

ADVANCED DBMS LAB

(20MCA134)

LAB RECORD

Submitted in partial fulfilment of the requirements for the award of the degree
of Master of Computer Applications of A P J Abdul Kalam Technological
University

Submitted by:

CHRISTEENA JOY (SJC22MCA-2020)



MASTER OF COMPUTER APPLICATIONS
ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY, PALAI
CHOONDACHERRY P.O, KOTTAYAM

KERALA

August 2023

ST. JOSEPH' S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI

(An ISO 9001: 2015 Certified College)

CHOONDACHERRY P.O, KOTTAYAM KERALA



CERTIFICATE

This is to certify that the ADVANCED DBMS Lab Record (20MCA134) submitted Christeena Joy student of second semester MCA at ST. JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI in partial fulfilment for the award of Master of Computer Applications is a bonafide record of the lab work carried out by her under our guidance and supervision. This record in any form has not been submitted to any other University or Institute for any purpose.

Prof. Rahul Shajan

Faculty In- Charge

Prof. Anish Augustine K

(HoD In Charge-MCA)

Submitted for the End Semester Examination held on _____

Examiner 1:

Examiner 2:

DECLARATION

I Christeena Joy, do hereby declare that the Advanced DBMS Lab Record (20 MCA 134) is a record of work carried out under the guidance of Dr. Rahul Shajan , Asst.Professor, Department of Computer Applications, SJCET, Palai as per the requirement of the curriculum of Master of Computer Applications Programme of A P J Abdul Kalam Technological University, Thiruvananthapuram. Further, I also declare that this record has not been submitted, full or part thereof, in any University / Institution for the award of any Degree / Diploma.

Place: Choondacherry

CHRISTEENA JOY

Date :

(SJC22MCA-2020)

CONTENT

Sl. No.	Program List	Page No.
1	Sql query operations on Employee table	1
2	Create the SAILORS, BOATS, RESERVES tables and execute the queries given below	17
3	Operations on tables salesman, customer, orders	29
4	DCL & TCL	35
5	Views	36
6	Joins	37
7	PL/SQL programs	40
8	PL/SQL procedure and functions	46
9	PL/SQL Cursor, trigger	48
10	MongoDB operations	51
11	Usage of aggregate functions	56
12	PyMongo operations	60

SET 1**1. Create an employee table 'EMP' with following fields :**

empno	NUMBER(4)
ename	VARCHAR2(25)
job	VARCHAR2(12)
salary	NUMBER(10,2)
commission	NUMBER(7,2)
deptno	NUMBER(2)

```
SQL> create table EMP(empno NUMBER(4), ename VARCHAR2(25), job
VARCHAR2(12), salary NUMBER(10,2), commission NUMBER(7,2), deptno
NUMBER(2));
```

Table created.

2. Display the structure of 'EMP'

```
SQL> desc EMP
```

Name	Null?	Type

EMPNO		NUMBER(4)
ENAME		VARCHAR2(25)
JOB		VARCHAR2(12)
SALARY		NUMBER(10,2)
COMMISSION		NUMBER(7,2)
DEPTNO		NUMBER(2)

3. Insert the following record into 'EMP'

EMPNO	ENAME	JOB	SAL	COMM	DEPTNO
7369	SMITH	CLERK	2000	800	20

SQL> insert into EMP values(7369,'SMITH','CLERK',2000,800,20);

1 row created.

4. Insert the rest of records using substitution variable.

EMPNO	ENAME	JOB	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	1600	300	30
7521	WARD	SALESMAN	1250	500	30
7566	JONES	MANAGER	2975		20
7654	MARTIN	SALESMAN	1250	1400	30
7698	BLAKE	MANAGER	2850		30
7782	CLARK	MANAGER	2450		10
7788	SCOTT	ANALYST	3000		20
7839	KING	PRESIDENT	5000		10
7844	TURNER	SALESMAN	1500		30
7876	ADAMS	CLERK	1100		20
7900	JAMES	NULL	950		30
7902	FORD	ANALYST	3000		20
7934	MILLER	CLERK	1300		10

SQL> insert into EMP values(7499,'ALLEN','SALESMAN',1600,300,30);

1 row created.

```
SQL> insert into EMP values(7521,'WARD','SALESMAN',1250,500,30);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
```

```
values(7566,'JONES','MANAGER',2975,20);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,commission,deptno)
```

```
values(7654,'MARTIN','SALESMAN',1250,1400,30);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
```

```
values(7698,'BLAKE','MANAGER',2850,30);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
```

```
values(7782,'CLARK','MANAGER',2450,10);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
```

```
values(7788,'SCOTT','ANALYST',3000,20);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
```

```
values(7839,'KING','PRESIDENT',5000,10);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
```

```
values(7844,'TURNER','SALESMAN',1500,30);
```

```
1 row created.
```

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
```

```
values(7876,'ADAMS','CLERK',1100,20);
```

1 row created.

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
values(7900,'JAMES','NULL',950,30);
```

1 row created.

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
values(7902,'FORD','ANALYST',300,20);
```

1 row created.

```
SQL> insert into EMP(empno,ename,job,salary,deptno)
values(7934,'MILLER','CLERK',1300,10);
```

1 row created.

```
SQL> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	SALARY	COMMISSION	DEPTNO
7369	SMITH	CLERK	2000	800	20
7499	ALLEN	SALESMAN	1600	300	30
7521	WARD	SALESMAN	1250	500	30
7566	JONES	MANAGER	2975		20
7654	MARTIN	SALESMAN	1250	1400	30
7698	BLAKE	MANAGER	2850		30
7782	CLARK	MANAGER	2450		10
7788	SCOTT	ANALYST	3000		20
7839	KING	PRESIDENT	5000		10
7844	TURNER	SALESMAN	1500		30
7876	ADAMS	CLERK	1100		20
7900	JAMES	NULL	950		30
7902	FORD	ANALYST	300		20
7934	MILLER	CLERK	1300		10

14 rows selected.

5. Insert job as 'CLERK' for all 'NULL' job types.

```
SQL> UPDATE EMP SET JOB='CLERK' WHERE JOB='NULL';
```

1 row updated.

```
SQL> select * from EMP;
```

EMPNO	ENAME	JOB	SALARY	COMMISSION	DEPTNO
-----	-----	-----	-----	-----	-----
7369	SMITH	CLERK	2000	800	20
7499	ALLEN	SALESMAN	1600	300	30
7521	WARD	SALESMAN	1250	500	30
7566	JONES	MANAGER	2975		20
7654	MARTIN	SALESMAN	1250	1400	30
7698	BLAKE	MANAGER	2850		30
7782	CLARK	MANAGER	2450		10
7788	SCOTT	ANALYST	3000		20
7839	KING	PRESIDENT	5000		10
7844	TURNER	SALESMAN	1500		30
7876	ADAMS	CLERK	1100		20
7900	JAMES	CLERK	950		30
7902	FORD	ANALYST	300		20
7934	MILLER	CLERK	1300		10

14 rows selected.

6. Add a new field 'date_join' with following values**date_join**

17-DEC-80

20-FEB-81

22-FEB-81

02-APR-81

28-SEP-81

01-MAY-81

09-JUN-81

19-APR-87

17-NOV-81

08-SEP-81

SQL> alter table EMP ADD date_join date ;

Table altered.

