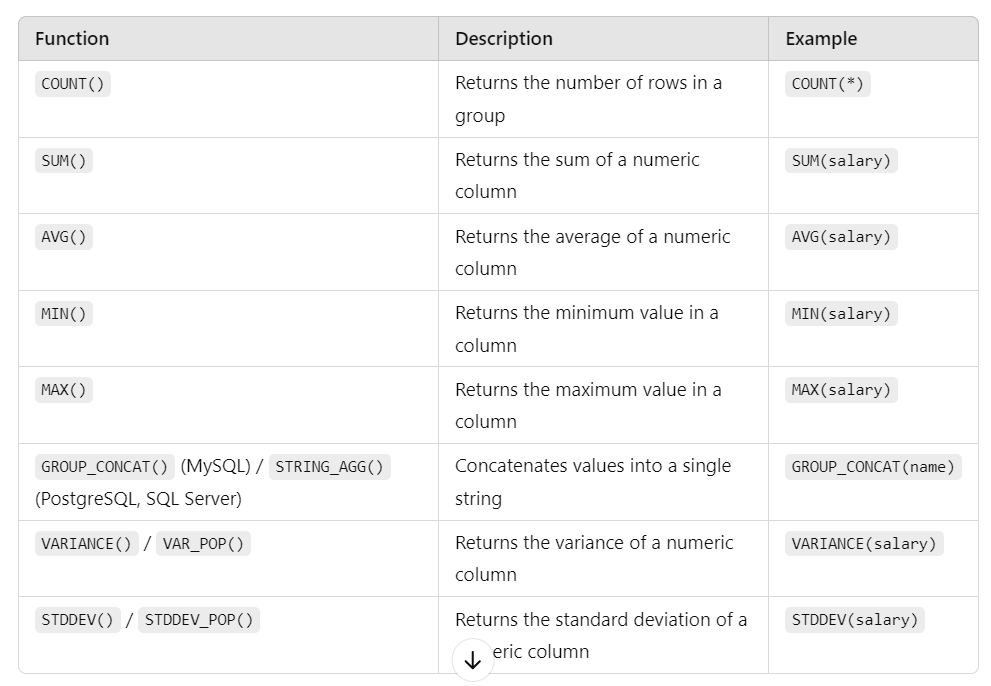
**-------------------**

**Group functions are applicable for multiple rows.**

**We have following list of group functions.**

**ex:**

**sum(), avg(), max(), min(), count(\*) and count(expression).**

****

**Q) Write a query to display sum of salary of each employee?**

**select sum(esal) from emp;**

**Q) Write a query to display average salary of each employee?**

**select avg(esal) from emp;**

**Q) Write a query to display highest salary from emp table?**

**select max(esal) from emp;**

**Q) Write a query to display least salary from emp table?**

**select min(esal) from emp;**

**Q) What is the difference between count(\*) and count(expression)?**

**count(\*)**

**--------**

**It will return number of records present in a database table.**

**It will return null records.**

**ex:**

**--**

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

**count(expression)**

**-------------------**

**It will return number of values present in a column.**

**It will not include null values.**

**ex:**

**--**

**select count(eid) from emp; // 6**

**select count(comm) from emp; // 5**

**userlist table**

**==============**

**drop table userlist;**

**create table userlist(uname varchar2(10),pwd varchar2(10));**

**insert into userlist values('raja','rani');**

**insert into userlist values('king','kingdom');**

**commit;**

**Q) Write a query to check given username and password valid or not?**

**select count(\*) from userlist where uname='raja' AND pwd='rani'; //1**

**select count(\*) from userlist where uname='raja' AND pwd='rani2'; //0**

**Dual table**

**===========**

**Dual table is a dummy table which is used to perform arithmetic operations and to see the**

**current system date.**

**Dual table contains one row and one clumn.**

**ex:**

**select 10+20 from dual;**

**select 10\*5 from dual;**

**select sysdate from dual;**

**select current\_date from dual;**

**2)Scalar Functions**

**==================**

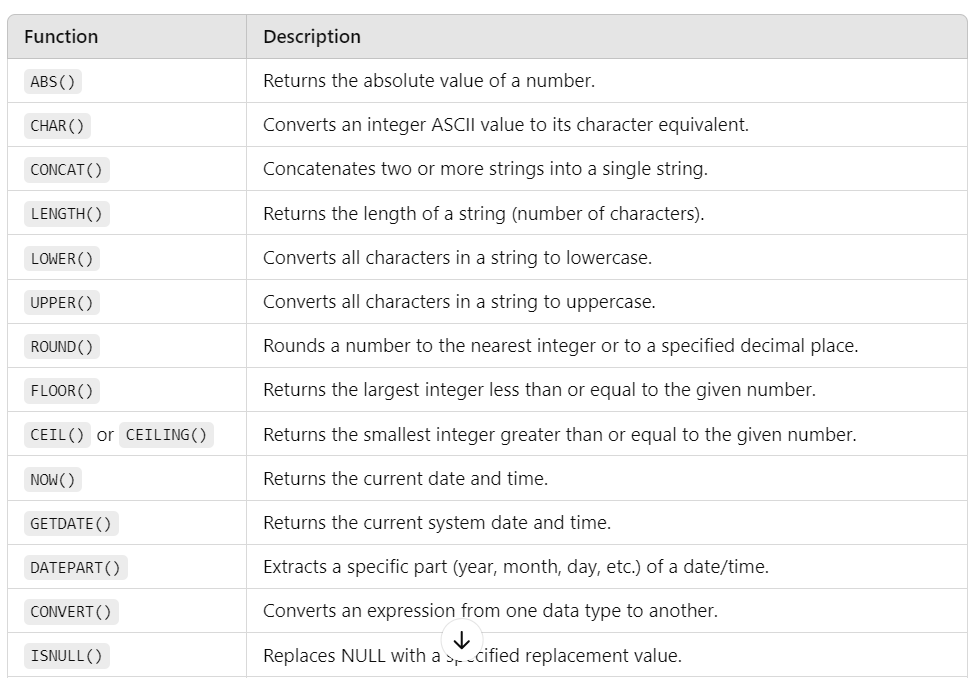
**We have following list of scalar functions.**

