ORACLE

Types of data

We have two types of data.

1) Unstructured Data

Data which is not in readable format is called unstructured data.

In general, meaning less data is called unstructured data.

ex: 302 Lakemba SYD NSW AUS

2) Structured Data

Data which is in readable format is called structured data.

In general, meaning full data is called structured data.

ex:

unit	locality	city	state	country
302	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 a case insensitive language.

Oracle is classified into two types.

Oracle

|

SQL	PL/SQL
(Structured Query Language)	(Procedural/Structured Query Language)
Develop by IBM	Develop by Oracle Corporation

Client-Server Architecture

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

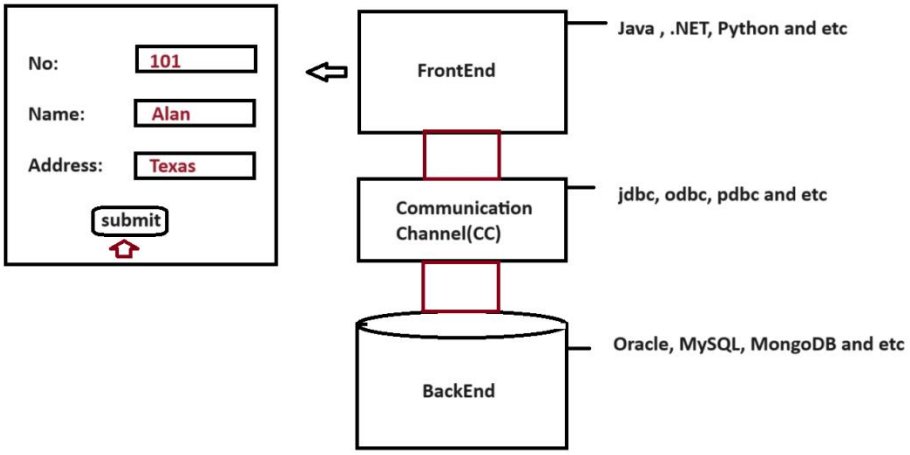

Diagram: oracle1.1

FrontEnd

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

ex: Java, .net, Python, Ruby, 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
and etc

BackEnd

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

ex: Oracle, MySQL, SQL Server, MongoDB, NoSQL, TeraData, DB2, Sybase and etc.

Management System

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

Management system will perform following activities very easily.

1) Inserting new data
2) Modifying existing data
3) Deleting unnecessary data
4) Selecting required data

DBMS

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

RDBMS

Any database which is developed based on relational theories which is used to manage the database is called relational database management system.

ex: Oracle, MySQL, SQL Server, PostgreSQL and etc.

SQL

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

This language is used to communicate with oracle database.

It is a command based language

It is a case insensitive language.

Every command must starts with verb.

ex: select, update, delete, drop and etc.

Every command must ends with semicolon.

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

Sub-Languages of SQL

We have five sub-language of SQL.

1) DDL (Data Definition Language)
2) DML (Data Manipulation Language)
3) DRL/DQL (Data Retrieve/Query Language)
4) TCL (Transaction Control Language)
5) DCL (Data Control Language)

1) DDL(Data Definition Language)

This language is used to maintain the objects in database.
It is a collection of five commands.
ex: create,alter,drop,truncate and rename.

2) DML(Data Manipulation Language)

This language is used to manipulate the data present in database.
It is a collection of four commands.
ex: insert,update,delete and merge.

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

This language is used to retrieve the data from database.
It is a collection of one command.
ex: select

4) TCL(Transaction Control Language)

This language is used to maintain the transaction of database.
It is a collection of three commands.
ex: commit,rollback and savepoint.

5) DCL(Data Control Language)

This language is used to control the access of data to the user.
It is a collection of two commands.
ex: grant and revoke.

Table

Table is an object which is used to represent the data.
Table is a collection of rows and columns.
Oracle is a case insensitive but data present in a table is a case sensitive.

ex:

SNO	SNAME	SADD
101	raja	hyd

---
102 | ravi | delhi
---
103 | ramana| vizag

Here we have 3 rows and 3 columns.

Oracle

Version	:	10g or 11g
Creator	:	Mr. Codd
Vendor	:	Oracle Corporation
Software	:	Expression Edition
website	:	www.oracle.com/in/database
Username	:	system (default)
Password	:	admin
Port No	:	1521

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

Establish the Connection with Database

To perform any operation on database we need to need to connect with database.
Once the work with database is completed then we need to disconnect with database.
There are following ways to connect and disconnect the 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

create command

It is used to create a table(object) 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(6));

Describe command

It is used to describe the structure of the table.

syntax: desc <table_name>;

ex: desc student;
desc dept;
desc emp;

insert command

It is used to insert the data in a database table.

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

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

A null is a keyword which represent undefined or unavailable.

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

Approach2

insert into student(sno,sname,sadd) values(103,'ramana','vizag');
insert into student(sno,sname) values(105,'rakesh');

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.

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,'CSE','PUNE');
insert into dept values(30,'EEE','VIZAG');
insert into dept values(40,'MEC','DELHI');
commit;
```

emp table

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

insert into emp values(201,'Alan',9000,10,'Clerk',null);
insert into emp values(202,'Kelvin',19000,10,'Clerk',500);
insert into emp values(203,'Nancy',29000,20,'HR',300);
insert into emp values(204,'Branda',18000,20,'HR',700);
insert into emp values(205,'Nelson',19000,30,'Manager',500);
insert into emp values(206,'Jose',49000,30,'Manager',200);
commit;
```

select command

It is used to select the records from database table.

syntax: select * from <table_name>; (Here '*' means all rows and columns.)

