# 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, Ino 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

1. 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;

1. 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=l)epositor.cname;

OR select c.name from Depositer union select c.name from Borrower;

1. 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;

1. 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;

1. Find average account balance at Akurdi branch.

select avg(balance), bname from Account where bname="Akurdi"

1. Find the average account balance at each branch select avg(balance),bname from Account group by bname;
2. Find no. of depositors at each branch.

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

1. Find the branches where average account balance > 12000.

select bname, avg(balance) from Account group by bname having avg(balance)>12000;

1. Find number of tuples in customer relation. select count(c name) from Customer;

OR

select count(\*) from Customer;

1. Calculate total loan amount given by bank.

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

OR select sum(amount) from Loan;

1. Delete all loans with loan amount between 1300 and 1500. delete from Loan where amount>=1300 and
2. 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.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.frame,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 Borrower1.bno,Borrower1.cname,Depositor1.Balance from Borrower 1 inner join Depositor 1 on Depositor 1.dno=Borrower1 .bno;

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

create view vupborrowerl as select bno,cname,bname,Amount from Borrower 1 •

# 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, Dateoflssue, 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(IO) |
| DATEOFISSUE | DATE |
| NAMEOFBOOK | CHAR(IO) |
| STATUS  SQL> desc fine; | CHAR(IO) |
| Name | Null? Type |

|  |  |
| --- | --- |
| ROLLNO | NUMBER(38) |
| FDATE | DATE |
| AMT | NUMBER(38) |

\*\*\*\*\*\*\* Insert values into Borrower table: \*\*\*\*\*\*\* SQL> Insert into borrower values (101,

1 row created.

SQL> Insert into borrower values (102, T); 1 row created.

SQL> Insert into borrower values (103,

'1');

1 row created.

SQL> Insert into borrower values (104, 1 row created.

SQL> Insert into borrower values (105,

1 row created.

SQL>

SQL> select \* from borrower;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

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

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 O 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 > O 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

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

SQL>

Enter value for rollno: [[1]](#footnote-1)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

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

SQL> select \* from fine;

ROLLNO FDATE AMT

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 1 104 Sai 25-AUG-17 SEPM 1

105 Ganesh Ol-SEP-17 IEEE 

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

1. Ram 23-SEP-17 DBMS R
2. Sai 10-SEP-17 CN R
3. Laxman 28-SEP-17 TOC 1
4. Sai 25-AUG-17 SEPM 105 Ganesh Ol-SEP-17 IEEE 

# 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 RollCa11.

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(l 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 andp total 990 THEN insert into result values(p roll no,p\_name,'Distinction'); Else ifp total andp total 900 THEN insert into result values(p roll no,p name,'First Class');

Else ifp total andp 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 ' I l var rollno I l ' 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

1. (var rollno in number,
2. p roll no out stud marks.roll no%type,
3. p name out stud marks.name%type, 5 p total out stud marks.total marks%type)
4. AS
5. BEGIN
6. SELECT roll no, name, total marks into p roll no, p name, p total from stud marks where roll no=var rollno;
7. IF p total andp total 990 THEN
8. insert into result values(p roll no,p name, 'Distinction') ;
9. Else if p total <=989 and p total >= 900 THEN
10. insert into result values(p roll no,p name,'First Class');
11. Else if p total <=899 and p total >= 825 THEN 14 insert into result values(p roll no,p name, 'HSC');
12. Else
13. insert into result values(p roll no,p name, 'fail');
14. End if;
15. End if;
16. End if;
17. EXCEPTION
18. WHEN no data found then
19. dbms output.put line('Roll no I l var rollno I l ' not found');
20. 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;
5. BEGIN
6. var rollno:=&var rollno;
7. Proc grade(var rollno,p roll no,p name,p total);
8. END

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

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;
5. BEGIN
6. var rollno:=&var rollno;
7. Proc grade(var rollno,p roll no,p name,p total);
8. END

Enter value for var rollno: 1 old 7: var rollno:=&var rollno; new 7: var rollno:=l;

PL/SQL procedure successfully completed.

