**Project Title: Loan Management System**

Team members:

1. Name: Suraj Paswan

Enrollment Number: 2002106

1. Name: Siddhant Waghmare

Enrollment Number: 2000933

1. Name: Abhishek Pawde

Enrollment Number: 2002230

**Problem Description**

**Issue of the system before the digital era**

All the accounts were maintained in hard copy through ledgers and the transactions were limited to stand alone branches.

The customers had to go to the branch holding the account to conduct transactions.

If a person wants to take the loan from the bank. first he has to open the new account in the bank. Before the digital era the process for opening the bank account was more difficult we have to provide number of identity proofs, address proof, declaration forms , voter id , pan card , tax receipt and the signature of an account holder of that bank. after that they will verify the documents and issue the account number the process generally take more than 15 days

For the process of the loan He has to write an application to the bank for the purpose of the loan then the bank gives the number of forms for filling after the account holder fills the form the bank tells them to submit the eligible documents

* 2 Years of IT return
* address prrof
* identity proof
* salary slip
* Annual income
* No objection Certificate

After Submiting the eligible documents the bank asks to get signature of three guarantor and nominee

after submitting all eligible documents the file is sent to processing centre where it take more than a month for processing. In that period if the error is found the process gets postpone by more than 15 days. sometimes it may take more than 2 months because of human error.

So the process of getting loan was more difficult than the current system.

**Improved Condition:**

The Customers who want a loan from the bank can go to the respective office for the enqiury of the loan, by knowing the terms and conditions the customer can fill the form and submit to the respective bank.

The eligible documents for the purposed loan is:

* 2 Years of IT return
* address prrof
* identity proof
* salary slip
* Annual income

Now, after the customers submit the eligible documents to the bank. The bank now starts the scrutiny of the documents which usually takes 1-2 days. After the scrutiny of documents the bank sends the soft copy of all forms and documents to the head office for verification.

The Hard copy of documents usually takes 10-15 days to reach the head office . Due to the hard copy takes 10-15 days, by the soft copy of documents the banks sends officials to the respective address of the customers for verification of documents. The officials asked to show Original Id proof and Address proof of customers and the purpose of taking the loan.

They all verify and submit the information on their website. Both verificatins takes 10-15 days of time. by that time the hard copy reach the head office and the officials starts verification of hard copy by successful verification of documents . they send the file to the sanction desk which usually takes 1-2 days

After that they call customer to the head office for verification. by 2- 3 days by successfully completing verification of documents and ID proofs. They release the money to the account of customer.

This Current system takes 1-2 months to sanction the loan by less paperwork and most of the work done in the online mode

**STUDY OF EXISTING SYSTEM:**

**LOAN MANAGEMENT SYSTEM PROJECT**

       Here the existing system is a manual one using which the banking agent can’t maintain the effectively by sharing across different branches with proper security and can’t track details easily. It doesn’t provide proper co-ordination between different departments of the company. It doesn’t allow the customer to check the status of his file in proper way which leads customer dis-satisfaction.

**DRAWBACKS OF EXISTING SYSTEM:**

* Doesn’t provide faster and effective system
* Doesn’t provide good co-ordination between departments
* Doesn’t provide effective forwarding system to move the file from one level to another
* Doesn’t user-friendly interface
* Difficulty in generating different reports as per the business requirement
* Doesn’t facilitate the services from online

**SCOPE OF THE PROPOSED SYSTEM:**

The online automated system with web-based architecture can support issues like.

* This system maintains the information related different departments and stored at a central DB, which leads easy accessibility and consistency
* Interest rates of different banks and the other details are also available at the click of a mouse.
* Customer can apply for a loan and track his file details from online.
* The decision process in faster and more consistent
* Provides good communication between two departments
* Provides a facility to generate the reports very easily.

**IDENTIFY END USER OF THE FUCNTION:**

Users:

The major functionality of this product is divided into Four categories.

