How to enter into ORACLE?

After entering into system click on the start button. Then
Start→ all programmers→Windows NT for oracle→SQL 8.0.

Then you will enter into oracle SQL * plus. Here the system asks for user name and password. Enter the username and password click on OK button.

User name: SCOTT / SYSTEM
Pass word: TIGER / MANAGER

Now you are connected to personal oracle of release 8.0.3.0.0 production. Now you are able to create the table and can store different information. SQL asking the questions to database to retrieve the information.
SQL*plus: Release 8.0.3.0.0 production on Tue Oct 06 10:10:50 2009

Commands	Category
a) CREATE b) ALTER c) DROP d) RENAME e) TRUNCATE	Data Definition Language (DDL)
a) SELECT b) INSERT c) UPDATE d) DELETE	Data Manipulation Language (DML) (Data retrieval)
a) COMMIT b) ROLLBACK c) SAVEPOINT	Transaction Control Language (TCL)
a) GRANT b) REVOKE	Data Control Language (DCL)

IMPLEMENTING DDL COMMANDS

CREATE TABLE:

To create tables we use CREATE TABLE command.

SYNTAX:

Create Table <table-name> (col_name1 data-type (size), col_name2 data-type (size)…);

1) EMP TABLE

```
SQL> create table emp1 (empno number (5), ename varchar2 (20),
        sal number (7, 2));
```

Table created.

DESCRIBE:

Display the structure of the table.

SYNTAX:

describe <tab name> [;]

```
SQL> desc emp1;
```

Name	Null?	Type
EMPNO		NUMBER(5)
ENAME		VARCHAR2(20)
SAL		NUMBER(7,2)

2) STSTUDENT TABLE

```
SQL> create table student (sno number (5), sname varchar2 (15), marks1 number
(3), marks2 number (3), marks3 number (3), total number (4));
```

TABLE CREATED.

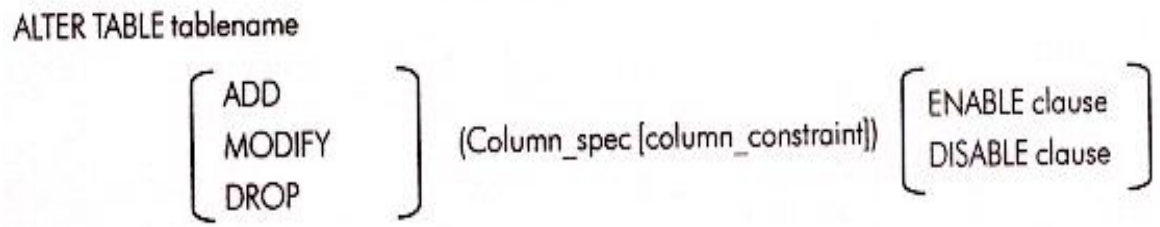
SQL> desc student;

Name	Null?	Type
SNO		NUMBER(5)
SNAME		VARCHAR2(15)
MARKS1		NUMBER(3)
MARKS2		NUMBER(3)
MARKS3		NUMBER(3)
TOTAL		NUMBER(4)

ALTER TABLE

For altering (changing) the definition of a table, i.e. adding, modifying and dropping one or more columns and/or constraints.

GENERAL SYNTAX FOR ALTER COMMAND:



SQL> ALTER TABLE EMP1
ADD (JOB VARCHAR2(20));

Table altered.

SQL> DESC EMP1;

Name	Null?	Type
EMPNO		NUMBER(5)
ENAME		VARCHAR2(20)
SAL		NUMBER(7,2)
JOB		VARCHAR2(20)

SQL> alter table emp1modify(ename char(20));

Table altered.

SQL> desc emp1;

Name	Null?	Type
EMPNO		NUMBER(5)
ENAME		CHAR(20)
SAL		NUMBER(7,2)
JOB		VARCHAR2(20)

DROP TABLE:

For dropping a column and/or constraint.

SYNTAX:

alter table table name drop(column/constraint . . .);

SQL> create table emp1(empno number(5),ename varchar2(20),
sal number(7,2));

Table created.

SQL> drop table emp1;

Table dropped.

SQL> desc emp1;

ERROR:
ORA-04043: object EMP1 does not exist.

IMPLEMENTING DML COMMANDS

INSERTION:

To insert rows into a table we use INSERT command.

SYNTAX:

insert into <tab name>(col1,col2,.....)values(val1,val2,val3.....);

```
SQL> create table emp1 (eno number (5), ename varchar2 (20), sal number (7, 2), job
varchar2 (10));

Table created.

SQL> insert into emp1 values (&eno ,' &ename ', &sal ,' &job');

Enter value for eno: 1
Enter value for ename: BHANU
Enter value for sal: 30000
Enter value for job: MANAGER
old   2: VALUES (&ENO,'&ENAME',&SAL,'&JOB')
new 2: VALUES (11,'BHANU',30000,'MANAGER')

1 row created.

SQL> /
Enter value for eno: 2
Enter value for ename: SATYA
Enter value for sal: 40000
Enter value for job: SOFTWARE
old   2: VALUES(&ENO,'&ENAME',&SAL,'&JOB')
new   2: VALUES(2,'SATYA',40000,'SOFTWARE')

1 row created.
SQL> SELECT * FROM EMP1;
```

ENO	ENAME	SAL	JOB
1	BHANU	30000	MANAGER
2	SATYA	40000	SOFTWARE

UPDATING TABLE:

It allows you to change values of rows in a table.

SYNTAX:

update table_name set column=expression [where condition];

SQL> update emp1 set sal=35000 where eno=1;

1 row updated.

SQL> select * from emp1;

ENO	ENAME	SAL	JOB
1	BHANU	35000	MANAGER
2	SATYA	40000	SOFTWARE

DELETING TABLE:

It allows you to remove one or more rows from a table.

SYNTAX:

delete from table_name [where condition];

SQL> delete from emp1 where eno=2;

1 row deleted.

SQL> select * from emp1;

ENO	ENAME	SAL	JOB
1	BHANU	35000	MANAGER

IMPLEMENTING DCL and TCL COMMANDS

These commands are used to controlling and accessing the oracle database.

1. CREATE USER:
It is used for creating an account in Database.

SYNTAX:
create user<user_name>identified by <password>;

NOTE:
Before using this command the user must have dba privileges .

2. GRANT:
In multi-user RDBMS you need to be a special user –DBA to get any operation even to run a simple query. You can use the SQL-DCL statements are used for securing your database. The "grant" and "revoke" statements are used to grant and revoke the permissions to/from the users.

SYNTAX :
GRANT CONNECT [, RESOURCE FILE] [, DBA] TO USER [USER,] [IDENTIFIED BY<PWD> [, <PWD>----]];

Oracle has two categories of privileges

1. System privileges.
2. Object privileges.

1. System privileges: these privileges enable an oracle user to connect and execute statements such as create user, create table...

```
create user pavan identified by bhanu;
grant connect to prakash;
grant connect, resource to pavan;
grant connect, resource, dba to pavan;
grant connect, resource to x, y identified by a, b;
grant connect, resource, dba to x, y identified by a, b ;
```

2. SHOW USER:

This command is used to display the user name we are currently working with.

SYNTAX:

SHOW<USER> [;]

3. EXIT OR QUIT

This command is used to exit from SQL* PLUS prompt.

SYNTAX: exit [quit][;]

4. CL SCR:

This command is used to clear the screen.

SYNTAX: cl scr;

5. CLEAR BUFFER:

This command is used to clear the buffer area.

6. REVOKE:

The "Revoke" is used to take away a privilege that was granted

REVOKE CONNECT [, RESOURCE] [, DBA] FROM <USER> [, USER,,,,];

revoke connect from pavan;
revoke dba from pavan;

7. GRANT -II FORM

This format of grant command grants privileges to users with respect to table or views. These privileges are known as object privileges.

SYNTAX:

GRANT INSERT[,DELETE][,UPDATE,SELECT,ALTER,INDEX]|ALL ON <TABLE_NAME> TO USER[,USER,,,,];

8. COMMIT:

To end your current transaction and make permanent all changes performed in the transaction. This command also erases all save points in the transaction and releases the transactions locks. You can also use this command manually commit an in-doubt distributed transaction.

SYNTAX:

Commit;

9. ROLLBACK:

The DBMS backs out, or undo unwanted changes to the databases. Before images of the records that have been changed are applied to the database. As a result, the database is returned to an earlier stage.

SYNTAX:
ROLLBACK [, TO SAVE POINT<SAVEPOINT_NAME>]

Note:

Commit and rollback commands are called as TCL category commands.

10. SAVEPOINT

This command is used to create the save point location. To identify a point in a transaction to which you can later roll back. Save point is the name of the Save point to be created.

SYNTAX: SAVEPOINT <SAVEPOINTNAME>

EX:

```
SQL>INSERT      2REC
SQL>UPDATE    2REC
SQL>COMMIT
SQL>INSERT      4REC
SQL>UPDATE    2REC
SQL>ROLLBACK
```

EXCERSIZE-1

Creation of tables.

```
SQL> create table sailors (
        2 Sid varchar (10) primary key,
        3 sname varchar (20),
        4 age number (5),
        5 rating number (5));
Table created.

SQL> create table boats (
        2   bid number (10) primary key,
        3   bname varchar (20),
        4   bcolor varchar (10));
Table created.

SQL> create table reserve (
        2   Sid varchar (10),
        3   bid number (10),
        4   day date,
        5   foreign key (Sid) references sailors,
        6   foreign key (bid) references boats);
Table created.

SQL> create table Branch (
        2 brname varchar (20),
        3 brcity varchar (15),
        4 assert number (15));
Table created.

SQL> create table customer (
        2 cname varchar (20),
        3 street varchar (15),
        4 city varchar (20));
Table created.

SQL> create table borrow (
        2 cname varchar (20),
        3 loanno number (15) primary key);
Table created.
```

```
SQL> create table loan (
          2 brname varchar (20),
          3 loanno varchar (15),
          4 amount varchar (15)
         Foreign key(loan no) references borrow);
Table created.

SQL> create table depositor (
          2 cname varchar (20),
          3 accno number (20) primary key);
Table created.

SQL> create table account (
           2 brname varchar (20),
           3 balance number (15),
           4 accno number (20),
           5 foreign key (accno) references depositor);
Table created.
```

EXCERSIZE-2

Insert into data from tables

SAILORS DATA

SQL> **insert into sailors values ('&sid','&sname','&age','&rating');**
Enter value for Sid: 22
Enter value for sname: Dustin
Enter value for age: 45
Enter value for rating: 7
old 1: insert into sailors values ('&sid','&sname','&age','&rating')
new 1: insert into sailors values ('22','Dustin','45','7')

1 row created.

SQL> /
Enter value for sid: 29
Enter value for sname: Brutus
Enter value for age: 33
Enter value for rating: 1
old 1: insert into sailors values('&sid','&sname','&age','&rating')
new 1: insert into sailors values('29','Brutus','33','1')

1 row created.

SQL> **select *from sailors;**

SID	SNAME	AGE	RATING
S22	Dustin	45	7
S29	Brutus	33	1
S31	Lubber	56	8
S32	Andy	25	8
S58	Rusty	35	10
S64	Horatio	35	7
S71	Zorba	16	10
S74	Horatio	35	9
S85	Arya	25	3
S95	Bob	63	3
S20	Rajesh	22	1
S26	Kumar	30	2
S25	Srinivas	27	5

BOATS DATA

SQL> **insert into boats values ('&bid','&bname','&bcolor');**
Enter value for bid: 101
Enter value for bname: interlake
Enter value for bcolor: blue
old 1: insert into boats values('&bid','&bname','&bcolor')
new 1: insert into boats values('101','interlake','blue')

1 row created.

SQL> /
Enter value for bid: 102
Enter value for bname: interlake
Enter value for bcolor: red
old 1: insert into boats values('&bid','&bname','&bcolor')
new 1: insert into boats values('102','interlake','red')

1 row created.