SQL> select \* from result;

ROLL NO NAME CLASS

2 Ram First Class

* 1. Ganesh 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;
  5. BEGIN
  6. var rollno:=&var rollno;
  7. Proc grade(var rollno,p roll no,p name,p total);
  8. END

Enter value for var rollno: 3 old 7: var rollno:=&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;
5. BEGIN
6. var rollno:=&var rollno;
7. Proc grade(var rollno,p roll no,p name,p total);
8. END

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

Insert into library values (102, 'Sai',to date('20170910', Insert into library values (103, 'Laxman',to

Insert into library values (104, 'Sai',to date('20170825',

Insert into library values (105, 'Ganesh',to date('20170901

Select \* from library;

Select \* from library\_audit;

SQL> select \* from library;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

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

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

Ram 23-SEP-17 DBMS

1. Sai 10-SEP-17 CN 1
2. Laxman 28-SEP-17 TOC 1 104 Sai 25-AUG-17 SEPM 1 105 Ganesh Ol-SEP-17 IEEE 1

SQL> select \* from library\_audit; no rows selected

SQL> Insert into library values (106, 'Gajanan',to date('20171001','YYYYMMDD'),'

DDA',

1 row created.

SQL> select \* from library;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

1. Ram 23-SEP-17 DBMS 1
2. Sai 10-SEP-17 CN 1
3. Laxman 28-SEP-17 TOC 1 104 Sai 25-AUG-17 SEPM 1 105 Ganesh Ol-SEP-17 IEEE 1 106 Gajanan Ol-OCT-17 DDA 1

6 rows selected.

SQL> select \* from library\_audit;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS TS

|  |  |
| --- | --- |
| 106 Gajanan Ol-OCT-17 DDA 1 | 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

1. Ram 23-SEP-17 DBMS 1
2. Sai 10-SEP-17 CN 1
3. Laxman 28-SEP-17 TOC 1
4. Sai 25-AUG-17 SEPM 1 105 Ganesh Ol-SEP-17 IEEE 1 Gajanan Ol-OCT-17 DDA 6 rows selected.

SQL> select \* from library\_audit;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS TS

106 Gajanan Ol-OCT-17 DDA 1 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 1 102 Sai 10-SEP-17 CN 1

1. Laxman 28-SEP-17 TOC 1
2. Sai 25-AUG-17 SEPM 1
3. Ganesh Ol-SEP-17 IEEE 1 106 Gajanan Ol-OCT-17 DDA 1

6 rows selected.

SQL> select \* from library\_audit;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS TS

-106 Gajanan Ol-OCT-17 DDA 1 02-OCT-17 01.07.25.375000 PM 101 Ram 23-SEP-17 DBMS 1 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

1. Ram 23-SEP-17 MongoDB 1
2. Sai 10-SEP-17 CN 1
3. Laxman 28-SEP-17 TOC 1 104 Sai 25-AUG-17 SEPM 1 105 Ganesh Ol-SEP-17 IEEE 1 Gajanan Ol-OCT-17 DDA 6 rows selected.

SQL> select \* from library\_audit;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS TS

106 Gajanan Ol-OCT-17 DDA 1 02-OCT-17 01.07.25.375000 PM 101 Ram 23-SEP-17 MongoDB 1 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 1

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

SQL> select \* from library\_audit;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS TS

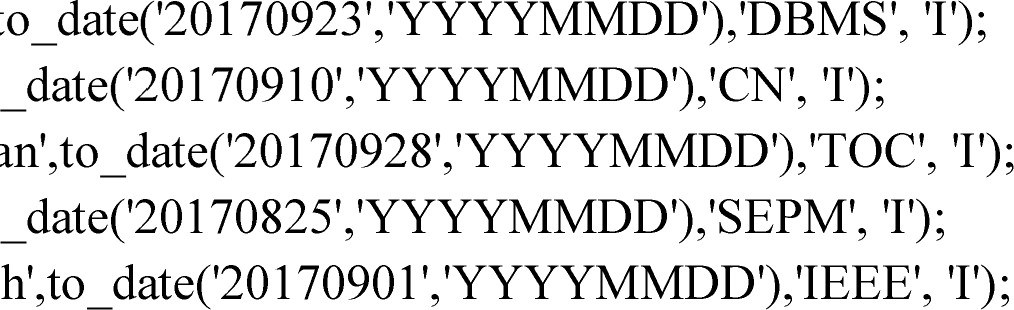
102 Sai 10-SEP-17 CN 1 02-OCT-17 02.15.24.618000 PM

106 Gajanan Ol-OCT-17 DDA 1 02-OCT-17 01.07.25.375000 PM 101 Ram 23-SEP-17 MongoDB 1 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

Insert into lib values (102, 'Sai',to

Insert into lib values (103, 'Laxman',to

Insert into lib values (104, 'Sai',to

Insert into lib values (105, 'Ganesh',to date('20170901

Select \* from lib;

Select \* from lib audit;

CREATE OR REPLACE TRIGGER ATI

AFTER INSERT OR DELETE OR UPDATE

17

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 date('20171001 1 row created.

