Problem Statement: Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

CREATE TABLE:

```
CREATE TABLE PLATFORM(
  PNO INT PRIMARY KEY,
  ARR  DECIMAL(4,2),
  DEPART DECIMAL(4,2)
);
```

Object Type **TABLE** Object **PLATFORM**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PLATFORM	PNO	Number	-	-	0	1	-	-	-
	ARR	Number	-	4	2	-	✓	-	-
	DEPART	Number	-	4	2	-	✓	-	-

1 - 3

```
CREATE TABLE PASSENGER(
   NO NUMBER(2),
   NAME VARCHAR2(10),
   GENDER VARCHAR2(6),
   AGE NUMBER(3),
   MOBILE_NO NUMBER(10),
   AADHAR_NO VARCHAR2(12) PRIMARY KEY
);
```

Object Type **TABLE** Object **PASSENGER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PASSENGER	NO	Number	-	2	0	-	✓	-	-
	NAME	Varchar2	10	-	-	-	✓	-	-
	GENDER	Varchar2	6	-	-	-	✓	-	-
	AGE	Number	-	3	0	-	✓	-	-
	MOBILE_NO	Number	-	10	0	-	✓	-	-
	AADHAR_NO	Varchar2	12	-	-	1	-	-	-

1 - 6

```
CREATE TABLE TRAIN(
TNO NUMBER(5) PRIMARY KEY,
T_NAME VARCHAR2(20),
ARR DECIMAL(4,2),
DEPART DECIMAL(4,2),
PNO NUMBER(3), FOREIGN KEY(PNO) REFERENCES PLATFORM(PNO)
);
```

Object Type **TABLE** Object **TRAIN**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TRAIN	TNO	Number	-	5	0	1	-	-	-
	T_NAME	Varchar2	20	-	-	-	✓	-	-
	ARR	Number	-	4	2	-	✓	-	-
	DEPART	Number	-	4	2	-	✓	-	-
	PNO	Number	-	3	0	-	✓	-	-
									1 - 5

CREATE TABLE RESERVATION

(

PNR NUMBER(10) PRIMARY KEY,

AADHAR VARCHAR2(12), FOREIGN KEY(AADHAR) REFERENCES PASSENGER(AADHAR_NO),

TNO NUMBER(5), FOREIGN KEY(TNO) REFERENCES TRAIN(TNO),

DT DATE,

P_FROM VARCHAR2(10),

P_TO VARCHAR2(10),

R_STATUS VARCHAR2(10),

R_CLASS VARCHAR2(10)

);

Object Type **TABLE** Object **RESERVATION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVATION	PNR	Number	-	10	0	1	-	-	-
	AADHAR	Varchar2	12	-	-	-	✓	-	-
	TNO	Number	-	5	0	-	✓	-	-
	DT	Date	7	-	-	-	✓	-	-
	P_FROM	Varchar2	10	-	-	-	✓	-	-
	P_TO	Varchar2	10	-	-	-	✓	-	-
	R_STATUS	Varchar2	10	-	-	-	✓	-	-
	R_CLASS	Varchar2	10	-	-	-	✓	-	-
									1 - 8

ALTERING TABLE:

ALTER TABLE PASSENGER ADD ADDRESSS VARCHAR2(10);

Object Type **TABLE** Object **PASSENGER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PASSENGER	NO	Number	-	2	0	-	✓	-	-
	NAME	Varchar2	10	-	-	-	✓	-	-
	GENDER	Varchar2	6	-	-	-	✓	-	-
	AGE	Number	-	3	0	-	✓	-	-
	MOBILE_NO	Number	-	10	0	-	✓	-	-
	AADHAR_NO	Varchar2	12	-	-	1	-	-	-
	ADDRESSS	Varchar2	10	-	-	-	✓	-	-
									1 - 7

DROP TABLE:

DROP TABLE RESERVATION;

```
Table dropped.
```

CREATE VIEW:

CREATE VIEW TICKET_2 AS SELECT RESERVATION.PNR, PASSENGER.NAME,RESERVATION. DT ,RESERVATION.TNO ,RESERVATION.P_FROM ,RESERVATION.P_TO FROM RESERVATION,PASSENGER WHERE RESERVATION.AADHAR=PASSENGER.AADHAR_NO;

```
View created.


0.27 seconds
```

SELECT *FROM TICKET_2 ORDER BY PNR;

PNR	NAME	DT	TNO	P_FROM	P_TO
1200600321	Shruti	28-JUN-21	13907	Pune	Mumbai
1200600506	Priyanka	01-JUL-21	12331	Latur	Mumbai
1200612345	Rohit	02-JUL-21	23410	Varanasi	Mumbai
1200700845	Mayur	10-JUL-21	10345	SC	Porbandar

4 rows returned in 0.19 seconds CSV Export

CREATE SEQUENCE:

CREATE SEQUENCE SEQUENCE_2

START WITH 1

INCREMENT BY 1

MINVALUE 0

MAXVALUE 50;

DESC SEQUENCE_2;

```
Object Type  SEQUENCE Object  SEQUENCE_2
```

CREATE INDEX:

CREATE INDEX IDX_RESERVATION ON RESERVATION(DT);

CREATE INDEX IDX_PASSENGER ON PASSENGER(NO);

CREATE INDEX IDX_TRAIN ON TRAIN(PNO);

```
Index created.
```

DROP INDEX:

DROP INDEX IDX_PASSENGER ;

```
Index dropped.
```

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

INSERT DATA INTO TABLE:

```
BEGIN
INSERT INTO TRAIN VALUES
 (12331,'LATUR CSTM','19.00','19.30',4);
 INSERT INTO TRAIN VALUES
 (13907,'INDRAYANI EXPRESS','19.20','20.20',6);
 INSERT INTO TRAIN VALUES
 (10345,'SC PBR SPECIAL','20.30','20.50',8);
 INSERT INTO TRAIN VALUES
 (23410,'BSB CSTM SPECIAL','21.15','21.30',3);
 INSERT INTO TRAIN VALUES
 (12232,'PUNE TPJ SPECIAL','21.30','21.56',2);
 INSERT INTO TRAIN VALUES
 (80876,'YPR JP EXPRESS','22.45','23.15',5);
 INSERT INTO TRAIN VALUES
 (22578,'KOP LTT SPECIAL','01.00','01.30',10);
 INSERT INTO TRAIN VALUES
 (30076,'PUNE TPL SPECIAL','2.00','2.30',8);
END;
/
```