SQL> **select *from boats;**

BID	BNAME	BCOLOR
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red
105	Sagar	blue

RESEVERS DATA

SQL>**insert into reserves values('&sid','&bid','&daydate');**
Enter value for sid: 22
Enter value for bid: 101
Enter value for daydate: 10-oct-2008
old 1: insert into reserve values('&sid','&bid','&daydate')
new 1: insert into reserve values('22','101','10-oct-2008')

1 row created.

SQL> /
Enter value for sid: 22
Enter value for bid: 102
Enter value for daydate: 10-oct-2008
old 1: insert into reserve values('&sid','&bid','&daydate')
new 1: insert into reserve values('22','102','10-oct-2008')

1 row created.

SQL> /
Enter value for sid: 22
Enter value for bid: 103
Enter value for daydate: 10-aug-2008
old 1: insert into reserve values('&sid','&bid','&daydate')
new 1: insert into reserve values('22','103','10-aug-2008'

1 row created.

SQL> /
Enter value for sid: 22
Enter value for bid: 104
Enter value for daydate: 10-jul-2009
old 1: insert into reserve values('&sid','&bid','&daydate')
new 1: insert into reserve values('22','104','10-jul-2009')

1 row created.

SQL> **select *from reserves;**

SID	BID	DAY
S22	**101**	**10-OCT-08**
S22	**102**	**10-OCT-08**
S22	**103**	**10-AUG-08**
S22	**104**	**10-JUL-09**
S22	**105**	**13-DEC-10**
S31	**103**	**08-DEC-10**
S31	**102**	**12-NOV-09**
S31	**105**	**11-DEC-09**
S64	**101**	**09-MAY-09**
S64	**104**	**09-AUG-10**
S74	**103**	**09-AUG-10**
S95	**105**	**13-DEC-10**

DEPOSITE DATA:

SQL> **insert into depositer values('&cname','&accno');**
Enter value for cname: Hayes
Enter value for accno: A-102
old 1: insert into depositer values('&cname','&accno')
new 1: insert into depositer values('Hayes','A-102')

1 row created.

SQL> /
Enter value for cname: Johnson
Enter value for accno: A-101
old 1: insert into depositer values('&cname','&accno')
new 1: insert into depositer values('Johnson','A-101')

1 row created.

SQL> /
Enter value for cname: Johnson
Enter value for accno: A-201
old 1: insert into depositer values('&cname','&accno')
new 1: insert into depositer values('Johnson','A-201')

1 row created.

SQL> **select *from depositer;**

CNAME	ACCNO
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

ACCOUNT DATA:

SQL> **insert into account values('&brname','&bal','&accno');**
Enter value for brname: Downtown
Enter value for bal: 5000
Enter value for accno: A-101
old 1: insert into account values('&brname','&bal','&accno')
new 1: insert into account values('Downtown','5000','A-101')

1 row created.

SQL> /
Enter value for brname: Perryridge
Enter value for bal: 4000
Enter value for accno: A-102
old 1: insert into account values('&brname','&bal','&accno')
new 1: insert into account values('Perryridge','4000','A-102')

1 row created.

SQL> /
Enter value for brname: Brighton
Enter value for bal: 9000
Enter value for accno: A-201
old 1: insert into account values('&brname','&bal','&accno')
new 1: insert into account values('Brighton','9000','A-201')

1 row created.

SQL> **select *from account;**

BRNAME	BAL	ACCNO
Downtown	5000	A-101
Perryridge	4000	A-102
Brighton	9000	A-201
Mianus	7000	A-215
Brighton	7500	A-217
Redwood	7000	A-222
Round Hill	3500	A-305

BRANCH DATA:

SQL> **insert into branch values('&brname','&brcity','&asserts');**
Enter value for brname: Brighton
Enter value for brcity: Brooklyn
Enter value for asserts: 71000
old 1: insert into branch values('&brname','&brcity','&asserts')
new 1: insert into branch values('Brighton','Brooklyn','71000')

1 row created.

SQL> /
Enter value for brname: Downtown
Enter value for brcity: Brooklyn
Enter value for asserts: 191000
old 1: insert into branch values('&brname','&brcity','&asserts')
new 1: insert into branch values('Downtown','Brooklyn','191000')

1 row created.

SQL> /
Enter value for brname: Mianus
Enter value for brcity: Horseneck
Enter value for asserts: 21000
old 1: insert into branch values('&brname','&brcity','&asserts')
new 1: insert into branch values('Mianus','Horseneck','21000')

1 row created.

SQL> **select *from branch;**

BRNAME	BRCITY	ASSERTS
Brighton	Brooklyn	71000
Downtown	Brooklyn	191000
Mianus	Horseneck	21000
North Town	Rye	351200
perryridge	Horseneck	40000
pownal	Bennington	5000
Redwood	Palo Alto	12340
Round Hill	Horseneck	1000

BORROW DATA:

SQL> **insert into borrow values('&cname','&loanno');**
Enter value for cname: Adams
Enter value for loanno: L-16
old 1: insert into borrow values('&cname','&loanno')
new 1: insert into borrow values('Adams','L-16')
1 row created.

SQL> /
Enter value for cname: Curry
Enter value for loanno: L-93
old 1: insert into borrow values('&cname','&loanno')
new 1: insert into borrow values('Curry','L-93')

1 row created.

SQL> /
Enter value for cname: Jackson
Enter value for loanno: L-15
old 1: insert into borrow values('&cname','&loanno')
new 1: insert into borrow values('Jackson','L-15')

1 row created.

SQL> **select *from borrow;**

CNAME	LOANNO
Adams	L-16
Curry	L-93
Jackson	L-15
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-14

LOAN DATA:

SQL> **insert into loan values('&brname','&loanno','&amount');**
Enter value for brname: Downtown
Enter value for loanno: L-14
Enter value for amount: 15000
old 1: insert into loan values('&brname','&loanno','&amount')
new 1: insert into loan values(' Downtown','L-14','15000')

1 row created.

SQL> /
Enter value for brname: perryridge
Enter value for loanno: L-15
Enter value for amount: 15000
old 1: insert into loan values('&brname','&loanno','&amount')
new 1: insert into loan values('perryridge','L-15','15000')

1 row created.

SQL> /
Enter value for brname: perryridge
Enter value for loanno: L-16
Enter value for amount: 13000
old 1: insert into loan values('&brname','&loanno','&amount')
new 1: insert into loan values('perryridge','L-16','13000')

1 row created.

SQL> **select *from loan;**

BRNAME	LOANNO	AMOUNT
Round Hill	L-11	9000
Downtown	L-14	15000
perryridge	L-15	15000
perryridge	L-16	13000
Downtown	L-17	10000
Redwood	L-23	20000
mianus	L-93	5000

EMP DATA:

SQL> **select *from EMP;**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		**10**

EXCERSIZE-3

DUAL-COMMANDS

1. SQL> select sysdate from dual;

SYSDATE

06-DEC-10

2. SQL> select 30+40 from dual;

30+40

70

3. SQL> select 60-26 from dual;

60-26

34

4. SQL> select 30*40 from dual;

30*40

1200

5. SQL> select 30/40 from dual;

30/40

.75

6. SQL> select abs (-60) from dual;

ABS (-60)

60

7. SQL> select power (2, 5) from dual;

POWER (2,5)

32

8. SQL> select sqrt (25) from dual;

SQRT (25)

5

9. SQL> select mod (15.56, 1) from dual;

MOD (15.56,1)

.56

10. SQL> select round (7.88) from dual;

ROUND (7.88)

8

11. SQL> select ceil (87.78) from dual;

CEIL (87.78)

88

12. SQL> select floor (87.78) from dual;

FLOOR (87.78)

87

13. SQL> select substr ('srinivasarao', 2, 4) from dual;

SUBSTR

---

rini

14. SQL> select length ('purna Chandra rao') from dual;

LENGTH ('PURNACHANDRARAO')

17

15. SQL> select lower ('RAGHAVA') from dual;

LOWER ('RAGHAVA')

---

raghava

16. SQL> select upper ('raghava') from dual;

UPPER ('raghava')

RAGHAVA

17. SQL> select ltrim ('srinivasarao','sr') from dual;
LTRIM (‘srinivasarao’,’sr’)
---
inivasarao
18. SQL> select rtrim ('srinivasarao','rao') from dual;
RTRIM ('srinivasarao’,’rao’)

Srinivasa
19. SQL> select lpad ('ARAVIND', 6,'$') from dual;
LPAD (‘ARAVIND’,6,’$’)

$$$ARAVIND

20. SQL> select rpad ('RAGHAVA', 8,'@')from dual;
RPAD ('RAGHAVA’,8,’@’)

RAGHVA@@@@@

21. SQL> select trunc (45.923, 1) from dual;
TRUNC (45.923, 1)

45.9
22. SQL> select exp(4) from dual;
EXP (4)

54.59815

23. SQL> select sysdate, next_day (sysdate,'friday') from dual;
SYSDATE NEXT_DAY (sysdate,’friday’)
--- ---
08-DEC-10 10-DEC-10

24. SQL> select to_char (sysdate,'day, ddth month yyyy') from sys.dual;
TO_CHAR (SYSDATE,'DAY, DDTH MONTH YYYY’)

Wednesday, 08th December 2010

25. SQL> select last_day('2-feb-2011') from dual;
LAST_DAY(‘2-feb-2011)

28-FEB-11

26. SQL> select chr(105) from dual;

CHR(105)

---

i

27. SQL> select concat('IN','DIA') from dual;

CONCAT('IN','DIA')

INDIA

28. SQL> select replace('Amit and Sumit','mit','zi') from dual;

REPLACE('AMIT AND SUMIT','MIT','ZI')

Azi and Suzi

29. SQL> select initcap('raghavulu') from dual;

INITCAP('raghavulu')

Raghavulu

30. SQL> select add_months('5-mar-2011','3') from dual;

ADD_MONTHS('5-mar-2011','3')

05-JUN-11

31. SQL> select months_between('5-mar-2011','5-aug-2011') from dual;

MONTHS_BETWEEN('5-MAR-2011','5-AUG-2011')

-5

32. SQL> select round('5-mar-2011','year') from dual;

Round('26-nov-05','year')

1-jan-06

33. SQL> select trunc('5-mar-2011','month') from dual;

TRUNC(26-nov-05','month')

1-nov-05

34. SQL> select uid from dual;

UID

20

35. SQL> select user from dual;

USER

SCOTT

EXCERSIZE-4

QUERIES:

1. List all departments' numbers, employee names and manager numbers in the EMP table.

SQL> **select deptno, ename, mgr from EMP;**

DEPTNO	ENAME	MGR
20	SMITH	7902
30	ALLEN	7698
30	WARD	7698
20	JONES	7839
30	MARTIN	7698
30	BLAKE	7839
10	CLARK	7839
20	SCOTT	7566
10	KING	
30	TURNER	7698
20	ADAMS	7788
30	JAMES	7698
20	FORD	7566
10	MILLER	7782

2. Display the column heading ANSAL for annual salary instead of Sal*12 using Column alias.

SQL> select ename, sal*12 as annsal from EMP;

ENAME	ANNSAL
SMITH	9600
ALLEN	19200
WARD	15000
JONES	35700
MARTIN	39000
BLAKE	34200
CLARK	29400
SCOTT	36000
KING	60000
TURNER	18000
ADAMS	13200
JAMES	11400
FORD	36000
MILLER	15600

3. Combine empno, ename and give alias employee to the expression**.**

SQL> select empno||ename as employee from EMP;

```
EMPLOYEE
-----------------------------------------
7369SMITH
7499ALLEN
7521WARD
7566JONES
7654MARTIN
7698BLAKE
7782CLARK
7788SCOTT
7839KING
7844TURNER
7876ADAMS
7900JAMES
7902FORD
7934MILLER
```