                 1. Administrative User Functions.

                 2. PickUp Dept User Functions.

                 3. Verification Dept User Functions.

4**.** Customer Functions

**Administrative User Functions:** Administrators can perform the following task.

* Create/Update/Delete New Banks Info
* View the list of banks
* Create/Update/Delete New Departments.
* View List of Departments
* Create/Update/Delete Employee Info
* View the list of Banks
* Manage loans info related to different banks
* View the list applications and assign it to pickup dept employees
* Sending messages to customers
* View the pending applications
* Update the status of the application
* Generate reports

**PickUp Dept User Functions:** This user can perform the following task

* View the list of applications assigned to him
* Store the list of documents info that the customer has submitted
* Forward the application to verification

**Verification Dept User Functions:** This user can perform the following task

* View the list applications forwarded by different employees of PickUp Dept
* Verify the details
* Forward applications to Administrator

**Customer Functions:** This user can perform the following task

* View the list of banks and their interest rates info
* Apply for loan in a bank
* Check the loan application status
* Messages

**Input Data to the System.**

A Database consist of Entities and for Creating a Entity in the database we give the following Input to the System.

Entity 1:Bank

Create table bank(Bank\_id int primary key,Bank\_Name varchar(12),Branch varchar(12),Address varchar(15));

Insert into bank values(‘801’,’SBI’,’Deccan’,’Deccan rd’);

Insert into bank values(‘802’,’HDFC’,’Main Branch’,’Sadashiv Peth’);

Insert into bank values(‘803’,’Axis Bank’,’Aundh’,’MG Road camp’);

Insert into bank values(‘804’,’Canara Bank’,’Shivajinagar’,’FC Rd’);

Insert into bank values(‘805’,’BOI’,’Hinjewadi’,’Phase-2 Pune’);

Insert into bank values(‘806’,’HDFC Bank’,’Kothrud’,’Karve Nagar’);

Entity 2:Debtor

Create table Debtor(Debtor\_Id int Primary key,Fname varchar(25),lname varchar(26),Address text,Age integer,Gender varchar(25));

Insert into Debtor values(‘23476’,’Rajesh’,’Singh’,’Ganeshkhind’,’35’,M);

Insert into Debtor values(‘23466’,’Rahul’,’Rajput’,’Aundh’,’32’,M);

Insert into Debtor values(‘23478’,’Vipul’,’Singh’,’MG Road’,’36’,M);

Insert into Debtor values(‘23475’,’Vignesh’,’Patil’,’Pimpri’,’33’,M);

Insert into Debtor values(‘23486’,’Vishal’,’Nikam’,’Sangvi’,’40’,M);

Insert into Debtor values(‘23468’,’Pravin’,’Kardak’,’Sangvi’,’44’,M);

Insert into Debtor values(‘23477’,’Samiksha’,’Kapoor’,’Kothrud’,’26’,F);

Insert into Debtor values(‘23488’,’Priyank’,’Pawar’,’Baner’,’28’,F);

Insert into Debtor values(‘23489’,’Sanyukta’,’Sharma’,’Pashan’,’30’,F);

Entity 3:Loan\_Offers

Create table Loan\_offers(Loan\_id int primary key,loan\_name varchar(25),Amount varchar(25),Loan\_range text,Interest varchar(25));

CREATE TABLE

Insert into Loan\_offers values(1023,’HOME LOAN’,’2400000’,’1000000 to 6000000’,’8 PERCENT’);

Insert into Loan\_offers values(1223,’HOME LOAN’,’3400000’,’1000000 to 6000000’,’10 PERCENT’);

Insert into Loan\_offers values(1123,’CAR LOAN’,’1000000’,’1000000 to 4000000’,’7 PERCENT’);

Insert into Loan\_offers values(1043,’PERSONAL LOAN’,’2400000’,’500000 to 3000000’,’12 PERCENT’);

Insert into Loan\_offers values(1323,’BIKE

LOAN’,’240000’,’50000 to 600000’,’12 PERCENT’);

Insert into Loan\_offers values(1423,’EDUCATION LOAN’,’750000’,’100000 to 8000000’,’8 PERCENT’);

Insert into Loan\_offers values(1523,’EDUCATION LOAN’,’600000’,’100000 to 8000000’,’7 PERCENT’);

Insert into Loan\_offers values(1723, EDUCATION LOAN’,’517000’,’1000000 to 8000000’,’6 PERCENT’);

Entity 4:Loan\_Information

Create table Loan\_Information(Info\_Id int Primary key,Debtor\_Id references Debtor(Debtor\_Id),Loan\_given\_date date,Loan\_Due\_date date,Monthly\_Payment Integer,Loan\_Amt money,Purpose varchar(30),Collateral varchar(25));

CREATE TABLE

Insert into Loan\_Information values(4243,’23486’,’2018-10-18,’2032-01-15’,’6000’,’750000’,’Gold Loan’,’Payment Slip/Vehicle’);