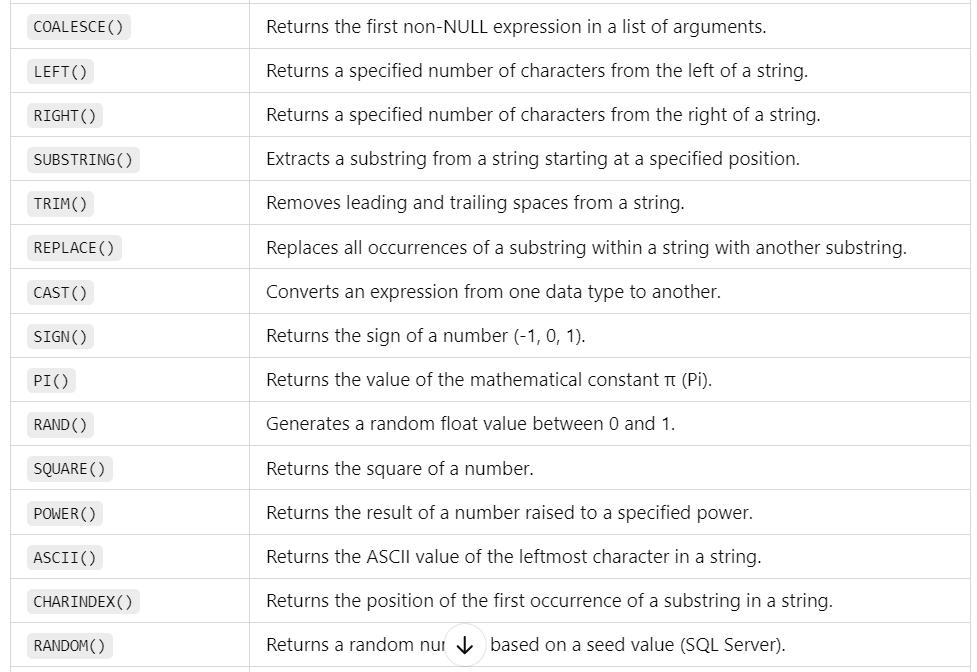
**i) Character functions**

**ii) Number functions**

**iii) Date functions**

**iv) Conversion functions**

****

****

**i) Character functions**

**----------------------**

**We have following list of character functions.**

**upper()**

**------**

**It is used to convert the string to uppercase.**

**ex:**

**select upper('oracle training') from dual;**

**lower()**

**---------**

**It is used to convert the string to lowercase.**

**ex:**

**select lower('ORACLE TRAINING') from dual;**

**initcap()**

**----------**

**It is used to display the string with initial capital letter.**

**ex:**

**select initcap('oracle training') from dual;**

**lpad()**

**-------**

**It is used to pad the characters to left side.**

**ex:**

**select lpad('oracle',10,'z') from dual; //zzzzoracle**

**rpad()**

**-------**

**It is used to pad the characters to right side.**

**ex:**

**select rpad('oracle',10,'z') from dual; //oraclezzzz**

**ltrim()**

**--------**

**It is used to trim the characters from left side.**

**ex:**

**select ltrim('zzoraclezz','z') from dual;// oraclezz**

**rtrim()**

**-------**

**It is used to trim the characters from right side.**

**ex:**

**select rtrim('zzoraclezz','z') from dual;// zzoracle**

**trim()**

**----**

**It is used to trim the characters from both the sides.**

**ex:**

**select trim('z' from 'zzoraclezz') from dual;**

**concat()**

**-------**

**It is used to concatinate two strings.**

**ex:**

**select concat('mega','star') from dual;**

**select concat(concat('mega','star'),'chiru') from dual;**

**replace()**

**-------**

**It is used to replace the characters.**

**ex:**

**select replace('gOOgle','O','oo') from dual;**

**ii) Number functions**

**---------------------**

**We have following list of number functions.**

**abs()**

**-----**

**It returns absolute value.**

**ex:**

**select abs(-45) from dual; //45**

**sqrt()**

**-----**

**It returns square root value.**

**ex:**

**select sqrt(25) from dual; // 5**

**power(A,B)**

**----------**

**It returns power value.**

**ex:**

**select power(5,3) from dual; //125**

**greatest()**

**----------**

**It return greatest value.**

**ex:**

**select greatest(6,9,2,4) from dual; // 9**

**least()**

**------**

**It returns least value.**

**ex:**

**select least(6,9,2,4) from dual; //2**

**ceil()**

**-----**

**It returns ceil value.**

**ex:**

**select ceil(10.9) from dual; // 11**

**select ceil(10.2) from dual; // 11**

**floor()**

**------**

**It returns floor value.**

**ex:**

**select floor(9.8) from dual; // 9**

**select floor(9.1) from dual; // 9**

**round()**

**-----**

**It returns nearest value.**

**ex:**

**select round(10.5) from dual; //11**

**select round(10.4) from dual; // 10**

**trunc()**

**-----------**

**It removes decimal numbers.**

**ex:**

**select trunc(10.56) from dual; // 10**

**Working with Date values**

**=======================**

**Every database software support date values.**

**Every database software support date values in different date patterns.**

**ex:**

**oracle - dd-MMM-yy**

**mysql - yyyy-MM-dd**

**emp1 table**

**----------**

**drop table emp1;**

**create table emp1(eid number(3),ename varchar2(10),edoj date);**

**insert into emp1 values(101,'Alan','01-JAN-24');**

**insert into emp1 values(102,'Jose',sysdate);**

**insert into emp1 values(103,'Lisa',current\_date);**

**commit;**

**Assignment**

**===========**

**Q) Write a java program to find out lucky number?**

**Class 05:**

**iii) Date functions**

**--------------------**

**We have following list of date functions.**

**ADD\_MONTHS()**

**-------------**

**It is used to add the months in a given date.**

**ex:**

**select ADD\_MONTHS('07-OCT-24',4) from dual;**

**MONTHS\_BETWEEN()**

**--------------**

**It returns number of months between two dates.**

**ex:**

**select MONTHS\_BETWEEN('01-JAN-24','07-OCT-24') from dual;**

**select MONTHS\_BETWEEN('01-JAN-24','01-NOV-24') from dual;**

**select ABS(MONTHS\_BETWEEN('01-JAN-24','01-NOV-24')) from dual;**

**NEXT\_DAY()**

**--------**

**It will return next date of a given day in a week.**

**ex:**

**select NEXT\_DAY(sysdate,'sunday') from dual;**

**select NEXT\_DAY(sysdate,'monday') from dual;**

**LAST\_DAY()**

**------------**

**It will return last date of a month.**

**ex:**

**select LAST\_DAY('13-FEB-24') from dual;**

**select LAST\_DAY(sysdate) from dual;**

**iv) convertion functions**

**--------------------------**

**convertion function is used to convert from one datatype to another datatype.**

**ex:**

**TO\_CHAR() function**

**We have two pseduo's for TO\_CHAR**

**1) number TO\_CHAR**

**----------------**

**It takes '9' in digits and dollar or euro's symbol.**

**ex:**

**select eid,ename,esal from emp;**

**select eid,ename,TO\_CHAR(esal,'9,999') from emp;**

**select eid,ename,TO\_CHAR(esal,'99,999') from emp;**

**select eid,ename,TO\_CHAR(esal,'$99,999') from emp;**

**select eid,ename,TO\_CHAR(esal,'$99,999') as ESAL from emp;**

**2) date TO\_CHAR**

**---------------**

**select TO\_CHAR(sysdate,'dd-MM-yyyy') from dual;**

**select TO\_CHAR(sysdate,'yyyy-MM-dd') from dual;**

**select TO\_CHAR(sysdate,'HH:MI:SS') from dual;**

**select TO\_CHAR(sysdate,'dd-MM-yyyy HH:MI:SS') from dual;**

**select TO\_CHAR(sysdate,'year') from dual;**

**select TO\_CHAR(sysdate,'month') from dual;**

**select TO\_CHAR(sysdate,'day') from dual;**

**Integrity Constraints**

**=====================**

**Constraints are the rules which are applied on the tables to achieve accuracy and quality of data.**

**We have five types of contraints.**

**1) NOT NULL**

**2) UNIQUE**

**3) PRIMARY KEY**

**4) FOREIGN KEY**

**5) CHECK**

**Constraints can be created at two levels.**

**i) column level**

**ii) table level**

**1) NOT NULL**

**------------**

**NOT NULL constraint does not accept null values.**

**NOT NULL constraint can accept duplicate values.**

**NOT NULL constraint can be created only at column level.**

**column level**

**-------------**

**drop table student;**

**create table student(sno number(3) NOT NULL,sname varchar2(10),sadd varchar2(12));**

**insert into student values(101,'raja','hyd');**

**insert into student values(null,'ravi','delhi'); //invalid**

**insert into student values(101,'ravi','delhi');**

**Note:**

**-----**

**NOT NULL constraint can be created for multiple columns.**

**ex:**

**---**

**drop table student;**

**create table student(sno number(3) NOT NULL,**

**sname varchar2(10) NOT NULL,**

**sadd varchar2(12) NOT NULL);**

**insert into student values(null,'raja','hyd'); //invalid**

**insert into student values(102,null,'delhi'); //invalid**

**insert into student values(103,'ravi',null); //invalid**

**insert into student values(104,'ramana','pune'); // valid**

**2) UNIQUE**

**----------**

**UNQIUE constraint does not accept duplicates.**

**UNIQUE constraint can accept null values.**

**UNIQUE constraint can be created at column level and table level.**

**column level**

**------------**

**drop table student;**

**create table student(sno number(3) UNIQUE,sname varchar2(10),sadd varchar2(12));**

**insert into student values(101,'raja','hyd');**

**insert into student values(null,'ravi','delhi');**

**insert into student values(101,'ravi','delhi'); //invalid**

**table level**

**------------**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12), UNIQUE(sno));**

**insert into student values(101,'raja','hyd');**

**insert into student values(null,'ravi','delhi');**

**insert into student values(101,'ravi','delhi'); //invalid**

**Note:**

**-----**

**UNIQUE constraint can be applied to multiple columns.**

**ex:**

**---**

**drop table student;**

**create table student(sno number(3) UNIQUE,sname varchar2(10) UNIQUE,sadd varchar2(12) UNIQUE);**

**insert into student values(101,'raja','hyd');**

**insert into student values(101,'ravi','delhi'); //invalid**

**insert into student values(102,'raja','delhi'); //invalid**

**insert into student values(103,'ramana','hyd'); //invalid**

**3) PRIMARY KEY**

**-------------------**

**Primary key is a combination of NOT NULL and UNIQUE constraint.**

**Primary key does not accept null values and duplicate values.**

**A table can have only one primary key.**

**Primary key constraint can be created at column level and table level.**

**column level**

**-----------**

**drop table student;**

**create table student(sno number(3) primary key,sname varchar2(10),sadd varchar2(12));**

**insert into student values(101,'raja','hyd');**

**insert into student values(null,'ravi','delhi'); //invalid**

**insert into student values(101,'ravi','delhi'); //invalid**

**table level**

**------------**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12), primary key(sno));**

**insert into student values(101,'raja','hyd');**

**insert into student values(null,'ravi','delhi'); //invalid**

**insert into student values(101,'ravi','delhi'); //invalid**

**4) FOREIGN KEY**

**----------------**

**Foreign key is used to establish the relationship between two tables.**

**This relationship is called parent and child relationship or master and details relationship.**

**To establish the relatinship between two tables a parent table must primary key or unique constraint and**

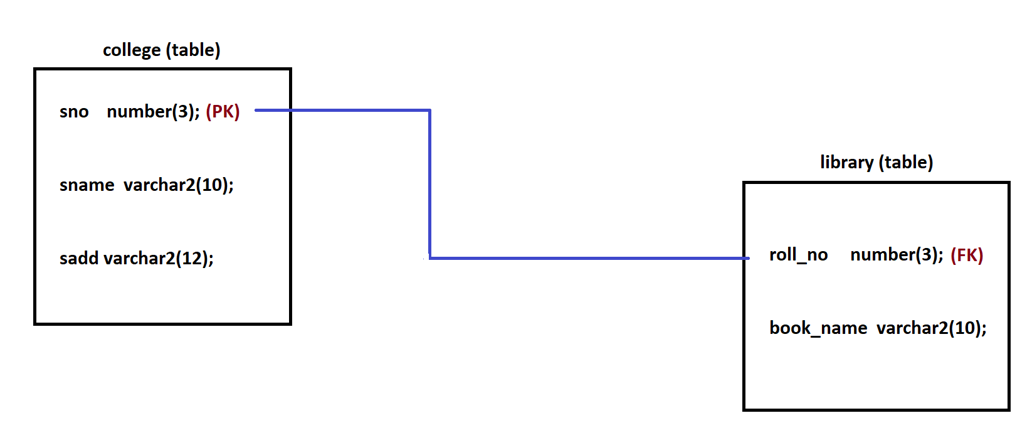
**child table must have foreign key.**

**Foreign key will accept only those values which are present in primary key.**

**Foreign key name may or may not match with primary key but datatype must match.**

**Foreign key will accept duplicates and null values.**

**Diagram: oracle5.1**

****

**college table**

**--------------**

**drop table college;**

**create table college(sno number(3) PRIMARY KEY,sname varchar2(10),sadd varchar2(12));**

**insert into college values(101,'raja','hyd');**

**insert into college values(102,'ravi','delhi');**

**insert into college values(103,'ramana','vizag');**

**commit;**

**library table**

**-------------**

**drop table library;**

**create table library(roll\_no number(3) REFERENCES college(sno), book\_name varchar2(10));**

**insert into library values(101,'java');**

**insert into library values(102,'oracle');**

**insert into library values(103,'html');**

**insert into library values(103,'CSS');**

**insert into library values(null,'Spring');**

**insert into library values(104,'hibernate'); //invalid**

**In order to drop the table, first we need to drop library table then college table.**

**ex:**

**drop table library;**

**drop table college;**

**5) CHECK**

**---------**

**CHECK constraint describes domain of column.**

**Here domain means what type of value a column must accept.**

**Check contraint can be created at column level and table level.**

**column level**

**------------**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),smarks number(3) check(smarks<=100));**

**insert into student values(101,'raja',78);**

**insert into student values(102,'ravi',100);**

**insert into student values(103,'ramana',289);**

**commit;**

**column level**

**------------**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),smarks number(3) check(smarks between 0 and 100));**

