Group A: Assignment No -2 & 3

Aim: Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator

Account(Acc_no, branch_name,balance)
branch(branch_name,branch_city,assets)
customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)

Solve following query:

Create above tables with appropriate constraints like primary key, foreign key, check constrains, not null etc.

1. **Find the names of all branches in loan relation.**
   select distinct bname from Loan;

2. **Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.**
   select bname, lno from Loan where bname='Akurdi' and amount>12000;

3. **Find all customers who have a loan from bank. Find their names,loan_no and loan amount.**
   select Borrower.cname,Loan.lno,Loan.amount from Loan inner join Borrower on Loan.lno=Borrower.lno
   OR
   select Borrower.cname,Loan.lno,Loan.amount from Loan, Borrower where Loan.lno= Borrower.lno

4. **List all customers in alphabetical order who have loan from Akurdi branch.**
   select Borrower.cname from Borrower inner join Loan on Borrower.lno=Loan.lno where Loan.bname='Akurdi' order by Borrower.cname asc;
   OR
   select Borrower.cname from Borrower,Loan where Borrower.lno=Loan.lno and Loan.bname='Akurdi' order by Borrower.cname;

5. **Find all customers who have an account or loan or both at bank**.
   select Depositor.cname from Depositor left join Borrower on Depositor.cname=Borrower.cname union select Borrower.cname from Depositor right join Borrower on Borrower.cname=Depositor.cname;
   OR
   select c.name from Depositer union select c.name from Borrower;

6. **Find all customers who have both account and loan at bank.**
   select Depositor.cname from Depositor inner join Borrower on Borrower.cname=Depositor.cname;
   OR
   select Depositer.cname, from Depositer, Borrower where Depositer.cname=Borrower.cname;

7. **Find all customer who have account but no loan at the bank.**
   select cname, from Depositer where cname not in (select cname from Borrower);

   OR
   select Depositor.cname from Depositor left join Borrower on Depositor.cname=Borrower.cname where Borrower.cname is null;

8. **Find average account balance at Akurdi branch.**
   select avg(balance), bname from Account where bname="Akurdi"

9. **Find the average account balance at each branch**
   select avg(balance),bname from Account group by bname;

10. **Find no. of depositors at each branch.**
    select bname,count(*) from Account group by bname;

11. **Find the branches where average account balance > 12000.**
    select bname, avg(balance) from Account group by bname having avg(balance)>12000;

12. **Find number of tuples in customer relation**.
    select count(c_name) from Customer;

    OR

    select count(*) from Customer;

13. **Calculate total loan amount given by bank.**
    select bname,sum(amount) from Loan group by bname;
    OR
    select sum(amount) from Loan;

14. **Delete all loans with loan amount between 1300 and 1500.**
    delete from Loan where amount>=1300 and amount<=1500;

15. **Delete all tuples at every branch located in Nigdi.**
    delete from Branch where bcity='Nigdi';

Group A: Assignment No -4

Aim: Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

1. **Reterieve the address of customer Fname as 'xyz' and Lname as 'pqr'.**
   select cust_mstr.fname,cust_mstr.lname,add_dets.add1,add_dets.add2,add_dets.state, add_dets.city,add_dets.pincode from cust_mstr inner join add_dets on cust_mstr.cust_no=add_dets.code_no where cust_mstr.fname="XYZ" and cust_mstr.lname="PQR";

2. **List the customer holding fixed deposit of amount more than 5000.**
   select fname,lname,amt from cust_mstr cust inner join acc_fd_cust_dets acc on cust.cust_no=acc.code_no inner join fd_dets fd on acc.acc_fd_no=fd.fd_sr_no where fd.amt>5000;

3. **List the employee details along with branch names to which they belong.**
   select * from emp_mstr inner join branch_mstr on emp_mstr.b_no=branch_mstr.b_no;

4. **List the employee details along with contact details using left outer join & right join.**
   select * from emp_mstr left join cntc_dets on emp_mstr.emp_no=cntc_dets.code_no union select * from emp_mstr right join cntc_dets on emp_mstr.emp_no=cntc_dets.code_no;

5. **List the customer who do not have bank branches in their vicinity.**
   select cust_mstr.fname,cust_mstr.lname from cust_mstr left join acc_fd_cust_dets on cust_mstr.cust_no=acc_fd_cust_dets.code_no where acc_fd_cust_dets.code_no is null;

6. **Create View on Borrower table by selecting any two columns and perform insert,update and delete operations.**
   create view view1 as select bname,sum(Amount) from Borrower group by bname;

7. **Create view on borrower and depositor table by selecting any one column from each table.Perform insert,delete and update operations.**
   create view view2 as select Borrower1.bno,Borrower1.cname,Depositor1.Balance from Borrower1 inner join Depositor1 on Depositor1.dno=Borrower1.bno;

8. **Create updateable View on Borrower table by selecting any two columns and perform insert,update and delete operations.**
   create view vupborrower1 as select bno,cname,bname,Amount from Borrower1;

Group A: Assignment No -5

Aim: Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory. Write a PL/SQL block of code for the following requirements:-

Schema:

1. Borrower(Rollno, Name, DateofIssue, NameofBook, Status)
2. Fine(Roll_no,Date,Amt)

- Accept roll_no & name of book from user.
- Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5 per day.
- If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
- After submitting the book, status will change from I to R.
- If condition of fine is true, then details will be stored into fine table.

*********Create table fine and Borrower:*********

```
SQL> create table borrower(rollno int, name char(10), dateofissue date, nameofbook
char(10), status char(10));
Table created.

SQL> create table fine(rollno int, fdate date, amt int);
Table created.

SQL> desc borrower;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ROLLNO                                             NUMBER(38)
 NAME                                               CHAR(10)
 DATEOFISSUE                                        DATE
 NAMEOFBOOK                                         CHAR(10)
 STATUS                                             CHAR(10)

SQL> desc fine;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ROLLNO                                             NUMBER(38)
 FDATE                                              DATE
 AMT                                                NUMBER(38)
```

********* Insert values into Borrower table: *********

```
SQL> Insert into borrower values (101, 'Ram',to_date('20170923','YYYYMMDD'),'DBM
S', 'I');
1 row created.

SQL> Insert into borrower values (102, 'Sai',to_date('20170910','YYYYMMDD'),'CN', 'I');
1 row created.

SQL> Insert into borrower values (103, 'Laxman',to_date('20170928','YYYYMMDD'),'TOC',
'I');
```

1 row created.

SQL> Insert into borrower values (104, 'Sai',to_date('20170825','YYYYMMDD'),'SEPM','I');
1 row created.

SQL> Insert into borrower values (105, 'Ganesh',to_date('20170901','YYYYMMDD'),'IEEE',
'I');
1 row created.
SQL>

SQL> select * from borrower;

ROLLNO	NAME	DATEOFISS	NAMEOFBOOK	STATUS
101	Ram	23-SEP-17	DBMS	I
102	Sai	10-SEP-17	CN	I
103	Laxman	28-SEP-17	TOC	I
104	Sai	25-AUG-17	SEPM	I
105	Ganesh	01-SEP-17	IEEE	I

