# Banking Database System

REVIEW REPORT

Submitted by

**Group Member1: Arshdeep Singh (19BCB0086)**

**Group Member2: Ananya Anand (19BCE2332)**

**Group Member3: Irene John (19BCE2437)**

Prepared For

**DATABASE MANAGEMENT SYSTEM (CSE2004)**

**PROJECT COMPONENT**

Submitted To

**Dr. Priya M**

**Associate Professor**

## School of Computer Science and Engineering


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# Table of contents

## ABSTRACT

In the current world where every application finds a place in the online platform, Banking System would be one of them. A bank is a financial intermediary involving several functionalities such as safeguarding, transferring, exchanging, or lending of money. Understanding these fundamental functionalities helps one to understand how banking systems work and helps to build modern trends in banking and finance.

Keeping in mind the functionalities and the need of the present generation we have developed a database management system for bank accounts for all the customers enrolled in bank.It involves the database that the bank maintains for all customer functionality.It keeps the day by day tally record as a complete banking involving information of Account type, Deposit, Withdrawal and Searching the transaction. It removes the need of physical movement to a bank to carry out basic services offered at the bank.

This project involves various concepts of database and web development. Our website has a user friendly interface and it is being created using Xampp control Panel wherein MYSQL has been used for database management to store and retrieve data from the database. In addition we have created multiple pages linked to each other. These pages have been developed and designed using HTML and CSS. PhP is used as a tool for connectivity between the front-end and the back-end. Indeed, this has been a great learning experience especially for the concepts of database and its practical application.

# 1. INTRODUCTION

During the past several decades' personnel function has been transformed from a relatively obscure record keeping staff too central and top level management function. There are many factors that have influenced this transformation like technological advances, professionalism, and general recognition of human beings as most important resources.

A computer based management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statement of any month. Separate database is maintained to handle all the details required for the correct statement calculation and generation.

This project intends to introduce more user friendliness in the various activities such as record updating, maintenance, and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updating can also be accomplished by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

The entire information has maintained in the database or Files and whoever wants to retrieve can't retrieve, only authorization user can retrieve the necessary information which can be easily be accessible from the file.

## 1.1 Background

In this section of our project we will like to mention background or the prerequisites skills that we need to have in order to carry out this project. The main area of knowledge in this project is the creation of databases. Thorough knowledge of ER diagram, functional dependencies, normalization etc. For the completion of the project we should have knowledge of the subjects listed –

1. HTML, CSS and PHP for front end development,
2. MYSQL in XAMMP SERVER for backend,
3. SQL for preparing the database.

## 1.2 Objective

The main objective of the project is to develop online Banking system for banks. In present system all banking works is done manually. User have to visit bank for Withdrawal or Deposit amount. In present bank system it is also difficult to find account information of account holder. In this bank management system we will automate all the banking process. In our bank management system user can check his balance online and he can also transfer money to other account online. In this software you can keep record for daily Banking transactions. The main purpose of developing bank management system is to design an application, which could store bank data and provide an interface for retrieving customer related details.

### 1.3 Motivation

Our major motivation of carrying out this project is that amidst the ongoing digital banking environment, the need for an efficient banking management solution is strongly felt. These systems are crucial for the growth of any sector and provide a multidisciplinary approach to the management as well as the governance of the bank related activities. These systems make the best use of computer systems as well as software for helping the bank and customer to create new account easily. The project makes a sincere effort to provide all the mentioned features to meet the requirements of the bank in terms of both the employees and the customer. Also, another source of motivation is Scams which our going on in the market. Thus, such factors piqued our interests to work on such project.

### 1.4 Contributions of the Project

Using this bank management system any information can be easily searched. User can view all the details of the customer, user can create new customer account and maintain its data efficiently and effectively. Which are maintained constantly update by system. Manage large number of customer details with ease. Particular account information can be modified particular customer record can be modified for one or more field's customer name, address by providing account number. Activities like updating, modification, deletion of records should be easier. A customer record can be easily deleted by authorize user by providing account number.