4. Write a statement that contains literals selected with Concatenation and a column Alias as employee

SQL> select empno||'-'|| ename as employee from EMP;

```
EMPLOYEE
---------------------------------------
7369-SMITH
7499-ALLEN
7521-WARD
7566-JONES
7654-MARTIN
7698-BLAKE
7782-CLARK
7788-SCOTT
7839-KING
7844-TURNER
7876-ADAMS
7900-JAMES
7902-FORD
7934-MILLER
```

5. Write a statement that displays NULL if any column values in expression are null.

SQL> select ename, nvl (to_char (Sal),'NULL') as sal from EMP;

ENAME	SAL
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	3250
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950
FORD	3000
MILLER	1300

6. Write a statement that uses NVL function null values to zero.

SQL> select ename, sal*12+nvl (comm, 0) as annsal from emp;

ENAME	ANNSAL
SMITH	9600
ALLEN	19500
WARD	15500
JONES	35700
MARTIN	40400
BLAKE	34200
CLARK	29400
SCOTT	36000
KING	60000
TURNER	18000
ADAMS	13200
JAMES	11400
FORD	36000
MILLER	15600

7. Display distinct values of deptno, job.

SQL> select distinct deptno, job from EMP;

DEPTNO	JOB
10	CLERK
10	MANAGER
10	PRESIDENT
20	ANALYST
20	CLERK
20	MANAGER
30	CLERK
30	MANAGER
30	SALESMAN

8. Write a statement to sort by ename.

SQL> select *from EMP order by ename;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7654	MARTIN	SALESMAN	7698	28-SEP-81	3250	1400	30
7934	MILLER	CLERK	7782	23-JAN-82	1300		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30

9. Reverse the order of Hire date column so that latest dates are displayed first.

SQL> select *from EMP order by hiredate desc;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7654	MARTIN	SALESMAN	7698	28-SEP-81	3250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7369	SMITH	CLERK	7902	17-DEC-80	800		20

10. Find all employees whose commission is greater than their salary.

SQL> select *from EMP where comm>sal;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30

11. Display the employees list those salary between 1000 and 2000.

SQL> select *from EMP where sal>=1000 and sal<=2000;

(Or)

SQL> select *from EMP where sal between 1000 and 2000;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

12. List all employees whose name starts with‘s’.

SQL> select *from EMP where ename like 'S%';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20

13. List the names, numbers, job and departments of all clerks.

SQL > select ename, empno, dname, job from EMP e, dept d where e.deptno=d.deptno and e.job='CLERK';

ENAME	EMPNO	DNAME	JOB
SMITH	7369	RESEARCH	CLERK
ADAMS	7876	RESEARCH	CLERK
JAMES	7900	SALES	CLERK
MILLER	**7934**	**ACCOUNTING**	**CLERK**

14. List all employees who have a name exactly of four characters in length.

SQL> select *from EMP where ename like ('____');

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	
7839	KING	PRESIDENT		17-NOV-81	5000		10
7902	FORD	ANALYST	7566	03-DEC-81	3000		20

15. Find all employees whose job does not start with “M”

SQL> select *from EMP where job not like 'M%';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

16. Find all employees who have a manager.

SQL> select *from EMP where job like 'M%';

(Or)

SQL> select *from EMP where job='MANAGER';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10

17. Display all employees name which have ‘TH’ or ‘LL’ in their names.

SQL> select *from EMP where ename like '%TH%' or ename like '%LL%';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

18. Display name and total remuneration for all employees.

SQL> select ename, sal*12+nvl (comm, 0) as remuneration from EMP;

ENAME	REMUNERATION
SMITH	9600
ALLEN	19500
WARD	15500
JONES	35700
MARTIN	16400
BLAKE	34200
CLARK	29400
SCOTT	36000
KING	60000
TURNER	18000
ADAMS	13200
JAMES	11400
FORD	36000
MILLER	15600

19. Write an SQL stmt that prompts the user for department number at runtime.

SQL> select *from dept where deptno='&departmentno';

```
Enter value for departmentno: 10
old   1: select *from dept where deptno='&departmentno'
new   1: select *from dept where deptno='10'
 DEPTNO    DNAME              LOC
--------- -------------- ------------------------------
  10             ACCOUNTING     NEW YORK
```

20. Define a variable to contain an arithmetic expression that calculate remuneration, in the subsequent statement use this variable no of times and empty it.

SQL> define exp=sal*12+nvl(comm,0)

SQL> select ename,&exp as remuneration from EMP;

```
old     1: select ename,&exp as remuneration from emp
new    1: select ename, sal*12+nvl(comm,0) as remuneration from emp
ENAME      REMUNERATION
---------- ---------------------------------
SMITH            9600
ALLEN           19500
WARD            15500
JONES            35700
MARTIN          16400
BLAKE            34200
CLARK           29400
SCOTT           36000
KING             60000
TURNER          18000
ADAMS            13200
JAMES            11400
FORD             36000
MILLER           15600
```

21. Display the department names and locations in mixed case.

SQL> select INITCAP (dname), INITCAP (loc) from dept;

```
INITCAP(DNAME)              INITCAP(LOC)
-------------- ----------------------------------
Accounting                  New York
Research                    Dallas
Sales                       Chicago
Operations                  Boston
```

22. Write a stmt that concatenates ename and job.

SQL> select concat (ename, job) from EMP;

CONCAT(ENAME,JOB)

SMITHCLERK
ALLENSALESMAN
WARDSALESMAN
JONESMANAGER
MARTINSALESMAN
BLAKEMANAGER
CLARKMANAGER
SCOTTANALYST
KINGPRESIDENT
TURNERSALESMAN
ADAMSCLERK
JAMESCLERK
FORDANALYST
MILLERCLERK

23. Write a command that will remove all trailing blanks.

SQL> select rtrim (ename,' ') from EMP;

RTRIM(ENAME,' ')

SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER

24. Translate the char's for given arrtibutes for deptno=10.

SQL> select ename, translate (ename,'C','P') as names, job, translate (job,'AR','IT') as jobs from EMP where deptno=10;

```
ENAME      NAMES      JOB                JOBS
---------- ---------- ------------------ --------------------------------------------------
CLARK      PLARK      MANAGER            MINIGET
KING       KING       PRESIDENT          PTESIDENT
MILLER     MILLER     CLERK              CLETK
```

25. Apply dual command by sal in employee table.

SQL> select trunc (sal/32, 2) from EMP where deptno=20;

```
TRUNC(SAL/32,2)
---------------------------------------
         25
      92.96
      93.75
      34.37
      93.75
```

26. List the employee name and salary increment by 15% and expressed as a whole numbers of dollars for deptno=30.

SQL> select ename, to_char (sal+sal*15/100,'$9999') as tsal from EMP where deptno=30;

```
ENAME      TSAL
---------- --------------------
ALLEN       $1840
WARD        $1438
MARTIN      $1438
BLAKE       $3278
TURNER      $1725
JAMES       $1093
```

27. Display the hire date operations of the EMP table like add and subtraction of dates.

SQL> select HIREDATE, HIREDATE+7, HIREDATE-7, sysdate-HIREDATE from EMP where HIREDATE like '%JUN%';

```
HIREDATE HIREDATE+7 HIREDATE-7 SYSDATE-HIREDATE
--------- --------- --------- --------------------------------------------------
09-JUN-81 16-JUN-81 02-JUN-81          10853.583
```

28. Display months between system date and given hire date; and display months bet'n given hire dates.

SQL> select months_between (sysdate, HIREDATE) as month1, months_between ('01- jan-81','05-nov-88') as month2 from EMP where months_between (sysdate, HIREDATE)>59;

```
 MONTH1    MONTH2
--------- ----------------------------
362.27687   -94.12903
360.18009   -94.12903
360.11558   -94.12903
358.76074   -94.12903
352.92203   -94.12903
357.793     -94.12903
356.53493   -94.12903
286.21235   -94.12903
351.27687   -94.12903
353.56719   -94.12903
285.08332   -94.12903
350.72848   -94.12903
350.72848   -94.12903
349.08332   -94.12903
```

29. Display the queries in the addition of months from hire date in a given department.

SQL> select hiredate, add_months (HIREDATE,4),add_months(HIREDATE,-4) from emp where deptno=20;

```
HIREDATE  ADD_MONTHS(HIREDATE,4) ADD_MONTHS(HIREDATE,-4)
--------- --------- -------------------------------------------------------------------------------
17-DEC-80 17-APR-81                       17-AUG-80
02-APR-81 02-AUG-81                       02-DEC-80
19-APR-87 19-AUG-87                       19-DEC-86
23-MAY-87       23-SEP-87                       23-JAN-87
03-DEC-81 03-APR-82                       03-AUG-81
```

30. Write a SQL stmt to display the salary increase according to job type.

SQL>select job,decode(job,'PRESIDENT','20%','MANAGER','17%', 'ANALYST','15%','SALESMAN','12%','CLERK','10%') as sal_inc from emp;

JOB	SAL_INC
CLERK	10 %
SALESMAN	12%
SALESMAN	12%
MANAGER	17%
SALESMAN	12%
MANAGER	17%
MANAGER	17%
ANALYST	15%
PRESIDENT	
SALESMAN	12%
CLERK	10 %
CLERK	10 %
ANALYST	15%
CLERK	10 %

31. Calculate the average salary of each different jobs.

SQL> select avg (sal), job from EMP group by job;

AVG(SAL)	JOB
3000	ANALYST
1037.5	CLERK
2758.3333	MANAGER
5000	PRESIDENT
1400	SALESMAN

32. Display the average salary for each job excluding managers.

SQL> select avg (sal), job from EMP where job! ='MANAGER' group by job;

AVG(SAL)	JOB
3000	ANALYST
1037.5	CLERK
5000	PRESIDENT
1400	SALESMAN

33. Display the average salary for all departments employing more than three people.

SQL> select deptno, avg (sal) from EMP group by deptno having count (*)>3;

DEPTNO	AVG(SAL)
20	2175
30	1566.6667

34. Display only those jobs where the maximum salary is greater than or equal to $3000.

SQL> select job, max (sal) from EMP having max (sal)>=3000 group by job;

JOB	MAX(SAL)
ANALYST	3000
PRESIDENT	5000

35. Find the minimum, maximum and average salaries of all employees**.**

SQL> select min (sal), max (sal), avg (sal) from EMP;

MIN(SAL)	MAX(SAL)	AVG(SAL)
800	5000	2073.2143

36. List the minimum, maximum salary for each job type.

SQL> select job, min (sal), max (sal) from EMP group by job;

JOB	MIN(SAL)	MAX(SAL)
ANALYST	3000	3000
CLERK	800	1300
MANAGER	2450	2975
PRESIDENT	5000	5000
SALESMAN	1250	1600

37 Join two tables EMP and dept based on deptno.

SQL > select empno, ename, dname from EMP e, dept d where e.deptno=d.deptno;

EMPNO	ENAME	DNAME
7369	SMITH	RESEARCH
7499	ALLEN	SALES
7521	WARD	SALES
7566	JONES	RESEARCH
7654	MARTIN	SALES

7698	BLAKE	SALES
7782	CLARK	ACCOUNTING
7788	SCOTT	RESEARCH
7839	KING	ACCOUNTING
7844	TURNER	SALES
7876	ADAMS	RESEARCH
7900	JAMES	SALES
7902	FORD	RESEARCH
7934	MILLER	ACCOUNTING

38. Evaluate an employee grade based on their salary between any pair of the low and high salary ranges.

SQL > select ename, sal, grade from EMP e, salgrade s where e.sal between s.losal and s.hisal;

ENAME	SAL	GRADE
SMITH	800	1
ADAMS	1100	1
JAMES	950	1
ALLEN	1600	3
TURNER	1500	3
JONES	2975	4
BLAKE	2850	4
CLARK	2450	4
SCOTT	3000	4
FORD	3000	4
KING	5000	5