Insert into Loan\_Information values(4315,’23489’,’2016-11-20,’2024-03-11’,’8000’,’500000’,’Business Loan’,’Aparment’);

Insert into Loan\_Information values(4253,’23477’,’2019-12-07,’2030-07-04’,’16000’,’1500000’,’Education Loan’,’Gold’);

Insert into Loan\_Information values(4263,’23488’,’2021-02-10,’2040-02-29’,’12000’,’2500000’,’Education Loan’,’Non Agricultural Land’);

Insert into Loan\_Information values(4213,’23476’,’2021-02-04,’2043-01-12’,’16000’,’3800000’,’Home Rennovation Loan’,’Payment Slip’);

Insert into Loan\_Information values(4313,’23466’,’2020-09-14,’2035-05-31’,’28333’,’4200000’,’Home Loan’,’Property Papers’);

Insert into Loan\_Information values(4223,’23475’,’2021-10-04,’2040-08-12’,’24000’,’500000’,’Home Loan’, ’Property Papers’);

Insert into Loan\_Information values(4233,’23478’,’2020-03-08,’2034-04-10’,’5833’,’800000’,’Personal Loan’, ’Gold Jewellery’);

Entity 5:Loan\_Transaction

Create Table Loan\_transaction(Transaction\_Id int primary key,Debtor\_id references Debtor(Debtor\_Id),Date date,Time time without time zone,Loan\_amt\_approved money,bank\_name varchar(20),Branch varchar(20),Account\_no bigint);

CREATE TABLE

Insert into Loan\_Transaction values(3223,’23466’,’2021-10-11’,’20:12:00’,’750000’,’BOI’,’Shivajinagar’,’10234265241’);

Insert into Loan\_Transaction values(3023,’23476’,’2021-10-01’,’07:30:45’,’800000’,’HDFC Bank’,’Kothrud’,’10223465211’);

Insert into Loan\_Transaction values(3123,’23478’,’2021-10-03’,’15:24:40’,’380000’,’HDFC Bank’,’Deccan Rd’,’20123415241’);

Insert into Loan\_Transaction values(3243,’23475’,’2021-10-03’,’07:30:45’,’500000’,’SBI’,’M.G Road’,’20023455233’);

Insert into Loan\_Transaction values(3323,’23486’,’2021-10-04’,’07:30:45’,’4200000’,’Axis bank’,’Aundh’,’10023445212’);

Insert into Loan\_Transaction values(3553,’23489’,’2020-11-12’,’12:56:32’,’1500000’,’BOI’,’Main Branch’,’40123442141’);

Insert into Loan\_Transaction values(3523,’23488’,’2021-10-08’,’21:42:12’,’2500000’,’SBI’,’Hinjewadi’,’32123212121’);

Insert into Loan\_Transaction values(3423,’23477’,’2021-10-07’,’16:34:24’,’800000’,’HDFC’,’Deccan’,’15514121232’);

Entity 6:Loan\_Report

Create table Report(Report\_id int Primary key,Debtor\_id references Debtor(Debtor\_Id), Info\_Id references Loan\_Information(Info\_Id),Transaction\_Id references Loan\_transaction(Transaction\_id),Date date);

CREATE TABLE

Insert into Report values(401,’23466’,’4243’,’3123’);

Insert into Report values(402,’23475’,’4253’,’3243’);

Insert into Report values(405,’23477’,’4223’,’3523’);

Insert into Report values(406,’23466’,’4213’,’3423’);

Insert into Report values(407,’23489’,’4263’,’3023’);

Insert into Report values(408,’23466’,’4223’,’3223’);

Insert into Report values(403,’23488’,’4213’,’3323’);

Insert into Report values(404,’23477’,’4243’,’3553’);

**OUTPUT FROM THE SYSTEM:**

**1st Entity Output from The System:**

postgres=# select \* from Bank;

bank\_id | bank\_name | branch | address

---------+-------------------+-------------------+---------------

802 | HDFC | Main Branch | Sadashiv Peth

803 | Axis Bank | Aundh | MG Road Camp

804 | Canara Bank | Shivajinagar | FC Road

805 | BOI | Hinjewadi | Phase-2 Pune

801 | SBI | Deccan | Deccan Rd

806 | HDFC Bank | Kothrud | Karve Nagar

(6 rows)

**2nd  Entity Output from The System:**

postgres=# select \* from Debtor;

debtor\_id | fname | lname | address | age | gender

-----------+----------+--------+------------------------+-----+--------

23466 | RAHUL | RAJPUT | AUNDH | 32 | M

23478 | VIPUL | SINGH | MG ROAD | 36 | M

23475 | VIGNESH | PATIL | PIMPRI | 33 | M

23486 | VISHAL | NIKAM | SANGVI | 40 | M

23468 | PRAVIN | KARDAK | SANGAVI | 44 | M

23477 | SAMIKSHA | KAPOOR | KOTHRUD| 26 | F

23488 | PRIYANKA | PAWAR | BANER | 28 | F

23470 | RAJESH | SINGH | GANESHKHIND| 35 | M

23489 | SANYUKTA | SHARMA | PASHAN | 30 | F

(9 rows)

**3rd Entity Output from The System:**

postgres=# select \* from Loan\_offers;

loan\_id | loan\_name | amount | loan\_range | interest

-----------+----------------------+-------------+-----------------------+------------

1023 | HOME LOAN | 2400000 | 1000000 TO 6000000 | 8 PERCENT

1223 | HOME LOAN | 3400000 | 1000000 TO 6000000 | 10 PERCENT

1123 | CAR LOAN | 1000000 | 100000 TO 4000000 | 7 PERCENT

1243 | PERSONAL LOAN | 2400000 | 50000 TO 3000000 | 12 PERCENT

1323 | BIKE LOAN | 240000 | 50000 TO 600000 | 5 PERCENT

1423 | EDUCATION LOAN | 750000 | 100000 TO 8000000 | 8 PERCENT

1523 | EDUCATION LOAN | 600000 | 100000 TO 8000000 | 7 PERCENT

1723 | EDUCATION LOAN | 517000 | 1000000 TO 8000000 | 6 PERCENT

(8 rows)

**4th Entity Output from The System:**

postgres=# select \* from Loan\_Information;

info\_id |debtor\_id |loan\_given\_date| loan\_due\_date|monthly\_payment

|loan\_amt\_approved|purpose|collateral

--------+------------+------------------+-----------------+-----------------+------------------+---------------+------------------

4243 | 23486 | 2018-10-18 | 2032-01-15 | 6000 | $750,000.00| Gold Loan | Payment Slip/vehicle

4315 | 23489 | 2016-11-20 | 2024-03-11 | 8000 | $500,000.00 | Business Loan | Apartment

4253 |23477 | 2019-12-07 | 2030-07-04 | 16000 | $1,500,000.00 | Education Loan | Gold

4263|23488 | 2021-02-10 | 2040-02-29| 12000 | $2,500,000.00 |Education Loan |Non Agricultural

Land 4313 |23476|2021-02-04 | 2035-05-31 |28333 | $3,800,000.00 | Home Renovation Loan| Property Papers

4223 | 23466 | 2020-09-14 | 2040-08-12 |24000 |$4,200,000.00|Home Loan| Property Papers

4233 |23475 |2021-10-04 |2034-04-10 | 5833 | $5,000,000.00 | Home loan| Gold Jewellery

4213 |23478 |2020-03-08 | 2043-01-12 | 16000 | $800,000.00 | Personal Loan | Payment slip

(8 rows)

**5th Entity Output from The System:**

postgres=# select \* from loan\_transaction;

transaction\_id|debtor\_id|date |time| loan\_amt\_approved |bank\_name|branch | account\_no

-----------+-------------+--------------+-------------+------------------------+-------------------+--------------+-------------

3123|23478 |2021-10-02| 15:24:40 | $3,800,000.00|HDFC Bank|Deccan Rd | 20123415241

3243 |23475|2021-10-04|07:30:45|$5,000,000.00 | SBI | Main branch | 20023455233

3323 |23486|2018-10-18| 07:30:45 |$4,200,000.00 | Axis bank | Aundh | 10023445212

3553| 23489|2016-11-20|12:56:32|$1,500,000.00 | BOI |Main branch| 40123442141

3523 |23488|2021-02-10|21:42:12|$2,500,000.00 | SBI | Hinjewadi | 32123212121

3423| 23477|2019-12-07|16:34:24| $800,000.00| HDFC | Deccan | 15514121232

3223|23477|2020-09-14|20:12:00 | $750,000.00 | BOI |Shivajinagar|10234265241

3023|23486|2021-02-14|07:30:45 |$800,000.00|HDFC Bank| Kothrud | 10223465211

(8 rows)

**6th Entity Output from The System:**

postgres=# select \* from loan\_Report;

report\_id | debtor\_id | info\_id | transaction\_id

-----------+-----------+---------+----------------

401 | 23466 | 4243 | 3123

402 | 23475 | 4253 | 3243

403 | 23488 | 4213 | 3323

404 | 23486 | 4243 | 3553

405 | 23477 | 4223 | 3523

406 | 23466 | 4213 | 3423

407 | 23489 | 4263 | 3023

408 | 23466 | 4223 | 3223

(8 rows)

**FUNCTIONAL OR PROCESSING REQUIREMENT OF THE SYSTEM:**

For Execution of above Database case study,the System requires windows 10 pro Operating System with 4GB RAM and AMD E1 Processor.The Space required is 446MB for the PgAdmin 4 Software for Database Creation having version 5.4 and 782MB for the PostgreSQL 13.

**ENTITY RELATIONSHIP MODELING**

**Identify the entities & the attributes.**

**1.)Entity Name:Bank**

**Attributes:Bank Id,Bank Name,Branch,Address.**

**2.)Entity Name:Debtors**

**Attributes-Debtor Id,Fname,Lname,Address,Age,Gender.**

**3.)Entity Name-Loan Offers**

**Attributes-Loan Id,Loan Name, Amount, Loan Range, Interest.**

**4.)Entity Name-Loan Transaction**

**Attributes-Transaction Id,Debtor\_Id, Date,Loan\_amt\_approved,Bank\_name,Branch,Account\_no.**

**5.)Entity Name-Loan Information**

**Attributes-Info Id, Debtor Id,Loan given\_date,Loan Due, Monthly\_payment,loan\_amt\_approved, ,Purpose,Collatrel.**

**6.)Entity Name-Loan\_Report**

**Attributes-Report Id,Debtor Id, Info\_Id,Transaction\_Id.**

**ENTITY RELATIONSHIP DIAGRAM**

**Interest**

**Loan Range**

**Amount**

**Loan Name**

**Loan\_ID**

**Loan\_Offers**

**1**

**M**

**Branch**

**Bank\_Id**

**Bank**

**Address**

**Bank Name**

**1**

**Debtor Id**

**Debtors Id**

**Transaction Id**

**M**

**Info Id**

**1**  **1**

**Debtor**

**1 M**

**Debtor Id**

**Fname**

**Loan\_Transaction**

**Loan\_Information**

**Lname**

**Loan date**

**Age**

**Date**

**Loan Due**

**Time**

**Address**

**Gender**

**Bank Name**

**Monthly Pytm**

**Loan Amt**

**Account No**

**Loan Amt**

**Branch**

**Purpose**

**Collateral**

**M N**

**Loan\_Transaction**

**Debtor**

**Loan\_Information**

**1 1 1**

**1 M M**

**Loan\_Report**

**Transaction\_Id**

**Info\_Id**

**Report\_Id**

**Debtor\_Id**

**Designing the Normalized Database**

|  |  |  |
| --- | --- | --- |
| Table 1 | Relationship | Table 2 |
| Bank | One to Many | Debtor |
| Loan Offers | One to Many | Bank |
| Loan\_Transaction | One to one | Debtor |
| Debtor | One to Many | Loan\_information |
| Loan\_Information | Many to Many | Loan\_Transaction |
| Loan\_Information | One to Many | Loan\_Report |
| Loan\_Trasanction | One to One | Loan\_Report |
| Loan\_Report | Many to One | Loan\_Information |
| Loan\_Report | Many to One | Debtor |

**Table name: Bank**

|  |  |  |
| --- | --- | --- |
| Field name | Field Type | Description |
| bank\_id | Integer | Id number |
| bank\_name | Character varying (12) | Name of Bank |
| branch | Character varying (12 | Branch of Bank |
| address | Character varying (12) | Address |

**Table name: Debtor**

|  |  |  |
| --- | --- | --- |
| Field name | Field Type | Description |
| Debtor\_Id | Integer | Id of Customer |
| Fname | character varying(25) | First Name |
| Lname | character varying(25) | Last Name |
| Address | Text | Address |
| Age | Integer | Age of Debtor |
| Gender | character varying(5) | Gender |

**Table name: Loan\_Offers**

|  |  |  |
| --- | --- | --- |
| Field name | Field Type | Description |
| Loan\_Id | Integer | Loan number |
| Loan\_Name | Character Varying() | Loan Name |
| Amount | Character Varying(26) | Amount Of Loan |
| Loan\_range | Text | Loan Range |
| Interest | Character Varying(25) | Loan Intersest |

**Table name:Loan\_Information**

|  |  |  |
| --- | --- | --- |
| Field name | Field Type | Description |
| info\_id | integer | Info Id |
| debtor\_id | integer | Customer Id |
| loan\_given\_date | date | Date of loan given |
| Loan\_due\_date | date | Loan Due Date |
| Monthly\_Payment | money | Monthly Payment given by Debtor |
| Loan\_amt | money | Loan Borrowed by Debtor |
| Purpose | Varchar(30) | Specifies Type of Loan |
| Collateral | Varchar(30) | Collateral Kept by Debtor |

**Table name: Loan\_Transaction**

|  |  |  |
| --- | --- | --- |
| Field name | Field Type | Description |
| Transaction\_Id | Integer | Transaction Id |
| Debtor\_Id | Integer | Id of Customer |
| Date | Date | Date of Transaction |
| Time | time | Time without time zone |
| Loan\_amt\_approved | Money | Loan amount approved for Debtor |
| Bank\_name | Character varying(20) | Name of the Bank |
| Branch | Character varying(20) | Branch Name |
| Account\_no | Bigint | Account no of the Debtor/Customer |

**Table name: Loan\_Report**

|  |  |  |
| --- | --- | --- |
| Field name | Field Type | Description |
| Report\_Id | Integer | Report Id |
| Debtor\_Id | Integer | Customer Id |
| Info\_Id | Integer | Information of Loan Id |
| Transaction\_Id | Integer | Date of Transaction Made |

**DESIGNING QUERIES RELATED TO FUNCIONAL REQUIREMENT:**

**Nested Queries**

Que-1)Functional Requirement Discription.

Display the Maximum & Minimum loan amount approved.

Query:

postgres=# select max(loan\_amt\_approved) from loan\_information;

max

---------------

$5,000,000.00

(1 row)

postgres=# select min(loan\_amt\_approved) from loan\_information;

min

-------------

$500,000.00

(1 row)

Que-2)Functional Requirement Description.

Write a query to display the Debtor Details and their bank transaction details,for the debtor who’s loan\_amt>’1500000’

Query:

postgres=# select debtor\_Id(Debtor),fname,lname,bank\_name,branch,loan\_amt,account\_no from debtor,loan\_transaction where debtor.debtor\_id=loan\_transaction.debtor\_id and loan\_amt>='1500000';

debtor\_id | fname | lname | bank\_name | branch | loan\_amt | account\_no

---------------+----------+-------------+----------------+----------------+---------------+-------------

23478 | VIPUL | SINGH | HDFC Bank | Deccan Rd | $3,800,000.00 | 20123415241

23475 | VIGNESH | PATIL | SBI | M.G Road | $5,000,000.00 | 20023455233

23486 | VISHAL | NIKAM | Axis bank | Aundh | $4,200,000.00 | 10023445212

23489 | SANYUKTA | SHARMA| BOI | Main branch | $1,500,000.00 | 40123442141

23488 | PRIYANKA | PAWAR | SBI | Hinjewadi | $2,500,000.00 | 32123212121

(5 rows)

**Views**

Ques1)Create a View Which contains details of Debtors who have applied for loan amount approved where loan amount approved<’1000000’

Query:

postgres=# create view Prj\_View1 as select Debtor\_id(Debtor),fname,lname,address,age,gender,loan\_amt\_approved from debtor,loan\_information where debtor.debtor\_id=loan\_information.debtor\_id and loan\_amt\_approved<'1000000';

CREATE VIEW

postgres=# select \* from Prj\_View1;

debtor\_id | fname | lname | address | age | gender | loan\_amt\_approved

-----------+----------+--------+---------+-----+--------+-------------------

23478 | VIPUL | SINGH | MG ROAD | 36 | M | $968,000.00

23486 | VISHAL | NIKAM | SANGVI | 40 | M | $907,500.00

23489 | SANYUKTA | SHARMA | PASHAN | 30 | F | $605,000.00

(3 rows)

Ques2) Create View to display the Details of the debtors who have applied for Education loan.

Query:

postgres=# create view Prj\_View2 as select debtor\_id(Debtor),fname,lname,address,age,gender,Purpose from Debtor,Loan\_information where debtor.debtor\_id=loan\_information.debtor\_id and purpose='Education Loan';

CREATE VIEW

postgres=# select \* from Prj\_View2;

debtor\_id | fname | lname | address | age | gender | purpose

-----------+----------+--------+---------+-----+--------+----------------

23477 | SAMIKSHA | KAPOOR | KOTHRUD | 26 | F | Education Loan

23488 | PRIYANKA | PAWAR | BANER | 28 | F | Education Loan

(2 rows)

**Stored Function**

Que1)Functional Requirement Description.

Write a Stored Function to find maximum loan amount approved.

Query:

create or replace function Maxloan\_amt() returns money as $$

declare

loan\_amt money;

begin

select Max(loan\_amt\_approved) into loan\_amt from Loan\_Information;

return loan\_amt;

end;

$$

language plpgsql;

**Output:**

postgres=# select Maxloan\_amt();

maxloan\_amt

---------------

$5,000,000.00

(1 row)

Que2) Functional Requirement Description

Write a stored function to count the total number of Debtors who have applied for Education Loan

Query:

create or replace function Count\_Debtor(Loan\_Purpose char(30)) returns int as $$

declare

total\_Debtors int;

begin

select count(debtor\_id(Loan\_Information)) into total\_Debtors from Debtor,Loan\_information where Debtor.debtor\_id=loan\_information.debtor\_id

and Purpose=Loan\_Purpose;

return total\_Debtors;

end;

$$

language plpgsql;

**Output:**

postgres=# select Count\_Debtor('Education Loan');

count\_debtor

--------------

2

(1 row)

postgres=# select Count\_Debtor('Home Loan');

count\_debtor

--------------

1

(1 row)

**Cursor**

Que1) Write Functional Requirement.

Write a stored function using Cursor to find the name of Debtors along with their Loan Amount where loan amount Approved is Greater than Rs 1500000.

Query:

Create or replace function Debtors() returns integer as $$

declare Debt Cursor for Select Fname,Lname,loan\_amt\_approved from Debtor,Loan\_information where debtor.debtor\_id=loan\_information.debtor\_id

and loan\_amt\_approved>'1500000';

f1 Debtor.fname%type;

l1 Debtor.Lname%type;

lamt loan\_Information.Loan\_amt\_approved%type;

begin

open Debt;

loop

fetch Debt into f1,l1,lamt;

exit when not found;

raise notice '% % %',f1,l1,lamt;

end loop;

close Debt;

return 0;

end;

$$ language plpgsql;

**Output:**

postgres=#

postgres=# select Debtors();

NOTICE: RAJESH SINGH $3,800,000.00

NOTICE: RAHUL RAJPUT $4,200,000.00

NOTICE: VIGNESH PATIL $5,000,000.00

NOTICE: PRIYANKA PAWAR $2,500,000.00

debtors

---------

0

(1 row)

Que2)Write Functional Requirement.

Write a Stored function using Cursor To display Debtors full name along with their loan amount who have loan Due in the Year 2022;

Query:

create or replace Function Loan\_due() returns integer as $$

declare Due\_Year cursor for select fname,lname,loan\_amt\_approved,loan\_due from Debtor,loan\_information where

debtor.debtor\_id=Loan\_information.debtor\_id and Extract(Year from loan\_due)='2022';

f1 debtor.fname%type;

l1 debtor.lname%type;

lamt loan\_information.loan\_amt\_approved%type;

d1 loan\_information.loan\_due%type;

begin

open Due\_Year;

loop

fetch Due\_Year into f1,l1,lamt,d1;

exit when not found;

raise notice '% % % %',f1,l1,lamt,d1;

end loop;

close Due\_Year;

return 0;

end;

$$ language plpgsql;

**Output:**

postgres=#

postgres=# select Loan\_Due();

NOTICE: SAMIKSHA KAPOOR $1,500,000.00 2022-08-10

NOTICE: RAHUL RAJPUT $4,200,000.00 2022-02-03

loan\_due

----------

0

(1 row)

**Exception**

Que1)Write a Functional Requirement.

Write a stored function to print total number of loan Transaction done for debtors through a particular branch.(Accept branch name as input parameter).In case the branch name is invalid,Raise an Exception for the same

Query:

create or replace function Brname\_Exception(Brname char(30)) returns integer as $$

declare

count\_debtor integer;

begin

raise notice 'The Function Begin';

select count(loan\_transaction.debtor\_id) into count\_debtor from loan\_transaction where branch=brname;

if count\_debtor=0 then

raise Exception 'No Loan Transaction done for debtors through this particular Branch or Branch name is Invalid';

end if;

return count\_debtor;

end;

$$ language 'plpgsql';

**Output:**

postgres=# Select Brname\_Exception('MG Road');

NOTICE: The Function Begin

ERROR: No Loan Transaction done for debtors through this particular branch or Branch name is Invalid

CONTEXT: PL/pgSQL function brname\_exception(character) line 9 at RAISE

postgres=#

Que2)Write a Functional Requirement.