### 1.5 Organization of the Project

- First, we started off with preparation of ER diagram taking into view all the possible attributes and relationships among them.
- Next, we converted our ER diagram to Relational mapping so that we can get an actual picture of all the relationships.
- After carrying out above mentioned steps we made list of all the tables and constraints of attributes.
- Next, we collected sample data to be stored in our database.
- After that we started with preparation of database in MYSQL with our collected data.
- After creation of initial Database, we normalized our tables so that inconsistency of our database can be reduced.
- After successful creation of our Final Database we operated few normal SQL queries and PL/SQL queries.
- After that we constructed our website and prepared its Front End and Back End and connected the Front end to our database.

### 1.6 Work Breakdown

| Registration Number | Team Member-Name | Work Assigned |
|---|---|---|
| 19BCB0086 | Arshdeep Singh | -Entire front-end implementation of the project and backend coding<br><br>-Compilation of the project<br><br>(connection between back-end and front-end using php ) |
| 19BCE2332 | Ananya Anand | Database implementation<br><br>(database relations creation)<br><br>-Documentation |
| 19BCE2437 | Irene John | Database implementation<br><br>(data collection and application of constraints)<br><br>-Documentation |

# 2. PROJECT RESOURCE REQUIREMENTS

## 2.1 Software Requirements

Front- End Software Used:

- PHP
- HTML
- CSS (materialize style sheet)

Back- End Software Used:

- MYSQL in XAMMP SERVER

Text Editor to type php files: SUBLIME TEXT

## 2.2 Hardware Requirements

Processor- Intel i3 OR ABOVE

Clock speed- 2.5 GHz

Ram 8 GB or above

Windows 10 or any other suitable OS

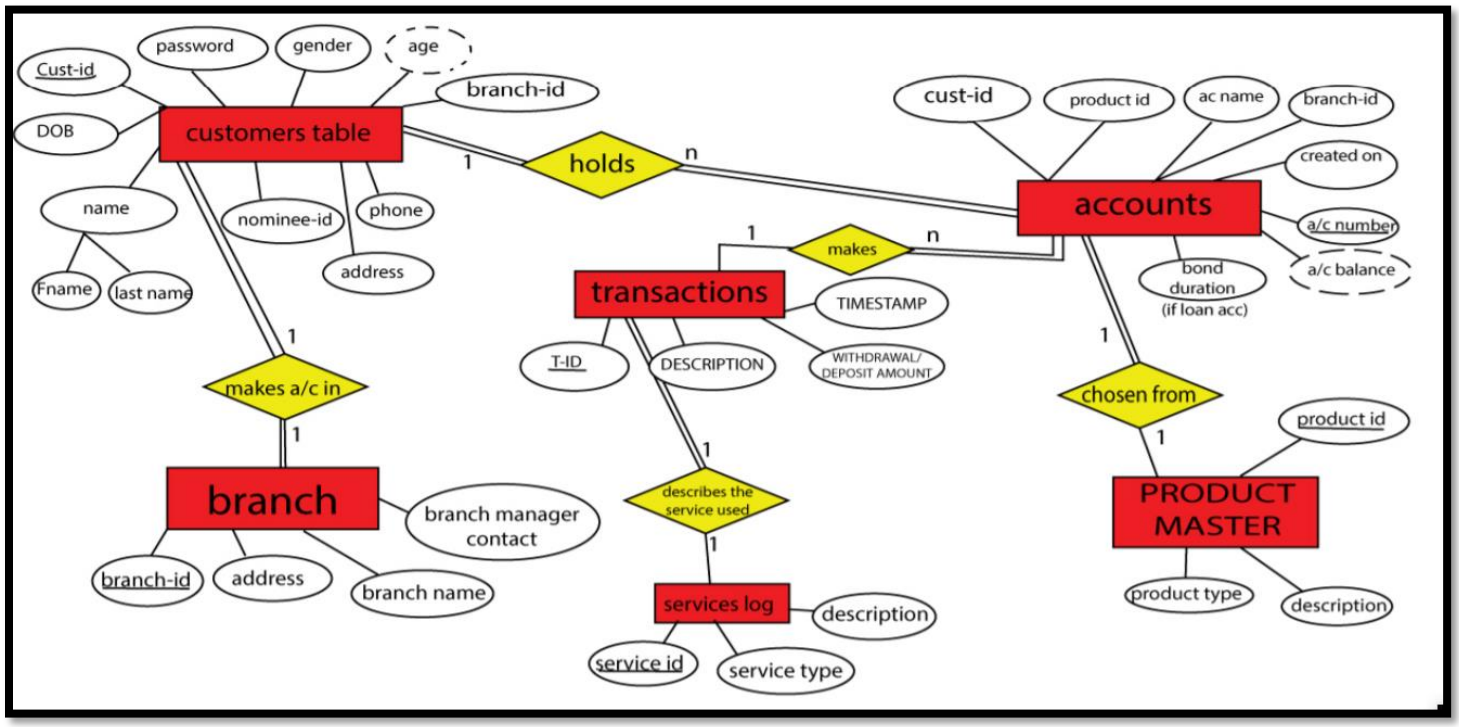
# 3 LITERATURE SURVEY

In this section, literature survey is given. Accordingly, research papers are reviewed and analyzed based on the prediction methods used.

| Authors | Method | Purpose | Advantages | Disadvantages |
|---|---|---|---|---|
| Momenul Ahmad | How to do web development | Giving general idea about web development and internet | It provides essential knowledge to beginner. | Knowledge provided is limited and it is not helpful to a more experienced developer. |
| Alijetro Castillo | Quantitative analysis on database contents | Provides knowledge about database contents and contains vast range of examples to analyze contents of database | It is vast and contains wide range of examples | It requires a pre requisite knowledge and not recommended for beginners |
| Me Me Khaing | 1.1.1.3 Why undergraduates should learn web development | Provides us with basic knowledge of HTML and CSS with some examples related to it | Provides HTML syntax and some examples regarding the same | It is nice document but it lacks some or the other syntax and contains less examples |
| Raghu Ramakrishnan | Introduction to DBMS | It is a textbook which helped us to understand ER diagram and conversion of it to relational mapping | Contains a nice overview of database | It would be better if it contains more examples and was better organized |
| Kavya .S | 1.1.1.1 A study on Database | Gives an overview on databases | Explains the general fundamentals and functioning | Lacks practical examples for reference |