TNO	T_NAME	ARR	DEPART	PNO
12331	LATUR CSTM	19	19.3	4
13907	INDRAYANI EXPRESS	19.2	20.2	6
10345	SC PBR SPECIAL	20.3	20.5	8
23410	BSB CSTM SPECIAL	21.15	21.3	3
12232	PUNE TPJ SPECIAL	21.3	21.56	2
80876	YPR JP EXPRESS	22.45	23.15	5
22578	KOP LTT SPECIAL	1	1.3	10
30076	PUNE TPL SPECIAL	2	2.3	8

8 rows returned in 0.06 seconds CSV Export

UPDATAE TABLE:

```
UPDATAE TRAIN
SET PNO=4
WHERE TNO=30076;
```

```
1 row(s) updated.
```

22578	KOP LTT SPECIAL	1	1.3	10
30076	PUNE TPL SPECIAL	2	2.3	4

DELETE DATA FROM TABLE:

DELETE FROM TRAIN

WHERE TNO=30076;

```
1 row(s) deleted.
```

TNO	T_NAME	ARR	DEPART	PNO
12331	LATUR CSTM	19	19.3	4
13907	INDRAYANI EXPRESS	19.2	20.2	6
10345	SC PBR SPECIAL	20.3	20.5	8
23410	BSB CSTM SPECIAL	21.15	21.3	3
12232	PUNE TPJ SPECIAL	21.3	21.56	2
80876	YPR JP EXPRESS	22.45	23.15	5
22578	KOP LTT SPECIAL	1	1.3	10

7 rows returned in 0.00 seconds CSV Export

FUNCTION:

COUNT:

SELECT COUNT(*) FROM TRAIN

COUNT(*)
7

SUM

SELECT SUM(PNO) FROM TRAIN

SUM(PNO)
38

MIN

SELECT MIN(PNO) FROM TRAIN

MIN(PNO)
2

1 rows returned in 0.02 seconds

MAX

SELECT MAX(PNO) FROM TRAIN

MAX(PNO)
10

SET OPERATION:

TABLE1:

ID	NAME
1	Rohit
2	Priyanka
3	Shruti

TABLE2:

ID	NAME
4	Manas
5	Mrunal
6	Ajit
3	Shruti

UNION:

SELECT *FROM TABLE1 UNION SELECT *FROM TABLE2;

ID	NAME
1	Rohit
2	Priyanka
3	Shruti
4	Manas
5	Mrunal
6	Ajit

6 rows returned in 0.11 seconds

UNION ALL:

SELECT *FROM TABLE1 UNION ALL SELECT *FROM TABLE2;

ID	NAME
1	Rohit
2	Priyanka
3	Shruti
4	Manas
5	Mrunal
6	Ajit
3	Shruti

7 rows returned in 0.03 seconds

INTERSECT:

SELECT *FROM TABLE1 INTERSECT SELECT *FROM TABLE2;

ID	NAME
3	Shruti

1 rows returned in 0.08 seconds

MINUS :

SELECT *FROM TABLE1 MINUS SELECT *FROM TABLE2;

ID	NAME
1	Rohit
2	Priyanka

2 rows returned in 0.00 seconds

Problem Statement: SQL Queries – all types of Join, Sub-Query and View: Write at least10 SQL queries for suitable database application using SQL DML statements.

INNER JOIN

SELECT TABLE1.ID, TABLE1.NAME, TABLE2.ID, TABLE2.NAME

FROM TABLE1

INNER JOIN TABLE2 ON TABLE1.ID = TABLE2.ID;

ID	NAME	ID	NAME
3	Shruti	3	Shruti

RIGHT OUTER JOIN

SELECT TABLE1.ID, TABLE1.NAME,TABLE2.ID, TABLE2.NAME

FROM TABLE1

RIGHT OUTER JOIN TABLE2 ON TABLE1.ID = TABLE2.ID;

ID	NAME	ID	NAME
3	Shruti	3	Shruti
-	-	6	Ajit
-	-	5	Mrunal
-	-	4	Manas

4 rows returned in 0.05 seconds

LEFT OUTER JOIN

SELECT TABLE1.ID, TABLE1.NAME, TABLE2.ID, TABLE2.NAME

FROM TABLE1

LEFT OUTER JOIN TABLE2 ON TABLE1.ID = TABLE2.ID;

ID	NAME	ID	NAME
3	Shruti	3	Shruti
1	Rohit	-	-
2	Priyanka	-	-

3 rows returned in 0.00 seconds

FULL OUTER JOIN

SELECT TABLE1.ID, TABLE1.NAME, TABLE2.ID, TABLE2.NAME

FROM TABLE1

FULL OUTER JOIN TABLE2 ON TABLE1.ID = TABLE2.ID;

ID	NAME	ID	NAME
3	Shruti	3	Shruti
1	Rohit	-	-
2	Priyanka	-	-
-	-	4	Manas
-	-	5	Mrunal
-	-	6	Ajit

6 rows returned in 0.30 seconds

CROSS JOIN

SELECT * FROM TABLE1 CROSS JOIN TABLE2;

ID	NAME	ID	NAME
1	Rohit	4	Manas
1	Rohit	5	Mrunal
1	Rohit	6	Ajit
1	Rohit	3	Shruti
2	Priyanka	4	Manas
2	Priyanka	5	Mrunal
2	Priyanka	6	Ajit
2	Priyanka	3	Shruti
3	Shruti	4	Manas
3	Shruti	5	Mrunal
More than 10 rows available. Increase rows selector to view more rows.			

10 rows returned in 0.16 seconds CSV Export

View:

Create A Sql View:

CREATE VIEW SIMPLEVIEW AS

SELECT NAME

FROM TABLE1

WHERE ID< 3;

```
View created.
```

SELECT * FROM SIMPLEVIEW;

NAME
Rohit

1 rows returned in 0.25 seconds

Update An Sql View

```
CREATE OR REPLACE VIEW SIMPLEVIEW  AS

SELECT NAME

FROM  TABLE1

WHERE ID > 2;

SELECT * FROM SIMPLEVIEW;
```

NAME
Shruti

1 rows returned in 0.00 seconds

Insert New Record In The View

```
INSERT INTO SIMPLEVIEW(NAME) VALUES ('AISHU');
```

Delete The Existing Row From The View

```
DELETE FROM SIMPLEVIEW WHERE NAME='ROHIT';
```

