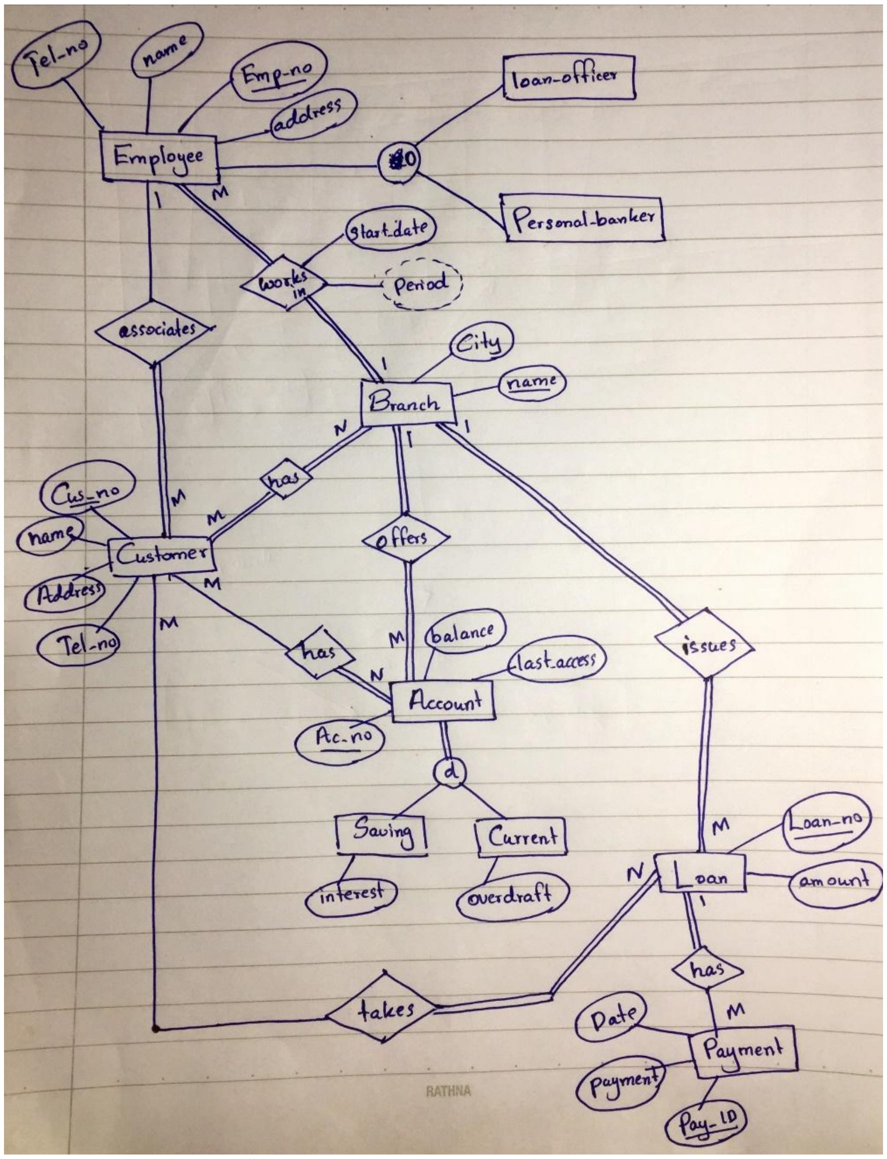
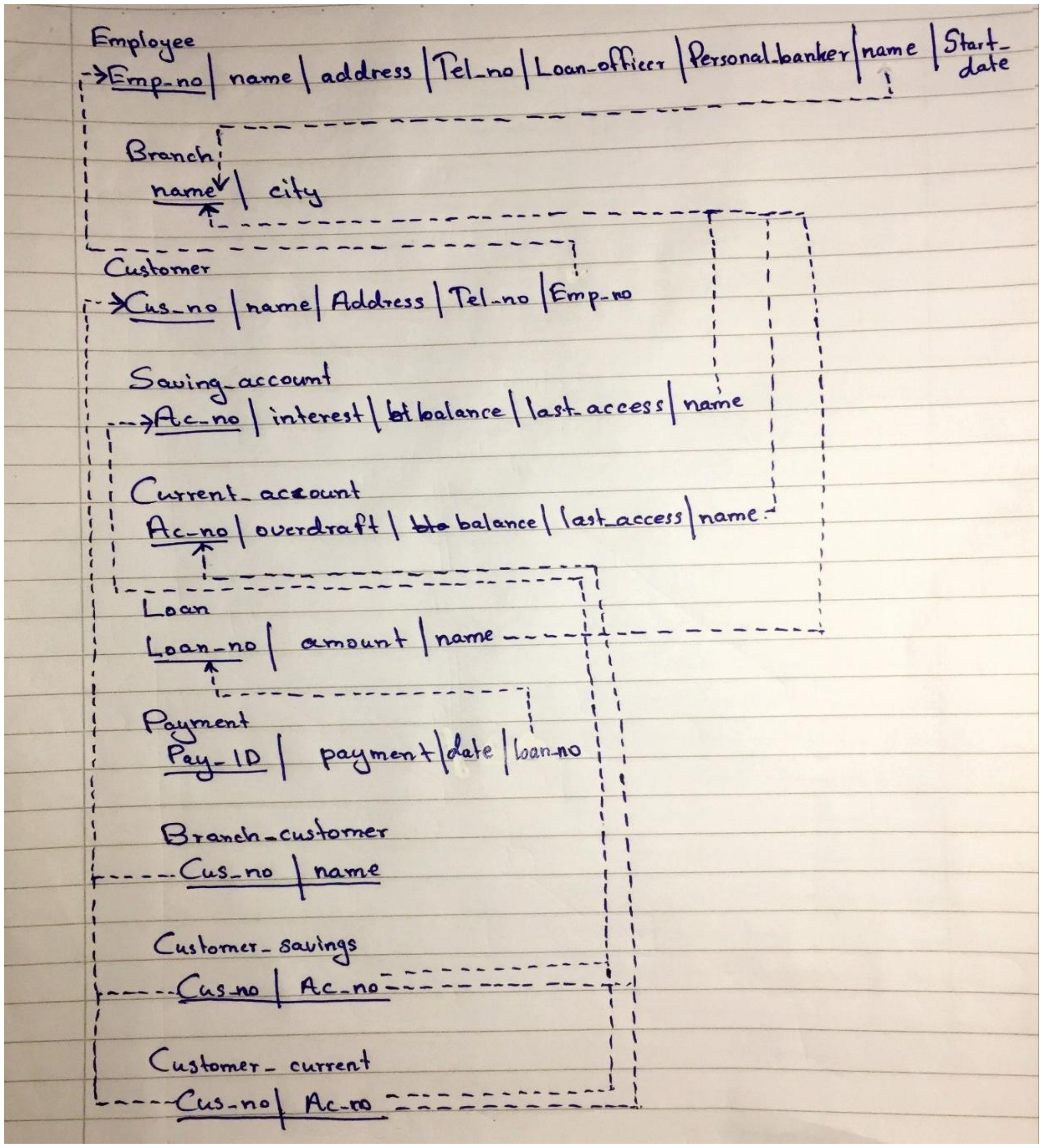
ADBMS sample paper answers

