ORACLE

Types of Data

We have two types of data.

1) Unstructured Data

2) Structured Data

1) Unstructured Data

Data which is not in readable format such type of data is called unstructured data.

In general , meaning less data is called unstructured data.

ex:

```
201   Lakemba    SYD    NSW    AUS
```

2) Structured Data

Data which is in readable format such type of data is called structured data.

In general, meaning full data is called structured data.

ex:

```
Unit  Locality   City   State  Country
---   -------    -----  ------ --------
201   Lakemba    SYD    NSW    AUS
```

Oracle

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

Oracle is a product of Oracle Corporation.

Oracle is classified into two types.

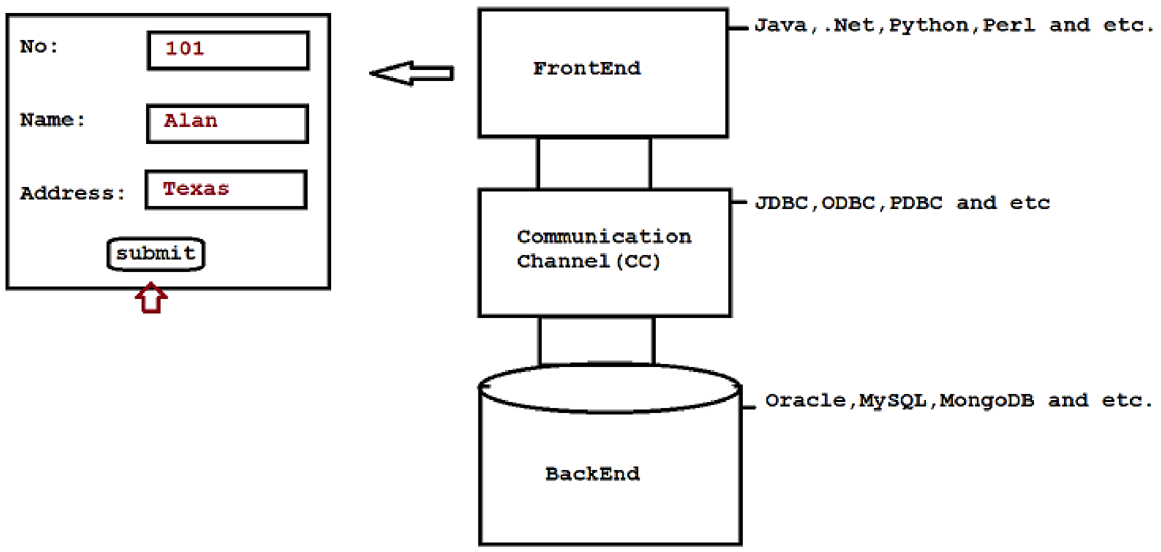
```
                    Oracle
                      |
     |-----------------------------------------------|
     SQL                                         PL/SQL
```

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

Client-Server Architecture

In this architecture we will see how our form data will store in a database.

Diagram: oracle1.1


No:
101
Name:
Alan
Address:
Texas
submit
FrontEnd
Java,.Net,Python,Perl and etc.
Communication Channel(CC)
JDBC,ODBC,PDBC and etc
BackEnd
Oracle,MySQL,MongoDB and etc.

FrontEnd

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

ex:

C++, C#, Java, Python, .Net , Perl 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 the enduser but it performs operations based on the instructions given by frontend is called backend.

ex:

Oracle , MySQL, Teradata, Sybase , MongoDB , NoSQL and etc.

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) Selecting the required data.

3) Updating the existing data.

4) Deleting the unnecessary.

DBMS

A database along with software which is used to manage the database is called database management system.

RDBMS

A database which is created based on relational theories such type of database is called relational database management system.

ex:

Oracle

MySQL

SQL Server

PostgreSQL

TeraData

Sybase

and etc.

Q)Difference between DBMS vs RDBMS?

DBMS	RDBMS
DBMS stands for Database Management System.	RDBMS stands for Relational Database Management System.
It stores the data in the form of files.	It stores the data in the form of tables.
It is used to handle small amount of data.	It is used to handle large amount of data.
It provides support for a single user at a time	It provides support for multiple users at a time.
Normalization is not possible for DBMS.	Normalization is possible for RDBMS
No security of data.	High security of data.

SQL

SQL stands for structured query language which pronounce as SEQUEL.

This language is used to communicate with oracle database.

It is a command based language.

Every command starts with verb.

ex:

```
select
insert
update
delete
merge
```

Every command ends with semicolon.

It is a case insensitive language.

ex:

```
select * from student;
SELECT * FROM STUDENT;
```

It is developed by Mr.Codd in 1972 (By IBM).

Sub Languages of SQL

We have five sub languages of SQL.

1)DDL (Data Definition Language)

2)DML (Data Manipualation 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.

2)DML (Data Manipualation Language)

This language is used to manipulate the database 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 present in 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 prevent a user to access the data.

It is a collection of two commands.

ex:

grant and revoke

Table

A table is an object which is used to represent the data.

A table is a collection of rows and columns.

ex:

SNO	SNAME	SADD
101	Alan	USA
102	Jose	UAE
103	Lisa	UK

Oracle is a case insensitive language.

But data present in a table is a case sensitive

Oracle

Version	**: 10g or 11g**
Creator	**: Mr.Codd**
Vendor	**: Oracle Corporation**
website	**: www.oracle.com/in/database/**
open source	**: open source**
username	**: system (default)**
password	**: admin**

Download link :

https://drive.google.com/file/d/0B9rC21sL6v0td1NDZXpkUy1oMm8/view?usp=drive_link&resourcekey=0-aKooR3NmAh_eLo_qGw_inA

Establish the connection with database software

To execute any SQL in a database we need to connect with database software.