39. Display all employee names and their departments name in department order.

SQL > select ename, dname from EMP e, dept d where e.deptno=d.deptno order by dname;

ENAME	DNAME
CLARK	ACCOUNTING
KING	ACCOUNTING
MILLER	ACCOUNTING
SMITH	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
SCOTT	RESEARCH

JONES RESEARCH
ALLEN SALES
BLAKE SALES
MARTIN SALES
JAMES SALES
TURNER SALES
WARD SALES

40. Display the name, location and departments of all employees whose salary is more than 1500.

SQL > select ename, loc, dname from EMP e, dept d where sal>1500 and e.deptno=d.deptno;

ENAME	LOC	DNAME
ALLEN	CHICAGO	SALES
JONES	DALLAS	RESEARCH
BLAKE	CHICAGO	SALES
CLARK	NEW YORK	ACCOUNTING
SCOTT	DALLAS	RESEARCH
KING	NEW YORK	ACCOUNTING
FORD	DALLAS	RESEARCH

41. Shows the employees on grade 3.

SQL > select ename, losal, hisal, grade from EMP e, salgrade s where s.grade=3 and e.sal between s.losal and s.hisal;

ENAME	LOSAL	HISAL	GRADE
ALLEN	1401	2000	3
TURNER	1401	2000	3

42. Show all employees in DALLAS.

SQL > select ename, loc from EMP e, dept d where d.loc='DALLAS' and e.deptno=d.deptno;

ENAME	LOC
SMITH	DALLAS
JONES	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
FORD	DALLAS

43. Find the average age of sailors?

SQL> select avg(age) from sailors;

(OR)

SQL> select avg(s.age) from sailors s;

```
AVG(AGE)
---------
34.384615
```

44. Find the average age of sailors with rating 10?

SQL> select avg(s.age) from sailors s where s.rating=10;

```
AVG(S.AGE)
---------
25.5
```

45. Find the name and age of the oldest sailor?

SQL> select sname,age from sailors where age=(select max(age) from sailors);

```
SNAME                AGE
-------------------- --------------------
Bob                  63
```

46. Count the number of sailors?

SQL> select count(*) from sailors;

```
COUNT(*)
--------------------
   13
```

47. Find the names and ages of all sailors?

SQL> select sname,age from sailors;

```
SNAME                AGE
-------------------- ---------
Dustin               45
Brutus               33
Lubber               56
Andy                 25
Rusty                35
Horatio              35
Zorba                16
Horatio              35
Arya                 25
Bob                  63
Rajesh               22
```

Kumar 30
Srinivas 27

48. Find all the sailors with rating above 7?

SQL> select *from sailors where rating>7;

SID	SNAME	AGE	RATING
S31	Lubber	56	8
S32	Andy	25	8
58	Rusty	35	10
S71	Zorba	16	10
S74	Horatio	35	9

49. Find the names of sailors who have reserved boat number=103?

SQL> select s.sname from sailors s,reserve r where r.bid=103 and s.sid=r.sid;

SNAME

Dustin
Lubber

50. Find the SID of sailors who have reserved red boat.

SQL> select r.sid from reserve r, boats b where b.bcolor='red' and r.bid=b.bid;

(OR)

SQL> select s.sid from sailors s, reserve r, boats b where s.sid=r.sid and r.bid=b.bid and b.bcolor='red';

SID

S22
S22
S31
S64

51. Find the color of boats reserved by ‘LUbbER’?

SQL>select b.bcolor from boats b, sailors s, reserve r where s.sname='Lubber' and r.sid=s.sid and r.bid=b.bid;

Bcolor

Red
Green
Red

52. Find the name of sailors who have reserved a red boat or a green boat?

SQL> select s.sname from sailors s, reserve r, boats b where s.sid=r.sid and r.bid=b.bid and (b.bcolor='red' or b.bcolor='green');

(OR)

SQL> select s.sname from sailors s, reserve r, boats b where s.sid=r.sid and r.bid=b.bid and b.bcolor='red' union
(select s1.sname from sailors s1, reserve r1, boats b1 where s1.sid=r1.sid and r1.bid=b1.bid and b1.color='green');

```
SNAME
--------------------
Dustin
Dustin
Dustin
Lubber
Lubber
Horatio
Horatio
```

53. Find the names of sailors who have reserved a red boat and green boat?

SQL> select s.sname from sailors s, reserve r, boats b where s.sid=r.sid and r.bid=b.bid and b.bcolor='red' intersect
select s1.sname from sailors s1, reserve r1, boats b1 where s1.sid=r1.sid and r1.bid=b1.bid and b1.bcolor='green';

```
SNAME
--------------------
 Dustin
Horatio
Lubber
```

54. Find the name of sailors who have reserved red boat but not green boat?

SQL> select s.sname from sailors s, reserve r, boats b where s.sid=r.sid and r.bid=b.bid and b.bcolor='red' minus
(select s1.sname from sailors s1, reserve r1, boats b1 where s1.sid=r1.sid and r1.bid=b1.bid and b1.bcolor='green');

```
 SNAME
--------------------
Dustin
Dustin
Lubber
Horatio
```

55. Find the name of sailors who have rating of 10 or have reserved boat 104?

SQL> select s.sname from sailors s, reserve r where (s.rating=10 or r.bid=104) and r.sid=s.sid;

```
SNAME
--------------------
 Rusty
Zorba
Dustin
```

56. Find the names and ages of all sailors?

SQL> select distinct s.sname, s.age from sailors's;

```
SNAME                AGE
-------------------- ----------------------
 Dustin                        45
Andy                           25
Arya                           25
Bob                            63
Brutus                         33
Horatio                        35
Kumar                          30
Lubber                         56
Rajesh                         22
Rusty                          35
Srinivas                       27
Zorba                          16
```

57. Find the names of sailors who have reserved at least one boat?

SQL> select s.sname from sailors s,reserve r where s.sid=r.sid;

```
SNAME
--------------------
 Dustin
 Dustin
 Dustin
 Dustin
 Dustin
Lubber
Lubber
Lubber
Horatio
Horatio
Bob
```

58. Compute increments for the rating's of persons who have sailed two different boats on the same day?

SQL> select s.sname, s.rating, s.rating+1 as ratinginc from sailors s,reserve r1,reserve r2 where s.sid=r1.sid and s.sid=r2.sid and r1.day=r2.day and r1.bid<>r2.bid;

Sname	rating	ratinginc
Dustin	7	8
Dustin	7	8

59. Find the ages of sailors whose name begins and ends with 'B' and has at least three characters.

SQL> select s.age from sailors s where s.sname like 'B_%B';

Age
63

60. Find the names of sailors who have reserved boat 03 using nested queries

SQL> select s.sname from sailors s where s.sid in(select r.sid from reserve r wherer.bid=103);

. sname
Dustin
Lubber

61. Find the names of sailors who have reserved for red boat?

SQL> select s.sname from sailors s where s.sid in(select r.sid from reserve r where r.bid in(select b.bid from boats b where b.bcolor='red'));

sname
Dustin
Lubber
Horatio

62. Find the names of sailors who have not reserved for red boat?

SQL> select s.sname from sailors s where s.sid not in(select r.sid from reserve r where r.bid in(select b.bid from boats b where b.bcolor='red'));

sname
Brutus
Andy

Rusty
Zorba
Horatio
Arya
Bob
Rajesh
Kumar
Srinivas

63. Find the names of sailors whose rating is better than some sailor called 'HORATIO'.

SQL> select s.sid,s.sname from sailors s where s.rating>any(select s1.rating from sailors s1 where s1.sname='horatio');

. sid	sname
S31	lubber
S32	andy
S58	rusty
S71	zorba
S74	horatio

64. Find the names of sailors whose rating is better than all sailor called 'HORATIO'.

SQL>select s.sid,s.sname from sailors s where s.rating>all(select s1.rating from sailors s1 where s1.sname='horatio');

sid	sname
S58	rusty
S71	zorba

65. Find the age of youngest sailor for each rating level?

SQL> select s.rating,min(s.age) from sailors s group by s.rating;

. rating	min(s.age)
1	22
2	30
3	25
5	27
7	35
8	25
9	35

66. Find the age of youngest sailor who is eligible for each rating level with at least two such sailors?

SQL> select s.rating,min(s.age) from sailors s where s.age>18 group by s.rating having count(*)>1;

. rating	min(s.age)
1	22
3	25
7	35
8	25

67. Find the average age of sailors for each rating level that has at least two sailors?

SQL>select s.rating,avg(s.age) from sailors s where s.age>18 group by s.rating having count(*)>1;

. rating	avg(s.age)
1	27.5
3	44
7	40
8	40.5

68. For each red boat find the number of reservations for this boat?

SQL> select b.bid,count(*) as rescount from boats b,reserve r where r.bid=b.bid and b.bcolor='red' group by b.bid;

. bid	rescount
102	2
104	2

69. Find the average of sailors who are voting age for each rating level that has at least two such sailors?

SQL> select s.rating,avg(s.age)as average from sailors s where s.age>18 group by s.rating having 1<(select count(*) from sailors s1 where s.rating=s1.rating and s1.age>=18);

rating	average
1	27.5
3	44
7	40
8	40.5

70. Find the minimum and maximum ages of all sailors

SQL>select min(s.age),max(s.age) from sailors s;

```
        . min(s.age)         max(s.age)
----------------------------------------------------------
                16              63
```

71. Find the names of all branches in the loan relation.

SQL> select l.brname from loan l;

```
 BRNAME
--------------------
Roundhill
Downtown
Perryridge
Perryridge
Downtown
Redwood
Mianus
```

72. Find the loan number for loans made at the perryridge

SQL> select l.loanno from loan l where l.amount>10000;

```
 LOANNO
--------------------
L-14
L-15
L-16
L-23
```

73. Find the loan numbers of these loans with loan amount>=10000 and <=15000.

SQL> select l.loanno from loan l where l.amount between 10000 and 15000;

```
LOANNO
--------------------
L-14
L-15
L-16
L-17
```

74. Find all customers who have a loan on the bank display their names and numbers?

SQL>select b.cname,b.loanno from borrow b,loan l where b.loanno=l.loanno;

CNAME	LOANNO
Smith	L-11
Williams	L-14
Jackson	L-15
Adams	L-16
Jones	L-17
Smith	L-23
Curry	L-93

75.Find the names and loan numbers of all the customers who have a loan at perryridge branch?

SQL>select b.cname,l.loanno from borrow b,loan l where b.loanno=l.loanno and l.brname='perryridge';

Cname	loanno
Jackson	L-15
Adams	L-16

76. Find the names of all branches that have asserts reater than at least one branch loacated in Horseneck.

SQL>select b.brname from branch b,branch b1 where b.asserts>b1.asserts and b1.brcity='Horseneck';

BRNAME

Brighton
Downtown
North Town
perryridge
Brighton
Downtown
North Town
Brighton
Downtown
Mianus
North Town
perryridge
pownal
Redwood

77. Find all customers whose name starts with ‘p’?

SQL>select c.cname from customer c where cname like ‘p%’;

CNAME

Johnson
Jones
Jackson

78. Find the names of all customers whose street address includes the substring ‘nag’;

SQL>select c.cname from customer c where c.street like ‘%nag%’;

CNAME

Jones
Lindsay
Turner
Adams
Jackson
Williams

79. Find the loanno’s and branch name and increments the amount of 100 times from loan relation.

SQL>select brname,loanno,amount*100 from loan;

. brname	loanno	amount*100
Roundhill	l-11	900000
Downtown	l-14	1500000
Perryridge	l-15	1500000
Perryridge	l-16	1300000
Downtown	l-17	1000000
Redwood	l-23	2000000
Mianus	l-93	500000

80. List the entire loan relation in desending order of amount if several loans have the same amount order then in ascending order by loan no?