**insert into student values(101,'raja',78);**

**insert into student values(102,'ravi',100);**

**insert into student values(103,'ramana',289); //invalid**

**commit;**

**column level**

**------------**

**drop table student;**

**create table student(sno number(3),sname varchar2(10) check (sname=lower(sname)),**

**smarks number(3));**

**insert into student values(101,'raja',78);**

**insert into student values(102,'RAVI',100); //invalid**

**insert into student values(103,'RaMaNa',289); //invalid**

**commit;**

**column level**

**------------**

**drop table student;**

**create table student(sno number(3),sname varchar2(10) check (sname=upper(sname)),**

**smarks number(3));**

**insert into student values(101,'raja',78); //invalid**

**insert into student values(102,'RAVI',100);**

**insert into student values(103,'RaMaNa',289); //invalid**

**commit;**

**column level**

**------------**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),smarks number(3),**

**check (sname=upper(sname)));**

**insert into student values(101,'raja',78); //invalid**

**insert into student values(102,'RAVI',100);**

**insert into student values(103,'RaMaNa',289); //invalid**

**commit;**

**Class 06:**

**Group by clause**

**===============**

**Group by clause is used to divide the rows into groups so that we can apply group functions.**

**A column which is used in select clause. Same column we need to use in group by clause.**

**Q) Write a query to perform sum of salary of each department?**

**select sum(esal),deptno from emp group by deptno;**

**Q) Write a query to display average salary of each job?**

**select avg(esal),job from emp group by job;**

**Q) Write a query to display maximum salary of each department?**

**select max(esal),deptno from emp group by deptno;**

**Q) Write a query to display minimum salary of each job?**

**select min(esal),job from emp group by job;**

**Q) Write a query to display number of employees working in each department?**

**select count(\*),deptno from emp group by deptno;**

**Having clause**

**=============**

**Having clause is used to filter the rows from group by clause.**

**First we need to declare group by clause then having clause.**

**Q) Write a query to display sum of salary of each department whose sum of salary is greater**

**then 35000?**

**select sum(esal),deptno from emp group by deptno having sum(esal)>35000;**

**Q) Write a query to display maximum salary of each job whose maximum salary greater then 25000?**

**select max(esal),job from emp group by job having max(esal)>25000;**

**Order by clause**

**===============**

**Order by clause is used to arrange the rows in a table.**

**By default it will arrange in ascending order.**

**ex:**

**select \* from emp order by eid;**

**select \* from emp order by ename;**

**select \* from emp order by esal;**

**select \* from emp order by esal desc;**

**We can write a query using where clause, group by , having clause and order by clause.**

**select max(esal),deptno from emp group by deptno having max(esal)>25000;**

**select max(esal),deptno from emp group by deptno having max(esal)>25000 order by deptno;**

**select max(esal),deptno from emp group by deptno having max(esal)>25000 order by deptno desc;**

**select max(esal),deptno from emp where deptno<>30 group by deptno**

**having max(esal)>25000 order by deptno ;**

**TCL commands**

**===========**

**1) commit**

**2) rollback**

**3) savepoint**

**1) commit**

**----------**

**It is used to make the changes permanent to database.**

**ex:**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12));**

**insert into student values(101,'raja','hyd');**

**insert into student values(102,'ravi','delhi');**

**commit;**

**select \* from student; //2 records**

**2) rollback**

**-------------**

**It is used to undo the changes which are not permanent.**

**ex:**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12));**

**insert into student values(101,'raja','hyd');**

**insert into student values(102,'ravi','delhi');**

**commit;**

**insert into student values(103,'ramana','delhi');**

**insert into student values(104,'ramulu','vizag');**

**select \* from student; //4 records**

**rollback;**

**select \* from student; //2 records**

**3) savepoint**

**-----------**

**It is used to make logical marking in database.**

**Instead of complete rollback we can rollback upto savepoint.**

**syntax:**

**savepoint <savepoint\_name>;**

**ex:**

**---**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12));**

**insert into student values(101,'raja','hyd');**

**insert into student values(102,'ravi','delhi');**

**savepoint sp1;**

**insert into student values(103,'ramana','delhi');**

**insert into student values(104,'ramulu','vizag');**

**savepoint sp2;**

**insert into student values(105,'Alan','Texas');**

**insert into student values(106,'Jose','Vegas');**

**select \* from student; // 6 records**

**rollback to sp2;**

**DCL commands**

**============**

**1) grant**

**2) revoke**

**Privileges**

**----------**

**Permission given to a user is called privileges.**

**Rights given to a user is called privileges.**

**We have two types of privileges.**

**1) System privilege : Permission given by DBA to user.**

**2) Object privilege : Permission given by one user to another user.**

**Schema**

**------**

**Schema is a memory location which is used to run SQL commands.**

**grant command**

**-----------**

**It is used to grant the permission to the user.**

**syntax:**

**-----**

**grant <privilege1>,<privilege2> to <user\_name>;**

**revoke command**

**------------**

**It is used to revoke the permission from the user.**

**syntax:**

**-------**

**revoke <privilege1>,<privilege2> from <user\_name>;**

**DBA> create user haritha identified by haritha;**

**DBA> create user rudra identified by rudra;**

**Rudra> conn rudra/rudra //logon denied**

**Haritha> conn haritha/haritha //logon denied**

**DBA> grant connect,resource to haritha,rudra;**

**Rudra> conn rudra/rudra //connected**

**Haritha> conn haritha/haritha //connected**

**Rudra>**

**create table employee(eid number(3),ename varchar2(10),esal number(10));**

**insert into employee values(201,'Kelvin',10000);**

**insert into employee values(202,'Jose',20000);**

**insert into employee values(203,'Mark',30000);**

**commit;**

**select \* from employee;**

**Haritha> select \* from rudra.employee; // table or view does not exist**

**Rudra> grant select on employee to haritha;**

**Haritha> select \* from rudra.employee;**

**Haritha> delete from rudra.employee; //insufficient privileges**

**rudra> grant delete on employee to haritha;**

**Haritha> delete from rudra.employee; //records delete**

**Haritha> commit;**

**Haritha> disc**

**Rudra> revoke select,delete on employee from haritha;**

**Rudra> disc**

**DBA> revoke connect,resource from haritha,rudra;**

**Class 07:**

**Index**

**======**

**Index is an object which is used to improve the performance of select statement.**

**Index in a database is similar to index in a book.**

**Index can be created only to those columns which are widely used in where clause.**

**When we create index, two columns will be generated i.e One is ROWID and second is indexed column**

**All records will store in ascending in a indexed column.**

**ex:**

**INDEX TABLE**

**--------------------------------------**

**ROWID INDEXED COLUMN**

**--------------------------------------**

**- | 9000**

**- | 19000**

**- | 23000**

**- | 27000**

**- | 31000**

**- | 37000**

**- | 42000**

**---------------------------------------**

**We have two types of indexes.**

**1) Simple index**

**2) Complex index**

**1) Simple index**

**----------------**

**If index is created only for one column is called simple index.**

**ex:**

**create index idx1 on emp(esal);**

**Here index is used if we use esal in where condition.**

**ex:**

**select \* from emp where esal=27000;**

**2) Complex index**

**-----------------**

**If index is created for multiple columns is called complex index.**

**ex:**

**create index idx2 on emp(eid,deptno);**

**Here index is used if we use eid and deptno in where condition.**

**ex:**

**select \* from emp where eid=201 and deptno=10;**

**Q) Write a query to display list of indexes present in database?**

**select index\_name from user\_indexes;**

**Q) Write a query to drop the index?**

**drop index idx1;**

**drop index idx2;**

**Q) What is the difference between ROWNUM and ROWID?**

**ROWNUM**

**------**

**ROWNUM value starts with '1' and it is increment by 1.**

**ROWNUM values are temperory.**

**ex:**

**select eid,ename,esal from emp;**

**select rownum,eid,ename,esal from emp;**

**select rownum,deptno,dname,dloc from dept;**

**ROWID**

**------**

**ROWID is a memory address where our records will store in a database.**

**ROWID is permanent.**

**ex:**

**select rowid,rownum,eid,ename,esal from emp;**

**select rowid,rownum,deptno,dname,dloc from dept;**

**Q) Write a query to display first three records from employee table?**

**select \* from emp where rownum<=3;**

**Q) Write a query to display 4th record from employee table?**

**select \* from emp where rownum<=4**

**minus**

**select \* from emp where rownum<=3;**

**Q) Write a query to display 5th record from employee table?**

**select \* from emp where rownum<=5**

**minus**

**select \* from emp where rownum<=4;**

**Synonyms**

**========**

**Alternate name given to a table is called synonym.**

**It provides an alternative name for another database object, referred to as the base object, that can exist on a local or remote server.**

**syntax:**

**------**

**create synonym <synonym\_name> for <table\_name>;**

**ex:**

**create synonym sy1 for emp;**

**We can use synonym instead of table name for all commands.**

**ex:**

**select \* from sy1;**

**delete from sy1;**

**select \* from emp; // no rows selected**

**rollback;**

**Q) Write a query to display list of synonyms present in database?**

**select synonym\_name from user\_synonyms;**

**Q) Write a query to drop the synonym?**

**drop synonym sy1;**

**Views**

**=====**

**View is a virtual representation of a data from one or more then one table.**

**A table which is used to create a view is called base table or above table.**

**View does not consumes memory because it does not contain data.**

**View gets the data from base table when we run select command.**

**We have following list of views.**

**1) Simple view**

**2) Complex view**

**3) With read only view**

**4) With check option view**

**5) Materialized view**

**1) Simple view**

**--------------**

**If a view is created by using one base table is called simple view.**

**ex:**

**create view v1 as select \* from emp;**

**select \* from v1; // 7 records**

**DML operations are allowed in simple view.**

**ex:**

**delete from v1;**

**select \* from v1; //no rows selected**

**select \* from emp; // no rows selected**

**2) Complex view**

**---------------**

**If a view is created by using more then one table is called complex view.**

**ex:**

**create view v2 as select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d;**

**DML operations are not allowed for complex view.**

**ex:**

**delete from v2; // cannot delete from view**

**3) With read only view**

**--------------------**

**If a view is created by using one base table and DML operations are not required then we need to use**

**with read only view.**

**ex:**

**create view v3 as select \* from emp with read only;**

**select \* from v3; // 7 records**

**delete from v3; //cannot delete from view**

**4) With check option view**

**---------------------**

**If a view is created by using one base table and DML operations are allowed only when condition is true then**

**we need to use with check option view.**

**ex:**

**create view v4 as select \* from emp where deptno=30 with check option;**

**select \* from v4;**

**insert into v4 values(207,'Lara',40000,40,'Salesman',200); // view WITH CHECK OPTION**

**insert into v4 values(207,'Lara',40000,30,'Salesman',200);**

**select \* from v4;**

**select \* from emp;**

**5) Materialized view**

**--------------------**

**Materialized view is also known as snapshot.**

**We can create materialized view when table have primary key or unique key.**

**ex:**

**alter table emp ADD primary key(eid);**

**create materialized view v5 as select \* from emp;**

**select \* from v5; // 7 records**

**select \* from emp; // 7 records**

**delete from emp where eid=207;**

**commit;**

**select \* from emp; // 6 records**

**select \* from v5; // 7 records**

**We need to refresh the materialized view.**

**ex:**

**exec DBMS\_SNAPSHOT.REFRESH('V5');**

**select \* from v5; // 6 records**

**Q) Write a query to see the list of views present in a database?**

**select view\_name from user\_views;**

**Q) Write a query to drop the view?**

**drop view v1;**

**drop view v2;**

**drop view v3;**

**drop view v4;**

**drop materialized view v5;**

**Class 08:**

**Sequence**

**========**

**Sequence is an object which is used to generate the numbers.**

**syntax:**

**------**

**create sequence <sequence\_name> start with value increment by value;**

**ex:**

**create sequence sq1 start with 1 increment by 1;**

**create sequence sq2 start with 101 increment by 1;**

**create sequence sq3 start with 10 increment by 10;**

**A sequence having two pseudo's.**

**1) NEXTVAL**

**--------**

**It is used to generate next number in a sequence.**

**ex:**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12));**

**create sequence sq1 start with 101 increment by 1;**

**insert into student values(sq1.NEXTVAL,'raja','hyd');**

**insert into student values(sq1.NEXTVAL,'ravi','delhi');**

**insert into student values(sq1.NEXTVAL,'ramana','vizag');**

**commit;**

**select \* from student;**

**2) CURRVAL**

**----------**

**It will return the last number generated by the sequence.**

**ex:**

**select sq1.CURRVAL from dual;**

**Q) Write a query to display list of sequences present in database?**

**select sequence\_name from user\_sequences;**

**Q) Write a query to drop the sequence?**

**drop sequence sq1;**

**Joins**

**======**

**Joins is used to retrieve the data from one or more then one table.**

**// join is used to the combine rows from two or more then two table**

**ex:**

**select \* from emp,dept; // 6\*4 = 24 records**

**select eid,ename,esal,deptno,dname,dloc from emp,dept; --column ambiguously defined**

**To overcome above limitation we need to use table.column name.**

**ex:**

**select emp.eid,emp.ename,emp.esal,dept.deptno,dept.dname,dept.dloc from emp , dept; --6\*4=24 records**

**Table alias**

**----------**

**A userdefined name given to a table is called table alias.**

**Using table alias length of the query will reduce mean while performance is maintained.**

**Table alias is temperory.Once the query is executed we will loss the table alias.**

**ex:**

**select e.eid,e.ename,e.esal,d.deptno,d.dname,d.dloc from emp e, dept d; -- 6\*4 = 24 records**

**We have following list of joins.**

**1) Equi Join**

**2) Non-Equi Join**

**3) Self Join**

**4) Cartisian Product**

**5) Inner Join**

**6) Outer Join**

**1) Equi Join**

**--------------**

**When two tables are joined based on common column is called equi join.**

**It uses the equality operator (=) to match records.**

**ex:**

**---**

**select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d**

**where (e.deptno=d.deptno); -- 6 records display**

**2) Non-Equi Join**

**----------------**

**When table tables joined without any equi join condition is called non-equi join.**

**Non-equi joins are often used in scenarios where you want to join tables based on ranges of values rather than exact matches.**