SQL> select * from fine;
no rows selected

*********Procedure for Calculating fine: *********

```
DECLARE
        p_nameofbook char(50);
        p_rollno number(3);
        p_dateofissue date;
        currentdate date;
        noofdays number(2);
        amount number;
        nodata EXCEPTION;
 BEGIN
        p_rollno := &rollno;
        p_nameofbook := '&nameofbook';
        currentdate := trunc(SYSDATE);
IF p_rollno <= 0 THEN
        RAISE nodata;
END IF;
        SELECT dateofissue into p_dateofissue FROM borrower WHERE rollno = p_rollno
        AND nameofbook =p_nameofbook;
        SELECT trunc(SYSDATE) - p_dateofissue  INTO noofdays from dual;
        dbms_output.put_line ('No of Days:' || noofdays);
 IF (noofdays > 30) THEN amount:= noofdays * 50;
 ELSIF (noofdays >= 15 AND noofdays <=30) THEN amount:= noofdays * 5;
 END IF;
 IF amount > 0 THEN
         INSERT INTO Fine values (p_rollno, sysdate, amount);
 END IF;
```

```
        UPDATE Borrower SET Status = 'R' WHERE rollno=p_rollno;
EXCEPTION
WHEN nodata THEN
        dbms_output.put_line('!!!!!Roll Number not found!!!!!');
 END;
 /
***** Output *****
Enter value for rollno: 101
old   9: p_rollno := &rollno;
new   9: p_rollno := 101;
Enter value for nameofbook: DBMS
old  10: p_nameofbook := '&nameofbook';
new  10: p_nameofbook := 'DBMS';
PL/SQL procedure successfully completed.

SQL> select * from fine;
no rows selected
SQL> select * from borrower;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
      101 Ram        23-SEP-17 DBMS       R
      102 Sai        10-SEP-17 CN         I
      103 Laxman     28-SEP-17 TOC        I
      104 Sai        25-AUG-17 SEPM       I
      105 Ganesh     01-SEP-17 IEEE       I
SQL>
================================================================
Enter value for rollno: 102
old   9: p_rollno := &rollno;
new   9: p_rollno := 102;
Enter value for nameofbook: CN
old  10: p_nameofbook := '&nameofbook';
new  10: p_nameofbook := 'CN';
PL/SQL procedure successfully completed.

SQL> select * from borrower;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
          ---------- ---------- --------- ---------- ----------
      101 Ram        23-SEP-17 DBMS       R
      102 Sai        10-SEP-17 CN         R
      103 Laxman     28-SEP-17 TOC        I
      104 Sai        25-AUG-17 SEPM       I
      105 Ganesh     01-SEP-17 IEEE       I

SQL> select * from fine;
   ROLLNO    FDATE           AMT
          ---------- --------- ----------
      102          28-SEP-17         90
```

```
==========================================================================
Enter value for rollno: 105
old   9: p_rollno := &rollno;
new   9: p_rollno := 105;
Enter value for nameofbook: IEEE
old  10: p_nameofbook := '&nameofbook';
new  10: p_nameofbook := 'IEEE';
PL/SQL procedure successfully completed.
SQL> select * from fine;
   ROLLNO FDATE           AMT
          ---------- --------- ----------
       102 28-SEP-17         90
       105 28-SEP-17        135

SQL> select * from borrower;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
          ---------- ---------- --------- ---------- ----------
       101 Ram        23-SEP-17 DBMS       R
       102 Sai        10-SEP-17 CN         R
       103 Laxman     28-SEP-17 TOC        I
       104 Sai        25-AUG-17 SEPM       I
       105 Ganesh     01-SEP-17 IEEE       R
==========================================================================
Enter value for rollno: 104
old   9: p_rollno := &rollno;
new   9: p_rollno := 104;
Enter value for nameofbook: SEPM
old  10: p_nameofbook := '&nameofbook';
new  10: p_nameofbook := 'SEPM';
PL/SQL procedure successfully completed.

SQL> select * from fine;
   ROLLNO FDATE           AMT
---------- --------- ----------
       102 28-SEP-17         90
       105 28-SEP-17        135
       104 28-SEP-17       1700

SQL> Select * from borrower;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
       101 Ram        23-SEP-17 DBMS       R
       102 Sai        10-SEP-17 CN         R
       103 Laxman     28-SEP-17 TOC        I
       104 Sai        25-AUG-17 SEPM       R
       105 Ganesh     01-SEP-17 IEEE       R
==========================================================
```

Group A: Assignment No -6

Aim: Write a PL/SQL block of code using parameterized Cursor that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall.

If the data in the first table already exist in the second table then that data should be skipped.

```
SQL> select * from oldt;
        ID          NAME
           ---------- ----------
         1           Prajakta
         3           Cristal

SQL> select * from newt;
        ID          NAME
           ---------- ----------
         2           Tanaz
         4           Sharvari


SQL> set serveroutput on
SQL>
DECLARE
        rollno number;
        flag int(2);
        cursor c_roll(rollno number)  is select * from  oldt where  id not in(select id from
        newt where newt.id=oldt.id) ;
        info newt%rowtype;
 BEGIN
        rollno := &rollno;
        flag :=0;
        open c_roll(rollno);
        loop fetch c_roll into info;
        exit when c_roll%notfound;
                if (info.id=rollno) then insert into newt values(info.id,info.Name);
                flag := 1;
                end if;
        end loop;
                if ( c_roll%rowcount = 0 or flag=0) then
                dbms_output.put_line('This record already exits in new table.');
                else dbms_output.put_line('Record updated in new table!');
                end if;
        close c_roll;
 END;
 /
```

******* OUTPUT *******

```
Enter value for rollno: 1
old   7: rollno := &rollno;
```

```
new   7: rollno := 1;
Record updated in new table!

PL/SQL procedure successfully completed.

SQL> select * from newt;
        ID          NAME
           ---------- ----------
         1          Prajakta
         2          Tanaz
         4          Sharvari

==========================================================================
Enter value for rollno: 3
old   7:     rollno := &rollno;
new   7:     rollno := 3;
Record updated in new table!

PL/SQL procedure successfully completed.

SQL> select * from newt;
        ID          NAME
           ---------- ----------
         1          Prajakta
         3          Cristal
         2          Tanaz
         4          Sharvari

==========================================================================
Enter value for rollno: 2
old   7:     rollno := &rollno;
new   7:     rollno := 2;
This record already exits in new table.

PL/SQL procedure successfully completed.

==========================================================================
Enter value for rollno: 1
old   7:     rollno := &rollno;
new   7:     rollno := 1;
This record already exits in new table.

PL/SQL procedure successfully completed.

==========================================================================
```

Group A: Assignment No -7

Aim: Write a Stored Procedure namely proc_Grade for the categorization of student.
If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class .
Write a PL/SQL block for using procedure created with above requirement.
stud_marks(roll_no, name, total_marks) result(Roll,Name, Class)

********* Create Table stud_marks and result: *********

```
create table stud_marks(roll_no number(20),name varchar2(20), total_marks number(20));
insert into stud_marks values(1,'Ganesh',1200);
insert into stud_marks values(2,'Ram',950);
insert into stud_marks values(3,'Sai',850);
insert into stud_marks values(4,'Laxman',800);
select * from stud_marks;
create table result (roll_no number(20),name varchar2(20), class varchar2(20));
select * from result;
```

******* Main Procedure proc_grade *******

```
Create or replace procedure proc_grade
(var_rollno in number,
p_roll_no out stud_marks.roll_no%type,
p_name out stud_marks.name%type,
p_total out stud_marks.total_marks%type)
AS
BEGIN
SELECT roll_no, name, total_marks into p_roll_no, p_name, p_total from stud_marks where
roll_no=var_rollno;
IF p_total <=1500 and p_total >= 990 THEN
        insert into result values(p_roll_no,p_name,'Distinction');
Else if p_total <=989 and p_total >= 900 THEN
        insert into result values(p_roll_no,p_name,'First Class');
Else if p_total <=899 and p_total >= 825 THEN
        insert into result values(p_roll_no,p_name,'HSC');
Else
        insert into result values(p_roll_no,p_name,'fail');
End if;
End if;
End if;
EXCEPTION
WHEN no_data_found then
dbms_output.put_line('Roll no  ' || var_rollno ||'  not found');
END;
/
```

******* Calling Procedure *******

```
DECLARE
        var_rollno number(20);
        p_roll_no stud_marks.roll_no%type;
```

```
        p_name stud_marks.name%type;
        p_total stud_marks.total_marks%type;
BEGIN
        var_rollno:=&var_rollno;
        Proc_grade(var_rollno,p_roll_no,p_name,p_total);
END;
/
```