SQL> update EMP set date_join='17-DEC-80' where empno=7369;

1 row updated.

SQL> update EMP set date_join='20-FEB-81' where empno=7499;

1 row updated.

SQL> update EMP set date_join='22-FEB-81' where empno=7521;

1 row updated.

SQL> update EMP set date_join='02-APR-81' where empno=7566;

1 row updated.

```
SQL> update EMP set date_join='28-SEP-81' where empno=7654;
```

1 row updated.

```
SQL> update EMP set date_join='01-MAY-81' where empno=7698;
```

1 row updated.

```
SQL> update EMP set date_join='09-JUN-81' where empno=7782;
```

1 row updated.

```
SQL> update EMP set date_join='19-APR-87' where empno=7788;
```

1 row updated.

```
SQL> update EMP set date_join='17-NOV-81' where empno=7839;
```

1 row updated.

```
SQL> update EMP set date_join='08-SEP-81' where empno=7844;
```

1 row updated.

```
SQL> update EMP set date_join='23-MAY-87' where empno=7876;
```

1 row updated.

```
SQL> update EMP set date_join='03-DEC-81' where empno=7900;
```

1 row updated.

```
SQL> update EMP set date_join='03-DEC-81' where empno=7902;
```

1 row updated.

SQL> update EMP set date_join='23-JAN-82' where empno=7934;

1 row updated.

7. Display details of all employees

SQL> select * from EMP;

EMPNO	ENAME	JOB	SALARY	COMMISSION	DEPTNO	DATE_JOIN
-----	-----	-----	-----	-----	-----	-----
7369	SMITH	CLERK	2000	800	20	17-DEC-80
7499	ALLEN	SALESMAN	1600	300	30	20-FEB-81
7521	WARD	SALESMAN	1250	500	30	22-FEB-81
7566	JONES	MANAGER	2975		20	02-APR-81
7654	MARTIN	SALESMAN	1250	1400	30	28-SEP-81
7698	BLAKE	MANAGER	2850		30	01-MAY-81
7782	CLARK	MANAGER	2450		10	09-JUN-81
7788	SCOTT	ANALYST	3000		20	19-APR-87
7839	KING	PRESIDENT	5000		10	17-NOV-81
7844	TURNER	SALESMAN	1500		30	08-SEP-81
7876	ADAMS	CLERK	1100		20	23-MAY-87
7900	JAMES	CLERK	950		30	03-DEC-81
7902	FORD	ANALYST	300		20	03-DEC-81
7934	MILLER	CLERK	1300		10	23-JAN-82

14 rows selected.

8. Display all the distinct job types in 'EMP'.

```
SQL> select distinct job from EMP;
```

```
JOB
```

```
-----
```

```
CLERK
```

```
SALESMAN
```

```
MANAGER
```

```
ANALYST
```

```
PRESIDENT
```

9. Display names of all employees in dept 20 and 30

```
SQL> select ename from EMP where deptno in(20,30);
```

```
ENAME
```

```
-----
```

```
SMITH
```

```
ALLEN
```

```
WARD
```

```
JONES
```

```
MARTIN
```

```
BLAKE
```

```
SCOTT
```

```
TURNER
```

```
ADAMS
```

JAMES

FORD

11 rows selected.

10. List name and Total of salary i.e sal+commission

SQL> select ename,sum(salary + commission) from EMP GROUP by ename;

ENAME	SUM(SALARY+COMMISSION)
-------	------------------------

SMITH	2800
-------	------

ALLEN	1900
-------	------

WARD	1750
------	------

JONES

MARTIN	2650
--------	------

BLAKE

CLARK

SCOTT

KING

TURNER

ADAMS

JAMES

FORD

MILLER

14 rows selected.

11. List name and Annual Salary i.e sal*12

SQL> select ename,sum(salary*12) from EMP group by ename;

ENAME	SUM(SALARY*12)
-------	----------------

SMITH	24000
-------	-------

ALLEN	19200
-------	-------

WARD	15000
------	-------

JONES	35700
-------	-------

MARTIN	15000
--------	-------

BLAKE	34200
-------	-------

CLARK	29400
-------	-------

SCOTT	36000
-------	-------

KING	60000
------	-------

TURNER	18000
--------	-------

ADAMS	13200
-------	-------

JAMES	11400
-------	-------

FORD	3600
------	------

MILLER	15600
--------	-------

14 rows selected.

12. List the employee who joined in the date '03-DEC-81'.

```
SQL> select ename from EMP where date_join='03-DEC-81';
```

ENAME

JAMES

FORD

13. Display the total salary of 'Miller'.

```
SQL> select salary from EMP where ename='MILLER';
```

SALARY

1300

14.Delete the employee 'Miller' from'EMP'

```
DELETE FROM EMPLOYEE WHERE ENAME='MILLER';
```

1 row deleted.

15. .Display name and deptno of all employees.

```
SQL> SELECT ENAME,DEPTNO FROM EMPLOYEE;
```

ENAME	DEPTNO
-------	--------

-----	-----
-------	-------

SMITH	20
-------	----

ALLEN	30
-------	----

WARD	30
JONES	20
MARTIN	30
BLAKE	30
CLARK	10
SCOTT	20
KING	10
TURNER	30
ADAMS	20
JAMES	30
FORD	20

13 rows selected.

16. Remove the field 'commission' from 'EMP' after updating salary with total salary, i.e sal+commission

EMPNO	ENAME	JOB	SALARY	DEPTNO	DATE_JOIN
7369	SMITH	CLERK	2800	20	17-DEC-80
7499	ALLEN	SALESMAN	1900	30	20-FEB-81
7521	WARD	SALESMAN	1750	30	20-FEB-81
7566	JONES	MANAGER	2975	20	22-FEB-81
7654	MARTIN	SALESMAN	2650	30	02-APR-81
7698	BLAKE	MANAGER	2850	30	28-SEP-81
7782	CLARK	MANAGER	2450	10	01-MAY-81
7788	SCOTT	ANALYST	3000	20	09-JUN-81
7839	KING	PRESIDENT	5000	10	19-APR-87
7844	TURNER	SALESMAN	1500	30	17-NOV-81

7876	ADAMS	CLERK	1100	20	08-SEP-81
7902	FORD	ANALYST	3000	20	03-DEC-81
7900	JAMES	CLERK	950	30	23-MAY-87

13 rows selected.

17. Display the name of employees having the same amount of salary (don't use subqueries)

SQL> SELECT ENAME,SALARY FROM EMPLOYEE WHERE SALARY IN
(SELECT salary FROM EMPLOYEE e WHERE EMPLOYEE.EMPNO < > e.EMPNO);

ENAME	SALARY
-----	-----
FORD	3000
SCOTT	3000

18. Display the name and employee no as 'name' and 'emp_id'

SQL> SELECT ENAME,EMPNO FROM EMPLOYEE;

ENAME	EMPNO
-----	-----
SMITH	7369
ALLEN	7499
WARD	7521
JONES	7566
MARTIN	7654
BLAKE	7698
CLARK	7782
SCOTT	7788
KING	7839
TURNER	7844

ADAMS	7876
FORD	7902
JAMES	7900

13 rows selected.