NAME
Shruti
AISHU

2 rows returned in 0.01 seconds

Drop A View

```
DROP VIEW SIMPLEVIEW ;
```

```
View dropped.
```

Problem Statement: Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory. Write a PL/SQLblock of code for the following requirements:- Schema: 1. Borrower(Roll,Name,DateofIssue, NameofBook, Status 2. Fine (Roll, Date, Amt) Accept Roll & Name of book from user. Check the number of days (from date of issue), if days are between 15 to 30 then fine amounts will be Rs 5/ 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 Borrower(
Roll_no number not null,
Name varchar2(20) not null,
DateofIssue date,
NameofBook varchar2(20),
Status varchar2(2)
);

insert into Borrower values (1,'Aishwarya', TO_DATE('08/09/2002','dd-mm-yyyy'),'CN','A');
insert into Borrower values (2,'Priyanka', TO_DATE('20/09/2002','dd/mm/yyyy'),'DBMS','A');
insert into Borrower values (3,'Madhav', TO_DATE('25/10/2002','dd/mm/yyyy'),'TOC','A');
insert into Borrower values (5,'Shruti', TO_DATE('27/10/2002','dd/mm/yyyy'),'SPOS','A');
insert into Borrower values (1,'Aishwarya', TO_DATE('03/10/2002','dd/mm/yyyy'),'TOC','A');
insert into Borrower values (4,'Shree', TO_DATE('25/09/2020','dd/mm/yyyy'),'DBMS','I');

select * from Borrower;
```

ROLL_NO	NAME	DATEOFISSUE	NAMEOFBOOK	STATUS
1	Aishwarya	08-SEP-02	CN	A
2	Priyanka	20-SEP-02	DBMS	A
3	Madhav	25-OCT-02	TOC	A
5	Shruti	27-OCT-02	SPOS	A
1	Aishwarya	03-OCT-02	TOC	A
4	Shree	25-SEP-20	DBMS	I

6 rows returned in 0.03 seconds CSV Export

```
CREATE TABLE Fine(
Roll_no NUMBER,
Date_Return DATE,
Amt NUMBER
);
```

Procedure Finer:

```
create or replace procedure finer(r in number, book_name in varchar2) as
  days number;
begin
  select trunc(sysdate - DateofIssue) into days
  from Borrower
  where Roll_no = r and NameofBook = book_name;

  if days < 15 then
    insert into Fine values (r, sysdate, 0);
  elsif days >= 15 and days < 30 then
    insert into Fine values (r, sysdate, 0);
    update Fine set Amt = Amt + (days * 5) where Roll_no = r;
  else
    days := days - 30;
    insert into Fine values (r, sysdate, 0);
    update Fine set Amt = Amt + (days * 50) + 75 where Roll_no = r;
  end if;

  update Borrower set Status = 'R' where Roll_no = r and NameofBook = book_name;
end;
/
```

Execution:

```
BEGIN
  finer(1, 'CN');
END;
/
select * from Fine;
```

ROLL_NO	DATE_RETURN	AMT
1	11-OCT-23	383725

1 rows returned in 0.03 seconds CSV Export

```
BEGIN
  finer(1, 'TOC');
  finer(2, 'DBMS');
END;
/
select * from Fine;
```

ROLL_NO	DATE_RETURN	AMT
1	11-OCT-23	766200
1	11-OCT-23	382475
2	11-OCT-23	383125

3 rows returned in 0.00 seconds CSV Export

Problem Statement: Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

CREATE TABLE areas (radius number, area number);

desc areas;

Object Type **TABLE** Object **AREAS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
AREAS	RADIUS	Number	-	-	-	-	✓	-	-
	AREA	Number	-	-	-	-	✓	-	-

1 - 2

```
declare
radius_var number;
area_var number;
pi number:= 3.14;
begin
dbms_output.put_line('Area of circle from radius 5 to 9');
for radius_var in 5..9 loop
area_var:=pi*radius_var*radius_var;
dbms_output.put_line(area_var);
insert into areas values(radius_var,area_var);
end loop;
end;
/
```

```
Area of circle from radius 5 to 9
78.5
113.04
153.86
200.96
254.34

Statement processed.
```

select * from areas;

RADIUS	AREA
5	78.5
6	113.04
7	153.86
8	200.96
9	254.34

5 rows returned in 0.26 seconds

Problem Statement: Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.

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 and900 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(name, total_marks) Result(Roll,Name, Class)

```
--student marks
create table Stud_Marks(
   STUD_NAME varchar2(20),
   TOTAL_MARKS number(5)
);

begin
insert into Stud_Marks values ('Aishwarya', 1400);
insert into Stud_Marks values ('Priyanka', 800);
insert into Stud_Marks values ('Madhav', 830);
insert into Stud_Marks values ('Shruti', 900);
insert into Stud_Marks values ('Pritam', 990);
insert into Stud_Marks values ('Ram', 300);
end;
/
```

RADIUS	AREA
5	78.5
6	113.04
7	153.86
8	200.96
9	254.34

5 rows returned in 0.26 seconds

```
-- Result table
create table Result(
   STUD_NAME varchar2(20),
   ROLL_NO number(5),
   CLASS varchar2(20)
);
```

```
create or replace PROCEDURE PROC_GRADE1 AS
BEGIN
	FOR i IN (SELECT * FROM Stud_Marks)
	LOOP
		DBMS_OUTPUT.PUT_LINE('Student Name: ' || i.Stud_Name || ' Student Marks: ' || i.Total_Marks);
		IF i.Total_Marks <=1500 AND i.Total_Marks >=990 THEN
			INSERT INTO Result (STUD_NAME,CLASS) VALUES (i.Stud_Name,'Distinction');
		ELSIF i.Total_Marks <=989 AND i.Total_Marks >=900 THEN
			INSERT INTO Result (STUD_NAME,CLASS) VALUES (i.Stud_Name,'First Class');
		ELSIF i.Total_Marks <=825 AND i.Total_Marks >=899 THEN
			INSERT INTO Result (STUD_NAME,CLASS) VALUES (i.Stud_Name,'Higher Second
Class');
		END IF;
	END LOOP;
	COMMIT;
END;
```

```
EXEC PROC_GRADE1;
```

```
SQL> EXEC PROC_GRADE1;
Student Name: Aishwarya Student Marks: 1400
Student Name: Priyanka Student Marks: 800
Student Name: Madhav Student Marks: 830
Student Name: Shruti Student Marks: 900
Student Name: Pritam Student Marks: 990
Student Name: Ram Student Marks: 300

PL/SQL procedure successfully completed.
```

```
select * from Result;
```

```
SQL> select * from Result;

STUD_NAME            CLASS
-------------------- --------------------
Aishwarya            Distinction
Priyanka             Fail
Madhav               Higher Second Class
Shruti               First Class
Pritam               Distinction
Ram                  Fail

6 rows selected.
```

Problem Statement: Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table Cust_New with the data available in the table Cust_Old. If the data in the first table already exist in the second table then that data should be skipped.

