1

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;

1. Find **all** loan numbers **for** loans made at Akurdi Branch **with loan amount** > **12000.**

select bname, lno from Loan where bname='Akurdi' and amounts12000;

1. 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, Borrower where Loan.lno= Borrower.lno

1. List all customers in alphabetical order who have loan from Akurdi branch.

select Borrower.cname from Borrower,Loan where Borrower.lno=Loan.lno and Loan.bname='Akurdi' order by Borrower.cname;

1. Find all customers who have both account and loan at bank.

select Depositer.cname, from Depositer, Borrower where Depositer.cname=Borrower.cname;

1. Find the average account balance at each branch

select avg(balance),bname from Account group by bname;

1. Find no. of depositors at each branch.

select bname,count(\*) from Account group by bname;

1. Calculate **total** loan amount given by **bank.**

select bname,sum(amount) from Loan group by bname; OR

select sum(amount) from Loan;

1. Find name of Customer and city where customer name starts with Letter P.

select cust\_name,cust\_city from Customer where cust\_name like 'P%';

1. Display distinct cities of branch.

select branch\_city from Branch group by branch\_name;

## 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.addl,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";

1. 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;

1. 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;

1. 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;
2. 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;

1. Create View on Borrower table by selecting any two columns and perform insert,update and delete operations.

create view viewl as select bname,sum(Amount) from Borrower group by bname;

1. 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 Borrowerl.bno,Borrower1.cname,Depositorl.Balance from Borrower 1 inner join Depositor 1 on Depositorl.dno=Borrowerl.bno;

1. Create updateable View on Borrower table by selecting any two columns and perform insert,update and delete operations.

create view vupborrower 1 as select bno,cname,bname,Amount from Borrower 1;

# 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(Ro1lno, Name, Dateodssue, 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

ROLLNO NAME DATEOFISSUE NAMEOFBOOK STATUS

SQL> desc fine; Name

ROLLNO FDATE AMT

Null? Type

NUMBER(38) CHAR(10) DATE CHAR(10) CHAR(10)

Null? Type

NUMBER(38) DATE 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

1. Ram 23-SEP-17 DBMS I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. 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 := &ro1lno;

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 ro11no=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 := &ro1lno; 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

1. Ram 23-SEP-17 DBMS R
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. Ganesh 01-SEP-17 IEEE I SQL>

Enter value for rollno: 102 old 9: p rollno := &ro1lno; 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

1. Ram 23-SEP-17 DBMS R
2. Sai 10-SEP-17 CN R
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. 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 := &ro1lno; 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

1. Ram 23-SEP-17 DBMS R
2. Sai 10-SEP-17 CN R
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. Ganesh 01-SEP-17 IEEE R

Enter value for rollno: 104 old 9: p rollno := &ro1lno; 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

1. Ram 23-SEP-17 DBMS R
2. Sai 10-SEP-17 CN R
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM R
5. 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 akeady 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=o1dt.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 ro11%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.

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(l,'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 +=l500 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

End if; End if; End if;

insert into result values(p roll no,p name,'fai1');

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 ro1lno:=&var rollno;

Proc grade(var ro1lno,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,

1. p roll no out stud marks.roll no%type,
2. p name out stud marks.name%type,
3. p total out stud marks.total marks%type) 6 AS
4. BEGIN
5. SELECT roll no, name, total marks into p roll no, p name, p total from stud marks where roll no=var rollno;
6. IF p total +=l500 and p total += 990 THEN
7. insert into result values(p roll no,p name,'Distinction');
8. Else if p total +=989 and p total += 900 THEN
9. insert into result values(p roll no,p name,'First Class');
10. Else if p total +=899 and p total += 825 THEN 14 insert into result values(p roll no,p name,'HSC');
11. Else
12. insert into result values(p roll no,p name,'fail');
13. End if;
14. End if;
15. End if;
16. EXCEPTION
17. WHEN no data found then
18. dbms output.put line('Roll no ' || var rollno ||' not found');
19. END;

24 /

Procedure created.