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 sno from student;

In select command we can perform arithmetic operations also.

ex: select sno+100 from student;
select sno-100 from student;
select sno*10 from student;

Column alias

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

Column alias can be applied 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 as roll_no ,sname as Name,sadd as City from student;

Interview Queries

Q) Write a query to display all the tables present in database?

select * from tab;

Q) Write a query to display logical database name?

select * from global_name; // XE

// ORCL

Q) Write a query to display employee 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 ANNUAL_SAL from emp table?

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

Where clause

Where clause is used to select specific records from database table.

syntax: select * from <table_name> where condition;

ex: select * from student where sno=101;

select * from student where sname='ravi';

select * from student where sadd='pune';

Interview Queries

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 whose who are working as a Manager?

select * from emp where job='Manager';

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

select * from emp where comm is null;

Update command

Update command is used to modify the data present in a database table.

syntax: update <table_name> set <column_name>=value where condition;

ex: update student set sname='rani' where sno=101;
update student set sname='alan',sadd='noida' where sno=102;

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

ex: update student set sname='raja';
commit;

Delete command

Delete command 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='ravi';
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 dept;
delete from emp;
commit;

Note: ALL DML operations are temperory.

Interview Queries

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

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

Q) Write a query to promote employee from clerk to salesman whose employee id is 202?

update emp set job='salesman' where eid=202;

Q) Write a query to terminate employees whose salary is greater then 30000?

delete from emp where esal>30000;

Q) Write a query to delete student record who is living in hyderabad?

delete from student where sadd='hyd';

Logical Operators

Logical operators are used to declare multiple conditions in where clause.

We have three logical operators.

1) AND
2) OR
3) NOT

1) AND

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

In AND operator all conditions must be from same row.

ex: select * from emp where ename='Alan' AND esal=9000;

select * from emp where ename='Alan' AND esal=19000; // no rows selected

2) OR

It will return the records if any condition is true.

In OR operator conditions can be from any row.

ex: select * from emp where ename='Alan' OR esal=9000;

select * from emp where ename='Alan' OR esal=19000;

select * from emp where ename='Alan' OR esal=79000;

3) NOT

It will return all 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';

Interview Queries

Q) Write a query to display employees information whose employee id is 203 and working as HR?

select * from emp where eid=203 AND job='HR';

Q) Write a query to display employee information whose emplyee 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 15000 and less then 30000?

select * from emp where esal>15000 and esal<30000;

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;

Between operator

Between operator will return the records those who are in the range of values.

Between operator will take the support of AND operator.

In Between operator first we need to declare lower limit then higher limit.

ex: select * from emp where eid between 201 AND 206;
select * from emp where eid between 206 AND 201; // no rows selected
select * from emp where esal between 15000 AND 30000;
select eid,ename,esal from emp where deptno between 10 AND 30;

IN operator

IN operator is a replacement of OR operator.

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

ex: select * from emp where eid IN(201,202,203);
select * from emp where deptno IN(10,20,30);
select * from emp where ename IN ('Alan','Jose','Nelson');

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 employee information whose employee name starts with 'A' letter?

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

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

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

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

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

2) Underscore(_)

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

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

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

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

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

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

Duplicate table or Copy of a table

Creating a duplicate table or copy of a table is a save side for the programmer because if something goes wrong we can recover that situation using duplicate table or copy of a table.

Using create and select command we can create duplicate table.

ex:
```
create table stud as select * from student;
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 deptno<>10;
create table employee as select * from emp eid IN(201,202,203);
create table employee as select * from emp esal between 15000 AND 30000;
create table employee as select * from emp name like 'A%';
```

cl scr

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

syntax: cl scr

DDL commands

```
create - (tables)
alter  - (columns)
drop   - (tables)
truncate - (rows/records)
rename - (tables)
```

alter command

Using alter command we can perform following activities very easily.

```
i) Adding the new columns
ii) Modifying the exiting columns
iii) Renaming the columns
iv) Droping the columns
```

i) <u>Adding the new columns</u>

Using alter command we can add new columns to 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));
```

alter table student ADD (state varchar2(10), pincode number(8));
update student set state='Telangana' where sno=101;

ii) Modifying the exiting columns

Using alter command we can modify the existing columns.

We can change size of a column only when existing values are fit into new size.

ex: desc student;
alter table student MODIFY (state varchar2(12));
desc student;

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

ex: desc student;
alter table student MODIFY (pincode varchar2(8));
desc student;

iii) Renaming a columns

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 pincode to country;
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 the columns

Using alter command we can drop the columns.

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

ex: alter table student DROP (state,country);

drop command

It is used to drop the tables from database.

syntax: drop table <table_name>;

ex: drop table student;
drop table emp;
drop table dept;

truncate command

A trunate command is used to delete the records permanently from database table.

syntax: truncate table <table_name>;

ex: truncate table emp;
truncate table student;

truncate table dept;

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

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

rename command

It is used to rename the table name.

syntax: rename <old_name> to <new_name>;

ex: rename emp to employee;
rename student to students;
rename dept to department;

Functions

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

We have two types of functions.

1) Group Functions / Multiple Rows Functions

2) Scalar Functions / Single Rows 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(express).

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 least/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.
ex: select count(*) from emp;

count(exp)

It will return number of values present in a column.
It will not include null values.
ex: select count(eid) from emp;
select count(comm) from emp;

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 username and password is valid or not?