SQL> select \* from lib;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

1. Ram 23-SEP-17 DBMS 1
2. Sai 10-SEP-17 CN 1
3. Laxman 28-SEP-17 TOC 1
4. Sai 25-AUG-17 SEPM 1
5. Ganesh Ol-SEP-17 IEEE 1
6. Gajanan Ol-OCT-17 DDA 1 6 rows selected.

SQL> select \* from lib audit;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS TS COMMAND

106 Gajanan Ol-OCT-17 DDA 1 02-OCT-17 11.12.03.791000PM 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

1. Ram 23-SEP-17 MongoDB 1
2. Sai 10-SEP-17 CN 1
3. Laxman 28-SEP-17 TOC 1
4. Sai 25-AUG-17 SEPM 1 105 Ganesh Ol-SEP-17 IEEE 1 106 Gajanan Ol-OCT-17 DDA 1

6 rows selected.

SQL> select \* from lib audit;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS TS COMMAND

106 Gajanan Ol-OCT-17 DDA 1 02-OCT-17 11.12.03.791000PM INSERT

101 Ram 23-SEP-17 DBMS 1 02-OCT-17 11.14.21.436000PM 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 Ol-OCT-17 DDA 1 02-OCT-17 11.12.03.791000 PM INSERT 101 Ram 23-SEP-17 MongoDB 1 02-OCT-17 11.14.21.436000 PM UPDATE

102 Sai 10-SEP-17CN 1 02-OCT-17 11.16.03.851000 PM DELETE

SQL> select \* from lib;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

101 Ram 23-SEP-17 MongoDB 1 103 Laxman 28-SEP-17 TOC 1 104 Sai 25-AUG-17 SEPM 1

105 Ganesh Ol-SEP-17 IEEE 1 106 Gajanan Ol-OCT-17 DDA 1

SQL>

# -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()

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( {$or:[ {Age:{$lt: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. {$ne: "Developer" } }).pretty()
2. 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()

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. Find all document with Exact Match on an Array having Expertise:

['Mongodb','Mysql', 'Cassandra'] db.emp. find( {Expertise: ["Mongodb" , " Mysql " , " Cassandra"] .pretty()

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

## -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 :  :

{totalSa1 : { $1te : 200000 } } } )

1. Find Employee with Total Salary for Each City with Designation="DBA" db.s.aggregate([{$match:{Designation:"DBA"}},{$group:{ id:"$Address",totalSal:{$sum:

### "Salary" } }

1. Find Total Salary of Employee with Designation="DBA" for Each Company db.s.aggregate([{$match:{Designation:"DBA"}},{$group:{ id:"$Company\_name",totalSal : {$sum: "Salary"} } }
2. Returns names and id in upper case and in alphabetical order.

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

1. Count all records from collection db.s.aggregate( [{$group: { id: null,count: { $sum: 1 }}}] )
2. For each unique Designation, find avg Salary and output is sorted by AvgSal db.s.aggregate( [{$group: { id: "$Designation",AvgSal: { $avg: "Salary" }}}, { $sort: { AvgSal: 1 } }] )
3. Return separates value in the Expertise array where Name of Employee="Swapnil" db. s. aggregate ([{$unwind: "$Expertise" } , {$match: {Name: " Swapnil" } }
4. Return separates value in the Expertise array and return sum of each element of array db.s.aggregate([{$unwind:"$Expertise"},{$group:{ id:"$Expertise",number:{$sum:1}}})
5. Return Array for Designation whose address is "Pune" id:"$Address", Array\_Designation: {$push: "$Designation" } } } ])
6. Return Max and Min Salary for each company.

db.s.aggregate([{$group:{ id:"$Company\_name",min:{$min:"$Salary"}, max: {$max: "Salary"} } }

-1.