SQL> DECLARE

1. var rollno number(20);
2. p roll no stud marks.roll no%type;
3. p name stud marks.name%type;
4. p total stud marks.total marks%type; 6 BEGIN
5. var ro1lno:=&var rollno;
6. Proc grade(var ro1lno,p roll no,p name,p total); 9 END;

10 /

Enter value for var rollno: 2

old 7: var ro1lno:=&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

1. var rollno number(20);
2. p roll no stud marks.roll no%type;
3. p name stud marks.name%type;
4. p total stud marks.total marks%type; 6 BEGIN
5. var ro1lno:=&var rollno;
6. Proc grade(var ro1lno,p roll no,p name,p total); 9 END;

10 /

Enter value for var rollno: 1

old 7: var ro1lno:=&var rollno; new 7: var rollno:=1;

PL/SQL procedure successfully completed.

SQL+ select \* from result;

ROLL NO NAME CLASS

2 Ram

1 Ganesh

First Class Distinction

SQL> DECLARE

1. var rollno number(20);
2. p roll no stud marks.roll no%type;
3. p name stud marks.name%type;
4. p total stud marks.total marks%type; 6 BEGIN
5. var ro1lno:=&var rollno;
6. Proc grade(var ro1lno,p roll no,p name,p total); 9 END;

10 /

Enter value for var rollno: 3

old 7: var ro1lno:=&var rollno; new 7: var rollno:=3;

PL/SQL procedure successfully completed.

SQL> DECLARE

1. var rollno number(20);
2. p roll no stud marks.roll no%type;
3. p name stud marks.name%type;
4. p total stud marks.total marks%type; 6 BEGIN
5. var ro1lno:=&var rollno;
6. Proc grade(var ro1lno,p roll no,p name,p total); 9 END;

10 /

Enter value for var rollno: 4

old 7: var ro1lno:=&var rollno; new 7: var rollno:=4;

PL/SQL procedure successfully completed. SQL+ select \* from result;

ROLL NO NAME

2 Ram

1 Ganesh

1. Sai
2. Laxman

CLASS

First Class Distinction

HSC

fail

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(ro1lno int, name char(10), dateofissue date, nameofbook char(10), status char(10));

Create table library\_audit(ro1lno int, name char(10), dateofissue date, nameofbook char(10), status char(10), ts timestamp);

Insert into library values (101, 'Ram',to date('20l70923','YYYYMMDD'),'DBMS', 'I'); Insert into library values (102, 'Sai',to date('20170910','YYYYMMDD'),'CN', 'I');

Insert into library values (103, 'Laxman',to date('20l70928','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

* 1. Ram 23-SEP-17 DBMS I
  2. Sai 10-SEP-17 CN I
  3. Laxman 28-SEP-17 TOC I
  4. Sai 25-AUG-17 SEPM I
  5. Ganesh 01-SEP-17 IEEE I

SQL+ Create table library audit(ro1lno int, name char(10), dateofissue date, nameofbook char(10), status char(10), ts timestamp);

Table created.

SQL+ select \* from library audit; no rows selected

AT"I“LR **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

1. Ram 23-SEP-17 DBMS I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. Ganesh 01-SEP-17 IEEE I

SQL+ select \* from library audit; no rows selected

SQL+ Insert into library values (106, 'Gajanan',to date('2017100l','YYYYMMDD'),' DDA', 'I');

1 row created.

SQL+ select \* from library;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

1. Ram 23-SEP-17 DBMS I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. Ganesh 01-SEP-17 IEEE I
6. 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

AT"I“LR UPDA"I“L **Trigger** — Row Level **Trigger**

CREATE OR REPLACE TRIGGER after\_update AFTER UPDATE

ON Library

FOR EACH ROW BEGIN

insert into library audit values(:o1d.rollno, :o1d.name, :o1d.dateofissue,

:o1d.nameofbook, :old.status, current timestamp);

END;

/

Trigger created.

SQL+ select \* from library;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

1. Ram 23-SEP-17 DBMS I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC 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

1. Ram 23-SEP-17 MongoDB I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. Ganesh 01-SEP-17 IEEE I
6. 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 |

### AT"I“LR DELE l“E Trigger — Row Level Trigger CREATE TRIGGER after\_delete

AFTER DELETE

ON Library

FOR EACH ROW BEGIN

insert into library audit values(:old.rollno, :o1d.name, :old.dateofissue,

:o1d.nameofbook, :old.status, current timestamp);

### END;

/

Trigger created.

SQL+ select \* from library;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

1. Ram 23-SEP-17 MongoDB I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC 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 ro1lno=l02; 1 row deleted.

SQL+ select \* from library;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

101 Ram 23-SEP-17 MongoDB I

1. Laxman 28-SEP-17 TOC I
2. Sai 25-AUG-17 SEPM I
3. Ganesh 01-SEP-17 IEEE I
4. 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> |  |  |  |  |  |

### AT"I“LR Trigger — Row Level Trigger (INSERT/UPDA l“E/DELE l“E)

Create table lib(ro1lno int, name char(10), dateofissue date, nameofbook char(10), status char(10));

Create table lib audit(ro1lno 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('20l70923','YYYYMMDD'),'DBMS', 'I'); Insert into lib values (102, 'Sai',to date('20l709l0','YYYYMMDD'),'CN', 'I');

Insert into lib values (103, 'Laxman',to date('20l70928','YYYYMMDD'),'TOC', 'I'); Insert into lib values (104, 'Sai',to date('20170825','YYYYMMDD'),'SEPM', 'I'); Insert into lib values (105, 'Ganesh',to date('20l7090l','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(:o1d.rollno, :o1d.name, :o1d.dateofissue,

:o1d.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(:o1d.rollno, :o1d.name, :o1d.dateofissue,

:o1d.nameofbook, :old.status, current timestamp, 'DELETE');

### END IF;

END;

/

Trigger created.

\*\*\*\*\* **OUTPUT** \*\*\*\*\*

\*\*\*\*\*Insert Operation\*\*\*\*\*

SQL+ Insert into lib values(l06,'Gajanan',to date('20171001','YYYYMMDD'),'DDA','I'); 1 row created.

SQL+ select \* from lib;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

1. Ram 23-SEP-17 DBMS I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. Ganesh 01-SEP-17 IEEE I
6. 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.ro1lno=l0l ; 1 row updated.

SQL+ select \* from lib;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

1. Ram 23-SEP-17 MongoDB I
2. Sai 10-SEP-17 CN I
3. Laxman 28-SEP-17 TOC I
4. Sai 25-AUG-17 SEPM I
5. Ganesh 01-SEP-17 IEEE I
6. 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 ro1lno=l02; 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

1. Laxman 28-SEP-17 TOC I
2. Sai 25-AUG-17 SEPM I
3. Ganesh 01-SEP-17 IEEE I
4. Gajanan 01-OCT-17 DDA I SQL>

12

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()

1. 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 })

1. 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:($1t:30}}, (Salary:{$gt:40000}} ]  ).pretty()

1. 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()

1. Finds all documents with Company\_name: "TCS" and modifies their salary field by 2000.

db.emp.update( (CName:"TCS"}, ($inc:(Salary:2000}}, (multi:true})

1. Find documents where Designation is not equal to "Developer". db.emp.find((Designation:($ne:"Developer"}}).pretty()
2. Find \_id, Designation, Address and Name from all documents where Company\_name is "Infosys".

db.emp.find( (CName:"Amazon"}, { id: l,Designation: l,Address:1,Name:1} ).pretty()

1. 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 ' ] 1 1 

1. Find all document with Exact Match on an Array having Expertise: ['Mongodb','Mysql', 'Cassandra']

db.emp.find( (Expertise:["Mongodb","Mysq1","Cassandra"]}).pretty()

1. Drop Single documents where designation=”Developer”. db.emp.remove((Designation:"Developer"},l)

## 13-1

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 : ( $1te : 200000 1 1 1 

1. Find Employee with Total Salary for Each City with Designation=”DBA” db.s.aggregate([($match:(Designation:"DBA"}},($group:( id:"$Address",totalsal:($sum: "$Salary"}}}])
2. Find Total Salary of Employee with Designation=”DBA” for Each Company db.s.aggregate([($match:(Designation:"DBA"}},{$group:{ id:"$Company name",totalSal

:($sum:"$Salary"}}}])

1. Returns names and \_id in upper case and in alphabetical order. db.s.aggregate([($project :(Name:{$toUpper:"$Name"}, id:1 }},( $sort : (Name :1}}] )
2. Count all records from collection

db.s.aggregate( [($group: ( id: null,count: { $sum: 1 



1. For each unique Designation, find avg Salary **and output is** sorted by AvgSal db.s.aggregate( [($group: ( id: "$Designation",AvgSa1: ( $avg: "$Salary" }}}, ( $sort: { AvgSal: 1  }] )
2. Return separates value in the Expertise array where Name of Employee=”Swapnil” db.s.aggregate ([($unwind:"$Expertise"},{$match:{Name:"Swapnil"}}]).pretty()
3. 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}}})