```
select count(*) from userlist where uname='raja' AND pwd='rani'; // 1
select count(*) from userlist where uname='raja' AND pwd='rani2'; // 0
```

Dual table

Dual table is a dummy table which is used to perform arithmetic operations and to see the current system date.
Dual table contains one row and one column.

```
ex: select 10+20 from dual;
    select 10*20 from dual;
    select sysdate from dual;
    select current_date from dual;
```

2) Scalar Functions

Scalar Functions are applicable for single row.
Scalar functions are divided into four types.

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;

select upper('oracle') as UPPER from dual;

lower()

It will convert uppercase to lowercase.

ex: select lower('ORACLE') from dual;

initcap()

It will convert initial letter to capital.

ex: select initcap('this is oracle class') from dual;

lpad()

It will pad the characters at left side.

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

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 characters from both the sides.

ex: select trim('z' from 'zzoraclezz') from dual; // oracle

concat()

It will concatinate the strings.

ex: select concat('mega','star') from dual;

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

replace()

It will replace the character in a string.

ex: select replace(esal,0,9) from emp;

ii) Number functions

abs()

It will return absoluate value.

ex: select abs(-10) from dual; //10

select abs(10) from dual; //10

power(A,B)

It will return power value.

ex: select power(5,3) from dual;

sqrt()

It will return exact square root value.

ex: select sqrt(25) from dual; //5

select sqrt(26) from dual; //5.09

ceil()

It will return ceil value.

ex: select ceil(10.2) from dual; // 11

select ceil(56.7) from dual; // 57

floor()

It will return floor value.

ex: select floor(10.2) from dual; //10

select floor(10.9) from dual; //10

round()

It will return nearest value.

ex: select round(10.4) from dual; // 10

select round(10.5) from dual; //11

trunc()

It will return decimal values.

ex: select trunc(10.56) from dual; //10

select trunc(56.78) from dual; // 56

greatest()

It will return greatest value.

ex: select greatest(101,102,103) from dual;

least()

It will return least value.

ex: select least(101,102,103) from dual;//101

Working with Date values

All database softwares support date values.

It is not recommanded to store date values in the form of varchar2.

Every database supports different date patterns.

ex: oracle -- dd-MMM-yy

MySQL -- yyyy-MM-dd

emp1 table

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

```
insert into emp1 values(101,'Alan','01-JAN-24');
insert into emp1 values(102,'Jose','15-NOV-23');
insert into emp1 values(103,'Nelson',sysdate);
commit;
```

iii) Date functions

We have following list of date functions.

ADD_MONTHS()

It will add the months in a given date.

```
ex:  select ADD_MONTHS(sysdate,6) from dual;
```

MONTHS_BETWEEN()

It will return number of months between two given dates.

```
ex:  select MONTHS_BETWEEN('01-JAN-23','11-JAN-24') from dual; //-12.3
     select ABS(MONTHS_BETWEEN('01-JAN-23','11-JAN-24')) from dual; //12.3
```

NEXT_DAY()

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

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

LAST_DAY()

It will return last date of a month.

```
ex:  select LAST_DAY(sysdate) from dual;
     select LAST_DAY('15-FEB-24') from dual;
```

iv) Conversion functions

It is used to convert from one type to another type.

ex: TO_CHAR()

We have two pseudo for TO_CHAR().

number to_char()

It will accept '9' in digits , dollar and euro symbols.

```
ex:  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;
```

date to_char()

```
     select  TO_CHAR(sysdate,'dd-MM-yyyy') from dual;
     select  eid,ename,TO_CHAR(sysdate,'dd-MM-yyyy') as DOJ from emp1;
     select  TO_CHAR(sysdate,'month') from dual;
     select  TO_CHAR(sysdate,'mon') from dual;
     select  TO_CHAR(sysdate,'year') from dual;
     select  TO_CHAR(sysdate,'yyyy') from dual;
```

```
select  TO_CHAR(sysdate,'mm') from dual;
select  TO_CHAR(sysdate,'dd') from dual;
select  TO_CHAR(sysdate,'day') from dual;
select  TO_CHAR(sysdate,'dy') 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

Group by clause is used to divide the rows into groups so that we can apply group functions.
A column which we used in select clause then same column must 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 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 lowest salary of each job?

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

Having clause

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

Q)Write a query to display sum of salary of each department whose salary is greater then 30000?

```
select sum(esal),deptno from emp group by deptno having sum(esal)>30000;
```

Q) Write a query to display average salary of each job where average salary is less then 40000?

```
select avg(esal),job from emp group by job having avg(esal)<40000;
```

Order by clause

It is used to arrange the rows in a table.
By default it will arrange in ascending order.

ex:

```
select * from emp order by eid;
select * from emp order by eid desc;
select * from emp order by ename;
select * from emp order by esal;
```

Q)Write a query to display sum of salary of each department whose salary is greater then 30000?

```
select sum(esal),deptno from emp group by deptno
having sum(esal)>30000
order by deptno;
```

Q)Write a query to display sum of salary of each department except 20 department whose salary is greater then 30000?

```
select sum(esal),deptno from emp where deptno<>20 group by deptno
having sum(esal)>30000
order by deptno;
```

Intigrity Constraints

Constraints are the rules which are applied on the tables.
We have following list of constraints.
1) NOT NULL
2) UNIQUE
3) PRIMARY KEY
4) FOREIGN KEY
5) CHECK
Every constraint can be created two levels.
i) Column level
ii) Table level

1) NOT NULL

NOT NULL constraint does not accept null values.
It will accept duplicate values.
NOT NULL constraint can be created at column level.
<u>column level</u>

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

Note: UNIQUE constraint can be created for multiple columns.