==========

```
SQL> create table stud_marks(Roll_no number(20),name varchar2(20), total_marks number(20));
Table created.
SQL> insert into stud_marks values(1,'Ganesh',1200);
1 row created.
SQL> insert into stud_marks values(2,'Ram',950);
1 row created.
SQL> insert into stud_marks values(3,'Sai',850);
1 row created.
SQL> insert into stud_marks values(4,'Laxman',800);
1 row created.
SQL> select * from stud_marks;
  ROLL_NO NAME                 TOTAL_MARKS
---------- -------------------- -----------
         1 Ganesh                      1200
         2 Ram                          950
         3 Sai                          850
         4 Laxman                       800

SQL> create table result (roll_no number(20),name varchar2(20), class varchar2(20));
Table created.
SQL> select * from result;
no rows selected
SQL> Create or replace procedure proc_grade
  2  (var_rollno in number,
  3  p_roll_no out stud_marks.roll_no%type,
  4  p_name out stud_marks.name%type,
  5  p_total out stud_marks.total_marks%type)
  6  AS
  7  BEGIN
  8  SELECT roll_no, name, total_marks into p_roll_no, p_name, p_total from stud
_marks where roll_no=var_rollno;
  9  IF p_total <=1500 and p_total >= 990 THEN
 10  insert into result values(p_roll_no,p_name,'Distinction');
 11  Else if p_total <=989 and p_total >= 900 THEN
 12  insert into result values(p_roll_no,p_name,'First Class');
 13  Else if p_total <=899 and p_total >= 825 THEN
 14  insert into result values(p_roll_no,p_name,'HSC');
 15  Else
 16  insert into result values(p_roll_no,p_name,'fail');
```

```
 17  End if;
 18  End if;
 19  End if;
 20  EXCEPTION
 21  WHEN no_data_found then
 22  dbms_output.put_line('Roll no  ' || var_rollno ||'  not found');
 23  END;
 24  /
Procedure created.
==========
SQL> DECLARE
  2  var_rollno number(20);
  3  p_roll_no stud_marks.roll_no%type;
  4  p_name stud_marks.name%type;
  5  p_total stud_marks.total_marks%type;
  6  BEGIN
  7  var_rollno:=&var_rollno;
  8  Proc_grade(var_rollno,p_roll_no,p_name,p_total);
  9  END;
 10  /
Enter value for var_rollno: 2
old   7: var_rollno:=&var_rollno;
new   7: var_rollno:=2;

PL/SQL procedure successfully completed.
==========
SQL> select * from result;
   ROLL_NO NAME                 CLASS
---------- -------------------- --------------------
         2 Ram                  First Class

SQL> DECLARE
  2  var_rollno number(20);
  3  p_roll_no stud_marks.roll_no%type;
  4  p_name stud_marks.name%type;
  5  p_total stud_marks.total_marks%type;
  6  BEGIN
  7  var_rollno:=&var_rollno;
  8  Proc_grade(var_rollno,p_roll_no,p_name,p_total);
  9  END;
 10  /
Enter value for var_rollno: 1
old   7: var_rollno:=&var_rollno;
new   7: var_rollno:=1;
PL/SQL procedure successfully completed.

SQL> select * from result;
   ROLL_NO NAME                 CLASS
```

```
---------- -------------------- --------------------
         2 Ram                  First Class
         1 Ganesh                Distinction
```

==========

```
SQL> DECLARE
  2  var_rollno number(20);
  3  p_roll_no stud_marks.roll_no%type;
  4  p_name stud_marks.name%type;
  5  p_total stud_marks.total_marks%type;
  6  BEGIN
  7  var_rollno:=&var_rollno;
  8  Proc_grade(var_rollno,p_roll_no,p_name,p_total);
  9  END;
 10  /
Enter value for var_rollno: 3
old   7: var_rollno:=&var_rollno;
new   7: var_rollno:=3;
PL/SQL procedure successfully completed.

SQL> DECLARE
  2  var_rollno number(20);
  3  p_roll_no stud_marks.roll_no%type;
  4  p_name stud_marks.name%type;
  5  p_total stud_marks.total_marks%type;
  6  BEGIN
  7  var_rollno:=&var_rollno;
  8  Proc_grade(var_rollno,p_roll_no,p_name,p_total);
  9  END;
 10  /
Enter value for var_rollno: 4
old   7: var_rollno:=&var_rollno;
new   7: var_rollno:=4;
PL/SQL procedure successfully completed.
SQL> select * from result;
   ROLL_NO NAME                 CLASS
---------- -------------------- --------------------
         2 Ram                  First Class
         1 Ganesh                Distinction
         3 Sai                 HSC
         4 Laxman                fail
```

==========

Group A: Assignment No -8

Aim: Database Trigger (All Types: Row level, before and After Triggers).Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

Create table library(rollno int, name char(10), dateofissue date, nameofbook char(10), status char(10));
Create table library_audit(rollno int, name char(10), dateofissue date, nameofbook char(10), status char(10), ts timestamp);

Insert into library values (101, 'Ram',to_date('20170923','YYYYMMDD'),'DBMS', 'I');
Insert into library values (102, 'Sai',to_date('20170910','YYYYMMDD'),'CN', 'I');
Insert into library values (103, 'Laxman',to_date('20170928','YYYYMMDD'),'TOC', 'I');
Insert into library values (104, 'Sai',to_date('20170825','YYYYMMDD'),'SEPM', 'I');
Insert into library values (105, 'Ganesh',to_date('20170901','YYYYMMDD'),'IEEE', 'I');
Select * from library;
Select * from library_audit;
SQL> select * from library;

ROLLNO	NAME	DATEOFISS	NAMEOFBOOK	STATUS
101	Ram	23-SEP-17	DBMS	I
102	Sai	10-SEP-17	CN	I
103	Laxman	28-SEP-17	TOC	I
104	Sai	25-AUG-17	SEPM	I
105	Ganesh	01-SEP-17	IEEE	I

SQL> Create table library_audit(rollno int, name char(10), dateofissue date, nameofbook char(10), status char(10), ts timestamp);
Table created.
SQL> select * from library_audit;
no rows selected

****

AFTER INSERT Trigger – Row Level Trigger

****

```
CREATE OR REPLACE TRIGGER after_insert
        AFTER INSERT
        ON library
        FOR EACH ROW
BEGIN
         insert into library_audit values(:new.rollno, :new.name, :new.dateofissue,
        :new.nameofbook, :new.status, current_timestamp);
END;
/
```

Trigger created.
SQL> select * from library;

ROLLNO	NAME	DATEOFISS	NAMEOFBOOK	STATUS

```
---------- ---------- --------- ---------- ----------
       101 Ram        23-SEP-17 DBMS       I
       102 Sai        10-SEP-17 CN         I
       103 Laxman     28-SEP-17 TOC        I
       104 Sai        25-AUG-17 SEPM       I
       105 Ganesh     01-SEP-17 IEEE       I

SQL> select * from library_audit;
no rows selected
SQL> Insert into library values (106, 'Gajanan',to_date('20171001','YYYYMMDD'),'
DDA', 'I');
1 row created.
SQL> select * from library;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
       101 Ram        23-SEP-17 DBMS       I
       102 Sai        10-SEP-17 CN         I
       103 Laxman     28-SEP-17 TOC        I
       104 Sai        25-AUG-17 SEPM       I
       105 Ganesh     01-SEP-17 IEEE       I
       106 Gajanan    01-OCT-17 DDA        I
6 rows selected.
SQL> select * from library_audit;
     ROLLNO       NAME       DATEOFISS       NAMEOFBOOK       STATUS       TS
 ---------------------------------------------------------------------------------------------------------------
     106      Gajanan    01-OCT-17          DDA       I      02-OCT-17 01.07.25.375000 PM
```