```
-- Create the tables

CREATE TABLE Cust_New (ID NUMBER, Name VARCHAR2(10), City VARCHAR2(10), Salary NUMBER);

CREATE TABLE Cust_Old (ID NUMBER, Name VARCHAR2(10), City VARCHAR2(10), Salary NUMBER);


-- Insert data into Cust_New

BEGIN

  INSERT INTO Cust_New VALUES (1, 'Ajay', 'Pune', 20000);

  INSERT INTO Cust_New VALUES (2, 'Ramesh', 'Pune', 15000);

  INSERT INTO Cust_New VALUES (3, 'Umesh', 'Pune', 40000);

  INSERT INTO Cust_New VALUES (4, 'Ram', 'Pune', 25000);

END;

/

select * from  Cust_New;
```

ID	NAME	CITY	SALARY
1	Ajay	Pune	20000
2	Ramesh	Pune	15000
3	Umesh	Pune	40000
4	Ram	Pune	25000

4 rows returned in 1.60 seconds

```
select * from  Cust_Old;
```

no data found

```
-- PL/SQL block to merge data from Cust_New into Cust_Old

DECLARE

  v_id NUMBER;

  v_name VARCHAR2(10);

  v_city VARCHAR2(10);
```

```
  v_salary NUMBER;
  v_exists NUMBER;

  CURSOR c_new_customers IS
    SELECT ID, Name, City, Salary
    FROM Cust_New;

BEGIN
  FOR cust_rec IN c_new_customers LOOP
    -- Check if the data already exists in Cust_Old
    SELECT COUNT(*) INTO v_exists
    FROM Cust_Old
    WHERE ID = cust_rec.ID
    AND Name = cust_rec.Name
    AND City = cust_rec.City
    AND Salary = cust_rec.Salary;

    -- If it doesn't exist, insert into Cust_Old
    IF v_exists = 0 THEN
      INSERT INTO Cust_Old (ID, Name, City, Salary)
      VALUES (cust_rec.ID, cust_rec.Name, cust_rec.City, cust_rec.Salary);
    END IF;
  END LOOP;
END;
/
```

```
1 row(s) inserted.
```

select * from Cust_Old;

ID	NAME	CITY	SALARY
1	Ajay	Pune	20000
2	Ramesh	Pune	15000
3	Umesh	Pune	40000
4	Ram	Pune	25000

4 rows returned in 0.00 seconds

Problem Statement: Database Trigger (All Types: Row level and Statement level triggers, 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 the Library table
CREATE TABLE Library (
    id NUMBER,
    name VARCHAR2(10),
    dept_no NUMBER
);

-- Insert values in table
begin
insert into Library values(1,'Aashi',1);
insert into Library values(2,'Nitin',15);
insert into Library values(3,'Nishi',8);
insert into Library values(4,'Aditi',7);
insert into Library values(5,'Roshni',1);
end;
/

select *from Library;
```

ID	NAME	DEPT_NO
1	Aashi	1
2	Nitin	15
3	Nishi	8
4	Aditi	7
5	Roshni	1

5 rows returned in 0.00 seconds

```
-- Create the Library_Audit table
CREATE TABLE Library_Audit (
    operation VARCHAR2(10),
    old_id NUMBER,
    old_name VARCHAR2(10),
    old_dept_no NUMBER
);
```

```
-- Create an AFTER DELETE trigger to log deleted records
CREATE OR REPLACE TRIGGER library_delete_trigger
AFTER DELETE ON Library
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (operation, old_id, old_name, old_dept_no)
    VALUES ('Delete', :old.id, :old.name, :old.dept_no);
END;
/

-- Create an AFTER UPDATE trigger to log updated records
CREATE OR REPLACE TRIGGER library_update_trigger
AFTER UPDATE ON Library
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (operation, old_id, old_name, old_dept_no)
    VALUES ('Update', :old.id, :old.name, :old.dept_no);
END;
/
```

Update Library set name='Aishu' where dept_no=7;

select *from Library_Audit;

OPERATION	OLD_ID	OLD_NAME	OLD_DEPT_NO
Update	4	Aditi	7

1 rows returned in 0.08 seconds CSV Export

delete from Library where id=3;

select *from Library_Audit;

OPERATION	OLD_ID	OLD_NAME	OLD_DEPT_NO
Update	4	Aditi	7
Delete	3	Nishi	8

2 rows returned in 0.00 seconds CSV Export

select *from Library;

ID	NAME	DEPT_NO
1	Aashi	1
2	Nitin	15
4	Aishu	7
5	Roshni	1

4 rows returned in 0.05 seconds

Problem Statement: Database Connectivity: Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