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

Note: UNIQUE constraint can be created for multiple columns.

```
ex:
drop table student;
create table student(sno number(3), sname varchar2(10),
                    sadd varchar2(12), UNIQUE(sno,sname,sadd));
```

3) Primary Key

Primary key is a combination of NOT NULL and UNIQUE constraint.
Primary key does not accept null values and duplicate values.
A table can have only one primary key.
Primary key 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'); //valid
insert into student values(101,'ravi','delhi'); // invalid
insert into student values(null,'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'); //valid
insert into student values(101,'ravi','delhi'); // invalid
insert into student values(null,'ramana','vizag'); //invalid
commit;
```

4) Foreign Key

Foreign key is used to establish the relationship between two tables.
This relationship is called parent and child relationship or master and detailed relationship.
To establish the relationship between two tables.A parent table must have primary key or unique key and child table must have foreign key.
A primary key name may or may not match with foreign key but datatype must match.
A foreign key will accept only those values which are present in primary key.
A foreign key will accept duplicate and null values also.

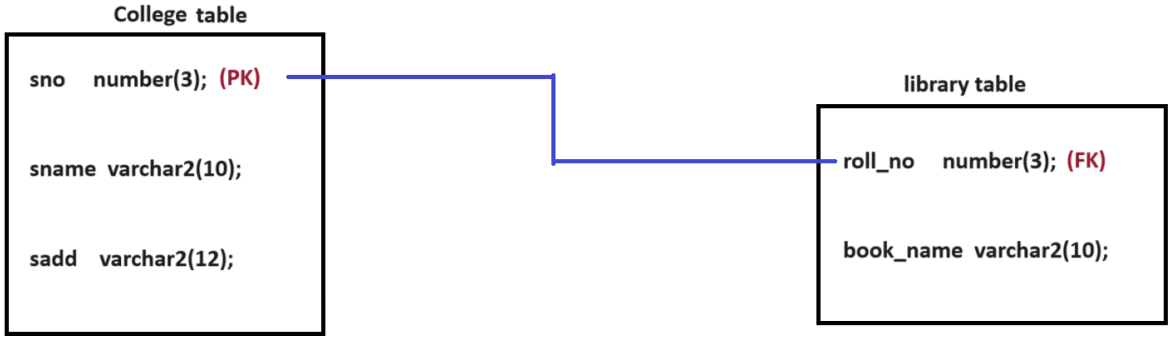

Diagram: oracle5.1

college table

```
drop table college;
create table college(sno number(3) primary key, sname varchar2(10), sadd varchar2(12));
insert into college values(101,'raja','hyd');
insert into college values(102,'ravi','delhi');
insert into college values(103,'ramana','vizag');
commit;
```

library table

```
drop table library;
create table library(roll_no number(3) REFERENCES college(sno), book_name varchar2(10));
insert into library values(101,'oracle');
insert into library values(102,'java');
insert into library values(103,'spring');
insert into library values(103,'reactjs');
insert into library values(null,'jdbc');
insert into library values(104,'servlets'); // invalid
commit;
```

5) CHECK

It is used to describe 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',70);
insert into student values(102,'ravi',120); //invalid
insert into student values(103,'ramana',999); //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',70);
insert into student values(102,'ravi',120); //invalid
insert into student values(103,'ramana',999); //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',70);
insert into student values(102,'RAVI',120); //invalid
```

```
insert into student values(103,'RaMaNa',999); //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',70);
insert into student values(102,'ravi',120); //invalid
insert into student values(103,'RaMaNa',999); //invalid
commit;
```

table level

```
ex:
drop table student;
create table student(sno number(3),sname varchar2(10), smarks number(3),
                                                    CHECK(sname=upper(sname)));
insert into student values(101,'RAJA',70);
insert into student values(102,'ravi',120); //invalid
insert into student values(103,'RaMaNa',999); //invalid
commit;
```

Note: Using constraints we can achieve accuracy and quality of data.

Q) How can we add constraint to 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;
```

TCL commands

1) commit
2) rollback
3) savepoint

1) commit

It is used to make the changes permanent to database.

```
ex:     drop table student;
        create table student(sno number(3),sname varchar2(10),sadd varchar2(12));
        insert into student values(101,'raja','hyd');
        insert into student values(102,'ravi','delhi');
        commit;
```

2) rollback

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

syntax: rollback;

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,'ramulu','pune');
select * from student; //4 records
rollback;
select * from student; //2 records
```

3) savepoint

It is used to make logical transaction in a database table.