1. Return Array for Designation whose address is "Pune" db.s.aggregate([($match:(Address:"Pune"}},($group:{ id:"$Address", Array Designation:($push:"$Designation"}}}])
2. Return Max and Min Salary for each company. db.s.aggregate([($group:{ id:"$Company name",min:($min:"$Salary"}, max:($max:"$Salary"}}}])

## 13-2

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({

"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" "1e05 l2f8453d103987f5fbfb87b7le9al3lc2a60"

"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" "1e05 l2f8453d103987f5fbfb87b7le9al3lc2a60" "ok" : 1

Before Index: Find userl9999 in collection

> db.user.find({username:”userl9999”}).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" : "1e05 l2f8453d103987f5fbfb87b7le9al3lc2a60" "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" : "1e05 l2f8453d103987f5fbfb87b7le9al3lc2a60" "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" If "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" "1e05 l2f8453d103987f5fbfb87b7le9al3lc2a60" "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,

"akeadyHasObj" : 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,

"seenlnvalidated" : 0,

"matchTested" : 0

"serverInfo" : {

"host" : "admin", "port" : 27017,

"version" : "3.0.10",

"gitVersion" "1e05l2f8453dl03987f5fbfb87b7l e9al31c2a60" "ok" : 1

**After Index: Find userl9999** in collection

> db.user.find({username:”userl9999”}).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,

"akeadyHasObj" : 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,

"seenlnvalidated" : 0,

"matchTested" : 0

"serverInfo" : {

"host" : "admin", "port" : 27017,

"version" : "3.0.10",

"gitVersion" "1e05 l2f8453d103987f5fbfb87b7le9al3lc2a60" "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,

"akeadyHasObj" : 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,

"seenlnvalidated" : 0,

"matchTested" : 0

"serverInfo" : {

"host" : "admin", "port" : 27017,

"version" : "3.0.10",

"gitVersion" : "1e05 l2f8453d103987f5fbfb87b7le9al3lc2a60"

"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.uti1.Date; import com.mongodb.\*; public class App

public static void main(String[] args) BufferedReader br=null;

int ch,eid,sa1;

String sq1,name,desig; try

br=new BufferedReader(new InputStreamReader(System.in));

/\*\*\*\* Connect to MongoDB \*\*\*\*/

MongoClient mongo = new MongoC1ient("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="); sa1=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());

want you to update=");

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

eid=Integer.parseInt(br.readLine()); System.out.println("Enter the new salary="); sa1=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("\nInva1id Choice");

delete=");

break; case 4:

System.out.println("\nDe1ete Record Options:"); System.out.println("1.Delete Particular data"); System.out.println("Enter the choice=");

int chl=Integer.parseInt(br.readLine()); switch(chl)

case 1:

System.out.println("Enter the emp id whoes record want you to

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("\nIsnva1id Choice");

break; case 5:

break; default:

System.out.println("\nInva1id 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.De1ete 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.De1ete document 5.Exit

Enter the choice=

2

( " id" ( "$oid" "5423l6lf9905d80f8dbd5290"} , "empid" 1 , "ename" : "Eshaa" , "salary" : 550000 , "designation" : "abc"}

( " id" ( "$oid" : "5423l643990570dbcac5dfdf'} , "empid" 1 , "ename" : "Eshaa" , "salary" : 50000 , "designation" : "Designer"}

( " id" : ( "$oid" : "5423l65c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" , "salary" : 600000 , "designation" : "Developer"}

( " id" : ( "$oid" : "5423l678990570dbcac5dfe1"} , "empid" : 3 , "ename" : "Deendayal" , "salary" : 550000 , "designation" : "Manager"}

( " id" : ( "$oid" : "5423l6db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" , "salary" : 450000 , "designation" : "Executive"}

Choices for User 1.Insert document 2.View document 3.Update document 4.De1ete 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" "5423l6lf9905d80f8dbd5290"} , "empid" 1 , "ename" : "Eshaa" , "salary" : 550000 , "designation" : "abc"}

( " id" ( "$oid" "5423l643990570dbcac5dfdf’} , "empid" 1 , "ename" : "Eshaa" , "salary" : 50000 , "designation" : "Designer"}

( " id" : ( "$oid" : "5423l65c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" , "salary" : 600000 , "designation" : "Developer"}