****

AFTER UPDATE Trigger – Row Level Trigger

****

```
CREATE OR REPLACE TRIGGER after_update
        AFTER UPDATE
        ON Library
        FOR EACH ROW
BEGIN
        insert into library_audit values(:old.rollno, :old.name, :old.dateofissue,
        :old.nameofbook, :old.status, current_timestamp);
END;
/
Trigger created.
SQL> select * from library;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
       101 Ram        23-SEP-17 DBMS       I
       102 Sai        10-SEP-17 CN         I
       103 Laxman     28-SEP-17 TOC        I
       104 Sai        25-AUG-17 SEPM       I
       105 Ganesh     01-SEP-17 IEEE       I
```

```
      106 Gajanan    01-OCT-17 DDA       I
6 rows selected.
SQL> select * from library_audit;
   ROLLNO   NAME      DATEOFISS       NAMEOFBOOK       STATUS                TS
---------- ---------- --------- ---------- ---------- --------- --------
      106         Gajanan    01-OCT-17          DDA       I     02-OCT-17 01.07.25.375000 PM

SQL> update library set nameofbook ='MongoDB' where library.rollno=101;
1 row updated.

SQL> select * from library;
   ROLLNO NAME      DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
      101 Ram       23-SEP-17 MongoDB    I
      102 Sai       10-SEP-17 CN         I
      103 Laxman    28-SEP-17 TOC        I
      104 Sai       25-AUG-17 SEPM       I
      105 Ganesh    01-SEP-17 IEEE       I
      106 Gajanan   01-OCT-17 DDA        I
6 rows selected.

SQL> select * from library_audit;
     ROLLNO  NAME      DATEOFISS       NAMEOFBOOK       STATUS                TS
---------- ---------- --------- ---------- ---------- --------- ------
-106    Gajanan    01-OCT-17          DDA       I       02-OCT-17 01.07.25.375000 PM
101     Ram        23-SEP-17          DBMS    I       02-OCT-17 01.58.22.372000 PM
```

****

AFTER DELETE Trigger – Row Level Trigger

****

```
CREATE TRIGGER after_delete
        AFTER DELETE
        ON Library
        FOR EACH ROW
BEGIN
        insert into library_audit values(:old.rollno, :old.name, :old.dateofissue,
        :old.nameofbook, :old.status, current_timestamp);
END;
/
Trigger created.
SQL> select * from library;
   ROLLNO NAME      DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
      101 Ram       23-SEP-17 MongoDB    I
      102 Sai       10-SEP-17 CN         I
      103 Laxman    28-SEP-17 TOC        I
      104 Sai       25-AUG-17 SEPM       I
      105 Ganesh    01-SEP-17 IEEE       I
```

```
      106 Gajanan    01-OCT-17 DDA        I
6 rows selected.

SQL> select * from library_audit;
ROLLNO       NAME       DATEOFISS       NAMEOFBOOK                STATUS        TS
---------- ---------- --------- ---------- ---------- ---------- --------- ------
-
 106     Gajanan    01-OCT-17          DDA       I             02-OCT-17 01.07.25.375000 PM
 101     Ram        23-SEP-17          MongoDB       I         02-OCT-17 01.58.22.372000 PM

SQL> delete from library where rollno=102;
1 row deleted.
SQL> select * from library;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
      101 Ram        23-SEP-17 MongoDB    I
      103 Laxman     28-SEP-17 TOC        I
      104 Sai        25-AUG-17 SEPM       I
      105 Ganesh     01-SEP-17 IEEE       I
      106 Gajanan    01-OCT-17 DDA        I

SQL> select * from library_audit;
  ROLLNO    NAME       DATEOFISS       NAMEOFBOOK                STATUS        TS
---------- ---------- --------- ---------- ---------- ---------- --------- ------
-
 102     Sai          10-SEP-17          CN              I         02-OCT-17 02.15.24.618000 PM
 106     Gajanan    01-OCT-17          DDA             I         02-OCT-17 01.07.25.375000 PM
 101     Ram        23-SEP-17          MongoDB       I           02-OCT-17 01.58.22.372000 PM
 SQL>
```

****

AFTER Trigger – Row Level Trigger (INSERT/UPDATE/DELETE)

****

Create table lib(rollno int, name char(10), dateofissue date, nameofbook char(10), status char(10));

Create table lib_audit(rollno int, name char(10), dateofissue date, nameofbook char(10), status char(10), ts timestamp,command varchar2(10));

```
Insert into lib values (101, 'Ram',to_date('20170923','YYYYMMDD'),'DBMS', 'I');
Insert into lib values (102, 'Sai',to_date('20170910','YYYYMMDD'),'CN', 'I');
Insert into lib values (103, 'Laxman',to_date('20170928','YYYYMMDD'),'TOC', 'I');
Insert into lib values (104, 'Sai',to_date('20170825','YYYYMMDD'),'SEPM', 'I');
Insert into lib values (105, 'Ganesh',to_date('20170901','YYYYMMDD'),'IEEE', 'I');
Select * from lib;
Select * from lib_audit;
```

==========

CREATE OR REPLACE TRIGGER AT1

```
        AFTER INSERT OR DELETE OR UPDATE
```

```
        ON lib
        FOR EACH ROW
BEGIN
IF UPDATING THEN
        insert into lib_audit values(:old.rollno, :old.name, :old.dateofissue,
        :old.nameofbook, :old.status, current_timestamp, ' UPDATE');
ELSIF INSERTING THEN
        insert into lib_audit values(:new.rollno, :new.name, :new.dateofissue,
        :new.nameofbook, :new.status, current_timestamp,'INSERT');
ELSIF DELETING THEN
        insert into lib_audit values(:old.rollno, :old.name, :old.dateofissue,
        :old.nameofbook, :old.status, current_timestamp, 'DELETE');
END IF;
END;
/
Trigger created.
```

==========

******* OUTPUT *******

*******Insert Operation*******

```
SQL> Insert into lib values(106,'Gajanan',to_date('20171001','YYYYMMDD'),'DDA','I');
1 row created.
SQL> select * from lib;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
       101 Ram        23-SEP-17 DBMS       I
       102 Sai        10-SEP-17 CN         I
       103 Laxman     28-SEP-17 TOC        I
       104 Sai        25-AUG-17 SEPM       I
       105 Ganesh     01-SEP-17 IEEE       I
       106 Gajanan    01-OCT-17 DDA        I
6 rows selected.

SQL> select * from lib_audit;
   ROLLNO   NAME      DATEOFISS    NAMEOFBOOK    STATUS    TS    COMMAND
---------- ---------- --------- ---------- ---------------------------------------------------------------------------
  106 Gajanan    01-OCT-17 DDA        I   02-OCT-17 11.12.03.791000 PM    INSERT
SQL>
```

==========

*******Update Operation*******

```
SQL> update lib set nameofbook ='MongoDB' where lib.rollno=101;
1 row updated.
SQL> select * from lib;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
       101 Ram        23-SEP-17 MongoDB    I
       102 Sai        10-SEP-17 CN         I
       103 Laxman     28-SEP-17 TOC        I
```

```
      104 Sai        25-AUG-17 SEPM       I
      105 Ganesh     01-SEP-17 IEEE       I
      106 Gajanan    01-OCT-17 DDA        I
6 rows selected.
SQL> select * from lib_audit;
   ROLLNO   NAME      DATEOFISS    NAMEOFBOOK    STATUS    TS    COMMAND
---------- ---------- --------- ---------- ----------------------------------------------------------------------
  106 Gajanan    01-OCT-17 DDA        I       02-OCT-17 11.12.03.791000 PM      INSERT
 101 Ram        23-SEP-17 DBMS    I 02-OCT-17 11.14.21.436000 PM      UPDATE
```