19. Rename table 'EMP' to 'EMPLOYEE'

Rename table 'EMP' to 'EMPLOYEE'

SQL> RENAME EMPLOYEE TO EMP;

Table renamed.

20. Create a new table 'EMP_TAB' from table 'EMPLOYEE'

SQL> CREATE TABLE EMP_TAB AS (SELECT * FROM EMPLOYEE);

Table created.

21. List the details of 'EMPLOYEE' and 'EMPTAB'

SELECT *FROM EMP;

EMPNO	ENAME	JOB	SALARY	DEPTNO	DATE_JOIN
7369	SMITH	CLERK	2800	20	17-DEC-80
7499	ALLEN	SALESMAN	1900	30	20-FEB-81
7521	WARD	SALESMAN	1750	30	20-FEB-81
7566	JONES	MANAGER	2975	20	22-FEB-81
7654	MARTIN	SALESMAN	2650	30	02-APR-81
7698	BLAKE	MANAGER	2850	30	28-SEP-81
7782	CLARK	MANAGER	2450	10	01-MAY-81

7788	SCOTT	ANALYST	3000	20	09-JUN-81
7839	KING	PRESIDENT	5000	10	19-APR-87
7844	TURNER	SALESMAN	1500	30	17-NOV-81
7876	ADAMS	CLERK	1100	20	08-SEP-81
7902	FORD	ANALYST	3000	20	03-DEC-81
7900	JAMES	CLERK	950	30	23-MAY-87

22.Delete all records from 'EMP'

```
SQL> DELETE FROM EMP_TAB
```

23. Delete the table 'EMP'

```
SQL>DROP TABLE EMP_TAB;
```

Table dropped.

SET 2

Create the following tables and execute the queries given below

SAILORS

sid	sname	rating	age
-----	-----	-----	-----
22	Dustin	7	45
29	Brutas	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	35
85	Art	3	26
95	Bob	3	64

BOATS

Bid	bname	color
-----	-----	-----
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

RESERVES

sid	bid	day
----	----	-----
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

```
SQL> CREATE TABLE SAILORS(sid INT NOT NULL PRIMARY KEY,sname  
varchar(25),rating number(2),age number);
```

Table created.

```
SQL> INSERT INTO SAILORS VALUES(22,'Dustin',7,45);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(29,'Brutas',1,33);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(31,'Lubber',8,55);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(32,'Andy',8,25);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(58,'Rusty',10,35);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(64,'Horatio',7,35);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(71,'Zorba',10,16);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(74,'Horatio',9,35);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(85,'Art',3,26);
```

1 row created.

```
SQL> INSERT INTO SAILORS VALUES(95,'Bob',3,64);
```

1 row created.

```
SQL> SELECT * FROM SAILORS;
```

SID	SNAME	RATING	AGE
-----	-------	--------	-----

22	Dustin	7	45
29	Brutas	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	35
85	Art	3	26
95	Bob	3	64

10 rows selected.

```
SQL> CREATE TABLE BOATS(Bid INT NOT NULL PRIMARY KEY,bname
varchar(25),color varchar(20));
```

Table created.

```
SQL> INSERT INTO BOATS VALUES(101,'Interlake','Blue');
```

1 row created.

```
SQL> INSERT INTO BOATS VALUES(102,'Interlake','Red');
```

1 row created.

```
SQL> INSERT INTO BOATS VALUES(103,'Clipper','Green');
```

1 row created.

```
SQL> INSERT INTO BOATS VALUES(104,'Marine','Red');
```

1 row created.

```
SQL> SELECT * FROM BOATS;
```

BID	BNAME	COLOR
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

```
SQL> CREATE TABLE RESERVES(sid INT REFERENCES SAILORS(sid),bid INT
REFERENCES BOATS(Bid),day DATE);
```

Table created.

```
SQL> INSERT INTO RESERVES VALUES(22,101,'10/oct/98');
```

1 row created.

```
SQL> INSERT INTO RESERVES VALUES(22,102,'10/oct/98');
```

1 row created.


```
SQL> INSERT INTO RESERVES VALUES(22,103,'10/aug/98');
```

```
1 row created.
```

```
SQL> INSERT INTO RESERVES VALUES(22,104,'10/jul/98');
```

```
1 row created.
```

```
SQL> INSERT INTO RESERVES VALUES(31,102,'11/oct/98');
```

```
1 row created.
```

```
SQL> INSERT INTO RESERVES VALUES(31,103,'11/jun/98');
```

```
1 row created.
```

```
SQL> INSERT INTO RESERVES VALUES(31,104,'11/dec/98');
```

```
1 row created.
```

```
SQL> INSERT INTO RESERVES VALUES(64,101,'09/may/98');
```

```
1 row created.
```

```
SQL> INSERT INTO RESERVES VALUES(64,102,'09/aug/98');
```

```
1 row created.
```

```
SQL> INSERT INTO RESERVES VALUES(74,103,'09/aug/98')
```

```
1 row created.
```

```
SQL> SELECT * FROM RESERVES;
```

SID	BID	DAY
22	101	10-OCT-98
22	102	10-OCT-98
22	103	10-AUG-98
22	104	10-JUL-98
31	102	11-OCT-98
31	103	11-JUN-98

31	104	11-DEC-98
64	101	09-MAY-98
64	102	09-AUG-98
74	103	09-AUG-98

10 rows selected.

1. Find the names and ages of all sailors

SQL> SELECT sname,age FROM SAILORS;

SNAME	AGE
-------	-----

Dustin	45
Brutas	33
Lubber	55
Andy	25
Rusty	35
Horatio	35
Zorba	16
Horatio	35
Art	26
Bob	64

10 rows selected.

2. Find all information of sailors who have reserved boat number 101.

SQL> SELECT * FROM SAILORS S,RESERVES R WHERE S.sid=R.sid AND
R.bid=101;

SID	SNAME	RATING	AGE	SID	BID	DAY
22	Dustin	7	45	22	101	10-OCT-98
64	Horatio	7	35	64	101	09-MAY-98

3. Find all sailors with rating above 7.

SQL> SELECT * FROM SAILORS WHERE rating>7;

SID	SNAME	RATING	AGE
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
71	Zorba	10	16
4	Horatio	9	35

4. Find the names of sailors who have reserved boat no 103.

SQL> SELECT S.sname FROM SAILORS S,RESERVES R WHERE S.sid=R.sid AND
R.bid=103;

SNAME

Dustin

Lubber

Horatio

5. Find the names of sailors who have reserved a red boat, and list in the order of age.

SQL> select distinct s.sname,s.age from SAILORS s,RESERVES r,BOATS b where
s.sid=r.sid and r.Bid=b.Bid and b.color='Red' order by s.age;

SNAME	AGE
-------	-----

Horatio	35
---------	----

Dustin	45
--------	----

Lubber	55
--------	----

6. Find the names of sailors who have reserved either a red or green boat.

SQL> select distinct s.sname from sailors s,reserves r,boats b where s.sid=r.sid and
r.bid=b.bid and (b.color='Red' or b.color='Green');

SNAME

Lubber

Dustin

Horatio

7. Find the colors of boats reserved by “Lubber”.

```
SQL> select distinct b.color from sailors s,reserves r,boats b where s.sid=r.sid and  
r.bid=b.bid and s.sname='Lubber';
```