# -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.ensurelndex( { "Designation": 1 } )
2. To Create Compound Indexes on Name: 1, Age: -1 db.emp.find().sort( { Name: 1, Age: -1 } ).explain("executionStats") db.emp.ensurelndex( { Name : 1, Age : -1 } )
3. To Create Multikey Indexes on Expertise array db. emp. "Expertise.2" : "Java" } explain(" executionStats") db. emp. ensurelndex( { "Expertise. Java" : 1 } )
4. Return a List of All Indexes on Collection db.emp.getlndexes()
5. Rebuild Indexes db.emp.relndex()
6. Drop Index on Remove Specific Index db.emp.droplndex( { "Designation": 1 } )
7. Remove All Indexes except for the id index from a collection db.emp.droplndexes()
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=o; i<200000; 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

"serverlnfo" : {

"host" : "admin"

"POW' : 27017,

23

"version" : "3.0.10",

"gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"

"ok" : 1



Before Index: Find user0 in collection



"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

"serverlnfo" : {

"host" : "admin"

"POW' : 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" : "user 19999"

"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"

"executionTimeMillisEstimate 60,

"works" : 200002,

"advanced" : 1

"needTime" : 200000,

"needFetch" : 0,

"saveState" : 1562,

"restoreState" • 1562,

"isEOF" : 1

"invalidates" : 0,

"direction" : "forward",

"docsExamined" : 200000

"serverlnfo" : { "host" : "admin"

"POW' : 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,

• • " : 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

"serverlnfo" : {

"host" : "admin"

"POW' : 27017,

"version" : "3.0.10",

"gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"

"0k" • 1

To create Single Field Index on Username > db.user.createlndex({username:l})

"createdCollectionAutomatically" : false,

"numIndexesBefore" . 

"numIndexesAfter" : 2,

"ok" : 1

> db.user.getlndexes()

"key" : { id" : 1

"name" id

"ns" : "mescoe.user"

"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

"serverlnfo" : {

"host" : "admin"

"POW' : 27017,

"version" : "3.0.10",

"gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"

"ok" : 1



After Index: Find user0 in collection



"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "mescoe.user"

"indexFilterSet" : false,

"parsedQuery" : {

"username" : {

"$eq" : "user0"

"winningPlan" : {

"stage" : "FETCH",

"mputStage"

"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" . 

"needT1me

"needFetch" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1

"invalidates" : 0,

"docsExamined" . 

"alreadyHasObj" : 0,

"mputStage"

"stage" : "IXSCAN",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0, "works" : 2,

"advanced" . 

"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

"serverlnfo" : { "host" : "admin"

"POW' : 27017, "version" : "3.0.10", "gitVersion" :

"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",

"mputStage"

"stage" : "IXSCAN",

"keyPattern" : {

"username" : 1

"indexName" : "username 1 "

"isMultiKey" : false,

"direction" : "forward",

"indexBounds"

"username" : [

" "user 19999\ , \ "user 19999\ "

"rejectedPlans" : [ ]

"executionStats" .

"executionSuccess" : true,

"nReturned" : 1,

"executionTimeMillis" : 0,

"totalKeysExamined" : 1

"totalDocsExamined" : 1

"executionStages"

"stage" : "FETCH",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0,

"works" : 2,

"advanced"

"needT1me

"needFetch" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1

"invalidates" : 0,

"docsExamined" . 

"alreadyHasObj" : 0,

"mputStage"

"stage" "IXSCAN",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0, "works" : 2,

"advanced" . 

"needTime" : 0,

"needFetch" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1

"invalidates" : 0,

"keyPattern" : {

"username" : 1

"indexName" : "username 1

"isMultiKey" : false,

"direction" : "forward",

"indexBounds"

"username" : [

" "user 19999\ , \ "user 19999\ "

"keysExamined" : 1

"dupsTested" : 0,

"dupsDropped" : 0,

"seenlnvalidated" : 0,

"matchTested" : 0

"serverlnfo" : { "host" : "admin"

"POW' : 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",

"mputStage"

"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" . 

"needT1me

"needFetch" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"docsExamined" . 

"alreadyHasObj" : 0,

"mputStage"

"stage" : "IXSCAN",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0, "works" : 2,

"advanced" . 