**ex:**

**select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d**

**where e.esal between 30000 AND 50000; -- 2\*4 = 8 records**

**select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d**

**where e.esal>30000 AND e.esal<50000;**

**3) Self Join**

**-------------**

**A table join to itself is called self join.**

**In self join we need to create two table alias for same table.**

**ex:**

**select e1.eid,e1.ename,e1.esal,e2.job,e2.comm from emp e1,emp e2**

**where (e1.deptno=e2.deptno); -- 6 + 6 = 12 records**

**4) Cartisian Product**

**--------------------**

**If two tables are join without any condition is called cartisian product.**

**It returns all possible combinations.**

**ex:**

**select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d; -- 6 \* 4 = 24 records**

**5) Inner Join**

**-------------**

**It is similar to equi join.**

**It is given by ANSI people.**

**ANSI stands for American National Standard Institute.**

**ex:**

**select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e INNER JOIN dept d**

**ON (e.deptno=d.deptno); -- 6 records**

**select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e JOIN dept d**

**ON (e.deptno=d.deptno); -- 6 records**

**6) Outer Join**

**-------------**

**It is a extension of equi join.**

**A '+' is a outer join operator.**

**It returns matching as well as not matching records.**

**We have following list of outer joins.**

**i) left outer join**

**-----------------**

**SQL**

**----**

**select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e,dept d**

**where(e.deptno=d.deptno(+));**

**ANSI**

**-----**

**select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e LEFT OUTER JOIN dept d**

**ON(e.deptno=d.deptno);**

**ii) right outer join**

**-------------------**

**SQL**

**----**

**select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e,dept d**

**where(e.deptno(+)=d.deptno);**

**ANSI**

**-----**

**select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e RIGHT OUTER JOIN dept d**

**ON(e.deptno=d.deptno);**

**iii) full outer join**

**--------------------**

**ANSI**

**-----**

**select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e FULL OUTER JOIN dept d**

**ON(e.deptno=d.deptno);**

**Class 09:**

**Merge command**

**=============**

**Merge command is possible by using update and insert command.**

**ex:**

**student10 table**

**---------------**

**drop table student10;**

**create table student10(sno number(3),sname varchar2(10),sadd varchar2(12));**

**insert into student10 values(101,'raja','hyd');**

**insert into student10 values(102,'ravi','delhi');**

**insert into student10 values(103,'ramana','vizag');**

**commit;**

**student20 table**

**---------------**

**drop table student20;**

**create table student20(sno number(3),sname varchar2(10),sadd varchar2(12));**

**insert into student20 values(103,'Alan','Florida');**

**insert into student20 values(104,'John','Texas');**

**commit;**

**merge into student10 s1**

**using student20 s2**

**ON(s1.sno=s2.sno)**

**when matched then**

**update set sname=s2.sname,sadd=s2.sadd**

**when not matched then**

**insert(sno,sname,sadd) values(s2.sno,s2.sname,s2.sadd);**

**select \* from student10;**

**select \* from student20;**

**Sub Queries**

**============**

**We will declare a query inside another query such concept is called sub query.**

**In sub query , first inner query will execute then outer query.**

**Sub queries are used to select the records based on unknown values.**

**We have following list of sub queries.**

**1) Single Row subquery**

**2) Multiple Row subquery**

**3) Multiple Column subquery**

**1) Single Row subquery**

**-----------------------**

**If a sub query returns only one row is called single row sub query.**

**Sub queries can be nested upto 32 levels.**

**ex:**

**SQL**

**----**

**select \* from emp where eid=201;**

**Subquery**

**--------**

**select \* from emp where eid=(select eid from emp where ename='Alan');**

**ex:**

**SQL**

**---**

**select \* from emp where eid=201 and ename='Alan';**

**Subquery**

**---------**

**select \* from emp where eid=(select eid from emp where ename='Alan')**

**and ename=(select ename from emp where esal=9000);**

**Q) Write a query to display employees information whose salary is greater then mark salary?**

**select \* from emp where esal>(select esal from emp where ename='Mark');**

**Q) Write a query to display second highest salary from emp table?**

**select max(esal) from emp where esal<(select max(esal) from emp);**

**2) Multiple Row subquery**

**---------------------**

**If a sub query returns more then one row is called multiple row sub query.**

**To perform multiple row sub query we need to use multiple row operators.**

**We have three multiple row operators.**

**i) ANY**

**ii) ALL**

**iii) IN**

**i) ANY**

**-----**

**select \* from emp where esal > ANY (select esal from emp where deptno=10);**

**select \* from emp where esal < ANY (select esal from emp where deptno=10);**

**ii) ALL**

**------**

**select \* from emp where esal > ALL (select esal from emp where deptno=10);**

**select \* from emp where esal < ALL (select esal from emp where deptno=10);**

**iii) IN**

**---------**

**select \* from emp where esal IN (select esal from emp where deptno=10);**

**3) Multiple Column subquery**

**--------------------------**

**If a sub query returns more then one column is called multiple column subquery.**

**ex:**

**select \* from emp where(eid,ename,esal) IN (select eid,ename,esal from emp where eid=201);**

**select eid,ename,esal from emp where(eid,ename,esal) IN (select eid,ename,esal from emp where eid=201);**

**select eid,ename,esal from emp where(eid,ename,esal) IN (select eid,ename,esal from emp);**

**Assignment**

**===========**

**Q) Write a query to display last three records from emp table?**

**SELECT \* FROM ( SELECT \* FROM emp ORDER BY eid DESC) WHERE ROWNUM <= 3 order by eid;**

**Q) What is the difference between RDBMS and MongoDB?**

**RDBMS MongoDB**

**------------- -----------**

**It is a relational database. It is a non-relational or document based database.**

**It can't store the data in key and value pair. It stores the data in key and value pair.**

**Not suitable for hierarchical data storage. Suitable for hierarchical data storage.**

**It has a predefined(static) schema. It has a dynamic schema.**

**It contains tables. It contains collections.**

**It is row based. It is document based.**

**It is column based. It is field based.**

**It is slower. It is faster.**

**It supports SQL query language. It supports JSON query language.**

**PL/SQL**

**=======**

**PL/SQL stands for Procedural Language extension to Structure Query Language.**

**It is a extension of SQL and gives following features.**

**1) We can achieve programming features like control statements ,loops and etc.**

**2) It reduces network traffic.**

**3) We can display custom error messages by using the concept of exception handling.**

**4) We can perform related operations by using the concept of triggers.**

**5) We can save the source code permanently to database for repeated execution.**

**PL/SQL block**

**============**

**A PL/SQL program is also known as PL/SQL block.**

**syntax:**

**DECLARE**

**-**

**- -- Declaration Section**

**-**

**BEGIN**

**-**

**- -- Execution Section**

**-**

**EXCEPTION**

**-**

**- -- Exception Section**

**-**

**END**

**/**

**Here '/' is used to submit the PL/sQL block into a database.**

**Class 10:**

**PL/SQL block**

**------------**

**A PL/SQL program is also known as PL/SQL block.**

**ex:**

**DECLARE**

**-**

**- -- Declaration Section**

**-**

**BEGIN**

**-**

**- -- Executable Section**

**-**

**EXCEPTION**

**-**

**- -- Exception Section**

**-**

**END**

**/**

**Declaration Section**

**-------------------------**

**It is used to declare variables, exceptions, cursors and etc.**

**It is optional section.**

**Executable Section**

**------------------------**

**It contains actual logic of PL/SQL program.**

**It is a mandatory section.**

**Exception Section**

**-----------------------**

**It contains set of lines which are executed when exception raised.**

**It is a optional section.**

**To see the output in PL/SQL we need to use below command.**

**ex:**

**SQL> set serveroutput on**

**Q) Write a PL/SQL program to display Hello World?**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Hello World');**