1) A)



Assumptions:

- Each employee is allocated to only one branch
- There may be employees other than loan officers and personal bankers and a single employee can work as both as a loan officer and a personal banker.
- A customer is associated by only one employee at a moment.
- One customer can be linked with more than one branch.
- There are no accounts in a branch other than savings and current accounts
- There is a payment ID to uniquely identify each payment.


Employee
Emp-no | name | address | Tel-no | Loan-officer | Personal-banker | name | Start-date
Branch
name | city
Customer
Cus-no | name | Address | Tel-no | Emp-no
Saving-account
Ac-no | interest | balance | last-access | name
Current-account
Ac-no | overdraft | balance | last-access | name
Loan
Loan-no | amount | name
Payment
Pay-ID | payment | date | loan-no
Branch-customer
Cus-no | name
Customer-savings
Cus-no | Ac-no
Customer-current
Cus-no | Ac-no

2) A)

A view is a virtual which contains rows and columns that belong to one or more real tables. The index of a view is determined by a result-set of an SQL statement.

B) ***General format :***

CREATE VIEW *view_name*
AS
SELECT *column1, column2….*
FROM *table_name(s)*

```
WHERE condition(s);
```

Example:

- Cement_stockin(stock_id, quantity, date, Supplier_id)
- Supplier (Supplier_id, Supplier_name, Address)

Creating a view including quantity, date columns from Cement_stockin table and Supplier_name from Supplier table.

```
CREATE VIEW Stock_info
AS
SELECT Supplier_name, quantity, date
FROM Cement_stockin, Supplier
WHERE Cement_stockin.Supplier_id = Supplier.Supplier_id;
```

C)

```
CREATE VIEW emp_contact
AS
SELECT emp_name, telephone, email
FROM employee_info;
```

3) A)

I)

```
SELECT COUNT(DISTINCT participated.driver_id) FROM participated, accident
WHERE accident.report_number = participated.report_number AND
accident.date = 2004;
```

OR

```
SELECT COUNT( DISTINCT participated.driver_id) FROM participated INNER JOIN
accident ON participated.report_number = accident.report_number WHERE
accident.date= 2004;
```

OR

```
SELECT COUNT( DISTINCT participated.driver_id) FROM participated INNER JOIN
accident ON participated.report_number = accident.report_number AND
accident.date= 2004;
```

OR

```
SELECT COUNT(DISTINCT participated.driver_id) FROM participated WHERE
report_number IN (SELECT report_number FROM accident WHERE date = 2004);
```

II)

```
SELECT COUNT(DISTINCT participated.report_number) FROM participated,
person WHERE participated.driver_id = person.driver_id AND person.name =
‘Tharaka’;
```