"needTime" : 0,

"needFetch" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1

33

"invalidates" 0,

"keyPattern" : {

"username" : 1

"indexName" . "username 1

"isMultiKey" : false,

"direction" : "forward",

"indexBounds"

"username" : [

\ "user9999\ "

"keysExamined" : 1

"dupsTested" : 0,

"dupsDropped" : 0,

"seenlnvalidated" : 0,

"matchTested" : 0

"serverlnfo" : { "host" : "admin"

"POW' : 27017,

"version" : "3.0.10",

"gitVersion" : "1e0512f8453d103987f5fbfb87b71e9a131c2a60"

"ok" : 1



## Group C: Assignment -1

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

ODBC/JDBC.

import java. net. UnknownHo stException; 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 .lnsert document");

System.out.println("2.View document");

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:

35

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. ... t '); 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: 1') ;

System.out.println(" 1. Update salary. t ') ; 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); newl)ocument.put("salary", sal);

updateObj.put("$set", newDocument); table.update(query, updateObj);

System.out.println("\nl)ocument 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("\nl)ocument Updated Successfully... break; default:

System.out.println("\nlnvalid Choice");

break; case 4:

System.out.println("\nDelete Record Options: t') ;

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 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("\nlsnvalid Choice");

break; case 5: break; default:

System.out.println("\nlnvalid Choice");

}while(ch!=5);

/\* \*\*\* Done \*\*\*\*/

System.out.println("Thank You... 1');

37

by:SHRINIWAS DESHMUKH..

catch (UnknownHostException e)

e.printStackTrace();

catch (MongoException e)

e.printStackTrace();

catch(IOException e)

e.printStackTrace();

/ output:

Choices for User

1 .lnsert document

1. View document
2. 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 .lnsert document

1. View document
2. Update document

4.Delete document

5.Exit

Enter the choice=

2 id" • { ' '$01d• " : "5423161f9905d80f8dbd5290"} "salary" : 550000 , "designation" : "abc"} id" • { ' '$01d• " : "54231643990570dbcac5dfdf'}

"salary" : 50000 , "designation" : "Designer"}

{ "\_id" : { "$01d• " : "5423165c990570dbcac5dfeO'

"salary" • 600000 "designation" : "Developer"}

{ "\_id" : { "$01d• " : "54231678990570dbcac5dfe1

"salary" : 550000 "designation" : "Manager"}

{ "\_id" : { ' '$01d• " : "542316db9905c6e3fd4bd76c'

"salary" : 450000 "designation" : "Executive"}

Choices for User

1 .lnsert document

1. View document
2. 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 .lnsert document

1. View document
2. Update document

4.Delete document

5.Exit

Enter the choice=

2 id" • { ' '$01d• " : "5423161f9905d80f8dbd5290"} "salary" : 550000 , "designation" : "abc"} id" • { ' '$01d• " : "54231643990570dbcac5dfdf'}

"salary" : 50000 , "designation" : "Designer"}

{ "\_id" : { "$01d• " : "5423165c990570dbcac5dfeO"}

"salary" : 600000 , "designation" : "Developer"} "empid""ename" : "Eshaa"

 "empid""ename" : "Eshaa"

"empid" • 2  "ename" : "Shriniwas"

"empid""ename l' : "Deendayal"

"empid" • 4  "ename" : "Kaustubh"

 "empid""ename" : "Eshaa"

 "empid""ename" : "Eshaa"

"empid" : 2 , "ename" : "Shriniwas"

39

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | { "\_id" : { ' '$01d• " : "54231678990570dbcac5dfe1"}  "salary" : 550000 "designation" : "Manager"} | "empid" • 3 | | "ename" : "Deendayal" | | { "\_id" : { ' '$01d• " : "542316db9905c6e3fd4bd76c"}  "salary" : 400000 "designation" : "Executive"}  Choices for User  1 .lnsert document   1. View document 2. 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 .lnsert document   1. View document 2. Update document   4.Delete document  5.Exit  Enter the choice=  2 | "empid' | • 4 | "ename" : "Kaustubh" | | id" • { ' '$01d• " : "5423161f9905d80f8dbd5290"}  "salary" : 550000 , "designation" : "abc"} | "empid" | | "ename" : "Eshaa" | | id" • { ' '$01d• " : "54231643990570dbcac5dfdf'}  "salary" : 50000 , "designation" : "Designer"} | "empid" | | "ename" : "Eshaa" | | { " id" • { ' '$01d• " : "5423165c990570dbcac5dfeO"}  "salary" • 600000 "designation" : "Developer"} | "empid" • 2 | | "ename" : "Shriniwas" | | { "\_id" : { "$01d• " : "54231678990570dbcac5dfe1 t ' }  "salary" : 550000 "designation" : "Manager"} | "empid" • 3 | | "ename" : "Deendayal" | | { "\_id" : { ' '$01d• " : "542316db9905c6e3fd4bd76c"}  "salary" : 400000 "designation" : "HR"} | "empid" • 4 | | "ename" : "Kaustubh" | |  |