Instead of complete rollback we can rollback upto savepoint.

syntax: savepoint <save_point_name>;

ex:
```
drop table student;
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));
insert into student values(101,'raja','hyd');
insert into student values(102,'ravi','delhi');
savepoint sp1;
insert into student values(103,'ramana','vizag');
insert into student values(104,'ramulu','pune');
savepoint sp2;
insert into student values(105,'Jack','USA');
insert into student values(106,'James','UK');
select * from student; // 6 records
rollback to sp2;
select * from student; // 4 records
rollback to sp1;
select * from student; // 2 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.

In general, 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.

1) grant

It is used to grant the permissions to the user.

syntax: grant <resource1>,<resource2> to <user_name>;

2) revoke

It is used to revoke the permissions from the user.

syntax: revoke <resource1>,<resource2> from <user_name>;

```
DBA> create user aravind identified by aravind;
DBA> create user manasa identified by manasa;
Aravind> conn aravind/aravind      --logon denied
Manasa> conn manasa/manasa         --logon denied
DBA> grant connect,resource to aravind,manasa;
Aravind> conn aravind/aravind
Manasa> conn manasa/manasa

Aravind>
create table employee(eid number(3),ename varchar2(10),esal number(10));
insert into employee values(201,'Alan',10000);
insert into employee values(202,'Jose',20000);
insert into employee values(203,'Mark',30000);
commit;
select * from employee;

Manasa>  select * from employee; //table or view does not exist

Aravind> grant select on employee to manasa;

Manasa> select * from aravind.employee;

Manasa> delete from aravind.employee;   --insufficient privileges

Aravind> grant delete,update on employee to manasa;

Manasa> delete from aravind.employee;
Manasa> commit;
```

Aravind> select * from employee; // no rows selected

Aravind> revoke select,update,delete on employee from manasa;

Manasa> disc
Aravind> disc

DBA> revoke connect,resource from aravind,manasa;

Pseudo Columns

A 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.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;
select rownum,sno,sname,sadd from student;

2) ROWID

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

ex: select rowid,eid,ename,esal from emp;
select rowid,rownum,eid,ename,esal from emp;

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?

drop user aravind cascade;
drop user manasa cascade;

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

select * from emp where rownum<=3;

Q) Write a query to display exact fourth record from employee table?

```
select * from emp where rownum<=4
minus
select * from emp where rownum<=3;
```

Synonyms

Alternate name given to a table is called synonym.

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

Using synonym, length of the table will reduce.

syntax: create synonym <synonym_name> for <table_name>;

ex: create synonym sy1 for student;

```
select * from sy1;
delete from sy1;
```

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

```
select synonym_name from user_synonyms;
```

Q) Write a query to drop the synonyms?

```
drop synonym  sy1;
```

Sequence

A 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 101 increment by 1;

```
create sequence sq2 start with 10 increment by 10;
```

We have two pseudo in a sequence.

1) NEXTVAL

It is used to generate next number in a 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 will return the last number which is generated by sequenced.

ex: select sq1.CURRVAL from dual;

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

```
select sequence_name from user_sequences;
```

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

```
drop sequence sq1;
```

Indexes

Index is used to improve the performance of select command

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

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

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

ex: Indexed table

ROWID	INDEXED_COLUMN
	9000
	13000
	23000
	28000
	49000

We have two types of indexes.

1) Simple Index

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

ex: create index idx1 ON emp(esal);

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

ex: select * from emp where esal=49000;

2) Complex Index

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

ex: create index idx2 ON emp(eid,deptno);

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

ex: select * from emp where eid=201 AND deptno=10;

Q) Write a query to see 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;
```

Joins

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

ex: 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 limitation we need to use table_name.column_name.

Select emp.eid,emp.ename,emp.esal,dept.deptno,dept.dname,dept.dloc from emp,dept;//6*4=24

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 meanwhile performance will be maintained.

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

We have following 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 records

2) Non-Equi join

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

ex: select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d
where esal>25000; // 2 * 4 = 8 records

3) Self join

When table is joined to itself is called self join.

In self join we will create two table alias for the 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

When tables are joined without using any condition is called cartisian product.

It will return all the possible combinations.

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

5) Inner join

Inner join is similar to equi join.

Inner join 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

SQL

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

ANSI

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

ii) right outer join

SQL

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

ANSI

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

iii) full outer join

ANSI

```
select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc
```

```
from emp e FULL OUTER JOIN dept d
ON(e.deptno=d.deptno);
```

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,'James','Florida');
commit;
```

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);
```

View

View is a logical 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.
A view does not consumes the memory.
A view does not have any data.
A view will get the data when we execute select command.

syntax: create view <view_name> as select stmt;

ex: create view v1 as select * from emp;
create view v1 as select eid,ename,esal from emp;
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 deptno<>10;
create view v1 as select * from emp where eid between 201 and 206;

We have following types of views.

1) Simple view
2) Complex view
3) With read only view
4) With check option view
5) Materialized view

1) Simple view

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

In simple view DML operations are allowed.

ex: create view v1 as select * from emp;
select * from v1;
select * from emp;
delete from v1 where eid=207;
select * from v1;
select * from emp;

2) Complex view

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

In complex view DML operations are not allowed.

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;
delete from v2;

3) With read only view

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

ex: create view v3 as select * from emp with read only;
select * from v3;
delete from v3;

4) With check option view

If a view is created by using one base table and DML operations are required only if our condition is true then we need to use with check option view.

ex: create view v4 as select * from emp where deptno=30 with check option;
insert into v4 values(207,'Lara',60000,50,'Salesman',500);//invalid
insert into v4 values(207,'Lara',60000,30,'Salesman',500);
select * from v4;
select * from emp;

5) Materialized view

Materialized view is also known as snapshot.

To create materialized view a table must have primary key.

ex: alter table emp ADD primary key(eid);

create materialized view v5 as select * from emp;

select * from v5;

delete from v5 where eid=207; // data manipulation not possible

delete from emp where eid=207; //record deleted

select * from v5; // record available here

exec DBMS_SNAPSHOT.REFRESH('v5');

Here DBMS_SNAPSHOT is a package name.

Here REFRESH is a procedure name.

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.

We have following list of sub queries.

1) Single row sub query

2) Multiple row sub query

3) Multiple column sub query

1) Single row sub query

When subquery returns only one row is called single row sub query.