==========

*******Delete Operation*******

```
SQL> delete from lib where rollno=102;
1 row deleted.
SQL>
SQL> select * from lib_audit;
   ROLLNO   NAME      DATEOFISS    NAMEOFBOOK    STATUS    TS    COMMAND
---------- ---------- --------- ---------- ----------------------------------------------------------------------
106 Gajanan    01-OCT-17 DDA        I       02-OCT-17 11.12.03.791000 PM      INSERT
101 Ram        23-SEP-17 MongoDB    I       02-OCT-17 11.14.21.436000 PM      UPDATE
102 Sai        10-SEP-17 CN         I            02-OCT-17 11.16.03.851000 PM      DELETE

SQL> select * from lib;
   ROLLNO NAME       DATEOFISS NAMEOFBOOK STATUS
---------- ---------- --------- ---------- ----------
      101 Ram        23-SEP-17 MongoDB    I
      103 Laxman     28-SEP-17 TOC        I
      104 Sai        25-AUG-17 SEPM       I
      105 Ganesh     01-SEP-17 IEEE       I
      106 Gajanan    01-OCT-17 DDA        I
SQL>
```

==========

Group B: Assignment -1

Aim: Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)

1. **Select all documents where the Designation field has the value "Programmer" and the value of the salary field is greater than 30000.**
   db.emp.find({"Designation":"Programmer","Salary":{$gt:30000}}).pretty()

2. **Creates a new document if no document in the employee collection contains {Designation: "Tester", Company_name: "TCS", Age: 25}**
   db.emp.update({Designation: "Tester", Company_name : "TCS" },{ $set : { Age: 25 } },{ upsert : true })

3. **Selects all documents in the collection where the field age has a value less than 30 or the value of the salary field is greater than 40000.**
   db.emp.find({$or:[{Age:{$lt:30}}, {Salary:{$gt:40000}}] }).pretty()

4. **Matches all documents where the value of the field Address is an embedded document that contains only the field city with the value "Pune" and the field Pin_code with the value "411001".**
   db.emp.find({"Address.PAddress":"Pune","Address.PinCode":"411001"}).pretty()

5. **Finds all documents with Company_name: "TCS" and modifies their salary field by 2000.**
   db.emp.update({CName:"TCS"}, {$inc:{Salary:2000}}, {multi:true})

6. **Find documents where Designation is not equal to "Developer".**
   db.emp.find({Designation:{$ne:"Developer"}}).pretty()

7. **Find _id, Designation, Address and Name from all documents where Company_name is "Infosys".**
   db.emp.find({CName:"Amazon"}, {_id:1,Designation:1,Address:1,Name:1}).pretty()

8. **Selects all documents in the employee collection where the value of the Designation is either "Developer" or "Tester".**
   db.emp.find({$or:[{Designation:"Developer"},{Designation:"Tester"}]}).pretty()
   OR
   db.emp.find({Designation: { $in: ['Developer', ' Tester '] } })

9. **Find all document with Exact Match on an Array having Expertise: ['Mongodb','Mysql', 'Cassandra']**
   db.emp.find({Expertise:["Mongodb","Mysql","Cassandra"]}).pretty()

10. **Drop Single documents where designation="Developer".**
    db.emp.remove({Designation:"Developer"},1)

Group B: Assignment -2

Aim: Implement aggregation operation on employee collection using MongoDB.

1. **Return Designation with Total Salary is Above 200000**

   db.s.aggregate({ $group :{ _id : "$Designation",totalSal : { $sum : "$Salary"}}},{$match : {totalSal : { $lte : 200000 } } })

2. **Find Employee with Total Salary for Each City with Designation="DBA"**

   db.s.aggregate([{$match:{Designation:"DBA"}},{$group:{_id:"$Address",totalSal:{$sum: "$Salary"}}}])

3. **Find Total Salary of Employee with Designation="DBA" for Each Company**

   db.s.aggregate([{$match:{Designation:"DBA"}},{$group:{_id:"$Company_name",totalSal :{$sum:"$Salary"}}}])

4. **Returns names and _id in upper case and in alphabetical order.**

   db.s.aggregate([{$project :{Name:{$toUpper:"$Name"}, _id:1 }},{ $sort : {Name :1}}])

5. **Count all records from collection**

   db.s.aggregate([{$group: {_id: null,count: { $sum: 1 }}}])

6. **For each unique Designation, find avg Salary and output is sorted by AvgSal**

   db.s.aggregate([{$group: {_id: "$Designation",AvgSal: { $avg: "$Salary" }}}, { $sort: { AvgSal: 1 } }])

7. **Return separates value in the Expertise array where Name of Employee="Swapnil"**

   db.s.aggregate ([{$unwind:"$Expertise"},{$match:{Name:"Swapnil"}}]).pretty()

8. **Return separates value in the Expertise array and return sum of each element of array**

   db.s.aggregate([{$unwind:"$Expertise"},{$group:{_id:"$Expertise",number:{$sum:1}}}])

9. **Return Array for Designation whose address is "Pune"**

   db.s.aggregate([{$match:{Address:"Pune"}},{$group:{_id:"$Address", Array_Designation:{$push:"$Designation"}}}])

10. **Return Max and Min Salary for each company.**

    db.s.aggregate([{$group:{_id:"$Company_name",min:{$min:"$Salary"}, max:{$max:"$Salary"}}}])

Group B: Assignment -3

Aim: Create Employee Collection using MongoDB and perform different Indexing operation.

1. **To Create Single Field Indexes on Designation**
   db.emp.find({Designation:"DBA"}).explain("executionStats")
   db.emp.ensureIndex({ "Designation": 1 })

2. **To Create Compound Indexes on Name: 1, Age: -1**
   db.emp.find().sort({ Name: 1, Age: -1 }).explain("executionStats")
   db.emp.ensureIndex({ Name : 1, Age : -1 })

3. **To Create Multikey Indexes on Expertise array**
   db.emp.find({"Expertise.2":"Java"}).explain("executionStats")
   db.emp.ensureIndex({"Expertise.Java":1})

4. **Return a List of All Indexes on Collection**
   db.emp.getIndexes()

5. **Rebuild Indexes**
   db.emp.reIndex()

6. **Drop Index on Remove Specific Index**
   db.emp.dropIndex({ "Designation": 1 })

7. **Remove All Indexes except for the _id index from a collection**
   db.emp.dropIndexes()

8. **Insert 200000 in user collection and create index on username. Observe the differences between in output before and after Index.**

Solution:

Inserting 200000 documents in user collection with help of for loop

```
for (i=0; i<200000; i++){
        db.user.insert({
                "i" : i,
                "username" : "user"+i,
                "age" : Math.floor(Math.random()*120),
                "created" : new Date() } ); }
```

****

Counting number of documents in user collection

```
> db.user.count()
200000
```

****
****

To see effective result of indexing we should execute particular find command with explain method before index and after creating index and observe executionStats of each find method.