**END;**

**/**

**Here DBMS\_OUTPUT is a package name.**

**PUT\_LINE is a procedure name.**

**Q) Write a PL/SQL program to perform sum of two numbers?**

**DECLARE**

**A number;**

**B number;**

**C number;**

**BEGIN**

**A:=10;**

**B:=20;**

**C:=A+B;**

**DBMS\_OUTPUT.PUT\_LINE(C);**

**END;**

**/**

**Declaration and Initialization using single line.**

**ex:**

**DECLARE**

**A number:=10;**

**B number:=20;**

**C number:=A+B;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(C);**

**END;**

**/**

**Using '&' symbol we can read dynamic inputs.**

**ex:**

**DECLARE**

**A number(3);**

**B number(3);**

**C number(6);**

**BEGIN**

**A:=&a;**

**B:=&b;**

**C:=A+B;**

**DBMS\_OUTPUT.PUT\_LINE('Sum of two numbers is '||C);**

**END;**

**/**

**In PL/SQL , DML operations are allowed.**

**Q) Write a PL/SQL program to insert a record into student table?**

**DECLARE**

**L\_sno number(3);**

**L\_sname varchar2(10);**

**L\_sadd varchar2(12);**

**BEGIN**

**L\_sno:=&sno;**

**L\_sname:='&sname';**

**L\_sadd:='&sadd';**

**insert into student values(L\_sno,L\_sname,L\_sadd);**

**DBMS\_OUTPUT.PUT\_LINE('Record Inserted');**

**END;**

**/**

**Q) Write a java program to update student name based on student number?**

**DECLARE**

**L\_sno number(3);**

**BEGIN**

**L\_sno:=&sno;**

**update student set sname='Alan' where sno=L\_sno;**

**DBMS\_OUTPUT.PUT\_LINE('Record Updated');**

**END;**

**/**

**Q) Write a PL/SQL program to delete student record based on student number?**

**DECLARE**

**L\_sno number(3);**

**BEGIN**

**L\_sno:=&sno;**

**delete from student where sno=L\_sno;**

**DBMS\_OUTPUT.PUT\_LINE('Record Deleted');**

**END;**

**/**

**In PL/SQL we can perform DRL operations.**

**To perform select queries we need to use "into" clause.**

**Q) Write a PL/SQL program to display employee name based on employee id?**

**DECLARE**

**L\_eid number(3);**

**L\_ename varchar2(10);**

**BEGIN**

**L\_eid:=&eid;**

**select ename into L\_ename from emp where eid=L\_eid;**

**DBMS\_OUTPUT.PUT\_LINE(L\_ename);**

**END;**

**/**

**Q) Write a PL/SQL program to display employee name and employee salary based on employee id?**

**DECLARE**

**L\_eid number(3);**

**L\_ename varchar2(10);**

**L\_esal number(10,2);**

**BEGIN**

**L\_eid:=&eid;**

**select ename,esal into L\_ename,L\_esal from emp where eid=L\_eid;**

**DBMS\_OUTPUT.PUT\_LINE(L\_ename||' '||L\_esal);**

**END;**

**/**

**Percentage(%) TYPE Attribute**

**==============================**

**It is used to declare a local variable with respect to the column of a table.**

**syntax:**

**variable\_name table\_name.col\_name%TYPE;**

**Q) Write a PL/SQL program to display employee name and employee salary based on employee id?**

**DECLARE**

**L\_eid emp.eid%TYPE;**

**L\_ename emp.ename%TYPE;**

**L\_esal emp.esal%TYPE;**

**BEGIN**

**L\_eid:=&eid;**

**select ename,esal into L\_ename,L\_esal from emp where eid=L\_eid;**

**DBMS\_OUTPUT.PUT\_LINE(L\_ename||' '||L\_esal);**

**END;**

**/**

**Percentage(%) ROWTYPE attribute**

**================================**

**It is used to declare a variable which holds complete row of a table.**

**A ROWTYPE variable we can't print directly.**

**To get the value from ROWTYPE attribute we need to use <variable\_name>.<column\_name>.**

**syntax:**

**variable\_name table\_name%ROWTYPE;**

**Q) Write a PL/SQL program to display employee information whose employee id is 202?**

**DECLARE**

**A emp%ROWTYPE;**

**BEGIN**

**select \* into A from emp where eid=202;**

**DBMS\_OUTPUT.PUT\_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job||' '||A.comm);**

**END;**

**/**

**Q) Write a PL/SQL program to display employee information based on employee id?**

**DECLARE**

**L\_eid emp.eid%TYPE;**

**A emp%ROWTYPE;**

**BEGIN**

**L\_eid:=&eid;**

**select \* into A from emp where eid=L\_eid;**

**DBMS\_OUTPUT.PUT\_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job||' '||A.comm);**

**END;**

**/**

**Control Statement**

**==================**

**We have three control statements in PL/SQL.**

**1) IF THEN**

**2) IF THEN ELSE**

**3) IF THEN ELSIF THEN ELSE**

**1) IF THEN**

**-----------**

**It evaluates the code if our condition is true.**

**ex:**

**DECLARE**

**A number:=5000;**

**BEGIN**

**IF A>2000 THEN**

**DBMS\_OUTPUT.PUT\_LINE('It is greatest');**

**END IF;**

**END;**

**/**

**ex:**

**DECLARE**

**A number:=2000;**

**BEGIN**

**IF A>5000 THEN**

**DBMS\_OUTPUT.PUT\_LINE('It is greatest');**

**END IF;**

**END;**

**/**

**2) IF THEN ELSE**

**-----------------**

**It will evaluate the code either our condition is true or false.**

**ex:**

**DECLARE**

**A number:=5000;**

**BEGIN**

**IF A>2000 THEN**

**DBMS\_OUTPUT.PUT\_LINE('It is greatest');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('It is least');**

**END IF;**

**END;**

**/**

**ex:**

**DECLARE**

**A number:=2000;**

**BEGIN**

**IF A>5000 THEN**

**DBMS\_OUTPUT.PUT\_LINE('It is greatest');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('It is least');**

**END IF;**

**END;**

**/**

**3) IF THEN ELSIF THEN ELSE**

**-------------------------**

**It will evaluate the code based on multiple conditions.**

**ex:**

**DECLARE**

**A number:=103;**

**BEGIN**

**IF A=100 THEN**

**DBMS\_OUTPUT.PUT\_LINE('It is police number');**

**ELSIF A=103 THEN**

**DBMS\_OUTPUT.PUT\_LINE('It is enquiry number');**

**ELSIF A=108 THEN**

**DBMS\_OUTPUT.PUT\_LINE('It is emergency number');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('It is invalid option');**

**END IF;**

**END;**

**/**

**Class 11:**

**To see the output in PL/SQL we need to use below command.**

**ex:**

**SQL> set serveroutput on**

**LOOPS**

**======**

**We have three types of loops in PL/SQL.**

**1) Simple loop**

**2) while loop**

**3) for loop**

**1) Simple loop**

**-----------**

**It evaluates the code how long our condition is true.**

**ex:**

**DECLARE**

**A number:=1;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Welcome');**

**LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Hello');**

**EXIT WHEN A=4;**

**A:=A+1;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Thank you');**

**END;**

**/**

**ex:**

**---**

**DECLARE**

**A number:=1;**

**BEGIN**

**LOOP**

**DBMS\_OUTPUT.PUT\_LINE(A);**

**EXIT WHEN A=10;**

**A:=A+1;**

**END LOOP;**

**END;**

**/**

**2) while loop**

**--------------**

**It will evaluate the code how long our condition is true.**

**ex:**

**DECLARE**

**A number:=1;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Welcome');**

**WHILE A<=4 LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Hello');**

**A:=A+1;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Thankyou');**

**END;**

**/**

**Q) Write a PL/SQL program to print 1 to 10 numbers in descending order?**

**DECLARE**

**A number:=10;**

**BEGIN**

**WHILE A>=1 LOOP**

**DBMS\_OUTPUT.PUT\_LINE(A);**

**A:=A-1;**

**END LOOP;**

**END;**

**/**

**3) for loop**

**----------**

**It evaluates the code how long our condition is true.**

**ex:**

**DECLARE**

**A number;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Welcome');**

**FOR A IN 1 .. 4 LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Hello');**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Thankyou');**

**END;**

**/**

**Q) Write a PL/SQL program to perform multiplication table of a given number?**

**DECLARE**

**N number;**

**A number;**

**BEGIN**

**N:=&no;**

**FOR A IN 1 .. 10 LOOP**

**DBMS\_OUTPUT.PUT\_LINE(N||' \* '||A||' = '||N\*A);**

**END LOOP;**

**END;**

**/**

**Exceptions**

**===========**

**Runtime errors are called exceptions.**

**We have two types of exceptions.**

**1) Predefined exceptions**

**2) Userdefined exceptions**

**1) Predefined exceptions**

**------------------------**

**We have following list of predefined exceptions.**

**i) NO\_DATA\_FOUND Exception**

**ii) TOO\_MANY\_ROWS Exception**

**iii) ZERO\_DIVIDE Exception**

**iv) VALUE\_ERROR Exception**

**v) DUP\_VAL\_ON\_INDEX Exception**

**vi) OTHERS**

**i) NO\_DATA\_FOUND Exception**

**--------------------------**

**This exception will raise when select statement does not return any record.**

**ex:**

**DECLARE**

**L\_Ename emp.ename%TYPE;**

**BEGIN**

**select ename into L\_Ename from emp where eid=209;**

**DBMS\_OUTPUT.PUT\_LINE(L\_Ename);**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('Please check employee Id');**

**END;**

**/**

**ii) TOO\_MANY\_ROWS Exception**

**----------------------------**

**This exception will raise when select statement returns more then one row.**

**ex:**

**DECLARE**

**L\_Ename emp.ename%TYPE;**

**BEGIN**

**select ename into L\_Ename from emp where deptno=20;**

**DBMS\_OUTPUT.PUT\_LINE(L\_Ename);**

**EXCEPTION**

**WHEN TOO\_MANY\_ROWS THEN**

**DBMS\_OUTPUT.PUT\_LINE('select statement returns multiple rows');**

**END;**

**/**

**iii) ZERO\_DIVIDE Exception**

**---------------------------**

**This exception will raise when are try to divide a number with zero.**

**ex:**

**DECLARE**

**A number;**

**BEGIN**

**A:=10/0;**

**DBMS\_OUTPUT.PUT\_LINE(A);**

**EXCEPTION**

**WHEN ZERO\_DIVIDE THEN**

**DBMS\_OUTPUT.PUT\_LINE('Dont divide by zero');**

**END;**

**/**

**iv) VALUE\_ERROR Exception**

**------------------------**

**This exception will raise when there is a mismatch with datatype or size.**

**ex:**

**DECLARE**

**A number(3);**

**BEGIN**

**A:=12345;**

**DBMS\_OUTPUT.PUT\_LINE(A);**

**EXCEPTION**

**WHEN VALUE\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('Please check the size');**

**END;**

**/**

**ex:**

**DECLARE**

**L\_Esal emp.esal%TYPE;**

**BEGIN**

**select ename into L\_Esal from emp where eid=207;**

**DBMS\_OUTPUT.PUT\_LINE(L\_Esal);**

**EXCEPTION**

**WHEN VALUE\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('Please check datatype');**

**END;**

**/**

**v) DUP\_VAL\_ON\_INDEX Exception**

**------------------------------**

**This exception will raise when we are trying to insert duplicate value in a primary key.**

**ex:**

**alter table emp add primary key (eid);**

**BEGIN**

**insert into emp values(207,'Ana',47000,60,'Salesman',800);**

**DBMS\_OUTPUT.PUT\_LINE('Thank you for inserting');**

**EXCEPTION**

**WHEN DUP\_VAL\_ON\_INDEX THEN**

**DBMS\_OUTPUT.PUT\_LINE('Please check the employee id ');**

**END;**

**/**

**vi) OTHERS**

**----------**

**It is a universal angular exception which is used to handle all types of exceptions.**

**ex:**

**DECLARE**

**L\_Ename emp.ename%TYPE;**

**BEGIN**

**select ename into L\_Ename from emp where eid=209;**

**DBMS\_OUTPUT.PUT\_LINE(L\_Ename);**

**EXCEPTION**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('Please check employee Id');**

**END;**

**/**

**2) Userdefined exceptions**

**-------------------------**

**Exceptions which are created by the user based on the application requirements are called custom exceptions.**

**Steps to work with userdefine exceptions.**

**-----------------------------------------**

**step1:**

**Declare the exception**

**ex:**

**<Exception\_Name> EXCEPTION;**

**step2:**

**Raise the exception**

**ex:**

**RAISE <Exception\_Name>;**

**step3:**

**Handle the exception**

**ex:**

**WHEN <Exception\_name> THEN**

**ex:**

**----**

**DECLARE**

**A number:=5000;**

**MY\_EX1 Exception;**

**BEGIN**

**IF A>2000 THEN**

**RAISE MY\_EX1;**

**END IF;**

**DBMS\_OUTPUT.PUT\_LINE(A);**

**EXCEPTION**

**WHEN MY\_EX1 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Number is too large');**

**END;**

**/**

**ex:**

**----**

**DECLARE**

**A number:=1000;**

**MY\_EX1 Exception;**

**BEGIN**

**IF A>2000 THEN**

**RAISE MY\_EX1;**

**END IF;**

**DBMS\_OUTPUT.PUT\_LINE(A);**

**EXCEPTION**

**WHEN MY\_EX1 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Number is too large');**

**END;**

**/**

**Cursors**

**=======**

**Cursor is a PL/SQL block which is used to run SQL commands.**

**We have two types of cursors.**

**1) Implicit cursor**

**2) Explicit cursor**

**1) Implicit cursor**

**--------------------**

**All the activities related to cursor like opening the cursor, processing the cursor and closing the cursor**

**which is done automatically is called implicit cursor.**

**We have four types of implicit cursor attributes.**

**i) SQL%ISOPEN**

**--------------**

**It is a boolean attribute which returns always false.**

**ii) SQL%FOUND**

**------------**

**It is a boolean attribute which returns true if SQL command is success and returns false if**

**SQL command is failed.**

**iii) SQL%NOTFOUND**

**------------------**

**It is completely inverse of SQL%FOUND.**

**It is a boolean attribute which returns false if SQL command is success and returns true if**

**SQL command is failed.**

**iv) SQL%ROWCOUNT**

**---------------**

**It will return number of records effecting in a database table.**

**ex:**

**---**

**BEGIN**

**IF SQL%ISOPEN THEN**

**DBMS\_OUTPUT.PUT\_LINE('TRUE');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('FALSE');**

**END IF;**

**END;**

**/**

**ex:**

**---**

**BEGIN**

**update student set sname='Jack' where sno=101;**

**IF SQL%FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('Record Updated');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Record Not Updated');**