COLOR

Red

Green

8. Find the names of sailors who have reserved both red and green boats.

```
SQL> select s.sname from SAILORS s,BOATS b,RESERVES r where s.sid=r.sid and  
r.Bid=b.Bid and b.color='Red' intersect select s.sname from SAILORS s,BOATS  
b,RESERVES r WHERE s.sid=r.sid and r.Bid=b.Bid and b.color='Green';
```

SNAME

Dustin

Horatio

Lubber

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

```
SQL> SELECT DISTINCT s.sname FROM SAILORS s, RESERVES r WHERE s.sid  
= r.sid;
```

SNAME

Lubber

Dustin

Horatio

10. Find the ids and names of sailors who have reserved two different boats on the same day.

```
SQL> SELECT DISTINCT s.sid,s.sname FROM SAILORS s,RESERVES r1,RESERVES r2
WHERE s.sid=r1.sid AND s.sid=r2.sid AND r1.day=r2.day AND r1.Bid<>r2.Bid;
```

SID	SNAME

22	Dustin

11. Find the name and the age of the youngest sailor.

```
SQL> select s.sname,s.age from sailors s where s.age<=all(select age from sailors);
```

SNAME	AGE

Zorba	16

12. Find the names and ratings of a sailor whose rating is better than some sailor called Horatio.

```
SQL> select s.sname,s.rating from sailors s where s.rating>any(select s2.rating from sailors
s2 where s2.sname='Horatio');
```

SNAME	RATING

Rusty	10
Zorba	10
Horatio	9
Lubber	8
Andy	8

13. Find the names of sailors who have reserved all boats.

```
SQL> select s.sname from sailors s where NOT EXISTS ( select b.bid from boats b where
NOT EXISTS ( select r.bid from reserves r where r.bid = b.bid and r.sid = s.sid
) );
```

SNAME

Dustin

14. Count the number of different sailor names.

```
SQL> select count(distinct s.sname)from sailors s;
```

COUNT(DISTINCTS.SNAME)

9

15. Calculate the average age of all sailors.

```
SQL> SELECT AVG(s.age) FROM SAILORS S;
```

AVG(S.AGE)

36.9

16. Find the average age of sailors for each rating level.

SQL> select s.rating,avg(s.age)as avg_age from SAILORS s group by s.rating;

RATING	AVG_AGE
1	33
8	40
7	40
3	45
10	25.5
9	35

6 rows selected.

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

SQL> select s.rating,avg(s.age)as avg_age from SAILORS s group by s.rating having count(*)>1;

RATING	AVG_AGE
1	33
8	40
7	40
3	45
10	25.5

SET 3

Consider the following schema for OrderDatabase:

SALESMAN (Salesman_id, Name, City, Commission)

CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)

ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

Write SQL queries to

SALESMAN_ID	NAME	CITY	COMMISSION
1000	JOHN	BANGALORE	25 %
2000	RAVI	BANGALORE	20 %
3000	KUMAR	MYSORE	15 %
4000	SMITH	DELHI	30 %
5000	HARSHA	HYDRABAD	15 %

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
10	PREETHI	BANGALORE	100	1000
11	VIVEK	MANGALORE	300	1000
12	BHASKAR	CHENNAI	400	2000
13	CHETHAN	BANGALORE	200	2000
14	HANATHA	BANGALORE	400	3000

SELECT * FROM ORDERS;

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
50	5000	04-MAY-17	10	1000
51	450	20-JAN-17	10	2000
52	1000	24-FEB-17	13	2000
53	3500	13-APR-17	14	3000
54	550	09-MAR-17	12	2000

SQL> CREATE TABLE SALESMAN (Salesman_id INT NOT NULL PRIMARY KEY,
Name VARCHAR (20), City VARCHAR (20), Commission number);

Table created.

```
SQL> INSERT INTO SALESMAN VALUES(1000,'JOHN','BANGALORE',25);
```

1 row created.

```
SQL> INSERT INTO SALESMAN VALUES(2000,'RAVI','BANGALORE',20);
```

1 row created.

```
SQL> INSERT INTO SALESMAN VALUES(3000,'KUMAR','MYSORE',15);
```

1 row created.

```
SQL> INSERT INTO SALESMAN VALUES(4000,'SMITH','DELHI',30);
```

1 row created.

```
SQL> INSERT INTO SALESMAN VALUES(5000,'HARSHA','HYDRABAD',15);
```

1 row created.

```
SQL> SELECT * FROM SALESMAN;
```

SALESMAN_ID	NAME	CITY	COMMISSION
1000	JOHN	BANGALORE	25
2000	RAVI	BANGALORE	20
3000	KUMAR	MYSORE	15
4000	SMITH	DELHI	30
5000	HARSHA	HYDRABAD	15

```
SQL> CREATE TABLE CUSTOMER(Customer_id INT PRIMARY KEY,  
Cust_Name VARCHAR(20), City VARCHAR(20), Grade NUMBER,Salesman_id INT  
REFERENCES SALESMAN(SALESMAN_ID));
```

Table created.

```
SQL> INSERT INTO CUSTOMER
VALUES(10,'PREETHI','BANGALORE',100,1000);
```

1 row created.

```
SQL> INSERT INTO CUSTOMER
VALUES(11,'VIVEK','MANGALORE',300,1000);
```

1 row created.

```
SQL> INSERT INTO CUSTOMER
VALUES(12,'BHASKAR','CHENNAI',400,2000);
```

1 row created.

```
SQL> INSERT INTO CUSTOMER
VALUES(13,'CHETHAN','BANGALORE',200,2000);
```

1 row created.

```
SQL> INSERT INTO CUSTOMER
VALUES(14,'MAMATHA','BANGALORE',400,3000);
```

1 row created.

```
SQL> SELECT * FROM CUSTOMER;
```

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
10	PREETHI	BANGALORE	100	1000
11	VIVEK	MANGALORE	300	1000
12	BHASKAR	CHENNAI	400	2000
13	CHETHAN	BANGALORE	200	2000
14	MAMATHA	BANGALORE	400	3000

```
SQL> CREATE TABLE ORDERS(Ord_No NUMBER, Purchase_Amt NUMBER,
Ord_Date DATE, Customer_id REFERENCES
CUSTOMER(CUSTOMER_ID),Salesman_id REFERENCES
SALESMAN(SALESMAN_ID));
```

Table created.

```
SQL> INSERT INTO ORDERS VALUES(50,5000,'04-MAY-17',10,1000);
```

1 row created.

```
SQL> INSERT INTO ORDERS VALUES(51,450,'20-JAN-17',10,2000);
```

1 row created.

```
SQL> INSERT INTO ORDERS VALUES(52,1000,'24-FEB-17',13,2000);
```

1 row created.

```
SQL> INSERT INTO ORDERS VALUES(53,3500,'13-APR-17',14,3000);
```

1 row created.

```
SQL> INSERT INTO ORDERS VALUES(54,550,'09-MAR-17',12,2000);
```

1 row created.

```
SQL> SELECT * FROM ORDERS;
```

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
--------	--------------	----------	-------------	-------------

50	5000	04-MAY-17	10	1000
51	450	20-JAN-17	10	2000
52	1000	24-FEB-17	13	2000
53	3500	13-APR-17	14	3000
54	550	09-MAR-17	12	2000