Before Index: Find all documents in collection

> db.user.find({}).explain("executionStats")


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [ ]
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 200000,
                "executionTimeMillis" : 68,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 200000,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "nReturned" : 200000,
                        "executionTimeMillisEstimate" : 40,
                        "works" : 200002,
                        "advanced" : 200000,
                        "needTime" : 1,
                        "needFetch" : 0,
                        "saveState" : 1562,
                        "restoreState" : 1562,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 200000
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
```

```
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}
```


****

Before Index: Find user0 in collection

> db.user.find({username:"user0"}).explain("executionStats")


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "username" : {
                                "$eq" : "user0"
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "username" : {
                                        "$eq" : "user0"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 1,
                "executionTimeMillis" : 127,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 200000,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "username" : {
                                        "$eq" : "user0"
                                }
                        },
                        "nReturned" : 1,
                        "executionTimeMillisEstimate" : 110,
                        "works" : 200002,
                        "advanced" : 1,
                        "needTime" : 200000,
                        "needFetch" : 0,
                        "saveState" : 1562,
                        "restoreState" : 1562,
                        "isEOF" : 1,
```

```
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 200000
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}
```

****

Before Index: Find user19999 in collection

> db.user.find({username:"user19999"}).explain("executionStats")

```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "username" : {
                                "$eq" : "user19999"
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "username" : {
                                        "$eq" : "user19999"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 1,
                "executionTimeMillis" : 59,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 200000,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "username" : {
                                        "$eq" : "user19999"
                                }
                        },
                        "nReturned" : 1,
```

```
                        "executionTimeMillisEstimate" : 60,
                        "works" : 200002,
                        "advanced" : 1,
                        "needTime" : 200000,
                        "needFetch" : 0,
                        "saveState" : 1562,
                        "restoreState" : 1562,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 200000
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}
```


****

Before Index: Find user9999 in collection

> db.user.find({username:"user9999"}).explain("executionStats")


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "username" : {
                                "$eq" : "user9999"
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "username" : {
                                        "$eq" : "user9999"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 1,
                "executionTimeMillis" : 53,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 200000,
```

```
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "username" : {
                                        "$eq" : "user9999"
                                }
                        },
                        "nReturned" : 1,
                        "executionTimeMillisEstimate" : 30,
                        "works" : 200002,
                        "advanced" : 1,
                        "needTime" : 200000,
                        "needFetch" : 0,
                        "saveState" : 1562,
                        "restoreState" : 1562,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 200000
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}

**********************************************************************************
**********************************************************************************
```

To create Single Field Index on Username

```
> db.user.createIndex({username:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.user.getIndexes()
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "mescoe.user"
        },
        {
```

```
                "v" : 1,
                "key" : {
                        "username" : 1
                },
                "name" : "username_1",
                "ns" : "mescoe.user"
        }]
********************************************************************************
********************************************************************************
```

After Index: Find all documents in collection

db.user.find({}).explain("executionStats")

```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [ ]
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 200000,
                "executionTimeMillis" : 45,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 200000,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "nReturned" : 200000,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 200002,
                        "advanced" : 200000,
                        "needTime" : 1,
                        "needFetch" : 0,
                        "saveState" : 1562,
                        "restoreState" : 1562,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 200000
```

```
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}
```


****

After Index: Find user0 in collection

> db.user.find({username:"user0"}).explain("executionStats")


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "username" : {
                                "$eq" : "user0"
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "username" : 1
                                },
                                "indexName" : "username_1",
                                "isMultiKey" : false,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "username" : [
                                                "[\"user0\", \"user0\"]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 1,
                "executionTimeMillis" : 0,
                "totalKeysExamined" : 1,
                "totalDocsExamined" : 1,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 1,
```

```
                        "executionTimeMillisEstimate" : 0,
                        "works" : 2,
                        "advanced" : 1,
                        "needTime" : 0,
                        "needFetch" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "docsExamined" : 1,
                        "alreadyHasObj" : 0,
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "nReturned" : 1,
                                "executionTimeMillisEstimate" : 0,
                                "works" : 2,
                                "advanced" : 1,
                                "needTime" : 0,
                                "needFetch" : 0,
                                "saveState" : 0,
                                "restoreState" : 0,
                                "isEOF" : 1,
                                "invalidates" : 0,
                                "keyPattern" : {
                                        "username" : 1
                                },
                                "indexName" : "username_1",
                                "isMultiKey" : false,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "username" : [
                                                "[\"user0\", \"user0\"]"
                                        ]
                                },
                                "keysExamined" : 1,
                                "dupsTested" : 0,
                                "dupsDropped" : 0,
                                "seenInvalidated" : 0,
                                "matchTested" : 0
                        }
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}
```


****

After Index: Find user19999 in collection

> db.user.find({username:"user19999"}).explain("executionStats")


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "username" : {
                                "$eq" : "user19999"
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "username" : 1
                                },
                                "indexName" : "username_1",
                                "isMultiKey" : false,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "username" : [
                                                "[\"user19999\", \"user19999\"]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 1,
                "executionTimeMillis" : 0,
                "totalKeysExamined" : 1,
                "totalDocsExamined" : 1,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 1,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 2,
                        "advanced" : 1,
                        "needTime" : 0,
                        "needFetch" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "docsExamined" : 1,
                        "alreadyHasObj" : 0,
```

```
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "nReturned" : 1,
                                "executionTimeMillisEstimate" : 0,
                                "works" : 2,
                                "advanced" : 1,
                                "needTime" : 0,
                                "needFetch" : 0,
                                "saveState" : 0,
                                "restoreState" : 0,
                                "isEOF" : 1,
                                "invalidates" : 0,
                                "keyPattern" : {
                                        "username" : 1
                                },
                                "indexName" : "username_1",
                                "isMultiKey" : false,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "username" : [
                                                "[\"user19999\", \"user19999\"]"
                                        ]
                                },
                                "keysExamined" : 1,
                                "dupsTested" : 0,
                                "dupsDropped" : 0,
                                "seenInvalidated" : 0,
                                "matchTested" : 0
                        }
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}
```


****

After Index: Find user9999 in collection

> db.user.find({username:"user9999"}).explain("executionStats")


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mescoe.user",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "username" : {
                                "$eq" : "user9999"
                        }
```

```
        },
        "winningPlan" : {
                "stage" : "FETCH",
                "inputStage" : {
                        "stage" : "IXSCAN",
                        "keyPattern" : {
                                "username" : 1
                        },
                        "indexName" : "username_1",
                        "isMultiKey" : false,
                        "direction" : "forward",
                        "indexBounds" : {
                                "username" : [
                                        "[\"user9999\", \"user9999\"]"
                                ]
                        }
                }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 1,
        "totalDocsExamined" : 1,
        "executionStages" : {
                "stage" : "FETCH",
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 0,
                "works" : 2,
                "advanced" : 1,
                "needTime" : 0,
                "needFetch" : 0,
                "saveState" : 0,
                "restoreState" : 0,
                "isEOF" : 1,
                "invalidates" : 0,
                "docsExamined" : 1,
                "alreadyHasObj" : 0,
                "inputStage" : {
                        "stage" : "IXSCAN",
                        "nReturned" : 1,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 2,
                        "advanced" : 1,
                        "needTime" : 0,
                        "needFetch" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
```

```
                                "invalidates" : 0,
                                "keyPattern" : {
                                        "username" : 1
                                },
                                "indexName" : "username_1",
                                "isMultiKey" : false,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "username" : [
                                                "[\"user9999\", \"user9999\"]"
                                        ]
                                },
                                "keysExamined" : 1,
                                "dupsTested" : 0,
                                "dupsDropped" : 0,
                                "seenInvalidated" : 0,
                                "matchTested" : 0
                        }
                }
        },
        "serverInfo" : {
                "host" : "admin",
                "port" : 27017,
                "version" : "3.0.10",
                "gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"
        },
        "ok" : 1
}
```