JDBC Statement:

```
import java.sql.*;
import java.util.Scanner;
import java.util.*;
import java.io.*;

public class JDBCPreparedStatementDemo
{
  static Connection con;
  static ResultSet rs;
  static PreparedStatement pstmt, pstmt1, pstmt2,pstmt3,pstmt4;
  static Scanner sc= new Scanner(System.in);

  public static void main(String args[])
  {
    JDBCPreparedStatementDemo obj = new JDBCPreparedStatementDemo();
    //Creating a scanner object
    String selQuery1="select * from customer";
    String selQuery2="insert into customer values(?,?,?)";
    String selQuery3="update customer set Name=? where id=?";
    String selQuery4="delete from customer where id=?";
    String selQuery5="select * from customer where id=?";
    int ch=1, op;

    try
    {
      //step1 load the driver class
      Class.forName("oracle.jdbc.driver.OracleDriver");

      //step2 create  the connection object
      con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","password");

      //step3 create the statement object
      pstmt=con.prepareStatement(selQuery1);
      pstmt1=con.prepareStatement(selQuery2);
      pstmt2=con.prepareStatement(selQuery3);
      pstmt3=con.prepareStatement(selQuery4);
      pstmt4=con.prepareStatement(selQuery5);

      do
      {
        System.out.println("\nWhich operation you want to do?");
        System.out.println("1. Display a table");
        System.out.println("2. Insert a record");
        System.out.println("3. Delete a record");
        System.out.println("4. Update a record");
        System.out.println("5. Search a record");
        System.out.println("\nEnter your Choice: ");
        ch = sc.nextInt();
        sc.nextLine();

        switch(ch){

          case 1:
```

```
        obj.display();
      break;

      case 2:
        obj.insert();
      break;

      case 3:
        obj.delete();
      break;

      case 4:
        obj.update();
      break;

      case 5:
        obj.search();
      break;

     } //End of switch

   }while(ch<5);

   //step5 close the connection object
   con.close();

  } //End of try block

  catch(Exception e)
  {
    System.out.println(e);
  }

} //End of main method
```

```
  public void display() throws SQLException
  {

     rs=pstmt.executeQuery();
     System.out.println("\nID\tName\tAmount");

     while(rs.next())
     {
       System.out.println(rs.getInt(1)+"
"+rs.getString(2)+"  "+rs.getInt(3));
     }
  } //End of display() method

  public void insert() throws SQLException
  {

     int id, amount;
     String name;
     System.out.println("Enter a id: ");
     id = sc.nextInt();
     sc.nextLine();
     System.out.println("Enter a Name: ");
     name=sc.nextLine();
     System.out.println("Enter a amount: ");
     amount = sc.nextInt();

     pstmt1.setInt(1,id);
     pstmt1.setString(2,name);
     pstmt1.setInt(3,amount);
     pstmt1.executeUpdate();

     System.out.println("Successfully insert a id
"+id);
  } //End of display() method

  public void update() throws SQLException
```

```
{
    int id;
    String name;
    System.out.println("Enter a id: ");
    id = sc.nextInt();
    sc.nextLine();
    System.out.println("Enter a Name: ");
    name=sc.nextLine();


    pstmt2.setString(1,name);
    pstmt2.setInt(2,id);
    pstmt2.executeUpdate();


    System.out.println("Successfully update a
name of id "+id);
 } //End of display() method


 public void delete() throws SQLException
 {
    int id;
    System.out.println("Enter a id: ");
    id = sc.nextInt();
    sc.nextLine();
    pstmt3.setInt(1,id);
    pstmt3.executeUpdate();


    System.out.println("Successfully delted id
"+id);
 } //End of display() method

public void search() throws SQLException
{

    int id;
    System.out.println("Enter a id: ");
    id = sc.nextInt();
    sc.nextLine();
    pstmt3.setInt(1,id);
    rs=pstmt.executeQuery();
    System.out.println("ID\tName\tAmount");


    if(rs.next()==true)
    {
      System.out.println(rs.getInt(1)+"
"+rs.getString(2)+"  "+rs.getInt(3));
    }
    else
    {
      System.out.println("Invalid ID");
    }
 } //End of display() method


}
```

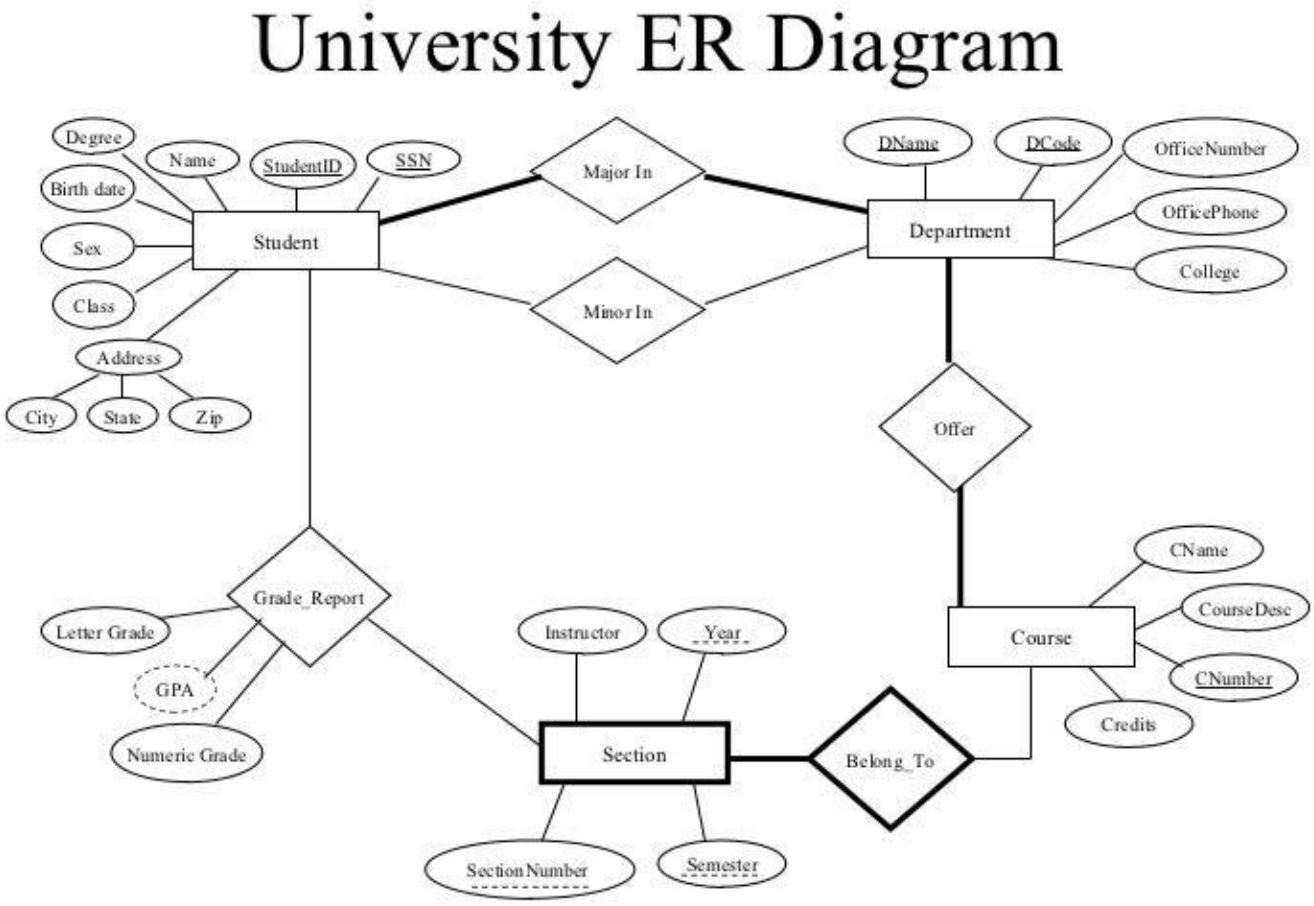
```
C:\Users\Ambition\Documents\Java Programming

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
4. Update a record
5. Search a record

Enter your Choice:
1

ID      Name    Amount

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
4. Update a record
5. Search a record

Enter your Choice:
2
Enter a id:
1
Enter a Name:
Aishu
Enter a amount:
200
Successfully insert a id 1

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
4. Update a record
5. Search a record

Enter your Choice:
2
```

```
Enter a id:
2
Enter a Name:
Shruti
Enter a amount:
400
Successfully insert a id 2

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
4. Update a record
5. Search a record

Enter your Choice:
1

ID      Name    Amount
1  Aishu  200
2  Shruti  400

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
4. Update a record
5. Search a record

Enter your Choice:
4
Enter a id:
2
Enter a Name:
Arjun
Successfully update a name of id 2

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
```

```
Enter your Choice:
1

ID      Name    Amount
1  Aishu  200
2  Arjun  400

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
4. Update a record
5. Search a record

Enter your Choice:
3
Enter a id:
2
Successfully delted id 2

Which operation you want to do?
1. Display a table
2. Insert a record
3. Delete a record
4. Update a record
5. Search a record

Enter your Choice:
1

ID      Name    Amount
1  Aishu  200
```