SQL>select *from loan order by amount desc, loanno asc;

BRNAME	LOANNO	AMOUNT*100
Roundhill	L-11	900000
Downtown	L-14	1500000
Perryridge	L-15	1500000

Perryridge	L-16	1300000
Downtown	L-17	1000000
Redwood	L-23	2000000
Mianus	L-93	500000

81.Find the all customers names having loan or account or both .

SQL>select cnmae from borrow union (select cnmae from depositer);

CNAME

Adams
Curry
Hayes
Jackson
Johnson
Jones
Lindsay
Smith
Turner
Williams

82. Find all customers who have an account but not no loan at the bank?

SQL> SQL> select cname from depositet minus(select cname from borrow);

CNAME

Hayes
Johnson
Lindsay
Turner

83. Find the average balance at perryridge branch.

SQL>select avg(balance) from account where brname='perryridge';

AVG(BAL)

6142.8571

84. Find the average account balance from each branch.

SQL>select brname, avg(bal) from account group by brname;

BRNAME	AVG(BAL)
Brithton	8250
Downtown	5000

Mianus	7000
Perryridge	4000
Redwood	7000
Roundhill	3500

85. Find the number of depositers for each branch?

SQL>select a.brname,count(distinct d.cname) depositer d,account a where a.accno=d.accno group by a.brname;

BRNAME	COUNT(DISTINCTD.CNAME)
Brithton	2
Downtown	1
Mianus	1
Perryridge	1
Redwood	1
Roundhill	1

86. Find the average balance for each branch ,avg bal of branch must be greater than 7000.

SQL>select brname,avg(bal) from account group by brname having avg(bal)>7000;

BRNAME	AVG(BAL)
Brithton	8250

87. Find the customers who have both a loan and an account at the bank?

SQL>select cname from borrow where cname IN(select cnmae from depositer);

CNAME
Jones
Smith
Smith

88. Find all the customers who have in the city pune

SQL>select *from customer c where c.city='pune';

Cname	street	city
Hayes	X-roads	pune
Smith	sardar road	pune
Curry	X-roads	pune

89. Find all customers who have a loan at bank but do not have an account at the bank?

SQL> select cname from borrow where cnmae NOT IN(select cnmae from depositer);

Cname

Jones
Smith
Smith

90. Find the names of all branches that have asserts greater than those of at least one branch located in Brooklyn

SQL>select b.brname from brach b where b.asserts>some(select s,asserts from branch b1 where b1.brcity='Brooklyn';

BRNAME

Downtown
North Town

91. Find all names of all branches that have asserts greater than that of each branch in 'BROOKLYN'.

SQL>select b.brname from branch b where b.asserts>all(select b1.assert from branch b1 where b1.city='Brookyln';

BRNAME

North Town

92. Find the branches that have highest average balance.

SQL>select brname having avg(balance)>=all(select avg(balance) from account group by bname);

BRNAME

Brithton

93. Find highest amount from each branch.

SQL>select brname,max(amount) from loan group by brname;

BRNAME	MAX(AMOUNT)
Downtown	15000
Mianus	5000
Perryridge	15000

Redwood	20000
Roundhill	9000

94. Find the customer name in the city pune.

SQL>select c.cname from customer c where c.city like 'pene';

CNAME

Hayes

95. Find all customer details.

SQL>select *from customer;

Cname	street	city
Hayes	X-roads	pune
Johnson	patelroad	Mumbai
Jones	nehrunagar	solapur
Lindsay	nashik	nagar nashik
Smith	sardar road	pune
Turner	sivajinagr	Mumbai
Adams	sivaji nagar	Mumbai
Curry	X-roads	pune
Jackson	nehrunagar	solapur
Williams	surya nagar	kohlapur

96. Write an sql statement that useruse an 'upper' function to user input in capitals.

SQL>select *from emp where job=upper('clerk');

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

97. Display customer names in lower case letters**.**

SQL>select *from customer where cname=lower(cname);

Cname	street	city
hayes	x-roads	pune
johnson	patel road	mumbai

98.Display customer names in upper case case lettwers.

SQL>select *from customer where cname=upper(cname);

Cname	street	city
HAYES	X-roads	pune
JOHNSON	patelroad	Mumbai
JONES	nehrunagar	solapur
LINDSAY	nashik	nagar nashik
SMITH	sardar road	pune
TURNER	sivajinagr	Mumbai
ADAMS	sivaji nagar	Mumbai
CURRY	X-roads	pune
JACKSON	nehrunagar	solapur
WILLIAMS	surya nagar	kohlapur

99. Display customer names in starting letter is capital letter.

SQL>select INITCAP(cname) from customer;

Cname
Hayes
Johnson
Jones
Lindsay
Smith
Turner
Adams
Curry
Jackson
Williams

100. Apply the customer names in LPAD and RPAD.

SQL>select LPAD(cname,10,'*'),RPAD(cname,10,'*') from customer;

Lpad	Rpad
*****hayes	hayes*****
***Johnson	Johnson***
*****jones	jones*****
***Lindsay	Lindsay***
*****smith	smith*****
****turner	turner****

EXCERSIZE-5

UPDATE QUERIES:

1. Supports that annual interest payments are being made and all balances are to be increased by 5%.

SQL> update account set balance=balance+(balance*0.05);

2 rows updated.

SQL> select *from acount;

BRNAME	BALANCE	ACCNO
downtown	5250	A-101
perryridge	4200	A-102

2. Suppose that account balances 5000/- receives 10% interest.

SQL> update account set balance=balance+(balance*0.1) where balance>5000;

1 row updated.

SQL> select *from acount;

BRNAME	BALANCE	ACCNO
downtown	5775	A-101
perryridge	4200	A-102

3. Pay 5% interest for all accounts balance whose balance is >=avg.

SQL> update account set balance=balance+(balance*0.05) where balance>=(select avg(balance) from acount);

1 row updated.

SQL> select *from acount;

BRNAME	BALANCE	ACCNO
downtown	6064	A-101
perryridge	4200	A-102

DELETION QUERIES:

1. Delete all tuples from loan relation**.**

SQL> delete from loan;

2 rows deleted.

2. Delete all of smith account records?

SQL> delete from depositor where cname='smith';

1 row deleted**.**

3. Delete all loan amounts between 2300 and 7500?

SQL> delete from loan where amount between 2300 and 7500;

rows deleted.

4. Delete all amounts with balances below the average at the branches?

SQL>delete from account where balance<(select avg(balance) from account);

3 rows deleted.

EXCERSIZE-6

VIEWS:

1. Create a view to retrieve the attributes of name, age, rating from sailors whose rating level is above 7.

SQL>create view v1 as select sname,age,rating from sailors where rating>7;
View created.

SQL> save v1.sql;
Created file v1.sql
SQL> select *from v1;

SNAME	SAGE	SRATING
mani	22	6
banu	23	7
sai	24	8
rani	25	9
raju	26	10

SQL> update v1 set sage=42 where srating=10 and sname='raju';
1 row updated.

SQL> select *from v1;

SNAME	SAGE	SRATING
mani	22	6
banu	23	7
sai	24	8
rani	25	9
raju	42	10

SQL> delete from v1 where srating=10;
1 row deleted**.**

SQL> select *from v1;

SNAME	SAGE	SRATING
mani	22	6
banu	23	7
sai	24	8
rani	25	9

2. Create a view to retrieve the rating and minimum age of all sailors for each rating level.

SQL>create view v2(rate,minimum) as select rating, min(age) from sailors group by rating;

View created.

SQL>select *from v2;

Rate	minimum
6	18
7	22
8	52
9	21
10	35

3. Create a view to select sailors name and bids of sailors.

SQL> create view v3(sname,bid) as select sname,bid from sailors s,reserves r where s.sid=r.sid;

View created.

SQL> select *from v3;

Sname	bid
Mani	101
Raju	103

4. Create a view to retrieve data from all relations whose selected for boat color is red and green.

SQL>create view v4 as select s.sname,s.rating from sailors ,reserves r,boats b where s.sid=r.sid and r.bid=b.bid and b.color='red' union select s.sname ,s.rating from sailors s, reserves r,boats b where s.sid=r.sid and r.bid=b.bid and b.color='green';

View created.

SQL>select *from v4;

Sname	rating
Mani	8
Rahul	10

5. Create a view to retrieve data from relations where reserve boats are selected by atleats two boats for sailors.

SQL>create view v5 as select s.rating,r.bid from sailors s,reserves r,reserves r1 where r.bid>r1.bid and r.day=r1.day and r.sid=s.sid and s.sid=r1.sid;

View created.

SQL>select *from v5;

Rating	bid
11	101
12	102
10	103
11	105

EXCERSIZE-7

PL/SQL PROGRAMS:

1. Write a PL/SQL program to display message.

```
SQL> declare
        begin
        dbms_output.put_line('WELCOME TO PL/SQL');
        end;
        /
```

OUPUT:

WELCOME TO PL/SQL

PL/SQL procedure successfully completed.

2. Write a pl/sql program to perform arithmetic operations.

```
SQL> declare
        a number(4):=&a;
         b number(4):=&b;
         c number(5);
         begin
         dbms_output.put_line('a value is:' ||a);
         dbms_output.put_line('b value is:' ||b);
         c:=a+b;
         dbms_output.put_line('Addition is:' ||c);
         c:=a-b;
         dbms_output.put_line('Subtraction is:' ||c);
         c:=a*b;
         dbms_output.put_line('Multipliction is:' ||c);
         c:=a/b;
         dbms_output.put_line('Division is:' ||c);
         c:=mod(a,b);
         dbms_output.put_line('Modulation is:' ||c);
         end;
         /
```

OUTPUT:

```
Enter value for a: 20
Old   2: a number(4):=&a;
New   2: a number(4):=20;
Enter value for b: 10
Old   3: b number(4):=&b;
New   3: b number(4):=10;
```

a value is:20

b value is:10

addition is:30
subtraction is:10
multipliction is:200
division is:2
modulation is:0

pL/SQL procedure successfully completed.

3. Program Find addition of N numbers.

```
SQL> declare
        s number(10):=0;
        i number(5);
        n number(5);
        begin
        n: =&n;
        for i in 1..n loop
        s:=s+i;
        end loop;
        dbms_output.put_line('Addition of N numbers is:'||s);
        end;
        /
```

OUTPUT:

```
Enter value for n: 10
old   6: n:=&n;
new   6: n:=10;
```

Addition of N numbers is:55

PL/SQL procedure successfully completed.

4. Program to write Swping of two values

```
SQL> declare
        a number(5):=&a;
        b number:=&b;
        temp number(5):=0;
        begin
        dbms_output.put_line('Before Swaping');
        dbms_output.put_line('a is:' ||a);
        dbms_output.put_line('b is:' ||b);
        temp:=a;
        a:=b;
        b:=temp;
```

```
        dbms_output.put_line('After Swaping:');
        dbms_output.put_line('a is:' ||a);
        dbms_output.put_line('b is:' ||b);
        end;
        /
```

OUTPUT:

```
Enter value for a: 10
old   2: a number(5):=&a;
new   2: a number(5):=10;
Enter value for b: 20
old   3: b number:=&b;
new   3: b number:=20;
```

Before Swaping
a is:10
b is:20
After Swaping:
a is:20
b is:10

PL/SQL procedure successfully completed.