1. Count the customers with grades above Bangalore's Average.

```
SQL> SELECT GRADE, COUNT (DISTINCT CUSTOMER_ID) FROM
CUSTOMER GROUP BY GRADE HAVING GRADE > (SELECT AVG(GRADE) FROM
CUSTOMER WHERE CITY='BANGALORE');
```

```
GRADE  COUNT(DISTINCTCUSTOMER_ID)
```

```
-----
```

```
300          1
400          2
```

2. Find the name and numbers of all salesmen who had more than one customer

```
SQL> SELECT SALESMAN_ID, NAME FROM SALESMAN A WHERE 1 <
(SELECT COUNT (*) FROM CUSTOMER WHERE
SALESMAN_ID=A.SALESMAN_ID);
```

```
SALESMAN_ID NAME
```

```
-----
```

```
1000      JOHN
2000      RAVI
```

3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNIONoperation.)

```
SQL> SELECT SALESMAN.SALESMAN_ID, NAME, CUST_NAME,
COMMISSION FROM SALESMAN, CUSTOMER WHERE SALESMAN.CITY =
CUSTOMER.CITY UNION SELECT SALESMAN_ID, NAME, 'NO MATCH',
COMMISSION FROM SALESMAN WHERE NOT CITY = ANY (SELECT CITY FROM
CUSTOMER) ORDER BY 2 DESC;
```

SALESMAN_ID	NAME	CUST_NAME	COMMISSION
4000	SMITH	NO MATCH	30
2000	RAVI	CHETHAN	20
2000	RAVI	MAMATHA	20
2000	RAVI	PREETHI	20
3000	KUMAR	NO MATCH	15
1000	JOHN	CHETHAN	25
1000	JOHN	MAMATHA	25
1000	JOHN	PREETHI	25
5000	HARSHA	NO MATCH	15

9 rows selected.

4. Create a view that finds the salesman who has the customer with the highest order of the day.

```
SQL> CREATE VIEW ELITSALESMAN AS SELECT B.ORD_DATE,
A.SALESMAN_ID, A.NAME FROM SALESMAN A, ORDERS B WHERE
A.SALESMAN_ID = B.SALESMAN_ID AND B.PURCHASE_AMT=(SELECT
MAX (PURCHASE_AMT) FROM ORDERS C WHERE C.ORD_DATE =
B.ORD_DATE);
```

View created.

5. Demonstrate the DELETE operation by removing salesmen with id 1000. All his orders must also be deleted.

```
SQL> DELETE FROM SALESMAN WHERE SALESMAN_ID=1000;
```

1 row deleted

SET 4**DCL AND TCL**

```
CREATE TABLE STUDENT(ROLLNO INTEGER(5),FIRSTNAME  
VARCHAR(2),LASTNAME VARCHAR(20));
```

Table created.

```
INSERT INTO STUDENT(60,'TOM',EMPHREM');
```

1 row created.

```
INSERT INTO STUDENT(18,'ANJU',SAJI');
```

1 row created.

```
INSERT INTO STUDENT(10,'AMMU',RAJU');
```

1 row created.

1. DCL

```
GRANT SELECT USERS TO 'TOM'@' LOCALHOST;
```

```
REVOKE SELECT,UPDATE ON STUDENT FROM BCA,MCA;
```

2. TCL

```
DELETE FROM STUDENT WHERE ROLLNO=18;
```

```
COMMIT;
```

```
DELETE FROM STUDENT WHERE ROLLNO=10;
```

```
SAVEPOINT ROLLNO;
```

SET 5**VIEWS**

```
SQL> CREATE TABLE employee(SSN VARCHAR2(20),FNAME  
VARCHAR2(20),LNAME VARCHAR2(20),ADDRESS VARCHAR2(20),SEX  
VARCHAR(1),SALARY NUMBER(38));
```

Table created.

```
SQL> insert into employee values('abc','Amrutha','biju','abc','F',25000);
```

1 row created.

```
SQL> insert into employee values('dbc','Anite','jose','jjjk','F',25000);
```

1 row created.

```
SQL> insert into employee values('cbc','Anna','maria','asd','F',25000);
```

1 row created.

```
SQL> insert into employee values('bbc','Bharathi','S','sss','F',25000);
```

1 row created.

1. Creating a views (with and without check option)

```
SQL> CREATE VIEW sales_staff AS SELECT FNAME,SSN FROM employee;
```

View created.

2. Selecting from a view.

```
SQL> select * from sales_staff;
```

FNAME	SSN
Amrutha	abc
Anite	dbc
Anna	cbc
Bharathi	bdc

SET 6**Joins****EmployeeDetail table**

	EmployeeID	FirstName	LastName	Salary	JoiningDate	Department	Gender
1	1	Vikas	Ahlawat	600000.00	2013-02-15 11:16:28.290	IT	Male
2	2	nikita	Jain	530000.00	2014-01-09 17:31:07.793	HR	Female
3	3	Ashish	Kumar	1000000.00	2014-01-09 10:05:07.793	IT	Male
4	4	Nikhil	Sharma	480000.00	2014-01-09 09:00:07.793	HR	Male
5	5	anish	kadian	500000.00	2014-01-09 09:31:07.793	Payroll	Male

ProjectDetail table

	ProjectDetailID	EmployeeDetailID	ProjectName
1	1	1	Task Track
2	2	1	CLP
3	3	1	Survey Managment
4	4	2	HR Managment
5	5	3	Task Track
6	6	3	GRS
7	7	3	DDS
8	8	4	HR Managment
9	9	6	GL Managment

- 1. Get employee name, project name order by firstname from "EmployeeDetail" and "ProjectDetail" for those employee which have assigned project already.**

```
SELECT FirstName,ProjectName FROM [EmployeeDetail] A INNER JOIN [ProjectDetail] B ON A.EmployeeID = B.EmployeeDetailID ORDER BY FirstName ;
```

- 2. Get employee name, project name order by firstname from "EmployeeDetail" and "ProjectDetail" for all employee even they have not assigned project.**

```
SELECT FirstName,ProjectName FROM [EmployeeDetail] A LEFT OUTER JOIN [ProjectDetail] B ON A.EmployeeID = B.EmployeeDetailID ORDER BY FirstName ;
```

3. Get employee name, project name order by firstname from "EmployeeDetail" and "ProjectDetail" for all employee if project is not assigned then display "-No Project Assigned".

```
SELECT FirstName, ISNULL(ProjectName, '-No Project  
Assigned') FROM [EmployeeDetail] A LEFT OUTER JOIN [ProjectDetail] B  
ON A.EmployeeID = B.EmployeeDetailID ORDER BY FirstName ;
```

4. Get all project name even they have not matching any employeeid, in left table, order by firstname from "EmployeeDetail" and "ProjectDetail".

```
SELECT FirstName, ProjectName FROM [EmployeeDetail] A RIGHT OUTER JOIN  
[ProjectDetail] B ON A.EmployeeID = B.EmployeeDetailID ORDER BY FirstName ;
```