# 4. DESIGN OF THE PROJECT

## 4.1 ER Diagram




**KEY:**

Entities: **Entities**

Relationships between Entities: Partial relation

Total relation

Relation: Relation

Structural Constraints: Primary key

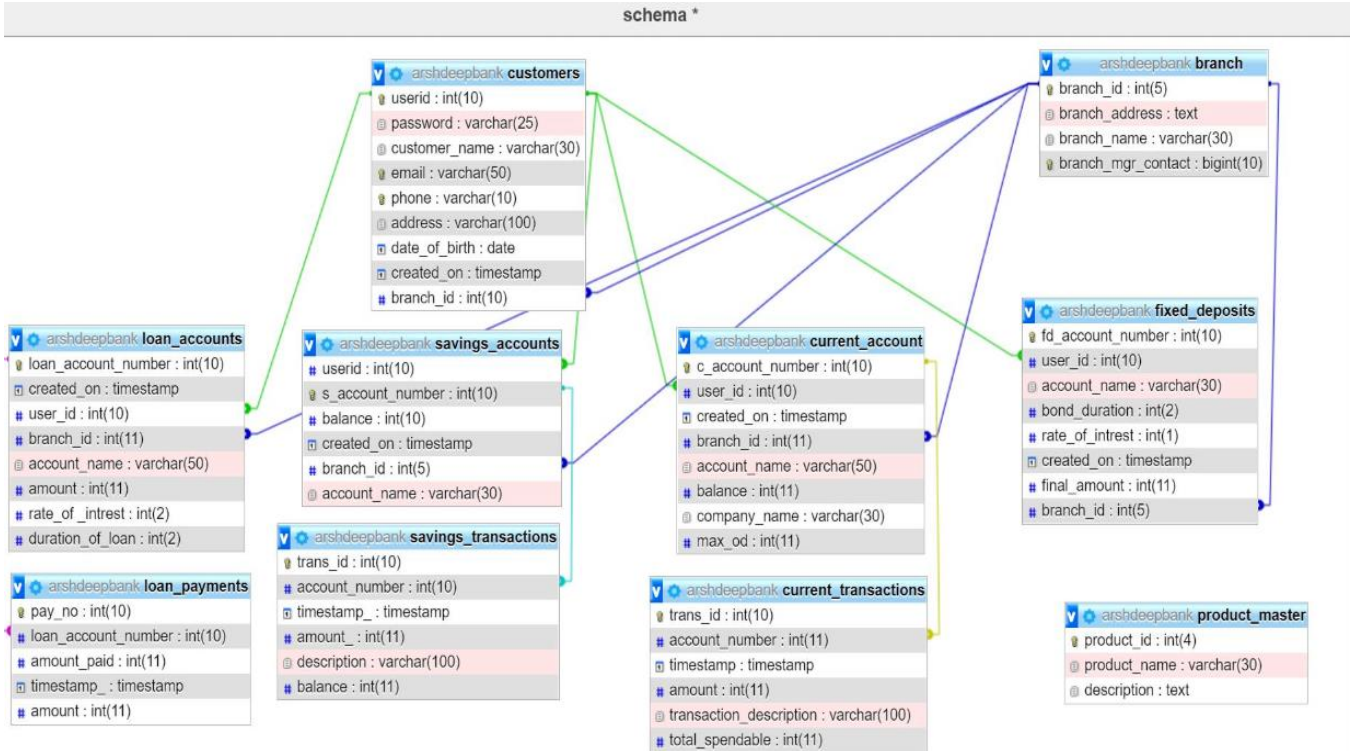
Cardinality ratios: 1:1- One to one

1: N – One to Many

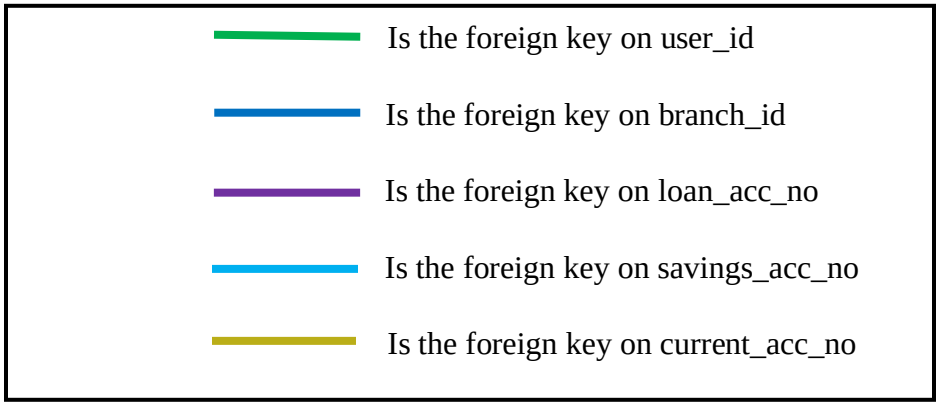
N: 1 – Many to one

N: N- Many to many

## 4.2 ER to Relational Mapping (Schema Diagram)




**KEY:**



**4.3 Normalization**
**Rules for Normalization:**

**1NF (FIRST NORMAL FORM)**

- Each table cell should contain a single value.
- Each record needs to be unique.

**2NF (SECOND NORMAL FORM)**

- Rule 1- Be in 1NF
- Rule 2- There should be no partial dependency.

**3NF (THIRD NORMAL FORM)**

- Rule 1- Be in 2NF
- Rule 2- There should be no transitive functional dependencies.

**BCNF**

- Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one Candidate Key.
- Sometimes is BCNF is also referred as 3.5 Normal Form.