( " id" : ( "$oid" : "5423l678990570dbcac5dfel"} , "empid" : 3 , "ename" : "Deendayal" , "salary" : 550000 , "designation" : "Manager"}

( " id" : ( "$oid" : "5423l6db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" , "salary" : 400000 , "designation" : "Executive"}

Choices for User 1.Insert document 2.View document 3.Update document 4.De1ete 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" : "5423161 f9905d80f8dbd5290"} , "empid" 1 , "ename" : "Eshaa" , "salary" : 550000 , "designation" : "abc"}

( " id" ( "$oid" "5423l643990570dbcac5dfdf’} , "empid" 1 , "ename" : "Eshaa" , "salary" : 50000 , "designation" : "Designer"}

( " id" : ( "$oid" : "5423 l65c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" , "salary" : 600000 , "designation" : "Developer"}

( " id" : ( "$oid" : "5423l678990570dbcac5dfel"} , "empid" : 3 , "ename" : "Deendayal" , "salary" : 550000 , "designation" : "Manager"}

( " id" : ( "$oid" : "5423l6db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" , "salary" : 400000 , "designation" : "HR"}

Choices for User 1.Insert document 2.View document 3.Update document 4.De1ete 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" "5423l6lf9905d80f8dbd5290"} , "empid" 1 , "ename" : "Eshaa" , "salary" : 550000 , "designation" : "abc"}

( " id" ( "$oid" "5423l643990570dbcac5dfdf’} , "empid" 1 , "ename" : "Eshaa" , "salary" : 50000 , "designation" : "Designer"}

( " id" : ( "$oid" : "5423l65c990570dbcac5dfe0"} , "empid" : 2 , "ename" : "Shriniwas" , "salary" : 600000 , "designation" : "Developer"}

( " id" : ( "$oid" : "5423l6db9905c6e3fd4bd76c"} , "empid" : 4 , "ename" : "Kaustubh" , "salary" : 400000 , "designation" : "HR"}

Choices for User 1.Insert document 2.View document 3.Update document 4.De1ete document 5.Exit

Enter the choice=

## 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/studentl";

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

1. UPDATE SOME VALUE\n
2. DISPLAY\n
3. DELETE\n
4. EXIT\n

ENTER YOUR OPTION : ";

System.out.println(menu); String query;

String sq1="update table employee set age=l;"; 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 : "); sa1ary=a.nextInt();

query="insert into

values("+empno+",”'+empname+"',"+age+",”’+designation+"',"+salary+");"; rs=stmt.executeUpdate(query);

if(rs==1)

System.out.println("\nData inserted succesfully! !\n");

employee

break;

case 2:System.out.println("Select the field you want to update

\nl.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!");

emp no ="+e+" ;";

break;

case 2: System.out.println("\nEnter the new designation : \n"); designation=b.nextLine();

query="update employee set designation = "’+designation+"" where

rs=stmt.executeUpdate(query); if(rs==1)

System.out.println("\n Updated successfully!");

break;

case 3: System.out.println("\nEnter the new salary : "); sa1ary=a.nextInt();

query="update employee set salary = "+sa1ary+" where emp no **="+e+";";**

rs=stmt.executeUpdate(query); if(rs==1)

System.out.println("\n Updated successfully!");

break;

default :System.out.println("\nP1ease enter a valid choice\n"); break;

break;

case 3:query="select \* from employee;"; ResultSet rsl=stmt.executeQuery(query);

System.out.println("Emp no\tEmp name\tAge\tDesgntn\tSalary"); while(rs1.next())

empno=rs1.getInt("emp no"); empname=rsl.getString("emp name"); age=rsl.getInt("age"); designation=rsl.getString("designation"); salary=rs1.getInt("salary");

System.out.println(empno+"\t"+empname+"\t"+age+"\t"+designation+"\t"+salary); break;

case 4 :System.out.println("\nl.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;"; rsl=stmt.executeQuery(query2); if(rsl==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 queryl="select \* from employee;"; rsl=stmt.executeQuery(queryl);

System.out.println("Emp no\tEmp name\tAge\tDesgntn\tSalary"); while(rs1.next())

empno=rsl.getInt("emp no"); empname=rsl.getString("emp name"); age=rsl.getInt("age"); designation=rsl.getString("designation"); salary=rsl.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

1. Employee name : kiran
2. Age :

20

1. Designation : executive
2. 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 |  |