**END IF;**

**END;**

**/**

**ex:**

**---**

**BEGIN**

**update student set sname='Jack' where sno=106;**

**IF SQL%FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('Record Updated');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Record Not Updated');**

**END IF;**

**END;**

**/**

**ex:**

**---**

**BEGIN**

**update student set sname='Jack' where sno=106;**

**IF SQL%NOTFOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('Record Updated');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Record Not Updated');**

**END IF;**

**END;**

**/**

**ex:**

**----**

**BEGIN**

**delete from student;**

**DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT||' Records Deleted');**

**END;**

**/**

**2) Explicit cursor**

**-------------------**

**All the activities related to cursor like opening the cursor, processing the cursor and closing the cursor which is done by a programmer is called explicit cursor.**

**We need to use explicit cursor when select statement returns more then one row.**

**We have following list of explicit cursor attributes.**

**i) %ISOPEN**

**--------------**

**It is a boolean attribute which returns true if cursor is open and returns false if cursor is closed.**

**ii) %FOUND**

**------------**

**It is a boolean attribute which returns true if SQL command is success and returns false if**

**SQL command is failed.**

**iii) %NOTFOUND**

**------------------**

**It is completely inverse of SQL%FOUND.**

**It is a boolean attribute which returns false if SQL command is success and returns true if**

**SQL command is failed.**

**iv) %ROWCOUNT**

**---------------**

**It will return number of records effecting in a database table.**

**Class 12:**

**To see the output in PL/SQL we need to use below command.**

**Ex:**

**SQL> set serveroutput on**

**Steps to work with cursor**

**-------------------------**

**step1:**

**------**

**Declare the cursor**

**step2:**

**-----**

**Open the cursor**

**step3:**

**-----**

**Fetch the data from cursor to local variables**

**step4:**

**-----**

**close the cursor**

**Q) Write a PL/SQL program to display employee names from employee table?**

**DECLARE**

**CURSOR C1 is select ename from emp;**

**L\_Ename emp.ename%TYPE;**

**BEGIN**

**OPEN C1;**

**LOOP**

**FETCH C1 into L\_Ename;**

**EXIT WHEN C1%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(L\_Ename);**

**END LOOP;**

**CLOSE C1;**

**END;**

**/**

**Q) Write a PL/SQL program to display employee name and employee salary from emp table?**

**DECLARE**

**CURSOR C2 is select ename,esal from emp;**

**L\_Ename emp.ename%TYPE;**

**L\_Esal emp.esal%TYPE;**

**BEGIN**

**OPEN C2;**

**LOOP**

**FETCH C2 into L\_Ename,L\_Esal;**

**EXIT WHEN C2%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(L\_Ename||' '||L\_Esal);**

**END LOOP;**

**END;**

**/**

**Q) Write a PL/SQL program to select employee information from employee table?**

**DECLARE**

**CURSOR C3 is select \* from emp;**

**A emp%ROWTYPE;**

**BEGIN**

**OPEN C3;**

**LOOP**

**FETCH C3 into A;**

**EXIT WHEN C3%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job);**

**END LOOP;**

**CLOSE C3;**

**END;**

**/**

**Procedures**

**===========**

**It is a named PL/SQL program which is compiled and permanently stored in a database for repeated execution.**

**syntax:**

**create or replace procedure <procedure\_name>**

**is**

**begin**

**-**

**- //code here**

**-**

**END;**

**/**

**Q) Write a procedure to display Hello World?**

**create or replace procedure p1**

**is**

**begin**

**DBMS\_OUTPUT.PUT\_LINE('Hello World');**

**END;**

**/**

**To execute the procedure we need to use below command.**

**ex:**

**exec p1;**

**Every procedure contains three parameters.**

**1) IN parameter**

**2) OUT parameter**

**3) IN OUT parameter**

**1) IN parameter**

**-----------------**

**IN parameter is used to accept the values from the user.**

**Q) Write a procedure to perform sum of two numbers?**

**create or replace procedure sum(A IN number,B IN number)**

**is**

**C number;**

**begin**

**C:=A+B;**

**DBMS\_OUTPUT.PUT\_LINE(C);**

**END;**

**/**

**We can execute the procedure by using below command.**

**ex:**

**exec sum(10,20);**

**Using procedure we can perform DML operations.**

**create or replace procedure deleteEmp(L\_Eid IN emp.eid%TYPE)**

**is**

**begin**

**delete from emp where eid=L\_Eid;**

**DBMS\_OUTPUT.PUT\_LINE('Record Deleted');**

**END;**

**/**

**To execute above procedure we need to use below command.**

**ex:**

**exec deleteEmp(207);**

**2) OUT parameter**

**-----------------**

**It is used to return the value to the user.**

**Q) Write a procedure to perform sum of two numbers and return sum?**

**create or replace procedure ret\_sum(A IN number,B IN number,C OUT number)**

**is**

**begin**

**C:=A+B;**

**END;**

**/**

**Steps to call a procedure having OUT parameter**

**----------------------------------------------**

**step1:**

**-----**

**Declare a bind variable**

**ex:**

**variable N number;**

**step2:**

**-----**

**Execute a procedure.**

**ex:**

**exec ret\_sum(10,40,:N);**

**step3:**

**-----**

**Print bind variable**

**ex:**

**print N;**

**3) IN OUT parameter**

**---------------------**

**It is used to accept and return the value from/to the user.**

**Q) Write a procedure to perform square of a given number?**

**create or replace procedure ret\_square(A IN OUT number)**

**is**

**begin**

**A:=A\*A;**

**END;**

**/**

**Steps to call a procedure having IN OUT parameter**

**-----------------------------------------------**

**step1:**

**------**

**Declare a bind variable.**

**ex:**

**variable N number;**

**step2:**

**-----**

**Initialize the bind variable.**

**ex:**

**BEGIN**

**:N := 5;**

**END;**

**/**

**step3:**

**------**

**Execute the procedure.**

**ex:**

**exec ret\_square(:N);**

**step4:**

**------**

**Print bind variable**

**ex:**

**print N;**

**Q) Write a query to display list of procedures present in database?**

**select object\_name from user\_objects where object\_type='PROCEDURE';**

**Q) Write a query to see the source code of a procedure?**

**select text from user\_source where name='P1';**

**Q) Write a query to drop the procedure?**

**drop procedure p1;**

**drop procedure sum;**

**drop procedure ret\_sum;**

**drop procedure ret\_square;**

**PL/SQL functions**

**=================**

**It is a named PL/SQL block which must and should returns a value.**

**syntax:**

**create or replace function <function\_name>**

**return datatype**

**is**

**begin**

**-**

**-**

**-**

**return datatype;**

**END;**

**/**

**Q) Write a function to perform sum of two numbers and return sum?**

**create or replace function f1(A number,B number)**

**return number**

**is**

**C number;**

**begin**

**C:=A+B;**

**return C;**

**END;**

**/**

**To call the function we need to use below command.**

**ex:**

**select f1(10,20) from dual;**

**Q) Write a function to accept salary and find out 10% of TDS?**

**create or replace function tax(salary number)**

**return number**

**is**

**TAX number;**

**begin**

**TAX:=salary\*10/100;**

**return TAX;**

**END;**

**/**

**To execute above function we need to use below command.**

**ex:**

**select tax(10000) from dual;**

**select eid,ename,esal, tax(esal) from emp;**

**select eid,ename,esal, tax(esal) as TDS from emp;**

**Q) Write a query to display list of functions present in database?**

**select object\_name from user\_objects where object\_type='FUNCTION';**

**Q) Write a query to see the source code of a function?**

**select text from user\_source where name='F1';**

**Q) Write a query to drop the function?**

**drop function F1;**

**drop function TAX;**

**Q) What is the difference between procedure and function?**

**Procedure function**

**---------- ------------**

**A procedure may or may not returns a value. A function always returns a value.**

**DML operations are allowed. DML operations are not allowed.**

**Can't be invoked by using select command. Can be invoked by using select command.**

**Packages**

**========**

**A package is a collection of logical related sub programs.**

**PL/SQL procedures and functions are called logical related sub programs.**

**Package creation involved in two steps.**

**1) package specification**

**------------------**

**It contains declaration of logical related sub programs.**

**2) package body**

**---------------**

**It contains definition of logical related sub programs.**

**ex:1**

**-----**

**package specification**

**----------------------**

**create or replace package pkg1**

**is**

**procedure sum(A IN number,B IN number);**

**END pkg1;**

**/**

**package body**

**------------**

**create or replace package body pkg1**

**is**

**procedure sum(A IN number,B IN number)**

**is**

**C number;**

**begin**

**C:=A+B;**

**DBMS\_OUTPUT.PUT\_LINE(C);**

**END;**

**END pkg1;**

**/**

**To call procedure we need to use below command.**

**ex:**

**exec pkg1.sum(10,60);**

**ex:2**

**-----**

**package specification**

**------------------**

**create or replace package pkg2**

**is**

**function f1(A number,B number)**

**return number;**

**end pkg2;**

**/**

**package body**

**-----------**

**create or replace package body pkg2**

**is**

**function f1(A number,B number)**

**return number**

**is**

**C number;**

**begin**

**C:=A+B;**

**return C;**

**END;**

**end pkg2;**

**/**

**To call the function we need to use below command.**

**ex:**

**select pkg2.f1(80,10) from dual;**

**Q) Write a query to display list of packages present in database?**

**select object\_name from user\_objects where object\_type='PACKAGE';**

**Q) Write a query to see the source code of a package?**

**select text from user\_source where name='PKG1';**

**Q) Write a query to drop the package?**

**drop package body pkg1;**

**drop package pkg1;**

**drop package body pkg2;**

**drop package pkg2;**

**Class 13**

**To see the output in PL/SQL we need to use below command.**

**ex:**

**SQL> set serveroutput on**

**Triggers**

**========**

**Trigger is a PL/SQL block which executes based on event.**

**Triggers events are insert, update and delete.**

**Triggers timings are before, after and insteadof.**

**syntax:**

**-------**

**create or replace trigger <trigger\_name> <timing> <event> on <object>**

**begin**

**-**

**-**

**-**

**end;**

**/**

**ex:**

**----**

**create or replace trigger trg1 before insert on student**

**begin**

**DBMS\_OUTPUT.PUT\_LINE('Thanks for Inserting');**

**END;**

**/**

**insert into student values(104,'ramulu','pune');**

**ex:**

**----**

**create or replace trigger trg2 after delete on student**

**begin**

**DBMS\_OUTPUT.PUT\_LINE('Thanks for deleting');**

**END;**

**/**

**delete from student where sno=104;**

**It is possible to declare multiple events in a single trigger.**

**ex:**

**create or replace trigger trg3 after update or delete or insert on student**

**begin**

**IF inserting then**

**DBMS\_OUTPUT.PUT\_LINE('Thanks for Insert');**

**elsif updating then**

**DBMS\_OUTPUT.PUT\_LINE('Thanks for update');**

**else**

**DBMS\_OUTPUT.PUT\_LINE('Thanks for delete');**

**END IF;**

**END;**

**/**

**update student set sname='rani' where sname='raja';**

**delete from student where sno=103;**

**insert into student values(104,'ramulu','pune');**

**Triggers are divided into two types.**

**1) Statement level trigger**

**2) Row level trigger**

**1) Statement level trigger**

**-----------------------**

**In statement level trigger , our trigger will execute only for one time irrespective of number of**

**records effecting in a database table.**

**By default every trigger is a statement level trigger.