OR

```
SELECT COUNT (DISTINCT participated.report_number) FROM participated
INNER JOIN person ON participated.driver_id = person.driver_id WHERE
person.name= ‘Tharaka’;
```

OR

```
SELECT COUNT (DISTINCT participated.report_number) FROM participated
INNER JOIN person ON participated.driver_id = person.driver_id AND
person.name= ‘Tharaka’;
```

OR

```
SELECT COUNT( DISTINCT participated.report_number) FROM participated
WHERE driver_id = (SELECT driver_id FROM person WHERE name = ‘Tharaka’);
```

III)

```
DELETE FROM car WHERE model = ‘Mazda’ AND license IN
        (SELECT license FROM owns WHERE driver_id =
                (SELECT driver-id FROM person
                WHERE name = ‘S shan’));
```

OR

```
DELETE FROM car WHERE license = (SELECT car.license FROM own
        INNER JOIN car ON car.license = owns.license
        INNER JOIN person ON own.driver_id = person.driver_id
        WHERE person.name =’S khan’ AND car.model = ‘Mazda’);
```

B)

I)

```
SELECT SHOW.Artist, THEATRE.City FROM SHOW,THEATRE
        WHERE SHOW.Hall = THEATRE.Name
        AND SHOW.Attendance >= 5000;
```

OR

```
              SELECT SHOW.Artist, THEATRE.City FROM SHOW
        INNER JOIN THEATRE ON SHOW.Hall = THEATRE.Name
        WHERE SHOW.Attendance >=5000;
```

II)
```
SELECT DISTINCT CITY.State FROM THEATRE
        INNER JOIN CITY ON THEATRE.City = CITY.Name
        INNER JOIN SHOW ON THEATRE.Name = SHOW.Hall
        WHERE SHOW.Artist = ‘Mr. X’ AND CITY.Country = ‘India’;
```

III)
```
SELECT DISTINCT SHOW.Artist FROM SHOW,THEATRE
        WHERE SHOW.Hall = THEATRE.Name
        AND City NOT IN(‘Colombo’);
```
OR

```
SELECT DISTINCT SHOW.Artist FROM SHOW
        INNER JOIN THEATRE ON SHOW.Hall = THEATRE.Name
        WHERE NOT THEATRE.City = ‘Colombo’;
```

IV)
```
SELECT name FROM THEATRE WHERE City = ‘Kandy’ AND Capacity > 5000;
```

4) a)
```
SET SERVERPUTPUT ON
SET ECHO ON

CREATE PROCEDURE ShowRecord( studntNo IN NUMBER)
AS
        Student_details Student%ROWTYPE;
BEGIN
        SELECT * INTO Student_details FROM Student
        WHERE SNO= studntNo;
        DBMS_OUTPUT.PUT_LINE(‘SNO : ’||Student_details.SNO ||’ Name: ’
        ||Student_details.Name||’ Marks : ’||Student_details.Marks);
END;
/
```

OR
```
SET SERVERPUTPUT ON
SET ECHO ON

CREATE PROCEDURE ShowRecord( studntNo IN NUMBER)
AS
```

```
        StudentNO Student.SNO%TYPE;
        SName Student.Name%TYPE;
        Mark Student.Marks%TYPE;
    BEGIN
        SELECT SNO, Name, Mark INTO StudentNO, SName, Mark FROM Student
        WHERE SNO= studntNo;
        DBMS_OUTPUT.PUT_LINE(‘SNO : ’||StudentNO ||’ Name: ’ ||SName||’
        Marks : ’||Mark);
    END;
    /
```

b)

```
    CREATE FUNCTION calMarkAvg
    RETURN NUMBER
    AS
        Average NUMBER;
    BEGIN
        SELECT AVG(Marks) INTO Average FROM Student;
        RETURN Average;
    END;
    /
```

c)

```
    CREATE PACKAGE pkgStudent
    AS
        FUNCTION deleteRecord(sno IN NUMBER) RETURN NUMBER;
        PROCEDURE insertRecord (sno Student.SNO%TYPE, name
        Student.Name%TYPE, marks Student.Marks%TYPE);
    END;
    /

    CREATE PACKAGE BODY pkgStudent
    AS
        FUNCTION deleteRecord(sno IN NUMBER)
        RETURN NUMBER
        AS
        BEGIN
            DELETE FROM Student WHERE SNO = sno;
            RETURN 1;
        END;

        PROCEDURE insertRecord
```

```
                (sno Student.SNO%TYPE,
                name Student.Name%TYPE,
                marks Student.Marks%TYPE)
        AS
        BEGIN
                INSERT INTO Student VALUES(sno, name, marks);
        END;
END;
/
```

d)

```
SET SERVEROUTPUT ON
SET ECHO ON

CREATE TRIGGER display_message
AFTER UPDATE ON Students
BEGIN
        DBMS_OUTPUT.PUT_LINE(‘Update successful !’);
END;
/
```