****

Group C: Assignment -1

Aim: Write a program to implement MongoDB database connectivity with PHP/ python/Java. Implement Database navigation operations (add, delete, edit etc.) using ODBC/JDBC.

```
import java.net.UnknownHostException;
import java.io.*;
import java.util.Date;
import com.mongodb.*;
public class App
{
public static void main(String[] args)
        {
        BufferedReader br=null;
        int ch,eid,sal;
        String sql,name,desig;
        try
        {
                br=new BufferedReader(new InputStreamReader(System.in));
                /**** Connect to MongoDB ****/
                MongoClient mongo = new MongoClient("localhost", 27017);

                /**** Get database ****/
                // if database doesn't exists, MongoDB will create it for you
                DB db = mongo.getDB("dmsa");

                /**** Get collection / table from 'testdb' ****/
                // if collection doesn't exists, MongoDB will create it for you
                DBCollection table = db.getCollection("emp");

                do
                {
                        System.out.println("\n\nChoices for User");
                        System.out.println("1.Insert document");
                        System.out.println("2.View document");
                        System.out.println("3.Update document");
                        System.out.println("4.Delete document");
                        System.out.println("5.Exit");
                        System.out.println("Enter the choice=");
                        ch=Integer.parseInt(br.readLine());
                        switch(ch)
                        {
                                case 1:
```

```
                System.out.println("\nINSERT RECORD:");
                System.out.println("Enter the emp_id=");
                eid=Integer.parseInt(br.readLine());
                System.out.println("Enter the emp_name=");
                name=br.readLine();
                System.out.println("Enter the emp_salary=");
                sal=Integer.parseInt(br.readLine());
                System.out.println("Enter the emp_designation=");
                desig=br.readLine();

                //To insert Data into DB
                BasicDBObject document = new BasicDBObject();
                document.put("empid", eid);
                document.put("ename", name);
                document.put("salary", sal);
                document.put("designation", desig);
                table.insert(document);
        System.out.println("\nDocumet inserted successfully....");
        break;
        case 2:
                BasicDBObject searchQuery = new BasicDBObject();
                //searchQuery.put();
                DBCursor cursor = table.find();
                while (cursor.hasNext())
                {
                        System.out.println(cursor.next());
                }
        break;
        case 3:
                BasicDBObject query = new BasicDBObject();
                BasicDBObject newDocument = new BasicDBObject();
                BasicDBObject updateObj = new BasicDBObject();

                System.out.println("\nUpdate Record Options:");
                System.out.println("1.Update salary.");
                System.out.println("2.Update designation.");
                System.out.println("Enter the choice=");
                int ch2=Integer.parseInt(br.readLine());
                switch(ch2)
                {
                case 1:
                System.out.println("Enter the emp_id whoes record
want you to update=");
                eid=Integer.parseInt(br.readLine());
                System.out.println("Enter the new salary=");
                sal=Integer.parseInt(br.readLine());
                query.put("empid", eid);
                newDocument.put("salary", sal);
```

```
                    updateObj.put("$set", newDocument);
                    table.update(query, updateObj);
                System.out.println("\nDocument Updated Successfully...");
                break;
                case 2:

        System.out.println("Enter the emp_id whoes record want you to update=");
                    eid=Integer.parseInt(br.readLine());
                    System.out.println("Enter the new designation=");
                    desig=br.readLine();
                    query.put("empid", eid);
                    newDocument.put("designation", desig);
                    updateObj.put("$set", newDocument);
                    table.update(query, updateObj);
        System.out.println("\nDocument Updated Successfully...");
            break;
            default:
        System.out.println("\nInvalid Choice");
                    }
            break;
            case 4:
                    System.out.println("\nDelete Record Options:");
                    System.out.println("1.Delete Particular data");
                    System.out.println("Enter the choice=");
                    int ch1=Integer.parseInt(br.readLine());
                    switch(ch1)
                    {
                    case 1:
                    System.out.println("Enter the emp_id whoes record want you to
delete=");
                    eid=Integer.parseInt(br.readLine());
                    System.out.println("\nRecord Deleted Successfully...");
                    BasicDBObject a = new BasicDBObject();
                    a.put("empid", eid);
                    table.remove(a);
                    break;
                    default:
                    System.out.println("\nIsnvalid Choice");
                    }
                    break;
                    case 5:
                    break;
                    default:
                    System.out.println("\nInvalid Choice");
                    }
                    }while(ch!=5);
        /**** Done ****/
        System.out.println("Thank You...");
```

```
                System.out.println("Programmed by:SHRINIWAS DESHMUKH..");
                }
                catch (UnknownHostException e)
                {
                        e.printStackTrace();
                }
                catch (MongoException e)
                {
                        e.printStackTrace();
                }
                catch(IOException e)
                {
                        e.printStackTrace();
                }
        }
}
```

/*** Output:**

```
Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
1

INSERT RECORD:
Enter the emp_id=
4
Enter the emp_name=
Kaustubh
Enter the emp_salary=
450000
Enter the emp_designation=
Executive

Documet inserted successfully....


Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
```

2
{ "_id" : { "$oid" : "5423161f9905d80f8dbd5290"} , "empid" : 1 , "ename" : "Eshaa" , "salary" : 550000 , "designation" : "abc"}
{ "_id" : { "$oid" : "54231643990570dbcac5dfdf"} , "empid" : 1 , "ename" : "Eshaa" , "salary" : 50000 , "designation" : "Designer"}
{ "_id" : { "$oid" : "5423165c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" , "salary" : 600000 , "designation" : "Developer"}
{ "_id" : { "$oid" : "54231678990570dbcac5dfe1"} , "empid" : 3 , "ename" : "Deendayal" , "salary" : 550000 , "designation" : "Manager"}
{ "_id" : { "$oid" : "542316db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" , "salary" : 450000 , "designation" : "Executive"}

Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
3

Update Record Options:
1.Update salary.
2.Update designation.
Enter the choice=
1
Enter the emp_id whoes record want you to update=
4
Enter the new salary=
400000

Document Updated Successfully...

Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
2
{ "_id" : { "$oid" : "5423161f9905d80f8dbd5290"} , "empid" : 1 , "ename" : "Eshaa" , "salary" : 550000 , "designation" : "abc"}
{ "_id" : { "$oid" : "54231643990570dbcac5dfdf"} , "empid" : 1 , "ename" : "Eshaa" , "salary" : 50000 , "designation" : "Designer"}
{ "_id" : { "$oid" : "5423165c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" , "salary" : 600000 , "designation" : "Developer"}

{ "_id" : { "$oid" : "54231678990570dbcac5dfe1"} , "empid" : 3 , "ename" : "Deendayal" , "salary" : 550000 , "designation" : "Manager"}
{ "_id" : { "$oid" : "542316db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" , "salary" : 400000 , "designation" : "Executive"}

Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
3

Update Record Options:
1.Update salary.
2.Update designation.
Enter the choice=
2
Enter the emp_id whoes record want you to update=
4
Enter the new designation=
HR

Document Updated Successfully...

Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
2
{ "_id" : { "$oid" : "5423161f9905d80f8dbd5290"} , "empid" : 1 , "ename" : "Eshaa" , "salary" : 550000 , "designation" : "abc"}
{ "_id" : { "$oid" : "54231643990570dbcac5dfdf"} , "empid" : 1 , "ename" : "Eshaa" , "salary" : 50000 , "designation" : "Designer"}
{ "_id" : { "$oid" : "5423165c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" , "salary" : 600000 , "designation" : "Developer"}
{ "_id" : { "$oid" : "54231678990570dbcac5dfe1"} , "empid" : 3 , "ename" : "Deendayal" , "salary" : 550000 , "designation" : "Manager"}
{ "_id" : { "$oid" : "542316db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" , "salary" : 400000 , "designation" : "HR"}

```
Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
4

Delete Record Options:
1.Delete Particular data
Enter the choice=
1
Enter the emp_id whoes record want you to delete=
3

Record Deleted Successfully...


Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
2
{ "_id" : { "$oid" : "5423161f9905d80f8dbd5290"} , "empid" : 1 , "ename" : "Eshaa" ,
"salary" : 550000 , "designation" : "abc"}
{ "_id" : { "$oid" : "54231643990570dbcac5dfdf"} , "empid" : 1 , "ename" : "Eshaa" ,
"salary" : 50000 , "designation" : "Designer"}
{ "_id" : { "$oid" : "5423165c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" ,
"salary" : 600000 , "designation" : "Developer"}
{ "_id" : { "$oid" : "542316db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" ,
"salary" : 400000 , "designation" : "HR"}


Choices for User
1.Insert document
2.View document
3.Update document
4.Delete document
5.Exit
Enter the choice=
5
*/
```