5. Get complete record(employee name, project name) from both tables([EmployeeDetail],[ProjectDetail]), if no match found in any table then show NULL.

```
SELECT FirstName, ProjectName FROM [EmployeeDetail] A FULL OUTER JOIN  
[ProjectDetail] B ON A.EmployeeID = B.EmployeeDetailID ORDER BY FirstName ;
```

6. Write a query to find out the employee name who has not assigned any project, and display "-No Project Assigned" (tables :- [EmployeeDetail],[ProjectDetail]).

```
SELECT FirstName, ISNULL(ProjectName, '-No Project Assigned') AS [ProjectName]  
FROM [EmployeeDetail] A LEFT OUTER JOIN [ProjectDetail] B ON A.EmployeeID =  
B.EmployeeDetailID WHERE ProjectName IS NULL ;
```

7. Write a query to find out the project name which is not assigned to any employee (tables :- [EmployeeDetail],[ProjectDetail]).

```
SELECT ProjectName FROM [EmployeeDetail] A RIGHT OUTER JOIN [ProjectDetail] B
ON A.EmployeeID = B.EmployeeDetailID
WHERE FirstName IS NULL ;
```

8. Write down the query to fetch EmployeeName & Project who has assign more than one project.

```
Select EmployeeID, FirstName, ProjectName from [EmployeeDetail] E INNER JOIN
[ProjectDetail] P ON E.EmployeeID = P.EmployeeDetailID WHERE EmployeeID IN
(SELECT EmployeeDetailID FROM [ProjectDetail] GROUP BY EmployeeDetailID
HAVING COUNT(*) >1 ) ;
```

9. Write down the query to fetch ProjectName on which more than one employee are working along with EmployeeName.

```
Select P.ProjectName, E.FName from ProjectDetails P INNER JOIN EmployeeDetails E on
p.EmployeeId = E.Id where P.ProjectName in(select ProjectName from ProjectDetails group
by ProjectName having COUNT(1)>1) ;
```

SET 7**PL/SQL****1. Write a PL/SQL program to print Hello world**

```
SQL> SET SERVEROUTPUT ON;
```

```
SQL> DECLARE
```

```
2  message varchar2(20):='Hello World!';
```

```
3  BEGIN
```

```
4  dbms_output.put_line(message);
```

```
5  END;
```

```
6  /
```

Hello World!

PL/SQL procedure successfully completed.

2. Write a PL/SQL block to find the maximum number from the given three numbers.

```
SQL> DECLARE
```

```
2  a number;
```

```
3  b number;
```

```
4  c number;
```

```
5  begin
```

```
6  a:=&a;
```

```
7  b:=&b;
8  c:=&c;
9  if(a>b and a>c)then
10 dbms_output.put_line('a is maximum'||a);
11 elsif(b>a and b>c)then
12 dbms_output.put_line('b is maximum'||b);
13 else
14 dbms_output.put_line('c is maximum'||c);
15 end if;
16 end;
17 /
```

Enter value for a: 4

old 6: a:=&a; new

6: a:=4; Enter value

for b: 2 old 7:

b:=&b; new 7:

b:=2; Enter value for

c: 5 old 8: c:=&c;

new 8: c:=5; c is

maximum5

PL/SQL procedure successfully completed.

3. Write a PL/SQL program to print integers from 1 to 10 by using PL/SQL FOR loop

```
SQL> DECLARE
```

```
2  n_times NUMBER:=10;
```

```
3  BEGIN
```

```
4  FOR n_i IN 1..n_times LOOP
```

```
5  DBMS_OUTPUT.PUT_LINE(n_i);
```

```
6  END LOOP;
```

```
7  END;
```

```
8  /
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

PL/SQL procedure successfully completed.

4. Write a program to accept a number and find the sum of the digits .

SQL> declare

2 n number(5):=&n;

3 s number:=0;

4 r number(2):=0;

5 begin

6 while n!=0

7 loop

8 r:=mod(n,10);

9 s:=s+r;

10 n:=trunc(n/10);

11 end loop;

12 dbms_output.put_line('sum of digits of given numbers is '||s);

13 end;

14 /

Enter value for n: 234 old

2: n number(5):=&n; new

2: n number(5):=234; sum
of digits of given numbers
is 9

PL/SQL procedure successfully completed.

5. Find the greatest number of inputs from the console.

SQL> declare

2 a number(2) :=&value_of_a;

3 b number(2) :=&value_of_b;

4 Begin

5 if a<b then

6 dbms_output.put_line(' Smaller Value is '||a);

7 elsif a>b then

8 dbms_output.put_line(' Smaller Value is '||b);

9 else

10 dbms_output.put_line(' Both no. are equal ');

11 end if;

12 END;

13 /

Enter value for value_of_a: 12 old 2:

a number(2) :=&value_of_a; new 2: a

number(2) :=12; Enter value for

value_of_b: 33 old 3: b number(2)

:=&value_of_b; new 3: b number(2)

:=33;

Smaller Value is 12

PL/SQL procedure successfully completed.

6. Reading the values from EMPLOYEE table.

Create table employee(ssn number(2),fname varchar(20),lname varchar(20),salary number(38));

Table created.

Insert into employee values(101,'amrutha','biju',75000);

1 row created.

Insert into employee values(102,'anite','jose',75000);

1 row created.

Insert into employee values(103,'anna','maria',75000);

1 row created.

Insert into employee values(104,'bharathi','s',75000);

1 row created.

1 Declare

2 v_name employee.fname%type;

3 v_job employee.lname%type;

4 v_sal employee.salary%type;

5 Begin

6 select fname,lname,salary into

v_fname, v_lname, v_salary from

employee where ssn =102;

7 dbms_output.put_line(v_fname||' '||v_lname||' '||v_salary);

8 End;

9 /

fname lname salary

anite jose 75000

SET 8**Named PL SQL Procedure and Functions****1. Procedure**

```
SQL> SET SERVEROUTPUT ON;
```

```
SQL> CREATE OR REPLACE PROCEDURE welcome_msg (p_name IN VARCHAR2)
```

```
2 IS
```

```
3 BEGIN
```

```
4 dbms_output.put_line ('Welcome' || p_name);
```

```
5 END;
```

```
6 /
```

Procedure created.

```
SQL> EXEC welcome_msg ('Guru99');
```

WelcomeGuru99

PL/SQL procedure successfully completed.

2. Procedure

```
SQL> CREATE OR REPLACE PROCEDURE welcome_msg (p_name IN  
VARCHAR2,salary out number)
```

```
2 IS
```

```
3 BEGIN    salary:=10000;
```

```
4 dbms_output.put_line ('Welcome ' || p_name);
```

```
5 END;
```

```
6 /
```

Procedure created.

```
SQL> var sal number;
```

```
SQL> EXEC welcome_msg ('Amrutha',:sal);
```

Welcome Amrutha

PL/SQL procedure successfully completed. Print sal;

3. Function

```
SQL> CREATE OR REPLACE FUNCTION welcome_msg_func ( p_name IN  
VARCHAR2) RETURN VARCHAR2
```

```
2 IS
```

```
3 BEGIN
```

```
4 RETURN ('Welcome '|| p_name);  END;
```

```
5 /
```

Function created.

```
SQL> DECLARE
```

```
2 lv_msg VARCHAR2(250);
```

```
3 BEGIN
```

```
4lv_msg:=welcome_msg_func('Amrutha');
```

```
5 dbms_output.put_line(lv_msg);
```

```
6 END;
```

```
7 /
```

Welcome Amrutha

PL/SQL procedure successfully completed.