Once the work with database is completed then we need to disconnect with database software.

ex:1

```
SQL> connect
        username : system
        password : admin
SQL> disconnect
```

ex:2

```
SQL> conn
        username : system
        password : admin
SQL> disc
```

ex:3

```
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 emp(eid number(3), ename varchar2(10), esal
number(10,2),
                        deptno number(3),job varchar2(10),comm
number(6));


create table dept(deptno number(3),dname varchar2(10),dloc
varchar2(12));
```

describe command

It is used to see the structure of a table.

syntax:

```
desc emp;

desc student;

desc dept;
```

insert command

It is used to insert a record/row into database table.

syntax:

```
insert into <table_name> values(val1,val2,...,valN);
```

ex:

```
insert into student values(101,'raja','hyd');
insert into student values('ravi',102,'delhi'); //invalid
insert into student values(102,'ravi'); //invalid
```

null

It is used to represent undefined or unavailable.

ex:

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

approach2

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

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

approach3

Using '&' symbol we can read dynamic inputs.

ex:

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

commit command

It is used to make the changes permanent to database.

ex:

```
        commit;
```

dept table

```
create table dept(deptno number(3),dname varchar2(10),dloc varchar2(12));
insert into dept values(10,'CSE','HYD');
insert into dept values(20,'EEE','DELHI');
insert into dept values(30,'ECE','PUNE');
insert into dept values(40,'MEC','VIZAG');
commit;
```

emp table

```
create table emp(eid number(3), ename varchar2(10), esal number(10,2),
                                deptno number(3),job varchar2(10),comm
number(6));

insert into emp values(201,'Alan',9000,10,'Clerk',null);
insert into emp values(202,'Jose',18000,10,'Clerk',100);

insert into emp values(203,'Kelvin',29000,20,'HR',200);
insert into emp values(204,'Lisa',19000,20,'HR',500);

insert into emp values(205,'Nelson',49000,30,'Manager',300);
insert into emp values(206,'Jack',30000,30,'Manager',900);

commit;
```

select command

It is used to read the records from database table.

syntax:

```
select * from <table_name>;
```

ex:

```
select * from emp;
select * from dept;
select * from student;
```

Projection

Selecting specific columns from the database table is called projection.

ex:

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

In select command we can perform arithmetic operations also.

ex:

```
select sno from student;
select sno+100 from student;
select sno-100 from student;
```

Column alias

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

Column alias we can apply to any column.

Column alias are temperory ,Once the query is executed we will loss the column alias.

ex:

```
select sno+100 as SNO  from student;
select sno-100 as SNO from student;
select sno,sname,sadd from student;
select sno as ROLLNO, sname as FIRSTNAME , sadd as CITY from student;
```

Interview Queries

Q)Write a query to display logical database name / schema?

```
select * from global_name;
```

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

```
select * from tab;
```

Q)Write a query to display all the employees information from emp 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 employee id, employee name, employee salary and annual salary from emp 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 ANNUL_SALARY from emp table?

```
elect eid,ename,esal,esal*12 as ANNUAL_SALARY from emp;
```

Where clause

It is used to retrive perticular records from database.

syntax:

```
select * from <table_name> where condition;

ex:

select * from student where sno=101;

select * from student where sname='raja';

select * from student where sname='RAJA'; // no rows selected
```

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 employees information those who ar working as a manager?

```
select * from emp where job='Manager';
```

Q)Write a query to display employees information those who not earn commission?

```
select * from emp where comm=null; // no rows selected

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

Q)Write a query to display employoees information whose salary is greater then 15000?

```
select * from emp where esal>15000;
```

Update command

It is used to update the record which is present in a database table.

syntax:

```
update <table_name> set <col_name>=value where condition;
```

ex:

```
update student set sname='rani' where sno=105;
update student set sadd='delhi' where sname='ravi';
update student set sname='raju',sadd='mumbai' where sno=104;
```

Note:

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

ex:

```
update student set sno=101;
update student set sname='raja';
update student set sadd='hyd';
```

Q)In how many ways we can insert the data in a table?

There are two ways to insert a data in a table.

1) Using insert command
2) Using update command

delete command

It is used to delete the records from database table.

syntax:

```
delete from <table_name> where condition;
```

ex:

```
delete from student where sno=101;
delete from student where sname='raju';
delete from student where sadd is null;
```

Note:

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

ex:

```
delete from student;
delete from emp;
delete from dept;
```

Q)Write a query to promote all the employee to Salesman those who are working on clerk designation?

```
update emp set job='Salesman' where job='Clerk';
```

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

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

Q)Write a query to terminate the employee whose salary is 30000?

```
delete from emp where esal=30000;
```

Note:

All DML commands are temperory.

Logical Operators

If we want to write more then one condition then we need to use logical operators.

We have three logical operators.

1) AND

2) OR

3) NOT

1) AND

It will return the records only if both the conditions are true.

Here conditions must be from same row.

ex:

```
select * from student where sno=101 AND sname='raja';
select * from student where sno=101 AND sname='ravi'; // no rows selected
```

2) OR

It will return the records if any condition is true.

Here conditions can be from any row.

ex:

```
        select * from student where sno=101 OR sname='raja';

        select * from student where sno=101 OR sname='ravi';
```

3) NOT

It is used to return the records except the condition.

A '<>' symbol denoted as not operator.

ex:

```
        select * from student where NOT sno=101;

        select * from student where sno<>101;

        select * from student where NOT sname='raja';

        select * from student where  sname<>'raja';
```

IN operator

It is a replacement of OR operator.

It will return the records those who are matching in the list of values.

ex:

```
select * from student where sno IN(101,102,103,108);


select * from student where sname IN ('raja','ravi','rakesh');
```

Between operator

It is used to return the records those who are in the range of values.

Between operator uses AND operator.

In between operator we will declare first lower limit then higher limit.

ex:

```
select * from student where sno between 101 AND 105;


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

Interview Queries

Q) Write a query to display employee information whose employee id is 204 and working as a HR?

```
select * from emp where eid=204 and job='HR';
```

Q)Write a query to display employees information those who are working in 10 ,20 and 30 department?

```
select * from emp where deptno IN(10,20,30);
```

Q)Write a query to display employees information whose salary greater then 35000 and less then 50000?

```
select * from emp where esal between 35000 and 50000;

select * from emp where esal>=35000 and esal<=50000;
```

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

```
select * from emp where NOT deptno=10;
select * from emp where deptno<>10;
```

Pattern Matching operators

Pattern matching operators are used to select the letter from database table.

Pattern matching operators will take the support of like keyword.

We have two types of pattern matching operators.