Group C: Assignment -2

Aim: Implement MYSQL/Oracle database connectivity with PHP/ python/Java. Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.

```
package mypack;
import java.sql.*;
import java.util.*;
public class connect{
	public static void connection()
	{
		String empname,designation;
		int empno,age,salary;
		try
		{
			Scanner a = new Scanner(System.in);
			Scanner b= new Scanner(System.in);
			int i,rs,e;
			String DRIVER_CLASS = "com.mysql.jdbc.Driver";
			Class.forName(DRIVER_CLASS);
			String UID="root";
			String PWD="admin123";
			String DB_URL="jdbc:mysql://localhost/student1";
			Connection
conn=DriverManager.getConnection(DB_URL,UID,PWD);
			Statement stmt=conn.createStatement();
			do
			{
			String menu="~~~~OPERATIONS~~~~~\n
			1.INSERT NEW ENTRY IN THE DATABASE\n
			2.UPDATE SOME VALUE\n
			3.DISPLAY\n
			4.DELETE\n
			5.EXIT\n
			ENTER YOUR OPTION  : ";
			System.out.println(menu);
			String query;
			String sql="update table employee set age=1;";
			i=a.nextInt();
			switch(i)
			{
			case 1:System.out.println("Enter the following information to be
inserted(Blank fields to be avoided)");
			System.out.println("1.Employee number : ");
			empno=a.nextInt();
```

```
                    System.out.println("2.Employee name : ");
                    empname=b.nextLine();
                    System.out.println("3.Age : ");
                    age=a.nextInt();
                    System.out.println("4.Designation : ");
                    designation=b.nextLine();
                    System.out.println("5.Salary : ");
                    salary=a.nextInt();
                    query="insert into employee values("+empno+","'+empname+"',"+age+",'"+designation+"',"+salary+");";
                    rs=stmt.executeUpdate(query);
                    if(rs==1)
                    {
                            System.out.println("\nData inserted succesfully!!\n");
                    }
                    break;
                    case 2:System.out.println("Select the field you want to update : \n1.Age\n2.Designation\n3.Salary\n");
                    int option=a.nextInt();
                    System.out.println("Enter the employee id for which you want to update data : ");
                    e=b.nextInt();
                    switch(option)
                    {
                    case 1 : System.out.println("\nEnter the new age : ");
                    age=a.nextInt();
                    query="update employee set age = "+age+" where emp_no = "+e+";";
                    rs=stmt.executeUpdate(query);
                    if(rs==1)
                    {
                            System.out.println("\nData has been updated successfully!");
                    }
                    break;
                    case 2: System.out.println("\nEnter the new designation : \n");
                    designation=b.nextLine();
                    query="update employee set designation = '"+designation+"' where emp_no ="+e+" ;";
                    rs=stmt.executeUpdate(query);
                    if(rs==1)
                    {
                            System.out.println("\n Updated successfully!");
                    }
                            break;
                    case 3: System.out.println("\nEnter the new salary : ");
                    salary=a.nextInt();
            query="update employee set salary = "+salary+" where emp_no ="+e+";";
            rs=stmt.executeUpdate(query);
            if(rs==1)
```

```
{
        System.out.println("\n Updated successfully!");
}
break;
default :System.out.println("\nPlease enter a valid choice\n");
break;
}
break;
case 3:query="select * from employee;";
ResultSet rs1=stmt.executeQuery(query);
System.out.println("Emp_no\tEmp_name\tAge\tDesgntn\tSalary");
                while(rs1.next())
                {

                empno=rs1.getInt("emp_no");
                empname=rs1.getString("emp_name");
                age=rs1.getInt("age");
                designation=rs1.getString("designation");
                salary=rs1.getInt("salary");

System.out.println(empno+"\t"+empname+"\t"+age+"\t"+designation+"\t"+salary);
                }
break;
case 4 :System.out.println("\n1.DELETE ALL RECORDS\n2.DELETE SELECTED DATA");
        option=a.nextInt();
        switch(option)
        {
                case 1:query="truncate table employee;";
                rs=stmt.executeUpdate(query);
                String query2="select * from employee;";
                rs1=stmt.executeQuery(query2);
                if(rs1==null)
System.out.println("\nAll records have been successfully deleted");
break;
case 2:System.out.println("Enter the employee id whose record you want to delete : ");
        e=a.nextInt();
        query="delete from employee where emp_no = "+e+";";
        rs=stmt.executeUpdate(query);
        if(rs==1)
        {
                System.out.println("\nThe specified record has been deleted!");
        }
                String query1="select * from employee;";
                rs1=stmt.executeQuery(query1);
                System.out.println("Emp_no\tEmp_name\tAge\tDesgntn\tSalary");
                while(rs1.next())
```

```
                        {

                        empno=rs1.getInt("emp_no");
                        empname=rs1.getString("emp_name");
                        age=rs1.getInt("age");
                        designation=rs1.getString("designation");
                        salary=rs1.getInt("salary");

        System.out.println(empno+"\t"+empname+"\t"+age+"\t"+designation+"\t"+salary);
                }
                break;
                }
        case 5:System.exit(0);
        }
        }while(i<=5);
        stmt.close();
        conn.close();

        }
        catch(Exception e)
        {
                e.printStackTrace();
        }
        }
        public static void main(String[] args) {
        // TODO Auto-generated method stub
        connection();
        }
}
```

/*** OUTPUT:**

```
~~~~OPERATIONS~~~~~
1.INSERT NEW ENTRY IN THE DATABASE
2.UPDATE SOME VALUE
3.DISPLAY
4.DELETE
5.EXIT
ENTER YOUR OPTION:
1
Enter the following information to be inserted(Blank fields to be avoided)
1.Employee number :
4
2.Employee name :
kiran
3.Age :
20
4.Designation :
executive
```

```
5.Salary :
500000

Data inserted succesfully!!

~~~~OPERATIONS~~~~~
1.INSERT NEW ENTRY IN THE DATABASE
2.UPDATE SOME VALUE
3.DISPLAY
4.DELETE
5.EXIT
ENTER YOUR OPTION:
3
Emp_no        Emp_name   Age       Desgntn            Salary
1         eshaa     19     CEO       60000000
2         varsha    19     manager           500000
3         kalpita   20     sales     200000
4         kiran     20     executive  500000
~~~~OPERATIONS~~~~~
1.INSERT NEW ENTRY IN THE DATABASE
2.UPDATE SOME VALUE
3.DISPLAY
4.DELETE
5.EXIT
ENTER YOUR OPTION:
2
Select the field you want to update:
1.Age
2.Designation
3.Salary

3
Enter the employee id for which you want to update data:
3

Enter the new salary:
300000

 Updated successfully!
~~~~OPERATIONS~~~~~
1.INSERT NEW ENTRY IN THE DATABASE
2.UPDATE SOME VALUE
3.DISPLAY
4.DELETE
5.EXIT
ENTER YOUR OPTION:
3
Emp_no        Emp_name   Age       Desgntn            Salary
```

```
1       eshaa     19    CEO       60000000
2       varsha    19    manager    500000
3       kalpita   20    sales      300000
4       kiran     20    executive  500000
~~~~OPERATIONS~~~~~
1.INSERT NEW ENTRY IN THE DATABASE
2.UPDATE SOME VALUE
3.DISPLAY
4.DELETE
5.EXIT
ENTER YOUR OPTION:
4

1.DELETE ALL RECORDS
2.DELETE SELECTED DATA
2
Enter the employee id whose record you want to delete:
4

The specified record has been deleted!
Emp_no        Emp_name  Age       Desgntn           Salary
1       eshaa     19     CEO      60000000
2       varsha    19     manager          500000
3       kalpita   20     sales     300000
*/
```