5. Find the biggest number among given three numbers.

```
SQL> declare
 n1 number(3):=&n1;
 n2 number(3):=&n2;
 n3 number(3):=&n3;
 begin
 dbms_output.put_line(n1);
 dbms_output.put_line(n2);
 dbms_output.put_line(n3);
 if n1>n2 and n1>n3 then
 dbms_output.put_line(n1 || ' is Largest number');
 end if;
 if n2>n1 and n2>n3 then
 dbms_output.put_line(n2 || 'is Largest number');
 end if;
 if n3>n2 and n2>n1 then
 dbms_output.put_line(n3 || 'is Largest number');
 end if;
 end;
 /
```

OUTPUT:

```
Enter value for n1: 10
Old   2: n1 number(3):=&n1;
New   2: n1 number(3):=10;
Enter value for n2: 12
Old   3: n2 number(3):=&n2;
New   3: n2 number(3):=12;
Enter value for n3: 5
Old   4: n3 number(3):=&n3;
New   4: n3 number(3):=5;
10
12
5
12 is Largest number

PL/SQL procedure successfully completed.
```

6. Program to find salary from employee where emp no=7782.

```
SQL> declare
        esal emp.sal%type;
        begin
        select sal into esal from emp where empno=7782;
        dbms_output.put_line('Salary is:'||esal);
        end;
        /
```

OUTPUT:

```
Salary is: 2450

PL/SQL procedure successfully completed.
```

7. Print the numbers in reverse order in a given range.

```
        SQL> declare
         x number(5);
         n number: ='&n';
         begin
         for x in reverse 1..n loop
         dbms_output.put_line(x);
         end loop;
         end;
         /
```

OUTPUT:

```
Enter value for n: 10
Old   3: n number: ='&n';
New   3: n number: ='10';
        10
        9
        8
        7
        6
        5
        4
        3
        2
        1
PL/SQL procedure successfully completed.
```

8. Program to fetch maximum salary from two employees.

```
SQL> declare
        esal1 emp.sal%type;
        esal2 emp.sal%type;
        begin
        select sal into esal1 from emp where empno=&number;
        select sal into esal2 from emp where empno=&number;
        if(esal1>esal2) then
        dbms_output.put_line('Max sal is:'||esal1);
        else
        dbms_output.put_line('Max sal is:'||esal2);
        end if;
        end;
        /
```

OUTPUT:

```
Enter value for number: 7566
old   5: select sal into esal1 from emp where empno=&number;
new   5: select sal into esal1 from emp where empno=7566;
Enter value for number: 7839
old   6: select sal into esal2 from emp where empno=&number;
new   6: select sal into esal2 from emp where empno=7839;
Max sal is: 5000

PL/SQL procedure successfully completed.
```

9. Display whether given number is EVEN or ODD

```
SQL>  declare
       n number:='&n';
       begin
       if mod(n,2)=0 then
       dbms_output.put_line(n ||' Is EVEN number');
       else
       dbms_output.put_line(n ||' Is ODD number');
       end if;
       end;
       /
```

OUTPUT:

Enter value for n: 25
old 2: n number:='&n';
new 2: n number:='25';
25 Is ODD number

PL/SQL procedure successfully completed.

10. Find the biggest digit in a given number.

```
SQL> declare
        n number:=&n;
        big number(10):=0;
        begin
        loop
        if mod(n,10)>big then
        big:=mod(n,10);
        end if;
        n:=trunc(n/10);
        exit when n=0;
        end loop;
        dbms_output.put_line('Biggest number is:'||big);
        end;
        /
```

OUTPUT:

Enter value for n: 125
Old 2: n number: =&n;
New 2: n number: =125;
Biggest number is:5

PL/SQL procedure successfully completed.

11. Program to print given string is reverse order.

```
SQL> declare
        str1 varchar2(30):='&str1';
        len number(5);
        rev_str varchar2(30);
        begin
        dbms_output.put_line('Given string is:');
        dbms_output.put_line(str1);
        len:=length(str1);
        while len>0
        loop
        rev_str:=rev_str||substr(str1,len,1);
        len:=len-1;
        end loop;
        dbms_output.put_line('Reverse string is:');
        dbms_output.put_line(rev_str);
        end;
        /
```

OUTPUT:

```
Enter value for str1: god
old   2: str1 varchar2(30):='&str1';
new   2: str1 varchar2(30):='god';
```

Given string is:
raghava
Reverse string is:
Avahgar
PL/SQL procedure successfully completed.

12. Find the number of Sunday's between given dates.

```
SQL> declare
        d1 date;
        d2 date;
        c number(10):=0;
        begin
        d1:='&d1';
        d2:='&d2';
        dbms_output.put_line(d1);
        dbms_output.put_line(d2);
        d1:=next_day(d1-1,'sunday');
        while (d1<=d2)
        loop
```

```
c: =c+1;
d1:=d1+7;
end loop;
dbms_output.put_line('No of sundays is:'|| c);
end;
/
```

OUTPUT:

```
Enter value for d1: 01-jan-2011
Old   6: d1:='&d1';
New   6: d1:='01-jan-2011';
Enter value for d2: 25-jan-2011
Old   7: d2:='&d2';
New   7: d2:='25-jan-2011';
```

01-JAN-11
25-JAN-11
No of sundays is: 4
PL/SQL procedure successfully completed.

13. Find given number is single digit or two digit or three digit or multi digit values.

```
SQL> declare
 x number:='&x';
 begin
 if x>=1 and x<=9 then
 dbms_output.put_line('Given number is single digit number');
 else if(x>=10 and x<=99) then
 dbms_output.put_line('Givne num is two digit number');
 else if(x>=100 and x<=999) then
 dbms_output.put_line('Given num is three digit number');
 else
 dbms_output.put_line('Given num is MULTI digit number');
 end if;
 end if;
 end if;
 end;
 /
```

OUTPUT:

```
Enter value for x: 45
Old   2: x number:='&x';
New   2: x number:='45';
```

Givne num is two digit number
PL/SQL procedure successfully completed.

14. Write a pl/sql program to generate prime numbers between 1 to n.

```
SQL> declare
        n number:=&n;
        c number;
        i number:=2;
        j number;
        num number:=0;
        begin
        while i<=n
        loop
        c:=0;
        j:=1;
        while j<=i
        loop
        if mod(i,j)=0 then
        c:=c+1;
        end if;
        j:=j+1;
        end loop;
        if c=2 then
        num:=i;
        dbms_output.put_line('Prime numbers are:'||num);
        end if;
        i:=i+1;
        end loop;
        end;
        /
```

OUTPUT:

```
Enter value for n: 15
old   2: n number:=&n;
new   2: n number:=15;
```

Prime numbers are:

2
3
5
7
11
13

PL/SQL procedure successfully completed.

15. Write pl/sql prog to check whether given num is Armstrong or not.

```
SQL> declare
        num number(5):=&num;
        rem number(5);
        s number(5):=0;
        temp number(5);
        begin
        temp: =num;
        while(num>0)
        loop
        rem:=mod(num,10);
        s:=s+power(rem,3);
        num:=trunc(num/10);
        end loop;
        if(s=temp) then
        dbms_output.put_line('Amstrong number');
        else
        dbms_output.put_line('NOT Amstrong number');
        end if;
        end;
        /
```

OUTPUT:

```
Enter value for num: 153
old   2: num number(5):=&num;
new   2: num number(5):=153;
```

Amstrong number

PL/SQL procedure successfully completed.

16. Write a pl/sql prog to check whether given string is palindrome or not.

```
SQL> declare
        str varchar2(20);
        str1 varchar2(20);
        len number(10);
        begin
        str:='&str';
        len:=length(str);
        while (len>0)
        loop
        str1:=str1||substr(str,len,1);
        len:=len-1;
```

```
        end loop;
        dbms_output.put_line('Reverse string is:'|| str1);
        if(str=str1) then
        dbms_output.put_line(str||' is Palindrome');
        else
        dbms_output.put_line(str||' is not Palindrome');
        end if;
        end;
        /
```

OUTPUT:

```
Enter value for str: vikatakavi
old   6: str:='&str';
new   6: str:='vikatakavi';
Reverse string is:ivakatakiv
```

Vikatakavi is not Palindrome

PL/SQL procedure successfully completed.

```
SQL> /
Enter value for str: sms
old   6: str:='&str';
new   6: str:='sms';
Reverse string is:sms
```

Sms is Palindrome

PL/SQL procedure successfully completed.

17. Program to find the number of even and numbers and find the sum of each.

```
SQL> declare
          n number(3):=&n;
          se number(10):=0;
          so number(10):=0;
          cte number(5):=0;
          cto number(5):=0;
          begin
          dbms_output.put_line(n);
          for i in 1..n loop
          if mod(i,2)=0 then
          cte:=cte+1;
          se:=se+i;
          else
```

```
        cto:=cto+1;
        so:=so+i;
        end if;
        end loop;
        dbms_output.put_line('Sum of even numbers is:'||se);
        dbms_output.put_line('no.of even numbers is:'||cte);
        dbms_output.put_line('Sum of odd numbers is:'||so);
        dbms_output.put_line('no. of odd numbers is:'||cto);
        end;
        /
```

OUTPUT:

```
Enter value for n: 10
old   2:  n number(3):=&n;
new   2:  n number(3):=10;
```

10
Sum of even numbers is: 30
no.of even numbers is:5
Sum of odd numbers is: 25
no. of odd numbers is:5

PL/SQL procedure successfully completed.

18. Find the area of circle in given radius from 1 to n.

```
SQL> create table area(
        radious number(10),
        area number(10,2));
```

Table created.

```
SQL> declare
     pi number(4,2):=3.14;
     rad number(5);
     cir_area number(14,2);
     begin
     rad:=1;
     while rad<=10
     loop
     cir_area:=pi*power (rad, 2);
      insert into area values(rad,cir_area);
      rad: =rad+1;
      end loop;
      end;
      /
```

PL/SQL procedure successfully completed.

OUTPUT:

SQL> select *from area;

RADIOUS	AREA
1	3.14
2	12.56
3	28.26
4	50.24
5	78.5
6	113.04
7	153.86
8	200.96
9	254.34
10	314

19. Write a PL/SQL program to generate Fibonacci series between 1 to N

```
SQL> declare
        n number(5):='&n';
        x number:=0;
        y number:=1;
        z number(5);
        begin
        dbms_output.put_line(x||' ');
        dbms_output.put_line(y||' ');
        for i in 3..n loop
        z:=x+y;
        dbms_output.put_line(' '||z);
        x:=y;
        y:=z;
        end loop;
         end;
        /
```

OUPUT:

```
Enter value for n: 10
old   2: n number(5):='&n';
new   2: n number(5):='10';
```

0 1 1 2 3 5 8 13 21 34

PL/SQL procedure successfully completed.

20. Write a pl/sql prog to check whether given number is Prime or not.

```
SQL> declare
        n number(3):=&n;
        i number(3);
        c number(3):=0;
        begin
        dbms_output.put_line(n);
        for i in 1..n loop
        if mod(n,i)=0 then
        c:=c+1;
        end if;
        end loop;
        if c=2 then
        dbms_output.put_line(n ||' is Prime number');
         else
         dbms_output.put_line(n ||' is not a Prime number');
         end if;
         end;
         /
```

OUTPUT:

```
Enter value for n: 10
old   2: n number(3):=&n;
new   2: n number(3):=10;
10
10 is not a Prime number
PL/SQL procedure successfully completed.
SQL> /
Enter value for n: 3
old   2: n number(3):=&n;
new   2: n number(3):=3;
3
3 is Prime number

PL/SQL procedure successfully completed.
```

21. Program to print reverses of a given number

```
SQL> declare
        n number(10):=&n;
        n1 number(10);
        m number(10):=0;
        begin
        dbms_output.put_line('Given number is:' ||n);
        n1:=n;
        while n1>0 loop
        m:=m*10+mod(n1,10);
        n1:=trunc(n1/10);
        end loop;
        dbms_output.put_line('Reverse number is:' ||m);
        end;
        /
```

OUTPUT:

```
Enter value for n: 12345
old   2: n number(10):=&n;
new   2: n number(10):=12345;
```

Given number is: 12345

Reverse number is: 54321

PL/SQL procedure successfully completed.