Problem Statement:

ER Modeling and Normalization: Decide a case study related to real time application in group of 2-3 students and formulate a problem statement for application to be developed. Propose a Conceptual Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities, generalization, specialization etc.) Convert the ER diagram into relational table's Relational data model.



Problem Stataement: Study of Open Source NOSQL Database: MongoDB (Installation, BasicCRUD operations, Execution)

---Creating and Inserting a Data in new Database

```
> show dbs
EventDB         0.014GB
FruitDB         0.000GB
StudentInfoDB   0.000GB
StudentLoginDB  0.000GB
admin           0.000GB
blogDB          0.000GB
blogsDB         0.000GB
config          0.000GB
fruitsDB        0.000GB
local           0.000GB
shopDB          0.000GB
todolistDB      0.000GB
userDB          0.000GB
> use blogDB
switched to db blogDB
> show Collection
uncaught exception: Error: don't know how to show [Collection] :
shellHelper.show@src/mongo/shell/utils.js:1211:11
shellHelper@src/mongo/shell/utils.js:838:15
@(shellhelp2):1:1
> show collections
logins
posts
> db.login.find()
> db.logins.find()
{ "_id" : ObjectId("640e9061f3b82e00cd750033"), "username" : "Aishwarya", "password" : "Aishu" }
> db.posts.find()
{ "_id" : ObjectId("61408413a14c8cd16e75cffb"), "title" : "Day2", "content" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.", "__v" : 0 }
```

{ "_id" : ObjectId("614088ccdf31ad3eb1015f0f"), "title" : "Day4", "content" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.", "__v" : 0 }

{ "_id" : ObjectId("652d77159361c61a7e92e319"), "title" : "online Daily Journal", "content" : "The online Daily Journal project aims to provide a user-friendly and convenient platform for a diverse community of writers, including bloggers, journal keepers, novelists, and storytellers. The central objective is to simplify the process of writing, storing, and managing thoughts and ideas without requiring any specialized technical knowledge. Leveraging a non-relational database, the system offers comprehensive blog management and publishing features. The application will be hosted online, accessible from any location and at any time, ensuring flexibility for writers. Most importantly, this platform offers its services free of charge, encouraging users to explore their creativity and document their imagination. The Daily Journal project seeks to democratize writing, making it accessible to all, and serves as a valuable tool for writers of all backgrounds.", "__v" : 0 }

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

--Create Collection

use StudentLoginDB

switched to db StudentLoginDB

> db.createCollection("StudentInfo")

{ "ok" : 1 }

> show collections

StudentInfo

--Insert

> db.StudentInfo.insert({_id:1, name:"Aishu "})

WriteResult({ "nInserted" : 1 })

> db.StudentInfo.insert({_id:2, name:"Priya "})

WriteResult({ "nInserted" : 1 })

> db.StudentInfo.insert({_id:3, name:"Snehal"})

WriteResult({ "nInserted" : 1 })

> db.StudentInfo.insert({_id:4, name:"Pritam"})

WriteResult({ "nInserted" : 1 })

--Find

> db.StudentInfo.find()

{ "_id" : 1, "name" : "Aishu " }

{ "_id" : 2, "name" : "Priya " }

{ "_id" : 3, "name" : "Snehal" }

{ "_id" : 4, "name" : "Pritam" }

--Update

> db.StudentInfo.update({_id:3}, {$set:{name:"Sita "}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.StudentInfo.find()

{ "_id" : 1, "name" : "Aishu " }

{ "_id" : 2, "name" : "Priya " }

{ "_id" : 3, "name" : "Sita " }

{ "_id" : 4, "name" : "Pritam" }

--Save

> db.StudentInfo.save({ "_id" : 4, "name" : "Gita" })

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.StudentInfo.find()

{ "_id" : 1, "name" : "Aishu " }

{ "_id" : 2, "name" : "Priya " }

{ "_id" : 3, "name" : "Sita " }

{ "_id" : 4, "name" : "Gita" }

--Delete

> db.StudentInfo.remove({ "_id" : 4 })

WriteResult({ "nRemoved" : 0 })

> db.StudentInfo.find()

{ "_id" : 1, "name" : "Aishu " }

{ "_id" : 2, "name" : "Priya " }

{ "_id" : 3, "name" : "Sita " }

LOGICAL OPERATORS:

AND:

> db.StudentInfo.find({ "_id" : 1, "name" : "Aishu " }).pretty()

{ "_id" : 1, "name" : "Aishu " }

OR

> db. StudentInfo.find({$or:[{ "_id" : 1},{"name" : "Aishu "}]}).pretty()

{ "_id" : 1, "name" : "Aishu " }

Problem Statement: MongoDB – Aggregation and Indexing: Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB.

```
> use StudentInfoDB
switched to db StudentInfoDB

> db.Student.find()
{ "_id" : 1, "name" : "Aishu ", "Marks" : 90 }
{ "_id" : 3, "name" : "Snehal", "Marks" : 93 }
{ "_id" : 4, "name" : "Pritam", "Marks" : 80 }
{ "_id" : 2, "name" : "Priya ", "Marks" : 68 }
```

Matching Documents ($match):

```
> db.Student.aggregate([
...     {
...         $match: { Marks: { $gte: 90 } }
...     }
... ])
{ "_id" : 1, "name" : "Aishu ", "Marks" : 90 }
{ "_id" : 3, "name" : "Snehal", "Marks" : 93 }
```