```
SQL> SELECT welcome_msg_func('Amrutha') FROM DUAL;
```

```
WELCOME_MSG_FUNC('Amrutha')
```

Welcome Amrutha

SET 9**PL/SQL Cursor , Triggor****1. PL/SQL procedure**

```
SQL> create table stud_file(sid number, name varchar(20), m1 number, m2 number);
```

Table created.

```
SQL> insert into stud_file values(1,'anu',40,45);
```

1 row created.

```
SQL> insert into stud_file values(2,'binu',48,45);
```

1 row created.

```
SQL> insert into stud_file values(3,'cini',30,45);
```

1 row created.

```
SQL> insert into stud_file values(4,'dini',30,25);
```

1 row created.

```
1.SQL> declare
```

```
2  id constant number :=1;
```

```
3  sname studs_file.name%type;
```

```
4  mark1 studs_file.m1%type;
```

```
5  mark2 studs_file.m2%type;
```

```
6  total number:=0;
```

```
7  begin
```

```
8  select name,m1,m2 into sname,mark1,mark2 from studs_file where sid=id;
```

```
9  total:=mark1+mark2;
```

```
10 dbms_output.put_line('Total marks of student '||sname||' with id '||id||' is:
    ||total);
```

```
11 end;
```

```
12 /
```

Output

Total marks of student anu with id 1 is: 85

PL/SQL procedure successfully completed.

2. Cursor

SQL> declare

```
2  cursor stud_cursor is select * from studs_file;
3  stud_rec stud_cursor%rowtype;
4  total number:=0;
5  begin
6  open stud_cursor;
7  loop
8  fetch stud_cursor into stud_rec;
9  exit when stud_cursor%notfound ;
10 total:=stud_rec.m1+stud_rec.m2;
11 dbms_output.put_line('Total marks of student '||stud_rec.name||' is: '||total);
12 end loop;
13 end;
14 /
```

Output

Total marks of student anu is: 85

Total marks of student binu is: 93

Total marks of student cini is: 75

Total marks of student dini is: 55

PL/SQL procedure successfully completed.

```
SQL> create table stud_mark(sid number,total number);
```

Table created.

3. Trigger

```
SQL> create or replace trigger stud_trig after insert on studs_file
```

```
2 for each row
```

```
3 declare
```

```
4 tot number:=0;
```

```
5 begin
```

```
6 tot:=:new.m1+:new.m2;
```

```
7 insert into stud_mark values(:new.sid,tot);
```

```
8 DBMS_OUTPUT.PUT_LINE('AFTER INSERT trigger activated:');
```

```
9 end;
```

```
10 /
```

Trigger created.

```
SQL> insert into studs_file values(5,'rani',40,45);
```

AFTER INSERT trigger activated:

1 row created.

```
SQL> select * from stud_mark;
```

Output

SID	TOTAL
-----	-------

5	85
---	----

SET 10**Mongo DB****1. Student Database**

Create database, Create collection, insert data, find, find one, sort , limit, skip, distinct, projection.

Create a student database with the fields: (SRN, Sname , Degree, Sem , CGPA)

```
> use student1
```

```
switched to db student1
```

```
>db.stud1coll.insert({srn:110,sname:"Rahul",degree:"BCA",sem:6, CGPA:7.9})
```

OR

```
> doc1=({srn:110,sname:"Rahul",degree:"BCA",sem:6,CGPA:7.9})
```

```
>db.studcoll.insert (doc1)
```

Note:

insert 10 documents.

1. display all the documents

```
>db.studcoll.find()
```

2. Display all the students in BCA

```
>db.studcoll.find({ degree:"BCA"})
```

3. Display all the students in ascending order

```
>db. studcoll.find().sort({ sname:1 })
```

4. Display first 5 students

```
>db. studcoll.find().limit(5)
```

5. display students 5,6,7

```
>db. studcoll.find().skip(4).limit(3)
```

6. list the degree of student "Rahul"

```
>db. studcoll.find({degree:1, sname:"Rahul"})
```

7. Display students details of 5,6,7 in descending order of percentage

```
>db. studcoll.find().sort({CGPA:-1}).skip(4).limit(3)
```

8. Display the number of students in BCA

```
>db. studcoll.find({degree:"BCA"}).count()
```

9. Display all the degrees without _id

```
>db. studcoll.find({}, {_id:0})
```

10. Display all the distinct degrees

```
>db. studcoll.distinct("degree")
```

11. Display all the BCA students with CGPA greater than 6, but less than 7.5

```
>db. studcoll.find(degree:"BCA",{CGPA:{$gt:6, $lt:7.5}})
```

12. Display all the students in BCA and in 6th Sem

```
>db. studcoll.find({$and:[{degree:"BCA"},{sem:6}]})
```


2. Employee Database

Update modifiers (\$set, \$unset, \$inc, \$push, \$pushAll, \$pull, \$pullAll, \$addToSet)

Create an employee database with the fields: {eid, ename, dept, desig,salary, yoj, address{ dno, street, locality, city}}

```
> use empdb9
```

switched to db empdb9

```
> doc1 = {eid:001, ename:"Rahul", dept:"production", desig:"developer", salary:30000, yoj:2015, address:{dno:397, street:2, locality:"rmnagar", city:"bangalore" } }
```

```
{
```

```
"eid" : 1,
```

```
"ename" : "Rahul",
```

```
"dept" : "production",
```

```
"desig" : "developer",
```

```
"salary" : 30000,
```

```
"yoj" : 2015,
```

```
"address" : {
```

```
"dno" : 397,
```

```
"street" : 2,
```

```
"locality" : "rmnagar",
```

```
"city" : "bangalore"
```

```
}
```

```
}
```

```
>db.emp09.insert(doc1)
```

```
WriteResult({ "nInserted" : 1 })
```

Note:

insert 10 documents.

1. Display all the employees with salary in range (50000, 75000)

```
>db.emp09.find({ salary:{$gt:50000, $lt:75000}})
```

2. Display all the employees with design developer

```
>db.emp09.find({ design:"developer"})
```

3. Display the Salary of “Rahul”

```
>db.emp09.find({ name:"Rahul" },{ salary:1 })
```

4. Display the city of employee “Rahul”

```
>db.emp09.find({ name:"Rahul" },{ "address.city":1 })
```

5. Update the salary of developers by 5000 increment

```
>db.emp09.update({ design:"developer" },{ $inc:{ "salary":5000} })
```

6. Add field age to employee “Rahul”

```
>db.emp09.update({ name:"Rahul" },{ $set:{ age:"22" } })
```

7. Remove Yoj from “Rahul”

```
>db.emp09.update({ name:"Rahul" },{ $unset:{ yoj:1 } })
```

8. Add an array field project to “Rahul”

```
>db.emp09.update({ename:"Rahul"},{$push:{projects:"p1"}})
```

9. Add p2 and p3 project to “Rahul”

```
>db.emp09.update({ename:"Rahul"},{$pushAll:{projects:["p2","p3"]}})
```

10. Remove p3 from “Rahul”