**

**ex:**

**create or replace trigger trg4 before delete on student**

**begin**

**DBMS\_OUTPUT.PUT\_LINE('Deleted!!!');**

**END;**

**/**

**delete from student;**

**2) Row level trigger**

**-----------------------**

**In row level trigger, our trigger will execute irrespective of number of records effected in a database table.**

**We can create a row level trigger by using "for each row" clause.**

**ex:**

**create or replace trigger trg5 before delete on student FOR EACH ROW**

**begin**

**DBMS\_OUTPUT.PUT\_LINE('Deleted!!!');**

**END;**

**/**

**delete from student; //trigger will execute multiple times**

**Q) Write a query to display list of triggers present in database?**

**select object\_name from user\_objects where object\_type='TRIGGER';**

**Q) Write a query to see the source code of a trigger?**

**select text from user\_source where name='TRG5';**

**Q) Write a query to drop the trigger?**

**drop trigger trg1;**

**drop trigger trg2;**

**drop trigger trg3;**

**drop trigger trg4;**

**drop trigger trg5;**

**imp questions for interview**

**=========================**

**1.What is SQL?**

**2. Sub languages of SQL?**

**3. What is table?**

**4. delete vs truncate command?**

**5. What is constraint and types of constraints?**

**6. What is sequence?**

**7. What is view?**

**8. What is index?**

**9. What is synonym?**

**10. What is join and types of joins?**

**11. ROWID vs ROWNUM?**

**12. What is group by clause, having clause and order by clause?**

**1. What is PL/SQL?**

**2. PL/SQL procedure vs function?**

**3. What is cursor in PL/SQL?**

**4. What is package in PL/SQL?**

**5. What is trigger in PL/SQL?**

**practice quries given by chat gpt**

**==================================**

**Sure! Here are the SQL-related questions without the answers or code examples:**

**1. Write a query to create a table named student.**

**2. Write a query to drop the table student.**

**3. Write a query to alter the student table and add a column state.**

**4. Write a query to rename the student table to students.**

**5. Write a query to create an index on the sname column of the student table.**

**6. Write a query to insert a record into the student table.**

**7. Write a query to update a record in the student table.**

**8. Write a query to delete a record from the student table.**

**9. Write a query to select all records from the student table.**

**10. Write a query to select a specific column sname from the student table.**

**11. Write a query to select all records from the student table where sname is 'Raja'.**

**12. Write a query to select records from the student table using the IN operator.**

**13. Write a query to select records using the LIKE operator in the student table.**

**14. Write a query to select records from the student table using the BETWEEN operator.**

**15. Write a query to select records with NULL values in the sadd column of the student table.**

**16. Write a query to count the total number of records in the student table.**

**17. Write a query to calculate the sum of a certain numeric column in the student table.**

**18. Write a query to calculate the average of a certain numeric column in the student table.**

**19. Write a query to group records in the student table by the sadd column.**

**20. Write a query with a HAVING clause to filter grouped records in the student table.**

**21. Write a query using an inner join to select columns from two tables.**

**22. Write a query using a left outer join.**

**23. Write a query using a right outer join.**

**24. Write a query to perform a self-join.**

**25. Write a query to perform a cross join.**

**26. Write a query to demonstrate a single-row subquery.**

**27. Write a query to demonstrate a multiple-row subquery using the IN operator.**

**28. Write a subquery within a SELECT statement.**

**29. Write a correlated subquery.**

**30. Write a query to demonstrate the use of the EXISTS operator.**

**31. Write a query to perform a commit operation.**

**32. Write a query to perform a rollback operation.**

**33. Write a query to create a savepoint.**

**34. Write a query to grant select permission on the student table to a user.**

**35. Write a query to revoke select permission on the student table from a user.**

**36. Write a query to create a function that adds two numbers.**

**37. Write a query to execute the function that adds two numbers.**

**38. Write a query to create a procedure to display 'Hello World'.**

**39. Write a query to execute the procedure that displays 'Hello World'.**

**40. Write a query to create a procedure to delete an employee record.**

**41. Write a query to create a procedure that accepts two numbers as parameters and returns their sum.**

**42. Write a query to create a row-level trigger that runs before an INSERT operation.**

**43. Write a query to test the previously created trigger by inserting a record.**

**44. Write a query to list all triggers present in the database.**

**45. Write a query to view the source code of a specific trigger.**

**46. Write a query to drop a trigger.**

**47. Write a query to create a materialized view based on a table.**

**48. Write a query to refresh a materialized view.**

**49. Write a query to demonstrate the use of implicit cursors.**

**50. Write a query to declare and use an explicit cursor to fetch employee names.**

**Some more practice quires given by sir**

**=================================**

##### Oracle Questions

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1007 | | **Alan** | | **70000** | | **10** | | **Clerk** | | **01-MAR-20** | | |
| 1210 | | **Jose** | | **35000** | | **10** | | **Clerk** | | **15-FEB-19** | | |
| 1163 | | **Mark** | | **13000** | | **20** | | **Manager** | | **23-DEC-23** | | |
| 1109 | | **Kelvin** | | **40000** | | **20** | | **Manager** | | **29-JUL-22** | | |
| 1021 | | **Lara** | | **25000** | | **30** | | **Hr** | | **07-SEP-21** | | |
| 1298 | | **Erick** | | **10000** | | **30** | | **Hr** | | **12-OCT-18** | | |
| 1345 | | **Brook** | | **18000** | | **40** | | **Salesman** | | **19-APR-22** | | |

1.  Write a query to create an employee table?

    create table employee (id number(3),name varchar2(10), salary number(10,2),

    deptno number(3),job varchar2(10), hiredate date);

2.  Write a query to create employee table using constraints?

    create table employee (id number(4) primary key,

    name varchar2(10) not null,

    salary number(10,2) check(salary between 5000 AND 50000), deptno number(3) NOT NULL,

    job varchar2(10) NOT NULL, hiredate date NOT NULL);

3.  Write a query to see the list of tables present in database?

    select\* from tab;

4.  Write a query to see the logical database name?

    select\* from global\_name;

5.  Write a query to see the list of users present in database?

    select username from all\_users;

6.  Write a query to change the password?

    ALTER USER system IDENTIFIED BY admin123;

7.  Write a query to display employee's information from employee table?

    select\* from employee;

8.  Write a query to display employee's information based on sorting order of employee id?

     select\* from employee order by id;

9.  Write a query to display current date and time?

     select current\_date from dual;

10. Write a query to see the list of records present in employee table?

     select count(\*) from employee;

11. Write a query to display first three records from employee table?

     select\* from employee where rownum<=3;

12. Write a query to display exact fourth record from employee table?

    select\* from employee where rownum<=4 minus

    select\* from employee where rownum<=3;

13. Write a query to display employee's information those who working in 10 department?

    select \* from employee where deptno=lO;

14. Write a query to display employee's information whose employee id is 1007, 1210 & 1163?

    select\* from employee where id IN(1007,1210,1163);

15. Write a query to display employee's information those who are not working in 20 department?

    select\* from employee where deptno <>20;

16. Write a query to display employee's information whose salary is null?

    select\* from employee where salary is null;

17. Write a query to display employee's information whose employee name start with 'A' letter?

     select\* from employee where name like 'A%';

18. Write a query to display employee id, employee name, employee salary and Annual salary?

     select id, name, salary, salary\*12 as Annual\_salary from employee;

19. Write a query to display employee's information whose salary between 30000 and 50000?

    select\* from employee where salary between 30000 AND 50000;

20. Write a query to see the list of views present in database?

    select view\_name from user\_views;

21. Write a query to see the list of synonyms present in database?

    select synonym\_name from user\_synonyms;

22. Write a query to see the list of indexes present in database?

    select index\_name from user\_indexes;

23. Write a query to display highest salary from employee table?

     select max{salary) from employee;

24. Write a query to display average salary of each employee?

    select avg{salary) from employee;

25. Write a query to display second highest salary from employee table?

    select max{salary) from employee where salary<{select max{salary) from employee);

26. Write a query to display sum of salary of each department?

     select sum{salary),deptno from employee group by deptno;

27. Write a query to display sum of salary of each job?

    select sum{salary),job from employee group by job;

28. Write a query to check username and password is valid or not?

    select count{\*) from userlist where uname='raja' and pwd='rani';

29. Write a query on group by clause, having clause and order by clause?

    select sum{salary),deptno from employee

    group by deptno

    having sum{salary)>36000

    order by deptno;

30. Write a query to display all the employees of hyderabad from highest to lowest salary?

    select id, name, salary FROM employee where city= 'Hyderabad' ORDER BY salary DESC;

31. Write a query to select all the names and their salary who has a role of developer?

    SELECT name, salary FROM employee WHERE job= 'Developer';

32. Write a query to select all the employees names of managers and developers who works as team moon?

    SELECT name FROM employee WHERE JOB IN{'Manager', 'Developer') AND team= 'Team Moon';

33. Write a query to display average salary of employees who are from Hyderabad?

    select avg{salary) as average\_salary from employee where city='Hyderabad';

34. Write a query to select employee names of manager and developers from hyderabad and salary more than 35000?

SELECT employee\_name FROM employees

    WHERE JOB IN{'Manager', 'Developer') AND city= 'Hyderabad' AND salary> 35000;

35. Write a query to display date and time?

    select TO\_CHAR(sysdate,'HH:Ml:SS YYYY-MM-DD') as Date\_And\_ Time from dual;

36. Write a query to replace '9' digit with '0'? select replace(salary,9,0) from employee;

37. Write a query to increase the salary of all developers of hyderabad by 20%?

    update employee set salary= salary\* 1.20

    where job= 'Developer' AND city= 'Hyderabad';

38. Write a query to promote all employees from clerk to manager?

    update employee set job='Manager' where job='Clerk';

39. Write a query to increment salary by 5000 based on employee id?

    update employee set salary=salary+5000 where id=ll09;

40. Write a query to terminate the employees whose employee id is 1109 and 1021?

    delete from employee where IN (1109, 1021);

41. Write a query to delete the records those who joined before 01-MAR-20?

    delete from employee where hiredate < TO\_DATE('2020-03-01', 'YYYY-MM-DD');