**-Normalization of the Tables**

**1) Customers**

[User_id, Password,Name,email_id,phone,address,DOB.created_on,Gender]

**Functional dependencies:**

User_id-> Password

User_id-> created_on

User_id->Name

User_id->Gender

User_id-> address

User_id->phone

User_id->DOB

User_id->email_id

**Candidate Ke**y: User_id

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Since the table is already normalized, the table Customers remains the same.

**2) Branch**

[br_id,br_name,address,mgr_contact]

**Functional dependencies:**

br_Id->address br_Id-> br_name

br_Id->mgr_contact

**Candidate Key:** br_Id

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Since the table is already normalized, the table Branch remains the same.


**3) Product master**

[product_id,acc_type,ac_info]

**Functional dependencies:**

product_Id->acc_type,ac_info

**Candidate Key:** product_Id

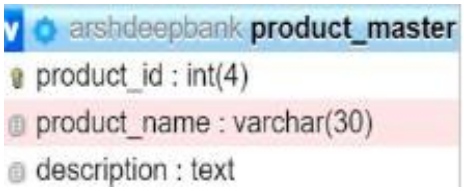
**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Since the table is already normalized, the table Product Master remains the same.




**Accounts**

The accounts table in the initial database was inconsistent as all account types were clubbed together with many null able domains.

Ex: If account number 1000010 has a savings account the values like bond duration or rate of interest which concerns with loan type will be null able

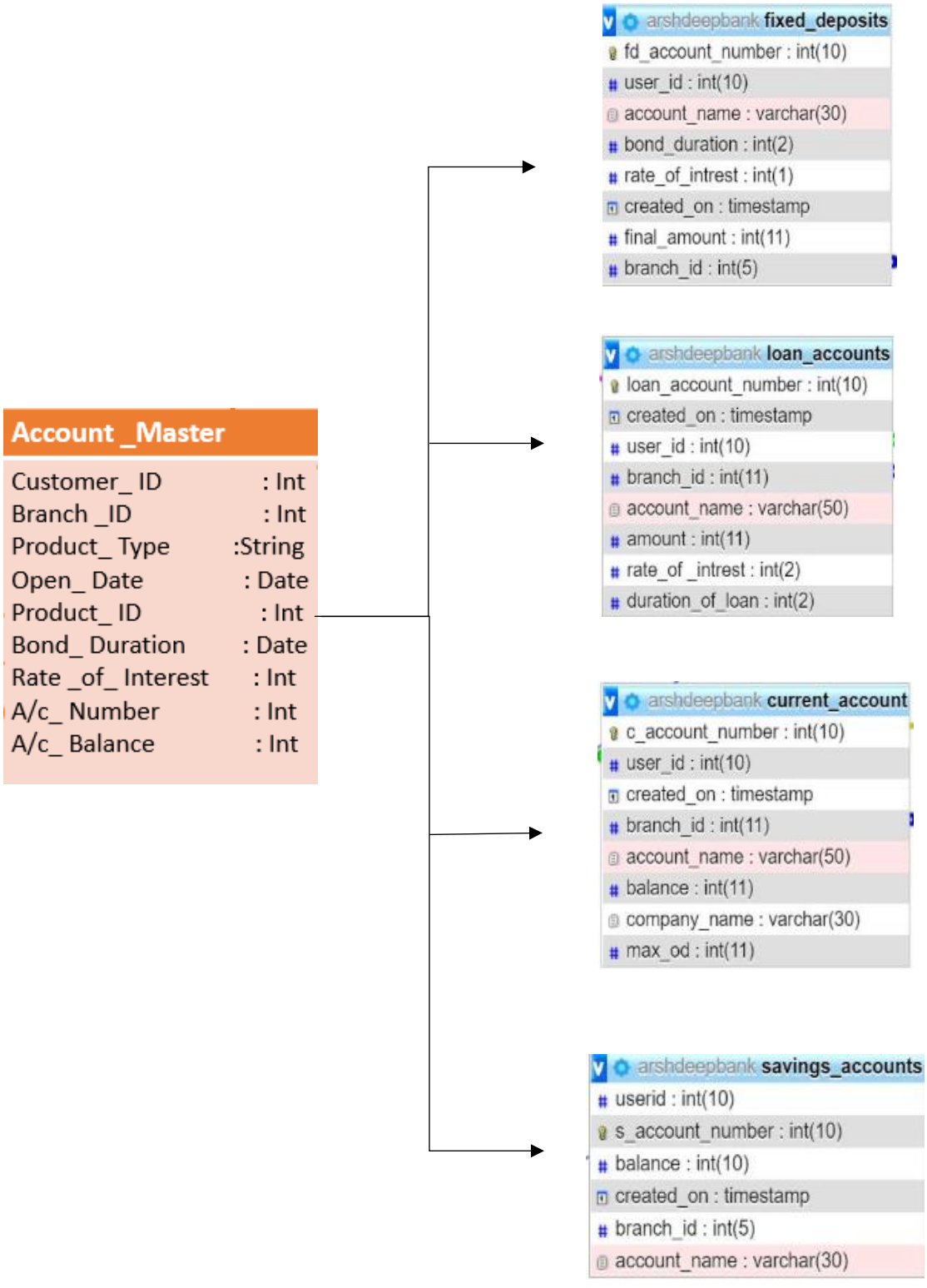
Thus accounts table has been parted into
-Savings account