Choices for User

1 .lnsert document

1. View document
2. 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 .lnsert document

1. View document
2. Update document

4.Delete document

5.Exit

Enter the choice=

2 id" • { ' '$01d• " : "5423161f9905d80f8dbd5290"} "salary" : 550000 , "designation" : "abc"} id" • { ' '$01d• " : "54231643990570dbcac5dfdf'}

"salary" : 50000 , "designation" : "Designer"}

{ "\_id" : { "$01d• " : "5423165c990570dbcac5dfeO"}

"salary" • 600000 "designation" : "Developer"}

{ "\_id" : { ' '$01d• " : "542316db9905c6e3fd4bd76c"}

"salary" : 400000 "designation" : "HR"}

Choices for User

1 .lnsert document

1. View document
2. Update document

4.Delete document

5.Exit

Enter the choice=

5

"empid""ename" : "Eshaa"

"empid""ename" : "Eshaa"

"empid" • 2  "ename" : "Shriniwas"

"empid' • 4  "ename" : "Kaustubh"

## 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

String DB URL="jdbc:mysql://localhost/student1 "

Connection conn=DriverManager.getConnection(DB URL,UID,PWD); Statement stmt=conn.createStatement(); do

String menu=" OPERATIONS\n

1.1NSERT 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=l i=a.nextlnt(); switch(i)

case 1:System.out.println("Enter the following information to be inserted(Blank fields to be avoided)");

.Employee number : t '); empno=a.nextlnt(); System.out.println("2.Employee name : empname=b.nextLine(); System.out.println("3.Age : t '); age=a.nextlnt();

System.out.println("4.Designation . designation=b.nextLine(); System.out.println("5.Salary : 1') ; salary=a.nextlnt(); query="msert 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 Salary\n"); int option=a.nextlnt();

System.out.println("Enter the employee id for which you want to update data :

e=b.nextlnt(); switch(option)

case 1 : System.out.println("\nEnter the new age : 1') ; age=a.nextlnt(); query="update employee set age "+age+" where emp no 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 rs=stmt.executeUpdate(query); if(rs==1)

System.out.println("\n Updated successfully! 1') ;

break; case 3: System.out.println("\nEnter the new salary : 1') ; salary=a.nextlnt(); query="update employee set salary = "+salary+" where emp\_no rs=stmt.executeUpdate(query); if(rs==1)

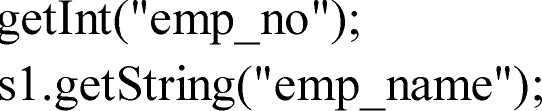
43

System.out.println("\n Updated successfully! 1') ;

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

break; case 3:query="select \* from employee;" ; ResultSet rs 1 =stmt.executeQuery(query) ;

System.out.println("Emp no\tEmp\_name\tAge\tDesgntn\tSalary"); while(rs 1 .next())

empno=rsl empname=rsl. get age=rsl.getlnt("age"); designation=rsl. get String("designation"); salary=rsl. getlnt(" salary");

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

break; case 4 ALL RECORDS\n2.DELETE SELECTED

DATA"); option=a.nextlnt(); switch(option)

case 1 table employee; "; rs=stmt.executeUpdate(query);

String query2="select \* from employee;" ; rs 1 =stmt. executeQuery(query2); if(rs 1 ==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.nextlnt(); query="delete from employee where emp no rs=stmt.executeUpdate(query); if(rs==1)

System.out.println("\nThe specified record has been deleted!

String queryl="select \* from employee;" ; rs 1 =stmt. executeQuery(query 1 ) ;

System.out.println("Emp no\tEmp\_name\tAge\tDesgntn\tSalary"); while(rs 1 .next())



empno=rsl no"); empname=rs 1. getString(" emp\_name") ; age=rsl.getlnt("age"); designation=rs 1. getString(" designation"); salary=rsl. getlnt(" 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

1.1NSERT 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

45

5.Salary :

500000

Data inserted succesfully! !

OPERATIONS

1.1NSERT 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

1.1NSERT 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

1.1NSERT 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

1.1NSERT 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  1 eshaa 19 CEO 60000000 2 varsha 19 manager 500000  3 kalpita 20 sales 300000 | Salary |

47

1. 28-SEP-17 90 [↑](#footnote-ref-1)