42. Write a query to truncate the table?

    truncate table employee;

43. Write a query to drop the table?

    drop table employee;

44. Write a query to create a synonym?

    create synonym syl for employee;

45. Write a query to drop the synonym?

    drop synonym synl;

46. Write a query to create a sequence?

    create sequence sql start with 100 increment by 1;

47. Write a query to drop the sequence?

    drop sequence sql;

48. Write a query to rename the column name from job to designation?

    alter table employee rename column job to designation;

49. Write a query to rename table name?

    rename employee to employees;

50. Write a query to create an index?

    create view vl as select \* from employee;

51. Write a query to drop a view?

    drop view vl;

52. Write a query to create a complex view?

    create view vl as select e.id, e.name, e.sal, d.dname, d.dloc from employee e, dept d

    where (e.deptno=d.deptno);

53. Write a query to display all employees' information whose salary is greater than Alan salary?

     select salary from employee where salary>(select salary from employee where name=' Alan');

54. Write a query to delete duplicate records from employee table?

    DELETE FROM employee WHERE ROWID NOT IN (

    SELECT MAX(ROWID) FROM employee group by id

    );

55. Write a query for equi-join?

    select e.id, e.name, e.salary, d.dname, d.dloc from employee e, dept d

    where(e.deptno=d.deptno);

56. Write a query for inner join?

    select e.id, e.name, e.salary, d.dname, d.dloc from employee e join dept d

    ON(e.deptno=d.deptno);

57. Write a query for outer join?

    select e.id, e.name, e.salary, d.dname, d.dloc from employee e LEFT OUTER JOIN dept d ON(e.deptno=d.deptno);

    Or

    select e.id, e.name, e.salary, d.dname, d.dloc from employee e Right OUTER JOIN dept d ON(e.deptno=d.deptno);

    Or

    select e.id, e.name, e.salary, d.dname, d.dloc from employee e FULL OUTER JOIN dept d ON(e.deptno=d.deptno);

58. Write a query for self-join?

    select el.id,el.name,el.salary,e2.deptno,e2.hiredate from employee el, employee e2

    where{el.deptno=e2 .deptno);

59. Create a procedure in PL/SQL?

    create or replace procedure sum (A IN number, BIN number) IS

    C number; BEGIN C:=A+B;

    DBMS\_OUTPUT.PUT\_LINE{C); END;

60. Create a function in PL/SQL?

    create or replace function ret\_sum {A number, B number) Return number

    Is

    C number; Begin C:=A+B;

    return C; END;

    I

61. Create a Trigger in PL/SQL?

    create or replace trigger trgl before insert on student begin

    DBMS\_OUTPUT.PUT\_LINE{'Thankyou for inserting the data'); END;

    I

62. Create a Cursor in PL/SQL?

    DECLARE

    CURSOR Cl is select\* from emp; A emp%ROWTYPE;

    BEGIN OPEN Cl; LOOP

    FETCH Cl into A;

    DBMS\_OUTPUT.PUT\_LINE{A.eidl I' 'I IA.enamel I' 'I IA.esall I' 'I IA.deptnol I' 'I IA.job); EXIT WHEN Cl%NOTFOUND;

    END LOOP; CLOSE Cl; END;

    I

63. Write a query to see the source code of a procedure?

    select text from user\_source where name='SUM';

64. Write a query to see the source code of a function?

select text from user\_source where name='RET\_SUM';

65. Write a query to see the list of procedures present in database?

    select object\_name from user\_objects where object\_type='PROCEDURE';

66. Write a query to see the list of functions present in database?

    select object\_name from user\_objects where object\_type='FUNCTION';

67. Write a query to see the list of packages present in database?

    select object\_name from user\_objects where object\_type='PACKAGE';

68. Write a query to see the list of triggers present in database?

    select object\_name from user\_objects where object\_type='TRIGGER';

69. Write a query to display employee id, first name, last name, call start time, call end time, customer id, customer name, customer address?

    select emp.id, emp.first\_name, emp.last\_name,

    c.start\_time, c.end\_time from employee e INNER JOIN call c ON(emp.id=c.employee\_id) INNER JOIN customer cust

    ON(cust.id=caII.customer \_id);

**Imp commands in sql/plsql**

**=======================**

**Here’s a comprehensive list of all major Oracle SQL commands with a brief one-line description for each:**

**1. DDL (Data Definition Language) Commands**

**These commands define and modify the database structure.**

1. **CREATE: Creates a new database object (table, view, index, etc.).**
2. **ALTER: Modifies an existing database object.**
3. **DROP: Deletes an existing database object (table, index, etc.).**
4. **TRUNCATE: Removes all rows from a table but keeps the structure intact.**
5. **RENAME: Changes the name of an existing database object.**
6. **COMMENT: Adds a comment to a database object for documentation.**
7. **FLASHBACK: Restores an object or table to a previous state.**
8. **ANALYZE: Collects statistics or validates the structure of database objects.**
9. **SET UNUSED: Marks a column or constraint as unused without dropping it.**
10. **CREATE TABLESPACE: Creates a new tablespace.**
11. **DROP TABLESPACE: Deletes an existing tablespace.**
12. **CREATE INDEX: Creates an index to improve query performance.**
13. **DROP INDEX: Deletes an existing index.**

**2. DML (Data Manipulation Language) Commands**

**These commands manage data within existing database objects.**

1. **SELECT: Retrieves data from one or more tables or views.**
2. **INSERT: Adds new rows of data into a table.**
3. **UPDATE: Modifies existing data in a table.**
4. **DELETE: Removes data rows from a table.**
5. **MERGE: Combines INSERT and UPDATE operations based on conditions (also called an "upsert").**
6. **TRUNCATE: Quickly removes all rows from a table (doesn't log individual row deletions).**
7. **LOCK TABLE: Explicitly locks a table to prevent access by other sessions.**

**3. DCL (Data Control Language) Commands**

**These commands manage access control to the database.**

1. **GRANT: Assigns privileges or roles to users or other database objects.**
2. **REVOKE: Removes privileges or roles from users or other database objects.**
3. **ALTER USER: Modifies user properties like password, profile, etc.**
4. **CREATE ROLE: Creates a new role.**
5. **DROP ROLE: Deletes a role from the database.**
6. **SET ROLE: Sets the role to be used for the current session.**

**4. TCL (Transaction Control Language) Commands**

**These commands manage the state of transactions in the database.**

1. **COMMIT: Saves all changes made during the current transaction.**
2. **ROLLBACK: Undoes all changes made during the current transaction.**
3. **SAVEPOINT: Sets a point in the transaction to which you can later roll back.**
4. **SET TRANSACTION: Defines specific settings for the current transaction, such as isolation level.**

**5. Query and Utility Commands**

**These commands assist with querying and managing the database environment.**

1. **EXPLAIN PLAN: Displays the execution plan for a SQL query to help optimize queries.**
2. **DESCRIBE: Displays the structure of a table or other database object.**
3. **SHOW: Displays various database settings, parameters, or session information (used in some tools like SQL\*Plus).**
4. **SELECT INTO: Selects data from one table and inserts it into another table (similar to INSERT INTO SELECT).**
5. **CREATE VIEW: Creates a virtual table based on the result of a query.**
6. **DROP VIEW: Deletes a view.**
7. **CREATE SYNONYM: Creates an alias for a database object (like a table).**
8. **DROP SYNONYM: Deletes a synonym.**
9. **CREATE SEQUENCE: Creates a sequence generator (useful for auto-incrementing numbers).**
10. **DROP SEQUENCE: Deletes an existing sequence.**
11. **CREATE PROCEDURE: Defines a stored procedure for business logic or reusable actions.**
12. **CREATE FUNCTION: Defines a function that returns a value based on input parameters.**
13. **CREATE TRIGGER: Creates a trigger to automatically perform actions when certain events occur on a table.**
14. **DROP PROCEDURE: Deletes a stored procedure.**
15. **DROP FUNCTION: Deletes a function.**
16. **DROP TRIGGER: Deletes a trigger.**
17. **CREATE PACKAGE: Defines a collection of related procedures, functions, and variables.**
18. **DROP PACKAGE: Deletes a package.**
19. **CREATE DATABASE LINK: Creates a link to another database for accessing remote objects.**
20. **DROP DATABASE LINK: Deletes a database link.**
21. **ALTER SESSION: Modifies session-level parameters and settings (e.g., timezone, NLS settings).**
22. **SHOW ERRORS: Displays compilation errors for PL/SQL code.**
23. **DESCRIBE: Provides the structure of a database object (e.g., table columns).**

**6. Data Integrity Commands**

**These commands enforce constraints and manage database integrity.**

1. **CREATE CONSTRAINT: Defines a constraint (e.g., primary key, foreign key) on a table.**
2. **DROP CONSTRAINT: Removes an existing constraint from a table.**
3. **ENABLE CONSTRAINT: Enables a constraint on a table.**
4. **DISABLE CONSTRAINT: Disables a constraint on a table.**
5. **ADD CONSTRAINT: Adds a constraint to an existing table.**
6. **CHECK: Defines a condition that data must meet to be entered into a table (part of constraints).**
7. **PRIMARY KEY: Defines a column (or set of columns) that uniquely identifies a record in a table.**
8. **FOREIGN KEY: Defines a relationship between two tables using a key from one table in another.**

**7. Backup and Recovery Commands**

**These commands are used for database backup and recovery.**

1. **BACKUP: Performs a backup operation on the database (via RMAN or other tools).**
2. **RESTORE: Recovers data or a database from a backup.**
3. **RECOVER: Applies logs to restore a database to a point in time.**

**8. Performance and Statistics Commands**

**These commands help analyze and improve the performance of queries and the database.**

1. **ANALYZE: Collects statistics or validates database objects for optimization.**
2. **DBMS\_STATS: Collects and manages statistics on database objects to optimize performance.**
3. **V$SESSION: Views session-level information for debugging or monitoring.**

**9. Other Miscellaneous Commands**

**These commands cover additional functionality and administrative tasks.**

1. **ALTER DATABASE: Alters database settings or performs certain operations like opening/closing the database.**
2. **ALTER SYSTEM: Modifies system-level parameters for Oracle database.**
3. **CREATE DIRECTORY: Defines a directory object to access the file system.**
4. **DROP DIRECTORY: Deletes a directory object.**
5. **CREATE CONTROLFILE: Creates a new control file for the database.**
6. **ALTER SYSTEM KILL SESSION: Terminates a session in the database.**
7. **CREATE USER: Creates a new database user.**
8. **DROP USER: Deletes an existing user from the database.**
9. **ALTER USER: Modifies user account attributes (e.g., password, default tablespace).**
10. **SHOW PARAMETER: Displays values for initialization parameters.**
11. **STARTUP: Starts the Oracle database instance.**
12. **SHUTDOWN: Shuts down the Oracle database instance.**

**10. PL/SQL Commands**

**PL/SQL is Oracle’s procedural extension to SQL, which includes the following commands:**

1. **DECLARE: Defines local variables in a PL/SQL block.**
2. **BEGIN: Begins a PL/SQL anonymous block or procedure.**
3. **END: Marks the end of a PL/SQL block.**
4. **EXCEPTION: Defines error handling in PL/SQL.**
5. **LOOP: Starts a loop within a PL/SQL block.**

**IF-ELSE: Performs conditional logic in PL/SQL**