Write a stored function to increase the loan approved amount for all loans by 10%.In case the initial loan approved amount was less than Rs 800000,then print a notice to the user,before updating the amount.

Query:

create or replace function loan\_Exception() returns integer as $$

declare

lamt loan\_information%rowtype;

loan\_Increment cursor for select \* from loan\_information;

begin

open loan\_Increment;

loop

fetch loan\_Increment into lamt;

exit when not found;

if

lamt.loan\_amt\_approved<'$800000'

then raise notice 'The loan Amt Approved was less than 800000';

update loan\_information set loan\_amt\_approved=loan\_amt\_approved+(loan\_amt\_approved\*0.1);

end if;

end loop;

close loan\_Increment;

return 0;

end;

$$ language 'plpgsql';

**Output:**

postgres=# select loan\_Exception();

NOTICE: The loan Amt Approved was less than 800000

NOTICE: The loan Amt Approved was less than 800000

loan\_exception

----------------

0

(1 row)

postgres=# select \* from Loan\_Information;

info\_id|debtor\_id|loan\_date|loan\_due|monthly\_payment|loan\_amt\_approved|purpose| collateral

---------+-----------+------------+------------+-----------------+-------------------+----------------------+-----------------------

4243 | 23486|2018-10-18 |2024-05-04 |6000 | $907,500.00 | Gold Loan | Payment Slip/vehicle

4315 | 23489 |2016-11-20| 2019-12-28 | 8000 | $605,000.00 | Business Loan | Apartment

4253 | 23477 | 2019-12-07 | 2022-08-10 | 16000 | $1,815,000.00 | Education Loan | Gold

4263 | 23488 | 2021-02-10 | 2023-07-24 | 12000 | $3,025,000.00 |Education Loan| Non Agricultural Land

4213|23476|2021-02-04|2024-01-10 | 16000 | $4,598,000.00 |Home Renovation Loan|Payment

Slip

4313|23466 |2020-09-14|2022-02-03 | 28333 | $5,082,000.00 | Home Loan | Property Papers

4223| 23475|2021-10-04|2024-08-12 | 24000 | $6,050,000.00 | Home loan | Property Papers

4233| 23478|2020-03-08 |2021-02-04 | 5833 | $968,000.00 | Personal Loan | Gold Jewellery

(8 rows)

**TRIGGER**

Ques1)Write a Functional Requirement.

Write a trigger on loan amount, if new loan amount entered by user exceed more than Rs 8000000 than raise an Error Stating ’The new Loan Amount Entered exceeds the Loan Range’.

Query:

create or replace function loan\_amt() returns trigger as $$

begin

if new.loan\_amt>'8000000'

then raise exception 'The New Loan amount Entered Exceeds the Loan Range';

end if;

return new;

end;

$$ language 'plpgsql';

**Output:**

postgres=#

postgres=# create trigger Prjct\_T3 before insert on loan\_transaction for each row execute procedure loan\_amt();

CREATE TRIGGER

postgres=#

postgres=# insert into loan\_transaction values(3385,'23475','2018-3-12','23:12:34','9000000','Kotak Bank','MG branch','4231673411');

ERROR: The New Loan amount Exceeds the Loan Range

CONTEXT: PL/pgSQL function loan\_amt() line 4 at RAISE

postgres=#

Ques2)Write Functional Requirement.

Write a trigger on loan Information table such that loan amount approved field of the table should not be updated.

Query:

create or replace function Trigger\_update() returns trigger as $$

begin

if new.loan\_amt\_approved is distinct from old.loan\_amt\_approved then

raise exception 'Update of "loan amount approved" is not allowed';

end if;

return new;

end;

$$ language 'plpgsql'

**Output:**

postgres=# create trigger Update before update on loan\_information for each row execute procedure Trigger\_update();

CREATE TRIGGER

postgres=#

postgres=# update loan\_Information set loan\_amt\_approved='1000000' where info\_id='4243';

ERROR: Update of "loan amount approved" is not allowed

CONTEXT: PL/pgSQL function trigger\_update() line 4 at RAISE

postgres=#