1)Percentage (%)

2)Underscore (_)

1)Percentage (%)

Q)Write a query to display employees information whose employee 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 'I'?

```
select * from emp where ename like '%I%';
```

2)Underscore (_)

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

```
select  * from emp where ename like '_I%';
```

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

```
select * from emp where ename like '%a_';
```

Comment in SQL

syntax:

```
-- comment here
```

Copy of a table or duplicate table

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

ex:

```
create table employee as select * from emp;
create table employee as select eid,ename,esal from emp;
create table employee as select * from emp where deptno=10;
create table employee as select * from emp where comm is null;
create table employee as select * from emp where eid in(201,202,203);
create table employee as select * from emp where eid between 201 and 206;
```

cl scr

It is used to clear th SQL command prompt.

ex:

```
cl scr
```

DDL commands

create	-	(tables)
alter	-	(columns)
drop	-	(tables)
truncate	-	(Rows/Records)
rename	-	(table)

alter command

Using alter command we can perform following operations.

i) Adding the new column

ii) Modifying a column

iii) Dropping a existing column

iv) renaming a column

i) Adding the new column

Using alter command we can add new column to a existing table.

All columns will be added at last in a 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 a column

Using alter command we can modify a column.

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

We can change datatype of a column only if that column is empty.

syntax:

```
	alter table <table_name> MODIFY(colname  datatype(size));
```

ex:

```
desc student;

alter table student modify(state varchar2(15));

select * from student;

desc student;

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

desc student;
```

iii) Dropping a existing column

Using alter command we can drop a existing column.

syntax:

```
alter table <table_name> DROP (col1,col2,..,colN);
```

ex:

```
alter table student DROP (state,pincode);
```

iv) renaming a column

Using alter command we can rename a column name.

syntax:

alter table <table_name> rename column <old_name> to <new_name>;

ex:

```
alter table student rename column  sadd to city;

alter table emp rename column esal to dailywages;

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

drop command

It is used to drop a table from database.

syntax:

```
drop table <table_name>;
```

ex:

```
drop table emp;

drop table student;

drop table dept;
```

truncate command

It is used to delete the records permenently from database table.

syntax:

```
truncate table <table_name>;
```

ex:

```
truncate table emp;

truncate table dept;

truncate table student;
```

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

delete	truncate
It deletes the data temperory.	It deletes the data permanently.
We can rollback the data.	We can't rollback the data.
Where clause can be used.	Where clause can't be used.
It is a DML command.	It is DDL command.

rename command

It is used to rename the table name.

syntax:

```
rename <old_name> to <new_name>;
```

ex:

```
rename  emp to employees;

rename dept to department;

rename student to students;
```

Functions

Functions are used to manipulate the data items 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 groups.

We have following list of group functions.

ex:

sum(), avg(), max(), min(), count(*) and count(exp).

Q)Write a query to display sum of salary of each employee from emp table?

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

Q)Write a query to display average salary of each employee form emp table?

```
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 minimum salary from emp table?

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

Q)Write a query to display number of records present in a table?

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

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

count(*)

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

It will include null records.

ex:

```
select count(*) from emp; // 6 records
```

count(exp)

It will return number of values present in a database table 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 invalid?

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

    select count(*) from userlist where uname='raja' and pwd='rani1'; // 0
```

Dual table

Dual table is a dummy table which contains 1 row and 1 column.

It is used to perform arithmetic operations and to see the current system date.

ex

```
    select 10+20 from dual;

    select 10*30 from dual;

    select sysdate from dual;

    select current_date from dual;
```

2)Scalar Functions

Scalar functions are applicable for single row.

We have following list of scalar functions.

i) Character functions

ii) Number functions

iii) Date functions

iv) Conversion functions

i) Character functions

abs()

It will return absoluate value.

ex:

```
select abs(-10) from dual; // 10
```

power(A,B)

It will return power value.

ex:

```
select power(2,3) from dual;
```

upper()

It will convert to upper case.

ex:

```
select upper('oracle') from dual;
```

lower()

It will convert to lower case.

ex:

```
select lower('ORACLE') from dual;
```

initcap()

It will return initial letter with capital.

ex:

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

lpad()

It will pad the characters at left side.

ex:

```
select lpad('oracle',10,'z') from dual;
```

rpad()

It will pad the characters at right side.

ex:

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

ltrim()

It will trim the characters from left side.

ex:

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

rtrim()

It will trim the characters from right side.

ex:

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

trim()

It will trim the charactrs from both the sides.

ex:

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

ii)Number function

abs()

It will return absolute value.

ex:

```
select abs(-56) from dual; //56
```

power(A,B)

It will return power value.

ex:

```
select power(2,3) from dual; // 2*2*2=8
```

ceil()

It will return ceil value.

ex:

```
select ceil(10.6) from dual; // 11


select ceil(9.2) from dual; //10
```

floor()

It will return floor value.

ex:

```
select floor(10.6) from dual; // 10


select floor(9.2) from dual; //9
```

round()

It will return nearest value.

ex:

```
select round(10.5) from dual; // 11
select round(10.4) from dual; //10
```

trunc()

It will remove decimals.

ex:

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


select trunc(56.341) from dual; //56
```

greatest()

It will return greatest value.

ex:

select greatest(10,20,30) from dual; //30

least()

It will return least value.

ex:

```
select least(10,20,30) from dual; //10
```

Working with Date values

Every database software support different date patterns.

ex:

Oracle - dd-MMM-yy

MySQL - yyyy-MM-dd

We need to insert date value inthe pattern which is supported by underlying database software.

ex:

```
drop table emp1;

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

insert into emp1 values(301,'Alan','01-JAN-22');


insert into emp1 values(302,'Jose',sysdate);

insert into emp1 values(303,'Lisa',current_date);

insert into emp1 values(304,'Jessi','15-MAR-23');

commit;
```

iii) Date functions

We have following list of date functions

ADD_MONTHS()

It will add months in a given date.

ex:

```
select ADD_MONTHS(sysdate,4) from dual;

select ADD_MONTHS('01-JAN-23',5) from dual;
```

MONTHS_BETWEEN()

It will return number of months between two given dates.

ex:

```
select MONTHS_BETWEEN('01-JAN-23','01-JUL-23') from dual;

select abs(MONTHS_BETWEEN('01-JAN-23','01-JUL-23')) from dual;

select abs(MONTHS_BETWEEN('01-JAN-23','15-JUL-23')) from dual;
```

NEXT_DAY()

It will return given day in a week.

ex:

```
select NEXT_DAY(sysdate,'SUNDAY') from dual;

select NEXT_DAY(sysdate,'TUESDAY') from dual;
```

LAST_DAY()

It will return last date of a month.

ex:

```
select LAST_DAY(sysdate) from dual;


select LAST_DAY('01-FEB-23') from dual;
```

iv) Conversion function

A conversion function is used to convert from one type to another type.

ex:

TO_CHAR()

We have two pseudo for TO_CHAR() .

1) number to_char()

It will take '9' in digits and $ 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') as ESAL 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,'day') from dual;

select TO_CHAR(sysdate,'month') from dual;

select TO_CHAR(sysdate,'year') from dual;

select TO_CHAR(sysdate,'dd') from dual;

select TO_CHAR(sysdate,'MM') from dual;

select TO_CHAR(sysdate,'YYYY') from dual;

select TO_CHAR(sysdate,'HH:MI:SS') from dual;

select TO_CHAR(sysdate,'yyyy-MM-dd HH:MI:SS') from dual;
```

Group by clause

It is used to divide the rows into multiple groups so that we can apply group functions.

A column which we used in a select clause then same column name we need to use in group by clause.

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

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

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

```
select max(esal),job from emp group by job;
```

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

```
select avg(esal),deptno from emp group by deptno;
```

Having clause

It is used to filter the rows in a group by clause.

Having clause must be declare after group by 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 minimum salary of each job whose minimum salary is less then 15000?

```
select min(esal),job from emp group by job having min(esal)<15000;
```

order by clause

It is used to filter the rows in a table.

Bydefault it will return the records 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;
```

Integrity Constraints

Constraints are a rules which are applied on a table to achieve accuracy and quality of data.

We have five types of contraitnts.

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.

But it will accept duplicate values.

NOT NULL constraint can be created 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,'ramana','vizag');

commit;
```

NOT NULL constraint can be created for multiple columns also.

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(101,'raja','hyd');

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

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

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

commit;
```

2) UNIQUE

UNIQUE constraint does not accept duplicate values.

But it will 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,'ramana','vizag'); //invalid
```

```
commit;
```

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,'ramana','vizag'); //invalid

commit;
```

UNIQUE constraint can be created for multiple columns also.

ex:

column level

```
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(102,'ravi','delhi');


insert into student values(101,'ramana','vizag'); //invalid


commit;
```

3)Primary key

It is a combination of NOT NULL and UNIQUE constraint.

It does not accept null values and duplicate values

A table can have only one primary key.

primary key can be created at column level and table level.

column level

ex:

```
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,'ramana','vizag'); //invalid
commit;
```

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,'ramana','vizag'); //invalid
commit;
```

Foreign key

It is used to establish the relationship between two tables.

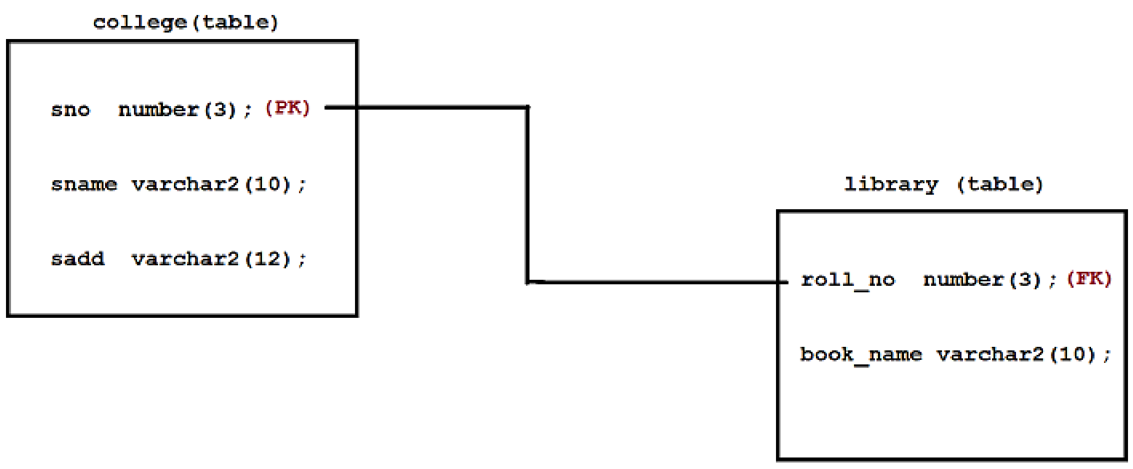
To establish the relationship a parent table must have primary key or unique key and child table must have foreign key.

A foreign key will accept only those values which are present in primary key column.

Foreign key values can be duplicated and it can be null.

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

Diagram: oracle6.1



college table

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

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,'React');

insert into library values(103,'Spring');

insert into library values(null,'HTML');

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

commit;
```

In order to drop the tables , first we need to drop child table then parent table.

ex:

```
        drop table library;

        drop table college;
```

5) check

It is used to check domain of a column.

Here domain means what type of values a column must accept.

Check constraint 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',39);

insert into student values(102,'ravi',104); //invalid

insert into student values(103,'ramana',200);//invalid

commit;
```

ex:

```
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',39);
insert into student values(102,'ravi',104); //invalid
insert into student values(103,'ramana',200);//invalid
commit;
```

ex:

```
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',39);
insert into student values(102,'ravi',104);  //invalid
insert into student values(103,'RaMaNa',200); //invalid
commit;
```

ex:

```
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',39); //invalid

insert into student values(102,'ravi',104);

insert into student values(103,'RaMaNa',200); //invalid

commit;
```

table level

ex:

```
drop table student;

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

insert into student values(101,'RAJA',39); //invalid

insert into student values(102,'ravi',104);

insert into student values(103,'RaMaNa',200); //invalid

commit;
```

Q)Write a query to add the constraint to a existing table?

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

Q)Write a query to drop the constraint from existing table?

```
alter table emp DROP primary key;
```

Q)What is the difference between ROWNUM vs ROWID ?

ROWNUM

ROWNUM values always starts with 1 and increment by 1.

ROWNUM values are temperory.

Once the query is executed we will loss the rownum values.

ex:

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


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

ROWID

ROWID is an address where our records will store in a database table.

ROWID's are permanent.

ex:

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

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

Interview Questions

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 6th record from employee table?

```
select * from emp where rownum<=6
minus
select * from emp where rownum<=5;
```

TCL commands

1) commit

It is used to makes 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;
```

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','vizag');

        insert into student values(104,'rakesh','pune');

        select * from student; // 4 records

        rollback;

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

3) savepoint

It is used to make logical marking in a database.

Instead of complete rollback we can rollback upto savepoint.

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,'john','USA');

        insert into student values(104,'hendry','UK');

        savepoint sp2;

        insert into student values(105,'Jerry','Japan');

        insert into student values(106,'Chan','China');

        select * from student; // 6 records

        rollback to sp2;

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

DCL commands

1) grant

2) revoke

Schema :

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

Privileges

Permissions given to a user is called privileges.

Rights given to a user is called privileges.

We have two types of privileges.

1) system privilege

Permissions given by DBA to user.

2) object privilege

Permissions given by one user to another user.

grant

It is used to grant the permissions to the user.

syntax:

```
grant privilege1,privilege2 to <user_name>;
```

revoke

It is used to revoke the permission from the user.

syntax:

```
revoke privilege1,privilege2 from <user_name>;
```

```
DBA> create user anil identified by anil123;

DBA> create user anjali identified by anjali;

Anil>  conn anil/anil123      // logon denied

Anjali> conn anjali/anjali    // logon denied

DBA> grant connect,resource to anil,anjali;

Anil>  conn anil/anil123

Anjali> conn anjali/anjali

Anil>

create table employee(eid number(3),ename varchar2(10),esal number(8));

insert into employee values(101,'Alan',10000);
insert into employee values(102,'Jose',20000);
insert into employee values(103,'Mark',30000);
commit;

Anjali> select * from employee;  //table or view does not exist
Anil> grant select on employee to anjali;
Anjali> select * from anil.employee;
Anjali> delete from anil.employee; //insufficient privileges
Anil> grant delete ,update on employee to anjali;
Anjali> delete from anil.employee;
Anjali> commit;
Anil> select * from employee;
Anil> revoke select,delete,update on employee from anjali;
Anjali> disc
Anil> disc
DBA> revoke connect,resource from anil,anjali;
```

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

```
        select username from all_users;
```

Q)Write a query to drop the user from database?

```
        drop user anil cascade;
        drop user anjali cascade;
```

Sequences

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

syntax:

create sequence <sequence_name> start with <val> increment by <val>;

ex:

create sequence sq1 start with 10 increment by 10;

create sequence sq2 start with 1 increment by 1;

We have two psudos for sequence.

1) NEXTVAL

It is used to generate the next number is sequence.

ex:

create sequence sq1 start with 101 increment by 1;

drop table student;

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

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;

2) CURRVAL

It is used to return the last number which is generated by sequence.

ex:

select sq1.CURRVAL from dual;

Q)Write a query to see the listen of sequence present in database?

select sequence_name from user_sequences;

Q)Write a query to drop the sequence present in database?

```
        drop sequence sq1;

        drop sequence sq2;
```

Synonyms

Alternate name given to a table is called synonym.

We can use synonym name in case of table name for all the commands.

syntax:

```
        create synonym <synonym_name> for <object_name>;
```

ex:

```
        create synonym sy1 for employee;


        select * from sy1;

        delete from sy1;
```

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

```
  select synonym_name from user_synonyms;
```

Q)Write a query to drop the synonym from database?

```
        drop synonym sy1;
```

Indexes

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

Index in a database is same as index in a book.

We can create index only to those columns which are widely used in where clause.

Whenever we create index , two columns will be created one is ROWID and another is index column . All the records will store in ascending order in a index column.

ex:

<u>INDEX TABLE</u>

ROWID	INDEX COLUMN
	9000
	18000
	19000
	29000
	30000
	49000

Indexes are divided into two types.

1)Simple index

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

syntax:

```
create index  <index_name> ON <object_name>(col_name);
```

ex:

```
create index idx1 ON emp(esal);


select * from emp where eid=201;
select * from emp where esal=19000;
```

Here index is used when we use esal in where clause.

2)Complex index

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

syntax:

```
create index <index_name> on <object_name>(col1,col2,..,colN);
```

ex:

```
create index idx2 on emp(eid,ename);
```

Here index is used when we use eid and ename in where clause.

```
select * from emp where eid=201 and ename='Alan';
```

Indexes are categories into two types.

1) Non-Unique index

If a index contains duplicate values is called non-unique index.

By default every index is a non-unique index.

ex:

```
create index idx3 on emp(deptno);
select * from emp where deptno=10;
```

2) Unique index

If a index contains unique values is called unique index.

ex:

```
	create unique index idx4 on emp(eid);

	select * from emp where eid=204;
```

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

```
	select index_name from user_indexes;
```

Q)Write a query to drop index ?

```
	drop index idx1;
	drop index idx2;
	drop index idx3;
	drop index idx4;
```

Joins

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

	select * from dept; // 4 records

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

	select eid,ename,esal,dname,dloc from emp,dept; // 6 * 4 =24 records

	select eid,ename,esal,deptno,dname,dloc from emp,dept;
	                                        //column ambiguously defined
```

To overcome this above limitation we need to use table_name.column_name.

ex:

```
        select
emp.eid,emp.ename,emp.esal,dept.deptno,dept.dname,dept.dloc
                from emp ,dept ;//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 will be maintained.

ex:

```
        select e.eid,e.ename,e.esal,d.deptno,d.dname,d.dloc
                from emp e,dept d; //24 records
```

Definition

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

We have following types 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.

ex:

```
select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e, dept d
where(e.deptno=d.deptno); // 6
```

2) Non-Equi Join

It is used to retrive the data based on non-equi join condition.

ex:

```
select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e, dept d
where e.esal<35000; // 5 * 4 = 20 records
```

3) Self Join

When table is joined 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

It will return all possible combinations.

ex:

```
select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e, dept
d;//6*4=24 rec
```

5)Inner join

It is similar to equi join.

Inner join is given by ANSI.

ANSI stands for American National Standards 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)Outer Join

It is a extension of equi join.

It will return matching as well as not matching records.

A '+' denoted as outer join operator.

We have following types of outer joins.

i)Left Outer join

SQL

```
select e.eid,e.ename,e.esal,d.dname,d.dloc from
emp e,dept d where(e.deptno=d.deptno(+));
```

ANSI

```
select e.eid,e.ename,e.esal,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,d.dname,d.dloc from
emp e,dept d where(e.deptno(+)=d.deptno);
```

ANSI

```
select e.eid,e.ename,e.esal,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,d.dname,d.dloc from
emp e FULL OUTER JOIN dept d ON(e.deptno=d.deptno);
```

Views

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

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

syntax:

```
create view <view_name> as select stmt;
```

ex:

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

View does not consume any memory.

View does not contains any data.

Whenever we write select statement then view will get the data from above table or base table.

We have different types of views.

1)Simple view

2)Complex view

3)With ready 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;

        create view v1 as select eid,ename,esal from emp;

        create view v1 as select * from where deptno=10;

        create view v1 as select * from where job<>='Manager';

        create view v1 as select * from emp where eid IN(201,202,203);

        create view v1 as select * from emp where ename like 'A%';

        create view v1 as select * from emp where esal between 5000 AND
50000;
```

DML operations are allowed in simple view.

ex:

```
        delete from v1;

        select * from v1; //no rows selected

        select * from emp; // no rows selected

        rollback;
```

2)Complex view

If a view is created by using more then one base 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 where(e.deptno=d.deptno);
```

```
select * from v2; // 6 records
```

DML operations are not allowed in complex view.

```
delete from v2;  //cannot delete from view
```

3)With ready only view

If we want to create a view by using one base table and DML operations are not allowed then we need to use with ready only view.

ex:

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



select * from v3; // 6 records display


delete from v3; // cannot delete from view
```

4)With check option view

If we want to create a view by using one base table and DML operation is allowed only if condition is satisfied then we need to use with check option value.

ex:

```
        create view v4 as select * from emp where deptno=10 with check
option;
```

```
insert into v4 values(208,'Chan',20000,60,'Salesman',200);
                                        //view WITH CHECK OPTION

insert into v4 values(208,'Chan',20000,10,'Salesman',200);

select * from v4;

select * from v4;
```

5) Materialized view

To create a materialized view a table must have primary key or unique key.

Materialized view is also known as snapshot.

ex:

```
alter table emp add primary key(eid);

create materialized view v5 as select * from emp;

select * from v5;

delete from emp where eid in (207,208);
commit;

select * from v5;
```

To get the changes in materialized view , we need to referesh the view.

ex:

```
exec DBMS_SNAPSHOT.REFRESH('v5');
```

Q)Write a query to see the list of views present in 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;
```

Sub Queries

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

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

ex:

Normal query :

```
select * from emp where eid=201;
```

sub query :

```
        select * from emp where eid=(select eid from emp where
ename='Alan');
```

Here first inner query will be executed then outer query.

Sub queries can be nested upto 32 levels.

Sub queries are different types.

1) Single row sub query

2) Multiple row sub query

3) Multiple column sub query

1) Single row sub query

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

ex:

```
        select * from emp where eid=(select eid from emp where
ename='Alan');
```

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

```
        select max(esal) from emp where esal<(select max(esal) from emp);
```

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

```
        select * from emp where esal>(select esal from emp where
ename='Lisa');
```

2) Multiple row sub query

When subquery 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.

1)ALL

2)ANY

3)IN

2)Multiple Row sub query

If 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.

1) ANY

2) ALL

3) IN

ex:

```
    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);
```

ex:

```
    select * from emp where esal > ALL (select esal from emp where
deptno=10);
```

ex:

```
    select * from emp where esal IN (select esal from emp where
deptno=10);
```

3)Multiple Column Sub query

If sub query returns more then one column then we need to use multiple column sub query.

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);
```

Merge Command

Merge command is a combination of insert and update command.

ex:

```
drop table student10;

create table student10(sno number(3),sname varchar2(10),smarks
number(3));

insert into student10 values(101,'raja',50);

insert into student10 values(102,'ravi',70);

insert into student10 values(103,'ramana',56);

commit;
```

ex:

```
drop table student20;

create table student20(sno number(3),sname varchar2(10),smarks
number(3));

insert into student20 values(101,'Jose',98);

insert into student20 values(104,'Lisa',82);

commit;
```

```
merge into student10 s1

using student20 s2

ON(s1.sno=s2.sno)

when matched

then update set sname=s2.sname,smarks=s2.smarks

when not matched

then insert(sno,sname,smarks) values(s2.sno,s2.sname,s2.smarks);
```

```
select * from student10;

select * from student20;
```

PL/SQL

PL/SQL stands for procedural or Structured Query Language.

It is a extension of SQL and gives following features.

1) We can achieve programming features like loops, control statements and etc.

2) It reduces network traffic.

3) We can perform related operations by using the concept of triggers.

4) It will save the source code permenently to database for repeated execution.

5) We can display our own exception messages by using the concept of exception handling.

PL/SQL Block

A PL/SQL program is also known as PL/SQL block.

syntax:

```
DECLARE

-

-        // Declaration section

-

BEGIN

-

-        // Executable section

-

EXCEPTION
```

```
-
-        // Exception section
-
END;
/
```

Declaration section

Declaration section is used to declare variables, exceptions , cursors and etc.

It is optional section.

Executable section

It contains actual logic which is used to complete a table.

It is mandatory section.

EXCEPTION

This section will execute when exception raised in our program.

It is optional section.

To see the output in PL/SQL we need to use below command.

ex:

```
        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.

Here 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

```
DECLARE
A number:=10;
B number:=20;
C number:=A+B;
BEGIN
DBMS_OUTPUT.PUT_LINE('sum of two numbers is '||C);
END;
/
```

Using '&' symbol we can read inputs at runtime.

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 we can perform DML operations.

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 PL/SQL program to update student name based on student number?

```
DECLARE
L_Sno number(3);
BEGIN
L_Sno:=&sno;
update student set sname='gogo' 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 select operations also.

To perform select operation in PL/SQL we need to use "into" clause.

ex:

Q)Write a PL/SQL program to display employee name whose employee id is 201?

```
DECLARE
L_ename varchar2(10);
BEGIN
select ename into L_ename from emp where eid=201;
DBMS_OUTPUT.PUT_LINE(L_ename);
END;
/
```

Q)Write a PL/SQL program to display employee name,employee salary based on employee id?

```
DECLARE
L_ename varchar2(10)
L_esal  number(10,2);
L_eid   number(3);
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 local variable with respect to the column type.

syntax:

```
variable_name   table_name.column_name%TYPE;
```

ex:

```
A   emp.eid%TYPE;
```

Q)Write a PL/SQL program to display employee name,employee salary based on employee id?

```
DECLARE

L_ename emp.ename%TYPE;

L_esal  emp.esal%TYPE;

L_eid   emp.eid%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 respect to complete row of a table.

Rowtype variable we can't display directly.

In order to access the data from rowtype variable we need to use variable_name.col_name.

syntax:

```
variable_name   emp%ROWTYPE;
```

ex:

```
A   emp%ROWTYPE;
```

Q)Write a PL/SQL program to display employee information based on employee Id?