Grouping Documents ($group):

```
> db.Student.aggregate([
...     {
...         $group: {
...             _id: null,
...             totalMarks: { $sum: "$Marks" },
...             avgMarks: { $avg: "$Marks" },
...             minMarks: { $min: "$Marks" },
...             maxMarks: { $max: "$Marks" }
...         }
...     }
... ])
{ "_id" : null, "totalMarks" : 331, "avgMarks" : 82.75, "minMarks" : 68, "maxMarks" : 93 }
```

Counting the Number of Documents in a Group:

```
> db.Student.aggregate([
...     {
...         $group: {
```

```
...             _id: "$name",
...             count: { $sum: 1 }
...         }
...     }
... ])
{ "_id" : "Pritam", "count" : 1 }
{ "_id" : "Priya ", "count" : 1 }
{ "_id" : "Snehal", "count" : 1 }
{ "_id" : "Aishu ", "count" : 1 }
```

Sorting the Results ($sort):

```
> db.Student.aggregate([
...     {
...         $sort: { Marks: -1 }
...     }
... ])
{ "_id" : 3, "name" : "Snehal", "Marks" : 93 }
{ "_id" : 1, "name" : "Aishu ", "Marks" : 90 }
{ "_id" : 4, "name" : "Pritam", "Marks" : 80 }
{ "_id" : 2, "name" : "Priya ", "Marks" : 68 }
```

Limiting the Number of Results ($limit):

```
> db.Student.aggregate([
...     {
...         $limit: 3
...     }
... ])
{ "_id" : 1, "name" : "Aishu ", "Marks" : 90 }
{ "_id" : 3, "name" : "Snehal", "Marks" : 93 }
{ "_id" : 4, "name" : "Pritam", "Marks" : 80 }
```

$first and $last Operators:

```
> db.Student.aggregate([
...     {
...         $sort: { Marks: 1 } // Sort by marks in ascending order
...     },
...     {
...         $group: {
```

```
...             _id: null,
...             highestMarksStudent: { $last: "$name" },
...             highestMarks: { $last: "$Marks" },
...             lowestMarksStudent: { $first: "$name" },
...             lowestMarks: { $first: "$Marks" }
...         }
...     }
... ])
{ "_id" : null, "highestMarksStudent" : "Snehal", "highestMarks" : 93, "lowestMarksStudent" : "Priya ", "lowestMarks" : 68 }
```

$skip:

```
> db.Student.aggregate([
...     {
...         $skip: 2
...     }
... ])
{ "_id" : 4, "name" : "Pritam", "Marks" : 80 }
{ "_id" : 2, "name" : "Priya ", "Marks" : 68 }
```

Problem Statement: MongoDB – Map-reduces operations: Implement Map reduces operation with suitable example using MongoDB.

Calculate the Average Marks Using Map-Reduce:

```
> var mapFunction = function() {
...     emit('average', this.Marks);
... };
>
> var reduceFunction = function(key, values) {
...     return Array.sum(values) / values.length;
... };
>
> db.Student.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: { inline: 1 } }
... )
{
        "results" : [
                {
                        "_id" : "average",
                        "value" : 82.75
                }
        ],
        "ok" : 1
}
```

Calculate the Total Marks Using Map-Reduce:

```
> var mapFunction = function() {
...     emit('total', this.Marks);
... };
>
> var reduceFunction = function(key, values) {
...     return Array.sum(values);
... };
>
```

```
> db.Student.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: { inline: 1 } }
... )
{ "results" : [ { "_id" : "total", "value" : 331 } ], "ok" : 1 }
```

Calculating Average Marks per Student Name Using Map-Reduce:

```
> var mapFunction = function() {
...     emit(this.name, this.Marks);
... };
>
> var reduceFunction = function(key, values) {
...     return Array.sum(values) / values.length;
... };
>
> db.Student.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: { inline: 1 } }
... )
{
        "results" : [
                {
                        "_id" : "Priya ",
                        "value" : 68
                },
                {
                        "_id" : "Pritam",
                        "value" : 80
                },
                {
                        "_id" : "Aishu ",
                        "value" : 90
                },
```

```
        {
            "_id" : "Snehal",
            "value" : 93
        }
    ],
    "ok" : 1
}
```

Problem Statement: Database Connectivity: Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations(add, delete, edit etc.)

Dependencies:

```
const express = require("express");

const bodyParser = require("body-parser");

const ejs = require("ejs");

const _ = require("lodash");

const mongoose = require('mongoose');

const alert = require('alert');

const notifier = require('node-notifier');
```

Login:

```
app.post("/login", function(req, res) {
 var login_name=req.body.login_button;
 Login.findOne({
  username: req.body.userName,
  password: req.body.passWord
 }, function(err, login) {
  if (!err) {
   if (login) {
    notifier.notify('Login Successful !!!!!!');


    if (login_name==="delete") {
     deleteFun();
     res.redirect("/");
    } else {
     res.render(login_name);
    }
```

```
    } else {
      notifier.notify({
        title: 'Login Unsuccessful !!!!!!!',
        message: 'Kindly enter correct username and password'
      });

      if (login_name==="delete") {
        res.redirect("/");
        // res.redirect("/posts/"+requestedPostId1);
      } else {
        res.redirect("/compose");
      }
    }
  }
 })
});
```