Subquery can be nested upto 32 levels.

ex: SQL

select * from emp where eid=201;

Subquery

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

Here sub query will execute first then parent query.

ex:

SQL

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

subquery

```
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 employees information whose salary is greater then Branda salary?

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

Q) Write a query to delete duplicate records from a database table ?

```
delete from student where rowid NOT IN(select max(rowid) from student group by sno);
```

2) Multiple row sub query

If a sub query returns more then one row is called multiple row sub query.

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

We have three types of multiple row operators.

1) ANY
2) ALL
3) IN

1) **ANY**

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

2) **ALL**

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

3) **IN**

```
select * from emp where esal IN (select esal from emp where deptno=10);
select * from emp where NOT esal IN (select esal from emp where deptno=10);
```

3) Multiple column subquery

If a sub query returns more then one column is called multiple column 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 Procedural and Structured Query Language.
It is a extension of SQL and having 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 error messages by using the concept of exception handling.
4. We can perform related operations by using the concept of triggers.
5. It will compile and store PL/SQL program permanent to database for repeated execution.

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
A declaration section is used to declare variables, exceptions , cursors and etc.
It is optional section.

Executable section
A executable section contains lines of code which are used to complete a table.
*It is a mandatory section.

Exception section
A exception section contains lines of code which are executed when exception is raised.
It is a optional section.
To see the output in PL/SQL we need to use below command.
ex: SQL> set serveroutput on

Q) Write a PL/SQL program to display Hello World?

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Hello World');
END;
/
```

*Note: Here DBMS_OUTPUT is a package name.
Here PUT_LINE is a procedure name.
Here '/' is used to submit the PL/SQL block into database.

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(C);
END;
/
```

Using '&' symbol we can provide dynamic values.

```
DECLARE
A number;
B number;
C number;
BEGIN
A := &a;
B := &b;
```

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

In PL/SQL , DML operations are allowed.

```
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 accept student number and update student name ?

```
DECLARE
L_Sno number(3);
BEGIN
L_Sno := &sno;
update student set sname='rani' where sno=L_Sno;
DBMS_OUTPUT.PUT_LINE('Record Updated');
END;
/
```

Q) Write a PL/SQL program to accept student number and delete the record?

```
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 , DRL operations are allowed.

If we perform select command in PL/SQL then we need to use "into" clause.

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_Eid  number(3);
L_Ename varchar2(10);
L_Esal number(10,2);
BEGIN
L_Eid := &eid;
select ename,esal into L_Ename,L_Esal from emp where eid=L_Eid;
DBMS_OUTPUT.PUT_LINE(L_Ename||' '||L_Esal);
END;
/
```

Percentage(%) TYPE Attribute

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

syntax: variable_name table_name.column_name%TYPE;

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

```
DECLARE
L_Eid  emp.eid%TYPE;
L_Ename emp.ename%TYPE;
L_Esal  emp.esal%TYPE;
BEGIN
L_Eid := &eid;
select ename,esal into L_Ename,L_Esal from emp where eid=L_Eid;
DBMS_OUTPUT.PUT_LINE(L_Ename||' '||L_Esal);
END;
/
```

Percentage(&) ROWTYPE attribute

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

ROWTYPE variable can't display directly.

We can access ROWTYPE variables values by using variable_name.column_name.

syntax: variable_name table_name%ROWTYPE;

Q) Write a PL/SQL to display employee information whose employee id is 205?

```
DECLARE
A  emp%ROWTYPE;
BEGIN
select * into A from emp where eid=205;
DBMS_OUTPUT.PUT_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.comm||' '||A.job);
END;
/
```

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

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

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

ex: SQL> set serveroutput on

Control Statements

We have three types of control statements in PL/SQL.

1. IF THEN
2. IF THEN ELSE
3. IF THEN ELSIF

1) <u>IF THEN</u>

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

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

```
IF A>2 THEN
DBMS_OUTPUT.PUT_LINE('It is greatest');
END IF;
DBMS_OUTPUT.PUT_LINE('thankyou');
END;
/
```

o/p:

```
welcome
It is greatest
thankyou
```

ex:

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

IF A>20 THEN
DBMS_OUTPUT.PUT_LINE('It is greatest');
END IF;

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

o/p:

```
welcome
thankyou
```

2) IF THEN ELSE

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

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

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

```
END IF;
DBMS_OUTPUT.PUT_LINE('thankyou');
END;
/
```

o/p:

```
welcome
It is positive
thankyou
```

ex:

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

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

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

o/p:

```
welcome
IT is negative
thankyou
```

3) IF THEN ELSIF

It will evalute the code based on multiple conditions.

```
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 in PL/SQL.

1. Simple Loop
2. While Loop
3. For Loop

1) Simple Loop

It will evalute the code untill our condition is true.

```
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('thankyou');
END;
/
```

o/p:

```
welcome
Hello
Hello
Hello
Hello
thankyou
```

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

```
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 evalute the code untill our condition is true.

```
DECLARE
A number:=1;
BEGIN
DBMS_OUTPUT.PUT_LINE('welcome');

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

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

Q) Write a PL/SQL program to display 10 naturals in descending order?