```
>db.emp09.update({ename:"Rahul"},{$pull:{projects:"p3"}})
```

11. Add a new embedded object “contacts” with “email” and “phone” as array objects to “Rahul”

```
>db.emp09.update({ename:"Rahul"},{$push:{contacts:{phone:"9036240380",  
email:"abc@gmail.com"}}})
```

12. Add two phone numbers to “Rahul”

```
>db.emp09.update({ename:"Rahul"},{$addToSet:{ "contact.phone":[9738751143,988073078  
4]}})
```

SET 11**1. Usage of aggregate functions****//USE DATABASE**

```
> use comp;
```

```
switched to db comp
```

//CREATE COLLECTION WEBSITE

```
> db.createCollection('website');
```

```
{ "ok" : 1 }
```

//INSERT VALUES IN WEBSITE

```
>db.website.insert({'roll':'1','name':'harsh','amount':1000,'url':'www.yahoo.com'});
```

```
WriteResult({ "nInserted" : 1 })
```

```
>db.website.insert({'roll':'2','name':'jitesh','amount':2000,'url':'www.yahoo.com'});
```

```
WriteResult({ "nInserted" : 1 })
```

```
>db.website.insert({'roll':'3','name':'rina','amount':3000,'url':'www.google.com'});
```

```
WriteResult({ "nInserted" : 1 })
```

```
>db.website.insert({'roll':'4','name':'ash','amount':4000,'url':'www.gmail.com'});
```

```
WriteResult({ "nInserted" : 1 })
```

```
>db.website.insert({'roll':'5','name':'ash','amount':1000,'url':'www.pvg.com'});
```

```
WriteResult({ "nInserted" : 1 })
```

//SUM AGGREGATE

```
> db.website.aggregate({$group:{_id:"$name","total":{$sum:"$amount"}}});
```

```
{ "_id" : "ash", "total" : 5000 }
```

```
{ "_id" : "rina", "total" : 3000 }
```

```
{ "_id" : "jitesh", "total" : 2000 }
```

```
{ "_id" : "harsh", "total" : 2000 }
```

//AVG AGGREGATE

```
> db.website.aggregate({$group:{_id:"$name","total":{$avg:"$amount"}}});
```

```
{ "_id" : "ash", "total" : 2500 }
```

```
{ "_id" : "rina", "total" : 3000 }
```

```
{ "_id" : "jitesh", "total" : 2000 }
```

```
{ "_id" : "harsh", "total" : 1000 }
```

//MIN AGGREGATION

```
> db.website.aggregate({$group:{_id:"$name","total":{$min:"$amount"}}});
```

```
{ "_id" : "ash", "total" : 1000 }
```

```
{ "_id" : "rina", "total" : 3000 }
```

```
{ "_id" : "jitesh", "total" : 2000 }
```

```
{ "_id" : "harsh", "total" : 1000 }
```

//MAX AGGREGATION

```
> db.website.aggregate({$group:{_id:"$name","total":{$max:"$amount"}}});

{ "_id" : "ash", "total" : 4000 }

{ "_id" : "rina", "total" : 3000 }

{ "_id" : "jitesh", "total" : 2000 }

{ "_id" : "harsh", "total" : 1000 }
```

//FIRST AGGREGATION

```
> db.website.aggregate({$group:{_id:"$name","total":{$first:"$amount"}}});

{ "_id" : "ash", "total" : 4000 }

{ "_id" : "rina", "total" : 3000 }

{ "_id" : "jitesh", "total" : 2000 }

{ "_id" : "harsh", "total" : 1000 }
```

//LAST AGGREGATION

```
> db.website.aggregate({$group:{_id:"$name","total":{$last:"$amount"}}});

{ "_id" : "ash", "total" : 1000 }

{ "_id" : "rina", "total" : 3000 }

{ "_id" : "jitesh", "total" : 2000 }

{ "_id" : "harsh", "total" : 1000 }
```

//PUSH AGGREGATION

```
> db.website.aggregate({$group:{_id:"$name","total":{$push:"$amount"}}});  
  
{ "_id" : "ash", "total" : [ 4000, 1000 ] }  
  
{ "_id" : "rina", "total" : [ 3000 ] }  
  
{ "_id" : "jitesh", "total" : [ 2000 ] }  
  
{ "_id" : "harsh", "total" : [ 1000, 1000 ] }
```

//COUNT AGGREGATION

```
> db.website.aggregate({$group:{_id:"$name","total": {$sum:1}}});  
  
{ "_id" : "ash", "total" : 2 }  
  
{ "_id" : "rina", "total" : 1 }  
  
{ "_id" : "jitesh", "total" : 1 }  
  
{ "_id" : "harsh", "total" : 2 }
```

//ADDTOSET AGGREGATE

```
> db.website.aggregate({$group:{_id:"$name","total":{$addToSet:"$amount"}}});  
  
{ "_id" : "ash", "total" : [ 1000, 4000 ] }  
  
{ "_id" : "rina", "total" : [ 3000 ] }  
  
{ "_id" : "jitesh", "total" : [ 2000 ] }  
  
{ "_id" : "harsh", "total" : [ 1000 ] }
```

SET 12**PyMongo**

PyMongo library, which is a Python driver for MongoDB, a popular NoSQL database.

1. Importing the necessary modules:

```
import pymongo
```

2. Creating a MongoDB client and connecting to a database:

```
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
```

3. Accessing a specific database:

```
mydb = myclient["mydatabase"]
```

4. Accessing a specific collection within the database:

```
mycol = mydb["customers"]
```

5. Inserting a single document into the collection:

```
myd = {"name": "Divya", "address": "highway37"}  
q = mycol.insert_one(myd) print(q.inserted_id)
```

6. Inserting multiple documents into the collection:

```
mydict = [  
    {"name": "John", "address": "highway 37"},  
    {"name": "Aby", "address": "Cross 30"},  
    {"name": "Jerry", "address": "River Road 45"}  
]  
x = mycol.insert_many(mydict)  
print(x.inserted_ids)
```


7. Retrieving a single document from the collection:

```
y = mycol.find_one()
print(y)
```

8. Retrieving all documents from the collection:

```
for z in mycol.find():
    print(z)
```

9. Retrieving documents while excluding the "name" field:

```
for a in mycol.find({}, {"name": 0}):
    print(a)
```

10. Deleting a document from the collection:

```
myquery = {"name": "John"}
mycol.delete_one(myquery)
```

11. Deleting multiple documents from the collection:

```
c = mycol.delete_many({})
print(c.deleted_count, "documents/rows deleted")
```

12. Updating a document in the collection:

```
myquery = {"address": "Canyon 123"}
newval = {"$set": {"address": "highway37"}}
f = mycol.update_one(myquery, newval)
print(f.modified_count, "Document updated")
```

13. Updating multiple documents in the collection:

```
myquery = {"address": "highway 37"}  
newval = {"$set": {"address": "Canyon123"}}  
f = mycol.update_many(myquery, newval)  
print(f.modified_count, "Document updated")
```

14. Limiting the number of retrieved documents to 2:

```
for p in mycol.find().limit(2):  
    print(p)
```

15. Sorting the retrieved documents by the "name" field:

```
mydoc = mycol.find().sort("name")  
for l in mydoc: print(l)
```

16. Dropping the collection (deleting all documents within it):

```
mycol.drop()
```