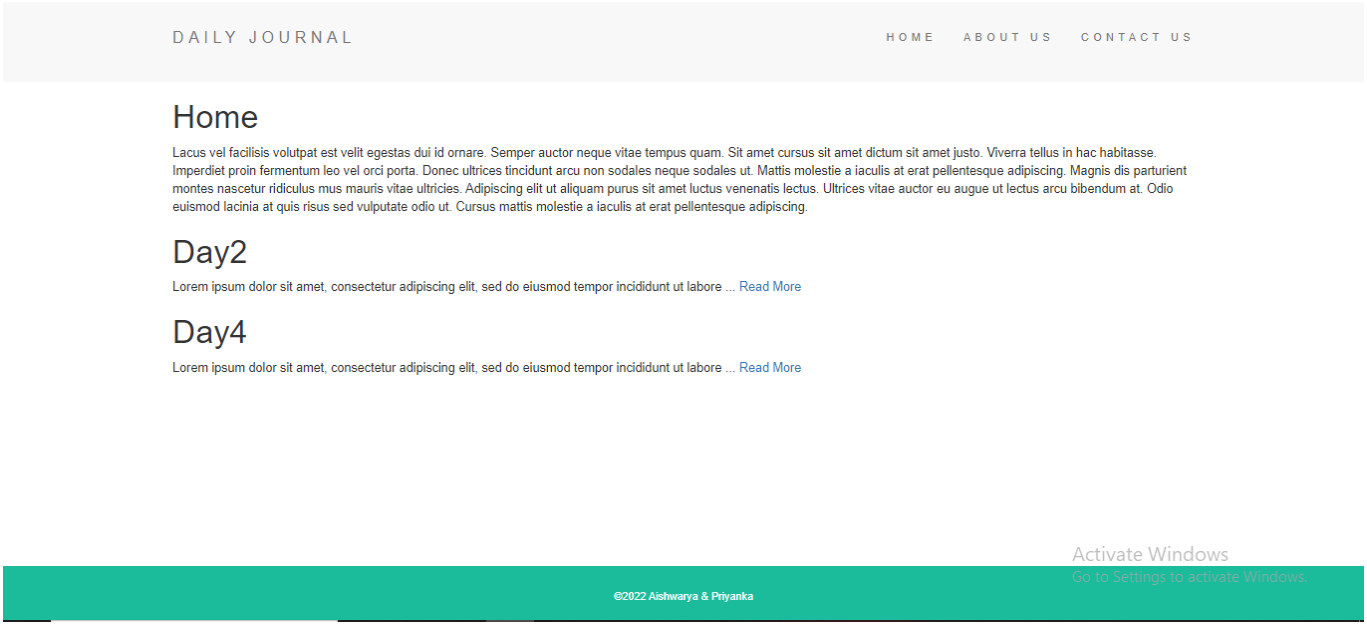
Publish Blog:

```
app.get("/posts/:postID", function(req, res) {
 const requestedTitle = req.params.postID;

 Post.findOne({
  _id: requestedTitle
 }, function(err, post) {

  if (!err) {
```

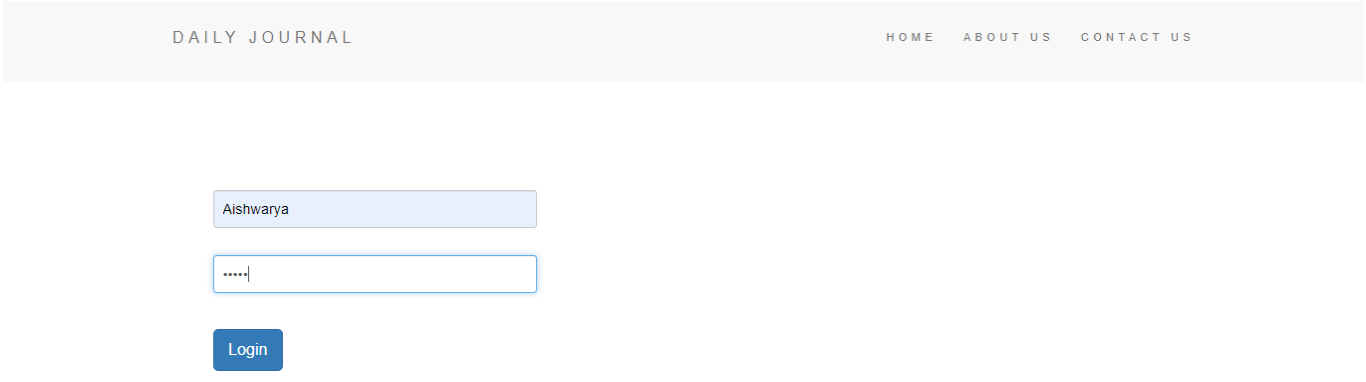
```
    res.render("post", {
      title: post.title,
      content: post.content,
      postId: requestedTitle
    });
  }
 });
});
```

SCREENSHOT

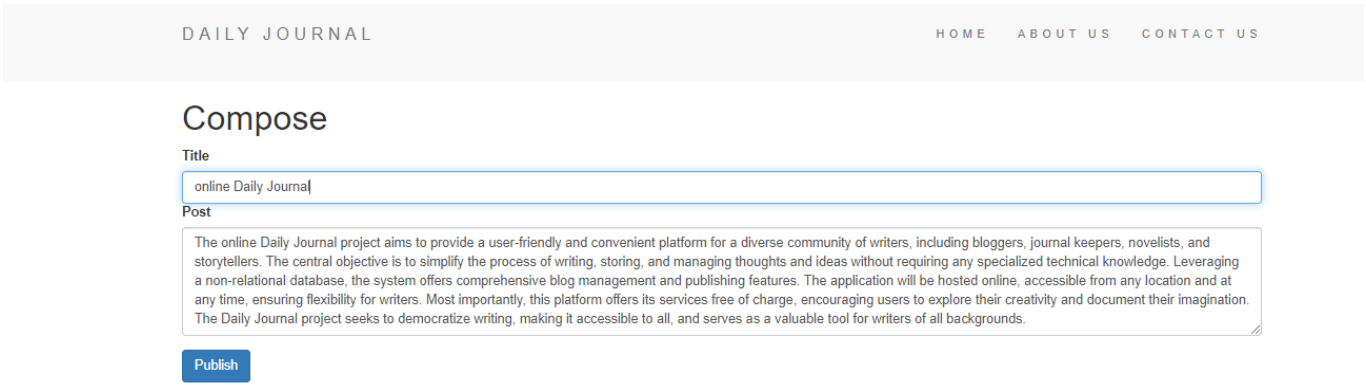
Home :



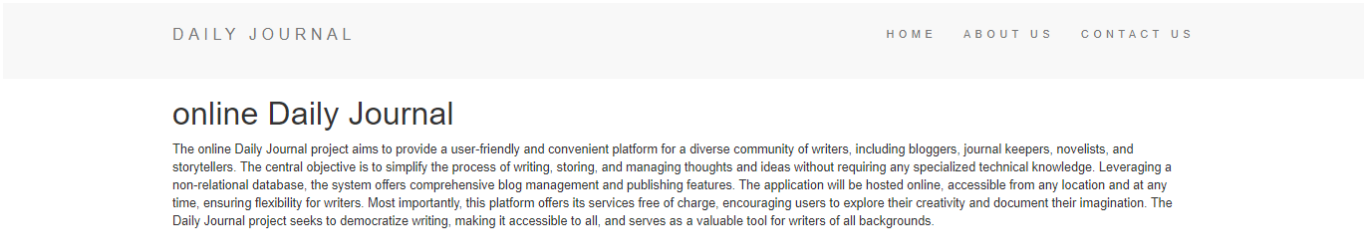
Login:



Publish Blog:



View Full Blog:



Delete Blog: