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 is called unstructured data.

In general, meaningless data is called unstructured data.

ex:

201 Lakemba SYD NSW AUS

2)Structured Data

Data which is in readable format is called structured data.

In general, meaningfull 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 (Structured Query Language)

PL/SQL (Procedural /Structured Query Language)

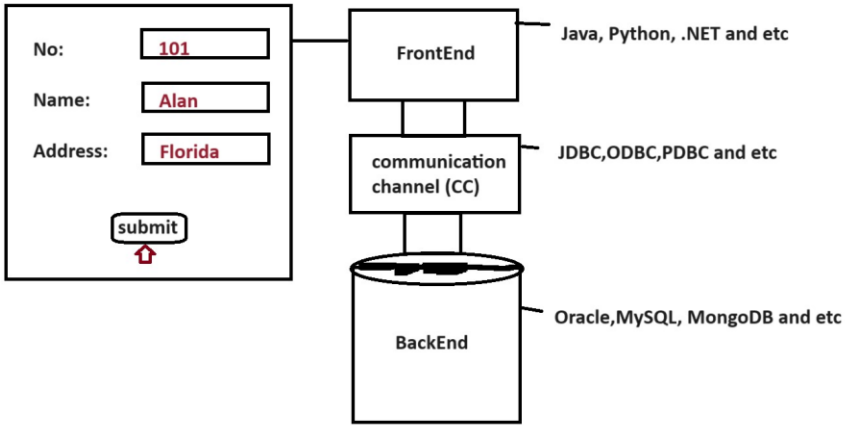
We have following list of oracle versions like 10g , 11g , 12c and 13i.

Client/Server Architecture

==========

In this architecture we will see ,how our data will store from frontend to backend.

Diagram: oracle1.1



FrontEnd

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

frontend.

ex:

Java, Python, .net, 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.

Management System

==========

It 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) Deleting the unnecessary data

4) Selecting the required data

DBMS

=====

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

database management system.

Lots of drawbacks of DBMS

> Single user

> Does not support Normalization

> It does allow any relationships(parent table and child table)

> waste of memory

RDBMS

======

If a database is created based on relational theories to manage the database is

called relational database management system.

ex:

Oracle, MySQL , SQL Server, Sybase, Teradata 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 case insensitive language.

ex:

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

It is a command based language.

Every command must starts with verbs like
select,update,delete,merge and etc.

Every command ends with semicolon(;).

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 Manipulation Language)

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

4) TCL (Transaction Control Language)

5) DCL (Data Control Language)

1) DDL

It is used to maintain the objects in database.

It is a collection of five commands.

ex:

create,alter,drop,truncate and rename

2) DML

This language is used to manipulate the present in database.

It is a collection of four commands.

ex:

insert,update,delete and merge

3) DRL/DQL

This language is used to retrieve the data from database.

It is a collection of one command.

ex:

select

4) TCL

This language is used to maintain the transaction of database.

It is a collection of three commands.

ex:

commit,rollback and savepoint.

5) DCL

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

It is a collection of two commands.

ex:

grant and revoke

Oracle

=======

Version : 10g or 11g

Vendor : Oracle Corporation

Creator : Mr. Codd

Port No : 1521

website : www.oracle.com/in/database

username : system (default)

password : admin

Download link :

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

Establish the connection with database software

==========

To perform or execute any query in database we need to establish the connection

with database software.

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

1)

SQL> connect

Username : system

Password : admin

SQL> disconnect

2)

SQL> conn

username : system

password : admin

SQL> disc

3)

SQL> conn system/admin

SQL> disc

Table

=======

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

A table is used to store the data in the form of rows and columns.

ex:

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

In above table we have 3 rows and 3 columns.

Oracle is a case insensitive but data which is present in a table is a case sensitive.

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(12));

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 see the structure of a table.

syntax:

desc table_name;

ex:

desc emp;

desc dept;

desc student;

Insert command

==========

It is used to insert a record/row in a table.

syntax:

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

ex:

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

insert into student values(102,'ravi'); //invalid not enough values

null

---

null is a operator which represent undefined or unavailable.

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

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 values.

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(12));

insert into dept values(10,'ECE','HYD');

insert into dept values(20,'EEE','PUNE');

insert into dept values(30,'CSE','DELHI');

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(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',29000,20,'HR',900);

insert into emp values(204,'Nelson',49000,20,'HR',500);

insert into emp values(205,'Lisa',23000,30,'Manager',800);

insert into emp values(206,'Brenda',36000,30,'Manager',600);

commit;

select command

==========

It is used to retrieve the records from database table.

syntax:

select * from <table_name>;

Here '*' means all columns and rows.

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 dname from dept;

In select command we can perform arithmetic operations also.

ex:

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

column alias

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

Column alias are temperory.

Once the query is executed we will loss the column alias.

Column alias we can applied to any column.

ex:

```
select sno-100 as SNO,sname,sadd from student;

select sno as rollno,
                sname as name,
                        sadd as city from student;
```

Interview Queries

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

```
select * from emp;
```

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

```
select * from global_name; // 10g --> XE
                                    // 11g --> ORCL
```

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

```
select * from tab;
```

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 as ANNUAL_SAL and

annual salary from emp table?

```
select eid,ename,esal,esal*12 as ANNUAL_SAL from emp;
```

where clause

==========

It is used to select specific rows from database table.

syntax:

select * from <table_name> where condition;

ex:

select * from student where sno=101;

select * from student where sname='ravi';

select * from student where sadd='pune';

is null

---

is null operator is used to select the records based on null values.

ex:

select * from student where sadd=null;//no rows selected

select * from student where sadd is null;

Interview Queries

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

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

Q) Write a query to display employees information those are working as Manager?

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

Q) Write a query to display employees information whose salary is greater then 25000?

```
select * from emp where esal>25000;
```

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

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

Q) Write a query to display employee id ,employee name, employee salary whose

commission is null?

```
select eid,ename,esal from emp where comm is null;
```

update command

==========

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

syntax:

---

update <table_name> set <col_name> where condition;

ex:

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

update student set sname='alan' where sadd='hyd';

update student set sno=501,sname='jojo' where sadd is null;

Note:

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

update student set sno=101;

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='ramulu';

delete from student where sadd='vizag';
```

Note:

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

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

Note:

All DML commands are temperory.

Interview Queries

Q)Write a query to terminate all the employees whose salary is greater then 40000?

```
delete from emp where esal>40000;
```

Q)Write a query to promote all employees from clerk to salesman?

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

Logical Operators

==========

Logical operators are used to declare more then one condition in a query.

We have three types of logical operators.

1) AND

2) OR

3) NOT

1) AND

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

Here both 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='Nelson'; // No rows selected

select * from emp where eid=209 AND ename='jojo'; // No rows selected

2) OR

It will return the records only if one condition is true.

Here both conditions can be from any row.

ex:

```
select * from emp where eid=201 OR ename='Alan';    // 1 records

select * from emp where eid=201 OR ename='Nelson';  // 2 records

select * from emp where eid=209 OR ename='jojo';    // no rows selected
```

3) NOT

It will 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 dept where NOT deptno=10;

select * from student where sadd<>'hyd';
```

Interview Queries

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

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

Q) Write a query to display employees information whose salary is greater then 20000 and

less then 40000?

```
select * from emp where esal>20000 AND esal<40000;
```

Q)Write a query to display employees information who are not working in 20 department?

```
select * from emp where deptno<>20;
```

BETWEEN operator

==========

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

In between operator first we need to write lower limit then higher limit.

Between operator we can use only for numbers.

Between operator will take the support of AND operator.

ex:

```
select * from emp where eid between 201 AND 206;

select * from emp where deptno between 10 AND 30;

select * from emp where esal between 20000 AND 40000;
```

IN operator

==========

IN operator is a replacement of OR operator.

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

ex:

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

select * from emp where ename IN('Alan','Nelson','Ana');

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

Pattern Matching Operators

==========

Pattern matching operators are used to select the letters 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 name starts with 'A' letter?

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

Q)Writea query to display employees information whose name ends with 'n' letter?

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

Q)Write a query to display employees information whose name having middle letter as 'l'?

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

2) Underscore(_)

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

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

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

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

Q)Write a query to display employees information whose employee name having third

letter as 's' ?

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

Duplicate table or Copy of table

==========

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

If something goes wrong then we can recover the situation by using duplicate table.

syntax:

```
create table <table_name> as select stmt;
```

ex:

```
create table employee as select * from emp;
```

create table employee 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 job<>'Manager';

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

create table employee as select * from emp where esal between 10000 AND 30000;

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

cl scr

=======

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

ex:

cl scr

DDL commands

==========

1) create (tables)

2) alter (columns)

3) drop (tables)

4) truncate (rows/records)

5) rename (tables)

2) alter command

Using alter command we can perform following activities very easily.

i)Adding the new column

ii)Modifying the existing column

iii)Renaming a column

iv)Dropping a exisitng column

i)Adding the new column

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

syntax:

alter table table_name ADD (col datatype(size));

ex:

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

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

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

ii)Modifying the existing column

Using alter command we can modify the existing column.

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 if that column is empty.

ex:

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

iii)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;

iv) Dropping a existing columns

Using alter command we can drop existing columns.

syntax:

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

ex:

alter table student drop (state,pincode);

3) drop command

A drop command is used to drop the tables from database.

syntax:

drop table <table_name>;

ex:

drop table emp;

drop table dept;

drop table student;

4) truncate

It is used to delete the records permanently 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 is used to delete the records temperory.	It is used to 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 use.

5) rename

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;

Functions

==========

Functions are used to manipulate the data items and gives 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(exp).

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 employee table?

select max(esal) from emp;

Q) Write a query to display lowest salary from employee table?

select min(esal) from emp;

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

count(*)

It will return number of records present in database table.

It will include null records also.

ex:

select count(*) from student;

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

count(exp)

It will return number of values present in datable table column.

It won't include null values.

ex:

```
select count(esal) from emp;//6

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

Userlist table

==========

```
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 username and password is valid or invalid?

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

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

Dual table

==========

A dual table is a dummy table which consist one row and one column.

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

ex:

select 10+20 from dual;

```
select 10*20-100 from dual;

select sysdate from dual;

select current_date from dual;
```

2)Scalar Functions

==========

Scalar functions are applicable for single row.

We have four types of scalar functions.

i) Character functions

ii) Number functions

iii) Date functions

iv) Conversion functions

i) Character functions

---

upper()

It will convert lowercase to uppercase.

ex:

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

lower()

It will convert uppercase to lowercase.

ex:

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

initcap()

It will display the output in initial letter capital.

ex:

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

lpad()

It is used to pad the characters towards left side.

ex:

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

rpad()

It is used to pad the characters towards 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;

rtrim()

It is used to trim the characters from right side.

ex:

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

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 concate the output.

ex:

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

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

Note:

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

select eid, upper(ename) as ENAME, esal, lower(job) as JOB from emp;

ii) Number functions

---

abs()

It will return absoluate value.

ex:

select abs(-20) from dual; // 20

select abs(-10.5) from dual; //10.5

select abs(50) from dual; //50

sqrt()

It will return square root value.

ex:

select sqrt(25) from dual; // 5

select sqrt(36) from dual; //6

select sqrt(37) from dual; //6.08

power(A,B)

It will return power value.

ex:

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

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

ceil()

It will return ceil value.

ex:

select ceil(10.5) from dual;//11

select ceil(8.3) from dual; //9

floor()

It will return floor value.

ex:

select floor(10.5) from dual;//10

select floor(8.3) from dual; //8

round()

It will take nearest value.

ex:

select round(10.5) from dual; // 11

select round(10.4) from dual; // 10

trunc()

It is used to remove the decimals.

ex:

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

select trunc(-19.56) from dual; // -19
```

greatest()

It will display greatest value.

ex:

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

least()

It will display least value.

ex:

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

Working with Date values

==========

Every database software supports different date patterns.

We need to insert date value in that format which is supported by underlying database software.

ex:

Oracle ---> dd-MMM-yy

MySQL ---> yyyy-MM-dd

and etc.

```
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,'Nelson',current_date);

insert into emp1 values(304,'Jack','15-JAN-21');
```

commit;

iii)Date functions

ADD_MONTHS()

We can add the months in a given date.

ex:

select ADD_MONTHS(sysdate,5) from dual; //21-APR-24

MONTHS_BETWEEN()

It will return exact months between two given dates.

ex:

select MONTHS_BETWEEN('01-JAN-23','01-NOV-23') from dual; //-10

select abs(MONTHS_BETWEEN('01-JAN-23','01-NOV-23')) from dual; //10

select abs(MONTHS_BETWEEN('01-JAN-23','21-NOV-23')) from dual; //10.64

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('16-DEC-23') from dual;

iv) Conversion functions

Convertion function is used to convert from one type to another type.

ex:

TO_CHAR()

TO_CHAR() function having two pseudos

a) number to_char()

It will acccept 9 in digits and '$' or Euros 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;
```

b) date to_char()

---

ex:

```
select TO_CHAR(sysdate,'dd-MM-yyyy') from dual;

select TO_CHAR(sysdate,'yyyy-MM-dd') from dual;

select TO_CHAR(sysdate,'day') from dual; //tuesday

select TO_CHAR(sysdate,'dy') from dual;//tue

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

select TO_CHAR(sysdate,'mon') from dual;//nov

select TO_CHAR(sysdate,'year') from dual; //twenty
twenty three

select TO_CHAR(sysdate,'dd') from dual; //21
```

```
select TO_CHAR(sysdate,'mm') from dual;//11

select TO_CHAR(sysdate,'yyyy') from dual; //2023

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

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

Group by clause

==========

It will divide the rows into multiple groups so that we can apply group functions.

In group by clause we can use same column name which we used in select 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 average salary of each job?

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

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

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

Q)Write a query to display highest salary of each deparment except 10 department?

select max(esal),deptno from emp where deptno<>10 group by deptno;

Having clause

==========

It is used to filter the rows from group by clause.

Having clause we need to use after group by clause.

Q)Write a query to display sum of salary of each deparment whose sum of salary is greater then

45000?

select sum(esal),deptno from emp group by deptno having sum(esal)>45000;

Q)Write a query to display maximum salary of each job whose maximum salary is less then

30000?

select max(esal),job from emp group by job having max(esal)<30000;

Order by clause

==========

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

By default it will arrange the records in ascending order.

ex:

```
select * from emp order by eid;

select * from emp order by eid desc;

select * from emp order by esal;

select * from emp order by ename;
```

Integrity Constraints

==========

A constraint is a rule which is applied on a table.

Using constraint we can achieve accuracy and quality of data.

We have five constraints in SQL.

1) NOT NULL

2) UNIQUE

3) PRIMARY KEY

4) FOREIGN KEY

5) CHECK

Constraint 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(101,'ravi','delhi');
insert into student values(null,'ramana','vizag'); //invalid
commit;
```

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

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

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

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

commit;

2) UNIQUE

UNIQUE 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(101,'ravi','delhi'); //invalid
insert into student values(null,'ramana','vizag');
commit;
```

Note:

UNIQUE constriant can be created for 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','vizag'); //invalid

insert into student values(103,'ramana','hyd'); //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(101,'ravi','delhi'); //invalid

insert into student values(null,'ramana','vizag');

commit;

3) PRIMARY KEY

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

PRIMARY KEY will not accept duplicates and null 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(101,'ravi','delhi');//invalid

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

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

4) FOREIGN KEY

A foreign key is used to establish the relationship between two tables.

This relationship is called parent and child relationship or master and detailed relationship.

To establish the relationship between two tables.A parent table must have

primary key or unique constraint and child table must have foreign key.

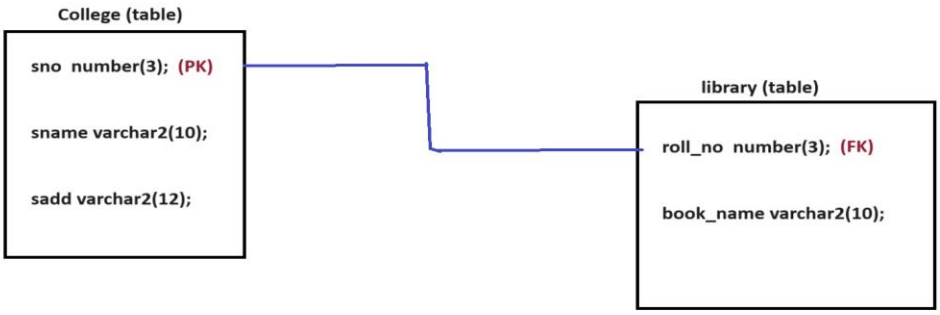
A foreign key will accept only those values which are present in primary key.

A primary key column name and foreign key column name may or may not match but

datatype must match.

A foreign key will accept duplicates and null.

Diagram: oracle6.1



college table (parent 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 (child 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,'angular');
insert into library values(104,'java'); //invalid
```

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

ex:

drop table library;

drop table college;

Q)How can we add constraint in a existing table?

alter table emp ADD primary key(eid);

Q)How can we remove constraint from existing table?

alter table emp DROP primary key;

5) CHECK

CHECK constraint describes domain of a column.

Here domain means what type of value 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',89);
insert into student values(102,'ravi',100);
insert into student values(103,'ramana',200);//invalid
insert into student values(104,'ramulu',999);//invalid
```

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',89);
insert into student values(102,'ravi',100);
insert into student values(103,'ramana',200);//invalid
insert into student values(104,'ramulu',999);//invalid
```

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',89); //invalid
insert into student values(102,'RAVI',100); //valid
insert into student values(103,'RaMaNa',200); //invalid
```

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',89); //valid
insert into student values(102,'RAVI',100); //Invalid
insert into student values(103,'RaMaNa',200); //invalid
```

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',89); //valid
insert into student values(102,'RAVI',100); //Invalid
insert into student values(103,'RaMaNa',200); //invalid
```

Pseudo columns

==========

Pseudo column means a column which is not real.

We have two pseudo columns.

1) ROWNUM

2) ROWID

1) ROWNUM

ROWNUM values always starts with 1 and increment by 1.

ROWNUM values are temperory.

ex:

```
select eid,ename,esal from emp;
select rownum,eid,ename,esal from emp;
select rownum,sno,sname,sadd from student;
```

2) ROWID

ROWID is a memory location where our records will store in a database table.

ROWID is permanent.

ex:

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

Q)Write a query to display first records from emp table?

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

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

```
select * from emp where rownum<=4
minus
```

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

Q)Write a query to display 6th record from emp table?

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

TCL commands

==========

commit

rollback

savepoint

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
```

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,'Jacky','Florida');
insert into student values(104,'James','Texas');

select * from student; // 4 records

rollback;

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

savepoint

It is used to maintain the logical marking in a database.

Instead of complete rollback we can rollback upto save point.

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,'Jacky','Florida');
insert into student values(104,'James','Texas');

savepoint sp2;

insert into student values(105,'Nelson','USA');
insert into student values(106,'Kelvin','UK');

select * from student; // 6 records

rollback to sp2;
```

select * from student; // 4 records

rollback to sp1;

select * from student; // 2 records

DCL commands

==========

grant

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>;

Revoke

It is usd to revoke the permissions from user.

syntax:

revoke <privilege1>,<privilege2> from <user>;

Sequences

==========

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 10 increment by 10;

create sequence s3 start with 201 increment by 1;

Sequence contains two pseudo's.

1) NEXTVAL

It is used to generate next number from 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;
select * from student;
```

2) CURRVAL

It will return the last number generated by sequence.

ex:

```
select sq1.CURRVAL from dual;
```

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

```
select sequence_name from user_sequences;
```

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

```
drop sequence sq1;
```

Synonyms

==========

Alternate name given to a table is called synonym.

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

Synonyms are used to reduce the length of the query.

syntax:

create synonym <synonym_name> for <object_name>;

ex:

create synonym stud for student;

select * from student;

select * from stud;

delete from stud;

select * from student; // no rows selected

select * from stud; // no rows selected

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 stud;
```

Indexes

=========

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

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 , we will get two columns one is rowid and second one is

indexed column. All the records will store in ascending order in indexed column.

ex:

Indexed table

ROWID | INDEXED column

| 9000

| 19000

| 23000

| 29000

| 36000

| 49000

We have two types of indexes.

1) Simple index

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

ex:

create index idx1 ON emp(esal);

select * from emp where esal=23000;

Here index will be used when we use esal in where clause.

2) Complex index

If index is created for more then one column is called complex index.

ex:

create index idx2 ON emp(eid,deptno);

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

Here index will be used when we use eid and deptno in where clause.

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 indexes from database?

drop index idx1;

drop index idx2;

Note:

By default every index is a non-unique index.

If we want unique index then we need to use below command.

ex:

create unique index idx3 ON emp(eid);

Interview Queries

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 bharath cascade;
drop user bhavana cascade;
```

Merge Command

==========

Merge command is a combination of insert and update command.

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','Texas');
insert into student20 values(104,'Jose','Florida');
commit;
```

ex:

```
        merge into student10
        using student20
        ON(student10.sno=student20.sno)
        when matched
        then update set
sname=student20.sname,sadd=student20.sadd
        when not matched
```

then insert(sno,sname,sadd)
values(student20.sno,student20.sname,student20.sadd);

ex:

ex:

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);

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 avoid above problem we will use tablename.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.

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

Using table alias ,length of the query will reduce and meanwhile performance is maintained.

ex:

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

Definition Joins

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

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

and etc.

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 records
```

2) Non-Equi Join

When tables are joined without equi join condition then it is called non-equi join.

ex:

```
select e.eid,e.ename,e.esal,d.dname,d.dloc
from emp e,dept d
where esal>=30000; // 2 * 4 = 8 records
```

3) Self Join

A table which joined to itself is called self join.

In self join we will 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 give you all possible combinations.

ex:

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

5) Inner Join

It is similar to equi-join.

It is given by ANSI people.

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
```

ex:

```
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 '+' symbol denoted as outer join operator.

We have three types of outer joins.

i)Left outer join

ii)Right outer join

iii)Full outer join

i)Left outer join

---

ex:

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

---

ex:

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);
```

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 base table or above table.

view does not consumes the memory.

view will get the data when we write select command.

syntax:

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

We have following list of views.

1) Simple view

2) Complex view

3) with read only view

4) with check option view

5) materized 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 * from emp where deptno=10;

create view v1 as select * from emp where comm is null;

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

create view v1 as select * from emp where esal between 10000 and 40000;

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

Note:

In simple view DML operations are allowed.

ex:

```
select * from v1;

delete from v1; // 7 records added

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
where(e.deptno=d.deptno);
```

Note:

In complex view DML operations are not allowed.

ex:

```
select * from v2;

delete from v2; // can't delete from view
```

3) with read only view

If a view is created by using one base table and DML operation are not required then we need to use with read only view.

ex:

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

Note:

DML operations are not allowed.

ex:

```
select * from v3;

delete from v3;// can't 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 satisfied then we need to use with check option view.

ex:

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

Note:

DML operations are allowed only if condition is true.

ex:

insert into v4 values(208,'Jojo',15000,40,'Salesman',100);//view WITH CHECK

insert into v4 values(208,'Jojo',15000,30,'Salesman',100); //record inserted

select * from v4;

select * from emp;

5) materialized view

A materilized view is also known as snapshot.

To create a materlized view a table must 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; // 8 records

delete from emp where eid=208;

commit;

select * from emp; // 7 records

select * from v5; // 8 records

In order to get updated records we need to referesh the materized view.

ex:

```
exec  DBMS_SNAPSHOT.REFRESH('V5');
```

Here DBMS_SNAPSHOT is a package name.

Here REFRESH is a procedure name.

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

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

==========

If we declare a query inside another query such concept is called sub query.

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

In sub queries first inner query will execute then outer query.

We have following list of sub queries.

1) Single Row Sub query

2) Multiple Row Sub query

3) Mutiple Column Sub query

1) Single Row Sub query

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

A sub query can be nested upto 32 levels.

ex:

SQL

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

SUB-QUERY

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

ex:

SQL

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

SUB-QUERY

```
select * from emp where
eid=(select eid from emp where esal=9000)
AND
ename=(select ename from emp where eid=201);
```

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 all the 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

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

To perform multiple row sub query we need to use multiple row operators.

We have following 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);

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

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 is called multiple column sub query.

In multiple column sub query we need to use IN operator.

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);

PL/SQL

=======

PL/SQL stands for Procedureal and Structured Query Language.

It is a extension of SQL and it gives following features.

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

2)It will reduce network traffic.

3)We can display our own exception messages by using the concept of exception

handling.

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

5)We can save the source code permanent to database for repeated execution.

PL/SQL Block

==========

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, cursors, exceptions and etc.

It is optional section.

Executable section

It contains lines of code which are used complete a table.

The actual logic we will write inside executable section.

IT is a mandatory section.

Exception section

It contains set of lines which are executed when exception is raised.

It is optional section.

In order to see the output in PL/SQL we need to set server output environment.

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.

Here PUT_LINE is a procedure name.

Here '/' is used to submit the block into the database.

Q)Write a PL/SQL program 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 Inialization using single line

---

ex:

```
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 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 number 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_no number(3);
```

```
L_name varchar2(10);
L_add  varchar2(12);
BEGIN
L_no:=&no;
L_name:='&name';
L_add:='&add';
insert into student values(L_no,L_name,L_add);
DBMS_OUTPUT.PUT_LINE('Record Inserted');
END;
/
```

Q)Write a PL/SQL program to update student name based on student number?

```
DECLARE
L_name varchar2(10);
L_no  number(3);
BEGIN
L_name:='&name';
L_no:=&no;
```

```
update student set sname=L_name where sno=L_no;
DBMS_OUTPUT.PUT_LINE('Record Updated');
END;
/
```

Q)Write a query to delete the student record based on student number?

```
DECLARE
L_no  number(3);
BEGIN
L_no:=&no;
delete from student where sno=L_no;
DBMS_OUTPUT.PUT_LINE('Record Deleted');
END;
/
```

IN PL/SQL, DRL operations are also allowed.

To perform DRL operations we need to use "into" clause.

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

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

Q)Write a PL/SQL program to display employee name, employee salary and

department number whose employee id is 202?

```
DECLARE
L_name varchar2(10);
L_sal  number(10,2);
L_dept number(3);
```

```
BEGIN
select ename,esal,deptno into L_name,L_sal,L_dept from emp
where eid=202;
DBMS_OUTPUT.PUT_LINE(L_name||' '||L_sal||'
'||L_dept);
END;
/
```

Q)Write a PL/SQL program to display employee name, employee salary and

department number based on employee id?

```
DECLARE
L_id   number(3);
L_name varchar2(10);
L_sal  number(10,2);
L_dept number(3);
BEGIN
L_id:=&id;
select ename,esal,deptno into L_name,L_sal,L_dept from emp
where eid=L_id;
```

```
        DBMS_OUTPUT.PUT_LINE(L_name||' '||L_sal||'
'||L_dept);
        END;
        /
```

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

ex:

```
        SQL> set  serveroutput  on
```

Q)Write a PL/SQL program to display employee name,employee salary,deptartment number from emp table based on employee id?

ex:

```
        DECLARE
        L_name  varchar2(10);
        L_sal   number(10,2);
        L_dept  number(3);
        L_id    number(3);
        BEGIN
        L_id:=&id;
```

```
    select ename,esal,deptno into L_name,L_sal,L_dept from
emp where eid=L_id;
    DBMS_OUTPUT.PUT_LINE(L_name||' '||L_sal||'
'||L_dept);
    END;
    /
```

Percentage(%) TYPE attribute

==========

It is used to declare a local variable with respect to column type.

syntax:

```
    <variable_name>  <table_name><column_name>%TYPE;
```

ex:

```
    DECLARE
    L_name  emp.ename%TYPE;
    L_sal   emp.esal%TYPE;
    L_dept  emp.deptno%TYPE;
```

```
L_id    emp.eid%TYPE;
BEGIN
L_id:=&id;
select ename,esal,deptno into L_name,L_sal,L_dept from
emp where eid=L_id;
DBMS_OUTPUT.PUT_LINE(L_name||' '||L_sal||'
'||L_dept);
END;
/
```

Q)Write a PL/SQL program to display employee name, employee salary, department number, job and commission based on employee id ?

```
DECLARE
L_name emp.ename%TYPE;
L_sal  emp.esal%TYPE;
L_dept emp.deptno%TYPE;
L_job  emp.job%TYPE;
L_comm emp.comm%TYPE;
L_id   emp.eid%TYPE;
```

```
BEGIN
L_id:=&id;
select ename,esal,deptno,job,comm into
L_name,L_sal,L_dept,L_job,L_comm
 from emp where eid=L_id;
DBMS_OUTPUT.PUT_LINE(L_name||' '||L_sal||' '||L_dept||'
'||L_job||' '||L_comm);
END;
/
```

Percentage(%) ROWTYPE attribute

==========

It is used to declare a local variable which holds complete row of a table.

ROWTYPE variable can't display directly.

In order to get the values from rowtype variable we need to use variable_name.column_name.

syntax:

<variable_name> <table_name>%ROWTYPE;

ex:

```
DECLARE
A    emp%ROWTYPE;
L_id  emp.eid%TYPE;
BEGIN
L_id:=&id;
select * into A from emp where eid=L_id;
DBMS_OUTPUT.PUT_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job);
END;
/
```

Control Statements

==========

1) IF THEN

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

ex:

```
DECLARE
A number:=5;
BEGIN
DBMS_OUTPUT.PUT_LINE('WELCOME');

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

DBMS_OUTPUT.PUT_LINE('ThankYou');
END;
/
```

ex:

--

```
DECLARE
A number:=5;
```

```
BEGIN
DBMS_OUTPUT.PUT_LINE('WELCOME');

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

DBMS_OUTPUT.PUT_LINE('ThankYou');
END;
/
```

2) IF THEN ELSE

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

ex:

```
DECLARE
A number:=4;
BEGIN
```

```
IF A>0 THEN
DBMS_OUTPUT.PUT_LINE('It is a positive number');
ELSE
DBMS_OUTPUT.PUT_LINE('It is a negative number');
END IF;

END;
/
```

ex:

```
DECLARE
A number:=-4;
BEGIN

IF A>0 THEN
DBMS_OUTPUT.PUT_LINE('It is a positive number');
ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('It is a negative number');
        END IF;

        END;
        /
```

3) IF THEN ELSIF THEN ELSE

It will evaluate the code based on multiple conditions.

ex:

--

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

        IF opt=100  THEN
        DBMS_OUTPUT.PUT_LINE('It is police number');
        ELSIF opt=103 THEN
```

```
DBMS_OUTPUT.PUT_LINE('It is enquiry number');
ELSIF opt=108 THEN
DBMS_OUTPUT.PUT_LINE('It is emergency number');
ELSE
DBMS_OUTPUT.PUT_LINE('Invalid option');
END IF;

END;
/
```

LOOPS

======

We have three types of loops.

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('HI');

LOOP
DBMS_OUTPUT.PUT_LINE('Hello');
EXIT WHEN A=4;
A:=A+1;
END LOOP;

DBMS_OUTPUT.PUT_LINE('BYE');
END;
/
```

Q)Write a PL/SQL program to display 10 natural numbers?

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('HI');

WHILE A<=4 LOOP
DBMS_OUTPUT.PUT_LINE('Hello');
A:=A+1;
END LOOP;

DBMS_OUTPUT.PUT_LINE('BYE');
END;
/
```

ex:

```
DECLARE
A number:=1;
N number:=5;
BEGIN

WHILE A<=10 LOOP
DBMS_OUTPUT.PUT_LINE(N||' * '||A||' = '||N*A);
A:=A+1;
END LOOP;

END;
/
```

3) For loop

It will evaluate the code untill our condition is true.

ex:

--

```
DECLARE
A number;
BEGIN
DBMS_OUTPUT.PUT_LINE('Hi');

FOR A IN 1 .. 4 LOOP
DBMS_OUTPUT.PUT_LINE('Hello');
END LOOP;

DBMS_OUTPUT.PUT_LINE('Bye');
END;
/
```

Exceptions in PL/SQL

==========

Runtime errors are called exceptions.

We have two types of exceptions.

1) Predefined exceptions

2) Userdefined exceptions

1) Predefined exceptions

Built-In exceptions are called predefined exceptions.

We have following list of predefined exceptions.

i) NO_DATA_FOUND

ii) TOO_MANY_ROWS

iii) ZERO_DIVIDE

iv) VALUE_ERROR

v) DUP_VAL_ON_INDEX

vi) OTHERS

i) NO_DATA_FOUND

This exception will raise when select statement does not return any value.

```
DECLARE
L_name  emp.ename%TYPE;
BEGIN
select ename into L_name from emp where eid=201;
DBMS_OUTPUT.PUT_LINE(L_name);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Please check employee id');
END;
/
```

ii) TOO_MANY_ROWS

This exception will raise when select statement returns more then one row.

ex:

```
DECLARE
L_name  emp.ename%TYPE;
BEGIN
select ename into L_name from emp where deptno=10;
DBMS_OUTPUT.PUT_LINE(L_name);
EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('more then one record');
END;
/
```

iii) ZERO_DIVIDE

This exception will raise when we 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('dont divide by zero');
END;
/
```

iv)VALUE_ERROR

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_sal  emp.esal%TYPE;
        BEGIN
        select ename into L_sal from emp where eid=201;
        DBMS_OUTPUT.PUT_LINE(L_sal);
        EXCEPTION
        WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Please check datatype');
        END;
        /
```

v)DUP_VAL_ON_INDEX

This exception will raise when we are trying to insert duplicate values in primary key.

ex:

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


BEGIN
insert into emp
values(201,'Jacky',30000,60,'Salesman',200);
DBMS_OUTPUT.PUT_LINE('Record Inserted');
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
DBMS_OUTPUT.PUT_LINE('Duplicate records not
allowed');
END;
/
```

vi) OTHERS

It is a universal angular exception which handles all types of exceptions.

ex:

```
DECLARE
L_name  emp.ename%TYPE;
BEGIN
select ename into L_name from emp where eid=209;
DBMS_OUTPUT.PUT_LINE(L_name);
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Please check employee id');
END;
/
```

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

ex:

```
SQL> set serveroutput on
```

2)Userdefined exceptions

==========

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

steps to work with userdefined exceptions

---

step1:

Declare the exception

step2:

Raise the exception

step3:

Handle the exception

ex:

```
DECLARE
L_SAL  number:=5000;
MY_EX1 EXCEPTION;
```

```
BEGIN
IF L_SAL>2000 THEN
RAISE MY_EX1;
END IF;
DBMS_OUTPUT.PUT_LINE(L_SAL);
EXCEPTION
WHEN MY_EX1 THEN
DBMS_OUTPUT.PUT_LINE('Salary is too high');
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, 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 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 records effected in a database table.

SQL%ISOPEN

```
BEGIN
IF SQL%ISOPEN THEN
DBMS_OUTPUT.PUT_LINE('Cursor is opened');
ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('Cursor is closed');
        END IF;
        END;
        /
```

SQL%FOUND

```
        BEGIN
        update student set sname='ramulu' where sno=103;
        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='ramulu' where sno=109;
```

```
        IF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Record updated');
        ELSE
        DBMS_OUTPUT.PUT_LINE('Record Not Updated');
        END IF;
        END;
        /
```

SQL%NOTFOUND

```
        BEGIN
        update student set sname='rani' where sno=109;
        IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Record updated');
        ELSE
        DBMS_OUTPUT.PUT_LINE('Record Not Updated');
        END IF;
        END;
        /
```

ex:

```
BEGIN
update student set sname='rani' where sno=103;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('Record updated');
ELSE
DBMS_OUTPUT.PUT_LINE('Record Not Updated');
END IF;
END;
/
```

SQL%ROWCOUNT

```
BEGIN
update student set sname='gogo';
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' records updated');
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 user is called explicit cursor.

We will use explicit cursor when select statement returns more then one row.

Explicit cursor having four types of attributes.

i) %ISOPEN

It is a boolean attribute which returns true if cursor is open and returns

false is cursor is closed.

ii) %FOUND

It is a boolean attribute which returns true if SQL command is success and

returns false is SQL command is failed.

iii) %NOTFOUND

It is completely reverse of %FOUND.

It is a boolean attribute which returns false if SQL command is success and

returns true is SQL command is failed.

iv) %ROWCOUNT

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

Steps to work with explicit 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 select employee name , employee salary from emp table?

```
DECLARE
CURSOR C1 is select ename,esal from emp;
L_name emp.ename%TYPE;
```

```
L_sal  emp.esal%TYPE;
BEGIN
OPEN C1;

LOOP
FETCH C1 into L_name,L_sal;
DBMS_OUTPUT.PUT_LINE(L_name||' '||L_sal);
EXIT WHEN C1%NOTFOUND;
END LOOP;

CLOSE C1;
END;
/
```

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

```
    DECLARE
    CURSOR C2 is select ename,esal,deptno,job from emp;
    L_name  emp.ename%TYPE;
```

```
L_sal   emp.esal%TYPE;
L_dept  emp.deptno%TYPE;
L_job   emp.job%TYPE;
BEGIN
OPEN C2;
LOOP
FETCH C2 into L_name,L_sal,L_dept,L_job;
DBMS_OUTPUT.PUT_LINE(L_name||' '||L_sal||'
'||L_dept||' '||L_job);
EXIT WHEN C2%NOTFOUND;
END LOOP;
CLOSE C2;
END;
/
```

Q)Write a PL/SQL program to display employees 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||' '||A.job||' '||A.comm);
EXIT WHEN C2%NOTFOUND;
END LOOP;
CLOSE C2;
END;
/
```

Procedures

==========

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

It is also known as stored PL/SL procedures.

syntax:

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

ex:

```
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

It will 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('sum of two numbers is ='||C);
END;
/
```

We can execute above procedure as follow.

ex:

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

2) OUT parameter

It will 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 execute the 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 the bind variable.

ex:

```
print N;
```

iii) IN OUT parameter

It will accept the value from the user and return the value to the user.

Q) Write a procedure to return square of a given number?

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

Steps to execute the 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 the bind variable.

ex:

```
print N;
```

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

ex:

SQL> set serveroutput on

In PL/SQL procedure, DML operations are allowed.

Q)Write a procedure to accept one employee id and delete the employee record?

```
create or replace procedure delete_record(L_id IN emp.eid%TYPE)
is
begin
delete from emp where eid=L_id;
DBMS_OUTPUT.PUT_LINE('Record Deleted');
end;
/
```

We can execute above procedure by using below command.

ex:

exec delete_record(207);

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

begin

-

-

-

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 function as follow

ex:

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

Q)Write a function to accept one salary and return 10% of TDS?

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

```
return number
is
TAX number;
begin
TAX:=SAL*10/100;
return TAX;
END;
/
```

We can execute above function as follow.

ex:

```
select TDS_RETURN(10000) from dual;
select eid,ename,esal,TDS_RETURN(esal) as TDS from emp;
```

If we find any error in procedure or function we need to use below command.

ex:

```
SQL> show errors;
```

Note:

In functions, DML operations are not allowed.

Q)What is the difference between procedure and function ?

Procedure	Function
Procedure may or may not returns a value.	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.

Logical related sub programs means procedures and functions.

In general, a package is a collection of procedures and functions.

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('sum of two numbers is '||C);
END;
END pkg1;
/
```

We can execute the procedure as follow.

ex:

```
exec   pkg1.sum(10,50);
```

ex:2

---

package specification

```
create or replace package pkg2
is
function ret_sum(A number,B number)
return number;
end pkg2;
/
```

package body

```
create or replace package body pkg2
```

```
is
function ret_sum(A number,B number)
return number
is
C number;
begin
C:=A+B;
return C;
END;
end pkg2;
/
```

We can execute above function as follow.

ex:

```
select  pkg2.ret_sum(40,60) from dual;
```

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 functions present in database?

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

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 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='F1';
```

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 procedure?

```
drop procedure p1;
```

Q)Write a query to drop the function?

```
drop function f1;
```

Q)Write a query to drop the package?

```
drop package pkg1;
```

Triggers

==========

Trigger is a PL/SQL block which is executed based on events.

Trigger events are insert,update and delete.

Triggers timings are before, after and insteadof.

syntax:

```
create or replace trigger trigger_name timming event on object
begin
-
-
end;
/
```

ex:

```
create or replace trigger trg1 before insert on student
begin
DBMS_OUTPUT.PUT_LINE('Thankyou for inserting the data');
END;
```

/

select * from student; // no trigger

insert into student values(104,'ramulu','pune'); // trigger will execute

We can create multiple triggers on a single table.

ex:

```
create or replace trigger trg4 after insert or update or delete on emp
begin
IF inserting then
DBMS_OUTPUT.PUT_LINE('Thankyou for inserting');
ELSIF updating then
DBMS_OUTPUT.PUT_LINE('Thankyou for updating');
ELSE
DBMS_OUTPUT.PUT_LINE('Thankyou for deleting');
END IF;
```

```
END;
/

delete from emp where eid=206; // Thankyou for deleting

update emp set ename='jojo' where eid=201;//Thankyou for updating

insert into emp values(207,'Maria',14000,30,'Salesman',200); //Thankyou for inserting
```

Triggers are classified into two types.

1) Statement level trigger

By default every trigger is a statement level trigger.

Trigger will execute only for one time irrespective of number of records effected in a database table.

ex:

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

select * from student; // no trigger

update student set sname='rani';
```

2) Row level trigger

In row level trigger, a trigger will execute irrespective number of records effecting in a database table.

To create a row level trigger we need to use "FOR EACH ROW" clause.

ex:

```
create or replace trigger trg2 after delete on student FOR EACH ROW
begin
DBMS_OUTPUT.PUT_LINE('Yahoo! deleted');
END;
/

select * from student; // no trigger

delete from student;
```

Q)Write a query to see the 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='TRG1';
```

Q)Write a query to drop the trigger?

```
drop trigger trg1;
```

Interview Question

==========

Q)Write a query to replace '0' with '9'?

```
select replace(esal,'0','9') as SALARY from emp;
```

Q)Write a query to display student number, student name, course name , course fee, book_name?

Diagram: oracle13.1

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,'spring');
commit;


administaration table
----------
create table administration(id number(3) REFERENCES
college(sno),
                        course varchar2(10), fee number(10,2));



insert into administration values(101,'CSE',10000);
insert into administration values(102,'ECE',20000);
insert into administration values(103,'MEC',30000);
commit;


    select c.sno,c.sname,a.course,a.fee,l.book_name
    from college c JOIN administration a
    ON(c.sno=a.id) JOIN library l
    ON(c.sno=l.roll_no);
```