22. Calculate FACTORIAL of a given number.

```
SQL> declare
        n number:='&n';
        fact number(5):=1;
        begin
        dbms_output.put_line ('Given number is:'|| n);
        while(n>0)
        loop
        fact:=fact*n;
        n:=n-1;
        end loop;
        dbms_output.put_line('Given num Factorial is:' || fact);
        end;
       /
```

OUTPUT:

```
Enter value for n: 6
old   2:  n number:='&n';
new   2:  n number:='6';
```

Given number is:6
Given num Factorial is:720

PL/SQL procedure successfully completed.

23. Write a pl/sql program for inserting rows into Emp_detal table with the following calculations.

HRA=50%of basic
DA=20%of basic
PF=7% of basic

NERPAY=basic+da+hra-pf.

```
SQL> create table emp_detail(
 eid                  number(5) primary key,
 name                 varchar2(20),
 deptno                number(5)
 base_sal              varchar2(20),
 hra                  varchar2(20),
 da                    varchar2(20),
 pf                    varchar2(20),
 net_pay              varchar2(20));

SQL> declare
        eno emp_detail.eid%type;
        ename emp_detail.name%type;
        dno emp_detail.deptno%type;
        basic emp_detail.base_sal%type;
        hra1 emp_detail.hra%type;
        da1 emp_detail.da%type;
        pf1 emp_detail.pf%type;
        netpay1 emp_detail.net_pay%type;
        begin
        eno:=&eno;
        ename: ='&ename';
        dno:=&dno;
        basic: =&basic;
        hra1 :=( basic*5)/100;
        da1 :=( basic*20)/100;
        pf1 :=( basic*7)/100;
        netpay1:=basic+hra1+da1-pf1;
        insert into emp_detail values(eno,ename,dno,basic,hra1,da1,pf1,netpay1);
        end;
        /
```

```
Enter value for eno: 104
old  11: eno:=&eno;
new  11: eno:=104;
Enter value for ename: Srinivas
old  12: ename:='&ename';
new  12: ename:='Srinivas';
Enter value for dno: 10
old  13: dno:=&dno;
new  13: dno:=10;
Enter value for basic: 8000
old  14: basic:=&basic;
new  14: basic:=8000;

PL/SQL procedure successfully completed.

SQL> /
Enter value for eno: 105
old  11: eno:=&eno;
new  11: eno:=105;
Enter value for ename: Shankar
old  12: ename:='&ename';
new  12: ename:='Shankar';
Enter value for dno: 20
old  13: dno:=&dno;
new  13: dno:=20;
Enter value for basic: 9000
old  14: basic:=&basic;
new  14: basic:=9000;

PL/SQL procedure successfully completed.

SQL> /
Enter value for eno: 103
old  11: eno:=&eno;
new  11: eno:=103;
Enter value for ename: Suresh
old  12: ename:='&ename';
new  12: ename:='Suresh';
Enter value for dno: 10
old  13: dno:=&dno;
new  13: dno:=10;
```

```
Enter value for basic: 6500
old  14: basic:=&basic;
new  14: basic:=6500;

PL/SQL procedure successfully completed.

SQL> /
Enter value for eno: 102
old  11: eno:=&eno;
new  11: eno:=102;
Enter value for ename: Santosh
old  12: ename:='&ename';
new  12: ename:='Santosh';
Enter value for dno: 20
old  13: dno:=&dno;
new  13: dno:=20;
Enter value for basic: 8500
old  14: basic:=&basic;
new  14: basic:=8500;

PL/SQL procedure successfully completed.

SQL> /
Enter value for eno: 101
old  11: eno:=&eno;
new  11: eno:=101;
Enter value for ename: Aravind
old  12: ename:='&ename';
new  12: ename:='Aravind';
Enter value for dno: 30
old  13: dno:=&dno;
new  13: dno:=30;
Enter value for basic: 8000
old  14: basic:=&basic;
new  14: basic:=8000;

PL/SQL procedure successfully completed.

SQL> /
Enter value for eno: 100
old  11: eno:=&eno;
```

```
new  11: eno:=100;
Enter value for ename: Raghava
old  12: ename:='&ename';
new  12: ename:='Raghava';
Enter value for dno: 30
old  13: dno:=&dno;
new  13: dno:=30;
Enter value for basic: 8000
old  14: basic:=&basic;
new  14: basic:=8000;

PL/SQL procedure successfully completed.

SQL> /
Enter value for eno: 107
old  11: eno:=&eno;
new  11: eno:=107;
Enter value for ename: Venkat
old  12: ename:='&ename';
new  12: ename:='Venkat';
Enter value for dno: 20
old  13: dno:=&dno;
new  13: dno:=20;
Enter value for basic: 8000
old  14: basic:=&basic;
new  14: basic:=8000;

PL/SQL procedure successfully completed.
```

OUTPUT:

```
SQL> select *from emp_detail;
```

EID	NAME	DEPTNO	BASE_SAL	HRA	DA	PF	NET_PAY
104	Srinivas	10	8000	400	1600	560	9440
105	Shankar	20	9000	450	1800	630	10620
103	Suresh	10	6500	325	1300	455	7670

102	Santosh	20	8500	425	1700	595	10030
101	Aravind	30	8000	400	1600	560	9440
100	Raghava	30	8000	400	1600	560	9440
107	Venkat	20	8000	400	1600	560	9440

24. Write the pl/sql program to find percentages of students and display Their grades and percentages.

```
SQL> declare
        java number(10);
         dbms number(10);
         co number(10);
         ppl number(10);
         se number(10);
         es number(10);
        total number(10);
         avgs float(10);
        per float(10);
        begin
        java:=&java;
        dbms:=&dbms;
        co:=&co;
        ppl:=&ppl;
        se:=&se;
        es:=&es;
        total:=(java+dbms+co+ppl+se+es);
        avgs:=(total/600);
        per:=avgs*100;
        if java<40 or dbms<40 or co<40 or ppl<40 or se<40 or es<40 then
        dbms_output.put_line('FAIL');
        end if;
        if per=60 then
        dbms_output.put_line('PASS');
        end if;
```

```
        if per>75 then
        dbms_output.put_line('GRADE "A"');
        else if per>65 and per<75 then
        dbms_output.put_line('GRADE "B"');
        else if per>55 and per<60 then
        dbms_output.put_line('GRADE "C"');
        else
        dbms_output.put_line('INVALID INPUT');
        end if;
        end if;
        end if;
        dbms_output.put_line('Percentage is:'||per);
        end;
         /
```

OUTPUT:

```
Enter value for java: 56
old  12: java:=&java;
new  12: java:=56;
Enter value for dbms: 65
old  13: dbms:=&dbms;
new  13: dbms:=65;
Enter value for co: 69
old  14: co:=&co;
new  14: co:=69;
Enter value for ppl: 58
old  15: ppl:=&ppl;
new  15: ppl:=58;
Enter value for se: 74
old  16: se:=&se;
new  16: se:=74;
Enter value for es: 75
old  17: es:=&es;
new  17: es:=75;
```

GRADE "B"

Percentage is: 66.17

PL/SQL procedure successfully completed.

25. Write a program to delete a selected tuple from given passing Parameter.

```
SQL> select *from salgrade;
   GRADE     LOSAL     HISAL
--------- --------- ---------
       1       700      1200
       2      1201      1400
       3      1401      2000
       4      2001      3000
       5      3001      9999

SQL> declare
          var number(3);
         begin
         var:=&grd;
         delete from salgrade where grade=var;
         dbms_output.put_line('row deleted');
         end;
         /
```

OUTPUT:

```
Enter value for grd: 2
old   4: var:=&grd;
new   4: var:=2;
row deleted

PL/SQL procedure successfully completed.

SQL> select *from salgrade;

   GRADE     LOSAL     HISAL
--------- --------- ----------
       1       700      1200
       3      1401      2000
       4      2001      3000
       5      3001      9999
```

EXCERSIZE-8

PROCEDURES

1. Write a procedure to display ename and salary from employee when user input is empno using in/out parameters and empno=7698.

```
SQL> create or replace procedure pro123(no in number,name out varchar,esal out
number)
  is
  begin
  select ename,sal into name,esal from emp where empno=no;
  end pro123;
  /
Procedure created.

SQL> declare
  name1 varchar2(20);
  esalary number(5);
  eno number(5);
  begin
  pro123(&eno,name1,esalary);
  dbms_output.put_line(name1||' '||esalary);
  end;
 /
```

OUTPUT:

```
Enter value for eno: 7698
old   6: pro123(&eno,name1,esalary);
new   6: pro123(7698,name1,esalary);
BLAKE 2850

PL/SQL procedure successfully completed.
```

2. Write a procedure to update salary of employee table taking empno as in parameter.

```
SQL> create or replace procedure pro12(no in number)
        is
        begin
        update emp set sal=(sal+1000) where empno=no;
        dbms_output.put_line('salary is updated');
        end;
        /
Procedure created.

SQL> declare
        begin
        pro12(&enum);
        end;
        /
```

OUTPUT:

```
Enter value for enum: 7654
old   3: pro12(&enum);
new   3: pro12(7654);
```

salary is updated

PL/SQL procedure successfully completed.

3. Program for swapping the values

```
SQL> Create or replace procedure SWAP(a in out integer,b in out integer) is
        t integer;
        begin
        t:=a;
        a:=b;
        b:=t;
        end;
        /

Procedure created.

SQL>  declare
        x integer:=&a;
        y integer:=&b;
        begin
        dbms_output.put_line('before procedure call');
        dbms_output.put_line('x = '||x);
        dbms_output.put_line('y = '||y);
        SWAP(x,y);
        dbms_output.put_line('after procedure call');
        dbms_output.put_line('x = '||x);
        dbms_output.put_line('y = '||y);
        end;
        /
```

Output:

```
Enter value for a: 23
old   2:  x integer:=&a;
new   2:  x integer:=23;
Enter value for b: 12
old   3:  y integer:=&b;
new   3:  y integer:=12;
before procedure call
x = 23
y = 12
after procedure call
x = 12
y = 23

PL/SQL procedure successfully completed.
```

EXCERSIZE-9

FUNCTIONS:

1. Write a function to check the validity of empno from employee table.

```
SQL> create or replace function fun1(eno in number)
        return number
        is
        no number;
        begin
        select empno into no from emp where empno=eno;
        return no;
        exception
        when no_data_found then
        return 1;
        end;
         /
```

Function created.

```
SQL> declare
        n number(4);
        begin
        n:=fun1(&n0);
        if(n=1) then
        dbms_output.put_line('no data found');
        else
        dbms_output.put_line('Valid data');
        end if;
        end;
        /
```

OUTPUT:

```
Enter value for n0: 7782
old   4:  n:=fun1(&n0);
new   4:  n:=fun1(7782);
```

Valid data

PL/SQL procedure successfully completed.

2. WRITE A PL/SQL BLOCK TO CREATE A FUNCTION.

```
SQL> CREATE OR REPLACE FUNCTION
     SUMSAL(DESIG EMP.JOB%TYPE) RETURN NUMBER AS
       VAR_SAL EMP.SAL%TYPE;
       TOT_SAL EMP.SAL%TYPE;
       VAR_COMM EMP.COMM%TYPE;
       BEGIN
       SELECT SUM(SAL),SUM(COMM) INTO VAR_SAL,VAR_COMM
       FROM EMP GROUP BY JOB HAVING JOB=DESIG;
        IF DESIG='SALESMAN' THEN
        TOT_SAL:=VAR_SAL+VAR_COMM;
       RETURN TOT_SAL;
       ELSE
       RETURN VAR_SAL;
       END IF;
      END;
```

Function created.

```
SQL> VARIABLE I NUMBER;
SQL> EXEC :I:=SUMSAL('SALESMAN');
```

PL/SQL procedure successfully completed.

```
SQL> PRINT I

         I
---------
     34000
```

EXCERSIZE-10

CURSORS:

1. program to fetch all data from sal grade using cursor for loop.

```
SQL> declare
        cursor salgrade_cur is select *from salgrade;
        begin
         for sal_rec in salgrade_cur
         loop
         dbms_output.put_line(sal_rec.grade||' '||sal_rec.losal||' '||sal_rec.hisal);
         end loop;
         end;
         /
```

Output:

1 700 1200
3 1401 2000
4 2001 3000
5 3001 9999

PL/SQL procedure successfully completed.