```
DECLARE
A number:=10;
BEGIN

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

END;
/
```

3) For Loop

It will evalute the code untill our condition is true.

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

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

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

o/p:

```
welcome
Hello
Hello
Hello
Hello
Thankyou
```

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

```
DECLARE
I number:=1;
N number;
BEGIN
N:=&n;
FOR I IN 1 .. 10 LOOP
DBMS_OUTPUT.PUT_LINE(N||' * '||I||' = '||N*I);
END LOOP;
END;
/
```

Exceptions

Runtime errors are called exceptions.

We have two types of exceptions in PL/SQL.

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 Exception
ii) TOO_MANY_ROWS Exception
iii) VALUE_ERROR Exception
iv) ZERO_DIVIDE Exception
v) DUP_VAL_ON_INDEX Exception
vi) OTHERS

i) NO_DATA_FOUND

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

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

ii) TOO_MANY_ROWS

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

```
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;
/
```

iii) VALUE_ERROR

This exception will raise when there is a mismatch with datatype or size.

```
DECLARE
A number(3);
BEGIN
```

```
A:=123456;
DBMS_OUTPUT.PUT_LINE(A);
EXCEPTION
WHEN VALUE_ERROR THEN
DBMS_OUTPUT.PUT_LINE('Please check the size');
END;
/
```

ex:2
```
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;
/
```

iv) ZERO_DIVIDE

This exception will raise when we are trying to divide a number with zero.

```
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;
/
```

v) DUP_VAL_ON_INDEX

This exception will raise when are trying to insert duplicate value in a primary key.

```
alter table student add primary key (sno);
BEGIN
insert into student values(101,'jose','florida');
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.

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

2) Userdefined Exceptions

Exceptions are created by the user are called userdefined exceptions.

steps to develop user defined exceptions

- step1: Declare the exception
- step2: Raise the exception
- step3: Handle the exception

```
DECLARE
A number:=5000;
MY_EX Exception;
BEGIN

IF A>2000 THEN
RAISE MY_EX;
END IF;
DBMS_OUTPUT.PUT_LINE(A);

EXCEPTION
WHEN  MY_EX THEN
DBMS_OUTPUT.PUT_LINE('Number is too large');
END;
/
```

Cursors

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

We have two types of cursors.

1) Implicit cursor
2) Explicit cursor

1) Implicit cursor

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

We have four types of implicit cursor attributes.

i) SQL%ISOPEN

It is a boolean attribute value 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 inverse of SQL%FOUND.

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

iv) SQL%ROWCOUNT

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

SQL%ISOPEN

```
BEGIN
IF  SQL%ISOPEN THEN
DBMS_OUTPUT.PUT_LINE('Cursor is open');
ELSE
DBMS_OUTPUT.PUT_LINE('Cursor is closed');
END IF;
END;
/
```

SQL%FOUND

```
BEGIN
update student set sname='rani' where sno=101;
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=105;
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='raja';
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' Records updated');
END;
/
```

2) Explicit cursor

All the activities related to cursor like open the cursor, processing the cursor and closing the cursor which is done by a programmer is called explicit cursor.

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

We have four types of explicit cursor attributes.

i) %ISOPEN

It is a boolean attribute which returns true if cursor is opened 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 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 effected in a database table.

Steps to work with explicit cursor

step1: Declare the cursor

ex: CURSOR <cursor_name> is select stmt;

step2: Open the cursor

ex: OPEN <cursor_name>;

step3: Fetch the data from cursor to local variables

ex: FETCH <cursor_name> into <variables>;

step4: Close the cursor

ex: CLOSE <cursor_name>;

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

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

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

```
DECLARE
CURSOR C2 is select eid,ename,esal from emp;
L_eid emp.eid%TYPE;
L_ename emp.ename%TYPE;
L_esal emp.esal%TYPE;
BEGIN
OPEN C2;
LOOP
FETCH C2 into L_eid,L_ename,L_esal;
DBMS_OUTPUT.PUT_LINE(L_eid||' '||L_ename||' '||L_esal);
```

```
EXIT WHEN C2%NOTFOUND;
END LOOP;
CLOSE C2;
END;
/
```

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

```
DECLARE
CURSOR C3 is select * from emp;
A emp%ROWTYPE;
BEGIN
OPEN C3;
LOOP
FETCH C3 into A;
DBMS_OUTPUT.PUT_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job||' '||A.comm);
EXIT WHEN C3%NOTFOUND;
END LOOP;
CLOSE C3;
END;
/
```

ex:

```
DECLARE
CURSOR C4 is select e.eid,e.ename,d.dname,d.dloc from emp e , dept d where
                    (e.deptno=d.deptno);
L_eid emp.eid%TYPE;
L_ename emp.ename%TYPE;
L_dname dept.dname%TYPE;
L_dloc dept.dloc%TYPE;
BEGIN
OPEN C4;
LOOP
FETCH C4 into L_eid,L_ename,L_dname,L_dloc;
DBMS_OUTPUT.PUT_LINE(L_eid||' '||L_ename||' '||L_dname||' '||L_dloc);
EXIT WHEN C4%NOTFOUND;
END LOOP;
CLOSE C4;
END;
/
```

Procedures

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

It is also known as stored PL/SQL procedures.

syntax:

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

Q) Write a procedure to display Hello World?

```
create or replace procedure p1
is
begin
DBMS_OUTPUT.PUT_LINE('Hello World');
END;
/
```

*Note: To execute the procedure we need to use below command.

```
ex:     SQL> exec  p1;
```

Every procedure may contains three parameters.

1) IN parameter
2) OUT parameter
3) IN OUT parameter

1) <u>IN parameter</u>

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 the procedure by using below command.

ex: SQL> exec sum(10,20);

Using procedures we can perform DML operations.

Q) Write a procedure to update student name based on student number?

```
create or replace procedure update_name(L_sno IN student.sno%TYPE)
is
begin
update student set sname='rani' where sno=L_sno;
DBMS_OUTPUT.PUT_LINE('Record updated');
END;
/
```

To execute the procedure we need to use below command.

ex: SQL> exec update_name(101);

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

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

To execute the procedure we need to use below command.

ex: exec delete_record(102);

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

ex: SQL> set serveroutput on

2) <u>OUT parameter</u>

It will return the output to the user.

Q) Write a procedure to perform sum of two numbers and return sum?

```
create or replace procedure ret_sum(A IN number,B IN number,C OUT number)
is
begin
C:=A+B;
END;
/
```

Steps to call a procedure having OUT parameter

step1: Declare a bind variable

ex: variable N number;

step2: Execute the procedure

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

step3: Print a bind variable

ex: print N;

3) IN OUT parameter

It is used to accept and return a value from/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 call a procedure having IN OUT parameter

step1: Declare a bind variable

ex: variable N number;

step2: Initialize the bind variable

ex:
```
BEGIN
:N := 5;
END;
/
```

step3: Execute the procedure

ex: exec ret_square(:N);

step4: Print a bind variable

ex: print N;

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 source code of a procedure?

```
select text from user_source where name='P1';
```

Q) Write a query to drop the procedure?

```
drop procedure p1;
```

```
drop procedure ret_sum;
drop procedure ret_square;
```

PL/SQL functions

It is a named PL/SQL block which must and should returns a value.

syntax:create or replace function <function_name>

```
return datatype
is
begin
-
-
-
end;
/
```

Q) Write a PL/SQL function to return sum of two numbers?

```
create or replace function f1(A number,B number)
return  number
is
C number;
begin
C:=A+B;
return C;
END;
/
```

To execute the function we need to use below command.

```
ex:     select f1(10,20) from dual;
        select f1(10,20) as SUM from dual;
```

Q) Write a PL/SQL function to accept one salary then find out 10% of tax?

```
create or replace function f2(SAL number)
return number
is
TAX number;
begin
TAX := SAL*10/100;
return TAX;
END;
/
```

To execute the function we need to use below command.

ex: select f2(10000) from dual;
select eid,ename,esal,f2(esal) as TAX from emp;

Note: In Functions DML operations are not allowed.

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 source code of a function?

select text from user_source where name='F1';

Q) Write a query to drop the function?

drop function f1;
drop function f2;

Q) What is the difference between procedures and functions?

Procedures	Functions
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 stmt.	Can be invoked by using select stmt.

Packages

A package is a collection of logical related sub programs.
PL/SQL procedures and functions are called logical related sub programs.
Package creation involved in two steps.

1) Package specification

It contains declaration of logical related sub programs.

2) Package body

It contains definition of logical related sub programs.

ex:

package specification

```
create or replace package pkg1
is
procedure sum(A IN number,B IN number);
end pkg1;
/
```

package body

```
create or replace package body pkg1
is
```

```
procedure sum(A IN number,B IN number)
is
C number;
begin
C:=A+B;
DBMS_OUTPUT.PUT_LINE(C);
END;
end pkg1;
/
```

To call the procedure we need to use below command.

```
ex:       exec  pkg1.sum(10,40);
```

ex:

package specification

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

package body

```
create or replace package body pkg2
is
function ret_fun(A number,B number)
return number
is
C number;
BEGIN
C:=A+B;
return C;
END;
end pkg2;
/
```

To call the function we need to use below command.

```
ex:       select  pkg2.ret_fun(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?

```
drop package body pkg1;
drop package pkg1;
drop package body pkg2;
drop package pkg2;
```

Triggers

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

Insert, Update and Delete are trigger events.

Before, After and Insteadof are the timings of trigger.

syntax:

```
create or replace trigger <trigger_name> <timing> <event> ON <object>
begin
-
-
-
end;
/
```

ex:

```
create or replace trigger trg1 after insert ON student
begin
DBMS_OUTPUT.PUT_LINE('Thank you for inserting !!!!');
END;
/
select * from student;
insert into student values(103,'ramana','vizag'); // trigger will execute
```

Trigger can be created for multiple events also.

ex:

```
create or replace trigger trg2 before insert OR update OR delete ON student
begin
IF inserting THEN
DBMS_OUTPUT.PUT_LINE('Thank you for inserting!!');
ELSIF updating THEN
DBMS_OUTPUT.PUT_LINE('Thank you for updating!!');
ELSE
DBMS_OUTPUT.PUT_LINE('Thank you for deleting!!');
END IF;
```

```
END;
/
select * from student; // no trigger
insert into student values(104,'ramulu','pune');
update student set sname='rani' where sno=104;
delete from student where sno=104;
```

Triggers are categories into two types.

1) Statement level trigger

In statement level trigger, trigger will execute only for one time irrespective of number of records effected in a database table.

By default every trigger is a statement level trigger.

```
create or replace trigger trg3 after delete on student
begin
DBMS_OUTPUT.PUT_LINE('Deleted!!');
END;
/
delete from student; // trigger will execute only for one time.
```

2) Row level trigger

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

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

```
create or replace trigger trg4 after delete on student FOR EACH ROW
begin
DBMS_OUTPUT.PUT_LINE('Deleted!!');
END;
/
delete from student; // trigger will execute here multiple times
```

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='TRG4';
```

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

```
drop trigger trg1;
drop trigger trg2;
drop trigger trg3;
drop trigger trg4;
```