```
DECLARE

A emp%ROWTYPE;

L_eid emp.eid%TYPE;

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);

END;
/
```

To see the output in PL/SQL we need to use below command.

ex:

```
SQL> set  serveroutput  on
```

Control Statements

1)IF THEN

It will evaluate the code only if our condition is true.

ex:

```
        DECLARE
        A number:=5;
        BEGIN

        IF A>2 THEN
        DBMS_OUTPUT.PUT_LINE('Welcome');
        END IF;

        END;
        /
```

ex:

```
        DECLARE
        A number:=5;
        BEGIN

        IF A>20 THEN
        DBMS_OUTPUT.PUT_LINE('Welcome');
        END IF;

        END;
        /
```

2) IF THEN ELSE

It will evaluate the code either our condition is true or false.

ex:

```
DECLARE
A number:=25;
BEGIN

IF A<18 THEN
DBMS_OUTPUT.PUT_LINE('You r not eligible to vote');
ELSE
DBMS_OUTPUT.PUT_LINE('You r eligible to vote');
END IF;

END;
/
```

ex:

```
DECLARE
A number:=5;
BEGIN

IF A<18 THEN
DBMS_OUTPUT.PUT_LINE('You r not eligible to vote');
ELSE
DBMS_OUTPUT.PUT_LINE('You r eligible to vote');
END IF;

END;
```

```
/
```

3) IF THEN ELSIF THEN ELSE

It will evaluate the code based on multiple conditions.

ex:

```
DECLARE
n number(3);
BEGIN
n:=&opt;

IF n=100 THEN
DBMS_OUTPUT.PUT_LINE('It is police number');
ELSIF n=103 THEN
DBMS_OUTPUT.PUT_LINE('It is enquiry number');
ELSIF  n=108 THEN
DBMS_OUTPUT.PUT_LINE('It is emergency number');
ELSE
DBMS_OUTPUT.PUT_LINE('Invalid optionn');
END IF;

END;
/
```

LOOPS

There are three types of LOOPS in PL/SQL.

1) Simple Loop

2) While Loop

3) For Loop

1) Simple Loop

It will evaluate the code untill 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 untill 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('Thank You');
END;
/
```

ex:

```
DECLARE
A number:=10;
BEGIN
```

```
WHILE A>=1 LOOP
DBMS_OUTPUT.PUT_LINE(A);
A:=A-1;
END LOOP;

END;
/
```

2) For Loop

It will evaluate the code untill 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('Thank You');
END;
/
```

ex:

```
DECLARE
A number;
```

```
N number:=5;
BEGIN

FOR A IN 1 .. 10 LOOP
DBMS_OUTPUT.PUT_LINE(N||' * '||A||' = '||N*A);
END LOOP;

END;
/
```

Exceptions

Runtime Errors are called Exceptions.

In PL/SQL exceptions are divided into two types.

1)Predefined Exceptions

2)Userdefined Exceptions

1)Predefined Exceptions

Built-In exceptions are called predefined exceptions.

We have following list of predefined exceptions.

1)NO_DATA_FOUND

2)TOO_MANY_ROWS

3)ZERO_DIVIDE

4)VALUE_ERROR

5)DUP_VAL_ON_INDEX

6)OTHERS

1)NO_DATA_FOUND

This exception will occur when select statement does not return any row.

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 the employee ID');
END;
/
```

2)TOO_MANY_ROWS

This exception will occur 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=10;
DBMS_OUTPUT.PUT_LINE(L_ename);
EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('select stmt returns more then one row');
END;
```

```
/
```

3)ZERO_DIVIDE

This exception will occur when we are trying to divide any 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('Please dont divide number with zero');
END;
/
```

4) VALUE_ERROR

This exception will occur 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=201;
    DBMS_OUTPUT.PUT_LINE(L_esal);
    EXCEPTION
    WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Please check the datatype');
    END;
    /
```

5)DUP_VAL_ON_INDEX

This exception will occur whenever we trying to insert duplicate value in a primary key.

ex:

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


        BEGIN
        insert into emp values(202,'Chan',34000,50,'Salesman',100);
        DBMS_OUTPUT.PUT_LINE('Record Inserted');
        Exception
        WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Duplicates are not allowed');
        END;
        /
```

6) OTHERS

It is a universal angular exception which handles 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 the employee ID');
    END;
    /
```

2)Userdefined Exceptions

Exceptions which are created by the user based on the application requirement are called userdefined exceptions.

steps to create userdefined 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
SAL number:=5000;
MY_EXP EXCEPTION ;
BEGIN

IF SAL>2000 THEN
RAISE MY_EXP;
END IF;
DBMS_OUTPUT.PUT_LINE(SAL);

EXCEPTION
WHEN MY_EXP THEN
DBMS_OUTPUT.PUT_LINE('Salary is too high');
END;
/
```

Cursors

Cursor is a memory location which is used to run SQL commands.

We have two types of cursors in PL/SQL.

1)Implicit cursor

2)Explicit cursor

1)Implicit cursor

All the operations related to cursor like opening the cursor , processing the cursor and closing the cursor which is done automatically is called implicit cursor.

Implicit cursor contains four types of cursor attributes.

i) SQL%ISOPEN

It is a boolean attribute which always returns 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 reverse 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 values effected in a database table.

SQL%ISOPEN

```
BEGIN
```

```
update student set sname='jojo' where sno=101;
IF SQL%ISOPEN THEN
DBMS_OUTPUT.PUT_LINE('cursor is open');
ELSE
DBMS_OUTPUT.PUT_LINE('cursor is closed');
END IF;
END;
/
```

ii) SQL%FOUND

```
BEGIN
update emp set ename='AntMan' where eid=201;
IF SQL%FOUND THEN
DBMS_OUTPUT.PUT_LINE('Record updated');
ELSE
DBMS_OUTPUT.PUT_LINE('Record not updated');
END IF;
END;
/
```

iii) SQL%NOTFOUND

```
BEGIN
update emp set ename='IronMan' where eid=201;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('Record updated');
ELSE
DBMS_OUTPUT.PUT_LINE('Record not updated');
END IF;
END;
```

```
/
```

iv) SQL%ROWCOUNT

```
BEGIN
delete from student;
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' Records Deleted');
END;
/
```

To see the output in PL/SQL we need to use below command.

ex:

```
SQL> set serveroutput on
```

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 user is called explicit cursor.