2. Write a prog to check whether cursor open or not with cursor is open display cursor is already open else open the cursor display the message just opned the cursor.

```
SQL> declare
        cursor name is select *from emp;
        begin
        if name%isopen
        then
        dbms_output.put_line('Already opened');
        else
        open name;
        dbms_output.put_line('Just opened');
        end if;
        close name;
        end;
        /
```

OUTPUT:

Just opened

PL/SQL procedure successfully completed.

3. write a Cursor to display the list of Employees and Total Salary Department wise.

```
SQL>  DECLARE
        Cursor c1 is select * from dept;
        Cursor c2 is select * from emp;
         s emp.sal%type;
         BEGIN
         for i in c1 loop
         s:=0;
         dbms_output.put_line('-----------------------------------------');
         dbms_output.put_line('Department is :' || i.deptno ||' Department name is:' ||
         i.dname);
        dbms_output.put_line('-----------------------------------------');
        for j in c2 loop
        if ( i.deptno=j.deptno) then
        s:=s+j.sal;
        dbms_output.put_line(j.empno|| ' '|| j.ename || ' '|| j.sal );
        end if;
        end loop;
        dbms_output.put_line('-----------------------------------------');
        dbms_output.put_line('Total salary is: '|| s);
        dbms_output.put_line('-----------------------------------------');
        end loop;
        END;
         /
```

OUTPUT:

```
---------------------------------------------------------------------------------------------
Department is: 10 Department name is: ACCOUNTING
---------------------------------------------------------------------------------------------
7782 CLARK 2450
7839 KING 5000
7934 MILLER 1300
---------------------------------------------------------------------------------------------
Total salary is: 8750
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
Department is: 20 Department name is: RESEARCH
---------------------------------------------------------------------------------------------
7369 SMITH 800
7566 JONES 2975
```

7788 SCOTT 3000
7876 ADAMS 1100
7902 FORD 3000

Total salary is: 10875

Department is: 30 Department name is: SALES

7499 ALLEN 1600
7521 WARD 1250
7654 MARTIN 1250
7698 BLAKE 2850
7844 TURNER 1500
7900 JAMES 950

Total salary is: 9400

Department is :40 Department name is: OPERATIONS

Total salary is: 0

PL/SQL procedure successfully completed.

4. Write a Cursor to display the list of employees who are working as a Managers or Analyst.

```
SQL> DECLARE
  cursor c(jb varchar2) is select ename from emp where job=jb;
  em emp.job%type;
  BEGIN
  open c('MANAGER');
  dbms_output.put_line(' EMPLOYEES WORKING AS MANAGERS ARE:');
  loop
  fetch c into em;
  exit when c%notfound;
  dbms_output.put_line(em);
  end loop;
```

```
  close c;
  open c('ANALYST');
  dbms_output.put_line(' EMPLOYEES WORKING AS ANALYST ARE:');
  loop
  fetch c into em;
  exit when c%notfound;
  dbms_output.put_line(em);
  end loop;
  close c;
  end;
  /
```

OUTPUT:

EMPLOYEES WORKING AS MANAGERS ARE:
JONES
BLAKE
CLARK
EMPLOYEES WORKING AS ANALYST ARE:
SCOTT
FORD

PL/SQL procedure successfully completed.

5. To write a Cursor to display List of Employees from Emp Table in PL/SQL block

```
SQL> DECLARE
    cursor c is select empno, ename, deptno, sal from emp ;
    i emp.empno%type;
    j emp.ename%type;
    k emp.deptno%type;
    l emp.sal%type;
    BEGIN
    open c;
    dbms_output.put_line('Empno, name, deptno, salary of employees are:= ');
  loop
  fetch c into i, j, k, l;
  exit when c%notfound;
  dbms_output.put_line(i||' '||j||' '||k||' '||l);
  end loop;
  close c;
 END;
  /
```

Output:

Empno, name, deptno, salary of employees are:=

7369	**SMITH**	**20**	**800**
7499	**ALLEN**	**30**	**1600**
7521	**WARD**	**30**	**1250**
7566	**JONES**	**20**	**2975**
7654	**MARTIN**	**30**	**1250**
7698	**BLAKE**	**30**	**2850**
7782	**CLARK**	**10**	**2450**
7788	**SCOTT**	**20**	**3000**
7839	**KING**	**10**	**5000**
7844	**TURNER**	**30**	**1500**
7876	**ADAMS**	**20**	**1100**
7900	**JAMES**	**30**	**950**
7902	**FORD**	**20**	**3000**
7934	**MILLER**	**10**	**1300**

PL/SQL procedure successfully completed.

6. To write a Cursor to find employee with given job and deptno.

```
SQL> DECLARE
        cursor c1(j varchar2, dn number) is select empno, ename from emp where
job=j and
        deptno=dn;
        row1 emp%rowtype;
        jb emp.job%type;
        d emp.deptno%type;
        BEGIN
        jb:='&jb';
        d:=&d;
        open c1(jb,d);
        fetch c1 into row1.empno,row1.ename;
        if c1%notfound then
        dbms_output.put_line('Employee does not exist');
        else
        dbms_output.put_line('empno is:'||row1.empno||' ' ||'employee name is:'||
        row1.ename);
        end if;
        END;
        /
```

Output:

Enter value for jb: SALES
old 8: jb:='&jb';
new 8: jb:='SALES';
Enter value for d: 10
old 9: d:=&d;
new 9: d:=10;
Employee does not exist

PL/SQL procedure successfully completed.

SQL> /
Enter value for jb: MANAGER
old 8: jb:='&jb';
new 8: jb:='MANAGER';
Enter value for d: 20
old : d:=&d;
new 9: d:=20;
empno is:7566 employee name is:JONES

PL/SQL procedure successfully completed.
SQL> /
Enter value for jb: CLERK
old 8: jb:='&jb';
new 8: jb:='CLERK';
Enter value for d: 40
old 9: d:=&d;
new 9: d:=40;
Employee does not exist

PL/SQL procedure successfully completed.

7. WRITE A PL/SQL BLOCK TO CREATE A EXPLICIT CURSOR.

```
SQL> CREATE TABLE DEPT10(
        EMPNO NUMBER(5),
        JOB VARCHAR2(20),
       SAL NUMBER(7,2),
       DEPTNO NUMBER(8));
```

Table created.

```
SQL>  DECLARE
          CURSOR C1 IS
          SELECT * FROM EMP ORDER BY DEPTNO;
          CURSOR_VAR C1%ROWTYPE;
          BEGIN
          OPEN C1;
          LOOP
         FETCH C1 INTO CURSOR_VAR;
          EXIT WHEN C1%NOTFOUND;
         IF CURSOR_VAR.DEPTNO=10 THEN
         INSERT INTO DEPT10
       VALUES(CURSOR_VAR.EMPNO,CURSOR_VAR.JOB,
       CURSOR_VAR.SAL,CURSOR_VAR.DEPTNO);
         END IF;
         END LOOP;
         CLOSE C1;
         END;
        /
PL/SQL procedure successfully completed.
SQL> SELECT * FROM DEPT10;
```

EMPNO	JOB	SAL	DEPTNO
7782	MANAGER	6450	10
7839	PRESIDENT	6450	10
7934	CLERK	6450	10

EXCERSIZE-11

TRIGGER

1. Write a program to display dept details using triggers

```
SQL> create or replace trigger trig1 before insert on dept for each row
        declare
        a number;
        begin
        if(:new.deptno is Null) then
        raise_application_error(-20001,'error::deptno cannot be null');
        else
        select count(*) into a from dept where deptno=:new.deptno;
        if(a=1) then
        raise_application_error(-20002,'error:: cannot have duplicate deptno');
        end if;
        end if;
        END;
        /

        Trigger created.
```

SQL> select *from dept;

DEPTNO	DNAME	LOC
10	**ACCOUNTING**	**NEW YORK**
20	**RESEARCH**	**DALLAS**
30	**SALES**	**CHICAGO**
40	**OPERATIONS**	**BOSTON**

```
SQL> insert into dept values(&deptnp,'&dname','&loc');
Enter value for deptnp: null
Enter value for dname: marktening
Enter value for loc: hyd
old   1: insert into dept values(&deptnp,'&dname','&loc')
new   1: insert into dept values(null,'marktening','hyd')
insert into dept values(null,'marktening','hyd')
            *
ERROR at line 1:
ORA-20001: error::deptno cannot be null
ORA-06512: at "SCOTT.TRIG1", line 5
```

ORA-04088: error during execution of trigger 'SCOTT.TRIG1'

SQL> /
Enter value for deptnp: 10
Enter value for dname: manager
Enter value for loc: hyd
old 1: insert into dept values(&deptnp,'&dname','&loc')
new 1: insert into dept values(10,'manager','hyd')
insert into dept values(10,'manager','hyd')
*
ERROR at line 1:
ORA-20002: error:: cannot have duplicate deptno
ORA-06512: at "SCOTT.TRIG1", line 9
ORA-04088: error during execution of trigger 'SCOTT.TRIG1'

SQL> /
Enter value for deptnp: 50
Enter value for dname: marktening
Enter value for loc: hyd
old 1: insert into dept values(&deptnp,'&dname','&loc')
new 1: insert into dept values(50,'marktening','hyd')

1 row created.

SQL> select *from dept;

DEPTNO	DNAME	LOC
10	**ACCOUNTING**	**NEW YORK**
20	**RESEARCH**	**DALLAS**
30	**SALES**	**CHICAGO**
40	**OPERATIONS**	**BOSTON**
50	**marktening**	**hyd**

2. WRITE PL/SQL BLOCK FOR CREATING A ROW LEVEL TRIGGER.

```
SQL> create or replace trigger emp_bouns
      after insert or delete or update of  ename,job,sal,comm on emp
     referencing old as o for each row
     begin
     if inserting then
     insert into bonus values(:new.ename,:new.job,:new.sal,:new.comm);
    end if;
    if deleting then
     insert into bonus values(:o.ename,:o.job,:o.sal,:o.comm);
    end if;
    if updating then
    insert into bonus values(:o.ename,:o.job,:o.sal,:o.comm);
    end if;
    end;
```

TRIGGER CREATED

```
SQL>:SELECT *FROM BONUS;
    NO ROWS SELECTED
SQL>:DELETE  FROM  EMP  WHERE DEPTNO=10;
      3 ROWS  SELECTED
SQL>:SELECT *FROM  BONUS;
```

NAME	JOB	SAL	COMM
7782	MANAGER	2450	
7839	PRESIDENT	5000	
7934	CLERK	1300	

3. WRITE PL/SQL BLOCK CREATES A STATEMENT LEVEL TRIGGER.

```
SQL> CREATE OR REPLACE TRIGGER STATE_TRIGGER
        AFTER DELETE OR INSERT OR UPDATE OF EMPNO ON EMP
        DECLARE
        VAR_DATE EMP.HIREDATE%TYPE;
        BEGIN
       SELECT SYSDATE INTO VAR_DATE FROM DUAL;
        IF DELETING OR INSERTING OR UPDATING THEN
        INSERT INTO TEMP VALUES('ON
     '||TO_CHAR(VAR_DATE,'DD:MM:YYYY     HH:SS   ')||SEQ.NEXTVAL||
        '   OPERATIONS PERFORMED');
       END IF;
      END;
     /

Trigger created.

SQL> SELECT *FROM TEMP;

no rows selected

SQL> DELETE FROM EMP WHERE JOB='MANAGER';

2 rows deleted.

SQL> SELECT *FROM TEMP;

MESSAGE
--------------------------------------------------------------------------------
ON 05:12:2008     02:20   1 OPERATION PERFORMED
```