We need to use explicit cursor when select statement returns more then one value.

We have four types 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 are effecting in a database table.

Q)Write a PL/SQL program to display employee name and employee salary from emp table?

```
DECLARE
CURSOR C1 is select ename,esal from emp;
L_ename emp.ename%TYPE;
L_esal emp.esal%TYPE;
BEGIN
OPEN C1;
LOOP
FETCH C1 into L_ename,L_esal;
DBMS_OUTPUT.PUT_LINE(L_ename||' '||L_esal);
EXIT WHEN C1%NOTFOUND;
END LOOP;
CLOSE C1;
END;
/
```

Q)Write a PL/SQL program to display employee information from emp table?

```
DECLARE
CURSOR C2 is select * from emp;
A emp%ROWTYPE;
BEGIN
OPEN C2;
LOOP
FETCH C2 into A;
DBMS_OUTPUT.PUT_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno);
EXIT WHEN C2%NOTFOUND;
END LOOP;
CLOSE C2;
END;
/
```

PL/SQL procedures

PL/SQL procedures are also known as stored PL/SQL procedures.

It is a named PL/SQL block which is compiled and execute in a database software for repeated execution.

syntax:

```
create or replace procedure <procedure_name>
is
begin
-
-
-
end;
```

```
/
```

Q)Write a pl/sql 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 may contains three parameters.

1)IN parameter

2)OUT parameter

3)IN OUT parameter

1)IN parameter

It is used to accept the values from the user.

Q)Write a PL/SQL program 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;
        /
```

To execute the procedure we need to use below command.

ex:

```
        exec  sum(10,20);
```

2)OUT parameter

It is used to return the value to the user.

Q)Write a PL/SQL program 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 the procedure.

ex:

```
        exec  ret_sum(10,20,:N);
```

step3:

Print bind variable.

ex:

```
        print N;
```

<u>3)IN OUT parameter</u>

It is used to accept the value from the user and it will return the value to the user.

Q)Write a PL/SQL program to display square of a given number?

```
create or replace procedure square(N IN OUT number)
is
begin
N:=N*N;
END;
/
```

<u>Steps to call a procedure having IN OUT parameter</u>

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  square(:N);
```

step4:

Print bind variable.

ex:

```
          print N;
```

PL/SQL procedures can perform DML operations.

Q)Write a PL/SQL program to delete student record based on student number?

```
     create or replace procedure delete_student(L_sno IN
student.sno%TYPE)
     is
     begin
     delete from student where sno=L_sno;
     DBMS_OUTPUT.PUT_LINE('Record Deleted');
     END;
     /
```

We can execute the procedure as follow.

ex:

```
exec  delete_student(103);
```

PL/SQL functions

A PL/SQL function is a 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 PL/SQL 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;

/
```

We can execute the function as follow.

ex:

```
select   f1(10,20) from dual;
```

Q)Write a PL/SQL program to accept one salary and find out 10% of TDS?

```
create or replace function find_tds(SAL number)

return number

is

TAX number;

BEGIN

TAX:=SAL*10/100;

RETURN TAX;

END;

/
```

We can call function as follow.

ex:

```
select find_tds(5000) from dual;

select eid,ename,esal,find_tds(esal) from emp;
```

Note:

DML operations are not allowed in Functions .

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 stmt.	Can be invoked by using select stmt.

Note:

PL/SQL procedures and functions are called logical related sub programs.

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

```
select object_name from user_objects where object_type='PROCEDURE';
```

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

```
select object_name from user_objects where object_type='FUNCTION';
```

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 see the source code of a function?

```
select text from user_source where name='SUM';
```

Q)WRite a query to drop the procedure from database?

```
drop procedure p1;

drop procedure sum;

drop procedure ret_square;
```

Q)WRite a query to drop the function from database?

```
drop function F1;

drop function find_tds;
```

PL/SQL packages

PL/SQL package is a collection of logical related sub programs.

PL/SQL package is a collection of procedures and functions.

Package creation involve in two things.

1) Package specification

It is a declaration of logical related sub programs.

2) Package body

It contains definition to logical related sub programs.

ex:1

package specification

```
create or replace package pkg1

is

procedure p1(A IN number,B IN number);

end pkg1;

/
```

package body

```
create or replace package body pkg1

is

procedure p1(A IN number,B IN number)

is

C number;

begin

C:=A+B;

DBMS_OUTPUT.PUT_LINE(C);

END;

end pkg1;

/
```

We can execute the procedure as follow.

ex:

```
exec  pkg1.p1(10,20);
```

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;
/
```

We can invoke the function as follow.

ex:

```
select  pkg2.f1(10,20) from dual;
```

Q)Write a query to see the 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 from database?

```
drop package body pkg1;
drop package pkg1;


drop package body pkg2;

drop package pkg2;
```

Triggers

Triggers are PL/SQL blocks which are execute based on the events.

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_name
is
begin
-
-
-
end;
```

ex:

```
create or replace trigger trg1 before delete on student
begin
DBMS_OUTPUT.PUT_LINE('Yahoo! You deleted');
END;
/

insert into student values(103,'ramana','vizag');
```

```
delete from student where sno=103; // trigger will occur
```

ex:

```
create or replace trigger trg2 after update or insert on student
begin
DBMS_OUTPUT.PUT_LINE('Yahoo! trigger executed');
END;
/

insert into student values(103,'ramana','vizag'); // trigger will occur

update student set sname='gogo' where sno=101; // trigger will occur
```

Every trigger is classified into two types.

1)Statement level trigger

If trigger is executed only for one time is called statement level trigger.

By default every trigger is a statement level trigger.

ex:

```
create or replace trigger trg3 after delete on student
begin
DBMS_OUTPUT.PUT_LINE('Yes, Deleted');
END;
/

delete from student; // trigger will execute only for one time.
```

2)Row level trigger

If a trigger will execute irrespective of number of records effected in a database table is called row level trigger.

We can create row level trigger by using "for each row" clause.

ex:

```
create or replace trigger trg3 after delete on student for each row
begin
DBMS_OUTPUT.PUT_LINE('Yes, Deleted');
END;
/

delete from student; // trigger will execute only for one time.
```

Q)Write a query to drop the triggers?

```
drop trigger trg1;
drop trigger trg2;
drop trigger trg3;
```

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

```
select object_name from user_objects where object_type='TRIGGER';
```