-Current

-Loan

-Fixed Deposit

Normalizing Accounts table


Account _Master
Customer_ ID : Int
Branch _ID : Int
Product_ Type :String
Open_ Date : Date
Product_ ID : Int
Bond_ Duration : Date
Rate _of_ Interest : Int
A/c_ Number : Int
A/c_ Balance : Int
arshdeepbank fixed_deposits
fd_account_number : int(10)
user_id : int(10)
account_name : varchar(30)
bond_duration : int(2)
rate_of_intrest : int(1)
created_on : timestamp
final_amount : int(11)
branch_id : int(5)
arshdeepbank loan_accounts
loan_account_number : int(10)
created_on : timestamp
user_id : int(10)
branch_id : int(11)
account_name : varchar(50)
amount : int(11)
rate_of _intrest : int(2)
duration_of_loan : int(2)
arshdeepbank current_account
c_account_number : int(10)
user_id : int(10)
created_on : timestamp
branch_id : int(11)
account_name : varchar(50)
balance : int(11)
company_name : varchar(30)
max_od : int(11)
arshdeepbank savings_accounts
userid : int(10)
s_account_number : int(10)
balance : int(10)
created_on : timestamp
branch_id : int(5)
account_name : varchar(30)

**4) Savings Account**

[Acc_id,cust_id,creationtimestamp,branch_id,Acc_name,balance]

**Functional dependencies:**

Acc_Id-> cust_id Acc_Id-> branch_id

Acc_Id-> created_timestamp Acc_Id->Acc_name

Acc_Id-> balance

**Candidate Key:** Acc_Id

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Hence the table is normalized.

**5) Current Account**

[Acc_id,cust_id,creationtimestamp,branch_id,Acc_name,balance,company_name,max_d]

**Functional dependencies:**

Acc_Id-> cust_id Acc_Id-> branch_id

Acc_Id-> created_timestamp Acc_Id->Acc_name

Acc_Id-> balance Acc_Id-> max od_amt

Acc_Id-> company name

**Candidate Key:** Acc_Id

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Hence the table is normalized.

**6) Loan**

[Ln_no,cust_id,created_timestamp,branch_id,Acc_name,rate_of_interest,duration,amount]

**Functional dependencies:**

Ln_no-> cust_id

Ln_no-> branch_id

Ln_no-> created_timestamp

Ln_no ->Acc_name

Ln_no-> duration

Ln_no ->rate_of_interest

**Candidate Key:** Ln_no

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Hence the table is normalized.

**7) Fixed Deposit**

[Acc_id,cust_id,name,duration,rate_of_interest,created_on,final_amount,Acc_name]

**Functional dependencies:**

Acc_Id-> cust_id Acc_Id->duration

Acc_Id-> created_on Acc_Id->Acc_name

Acc_Id-> rate_of_interest Acc_Id->final_amount

**Candidate Key:** Acc_Id

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Hence the table is normalized.

**Transactions**

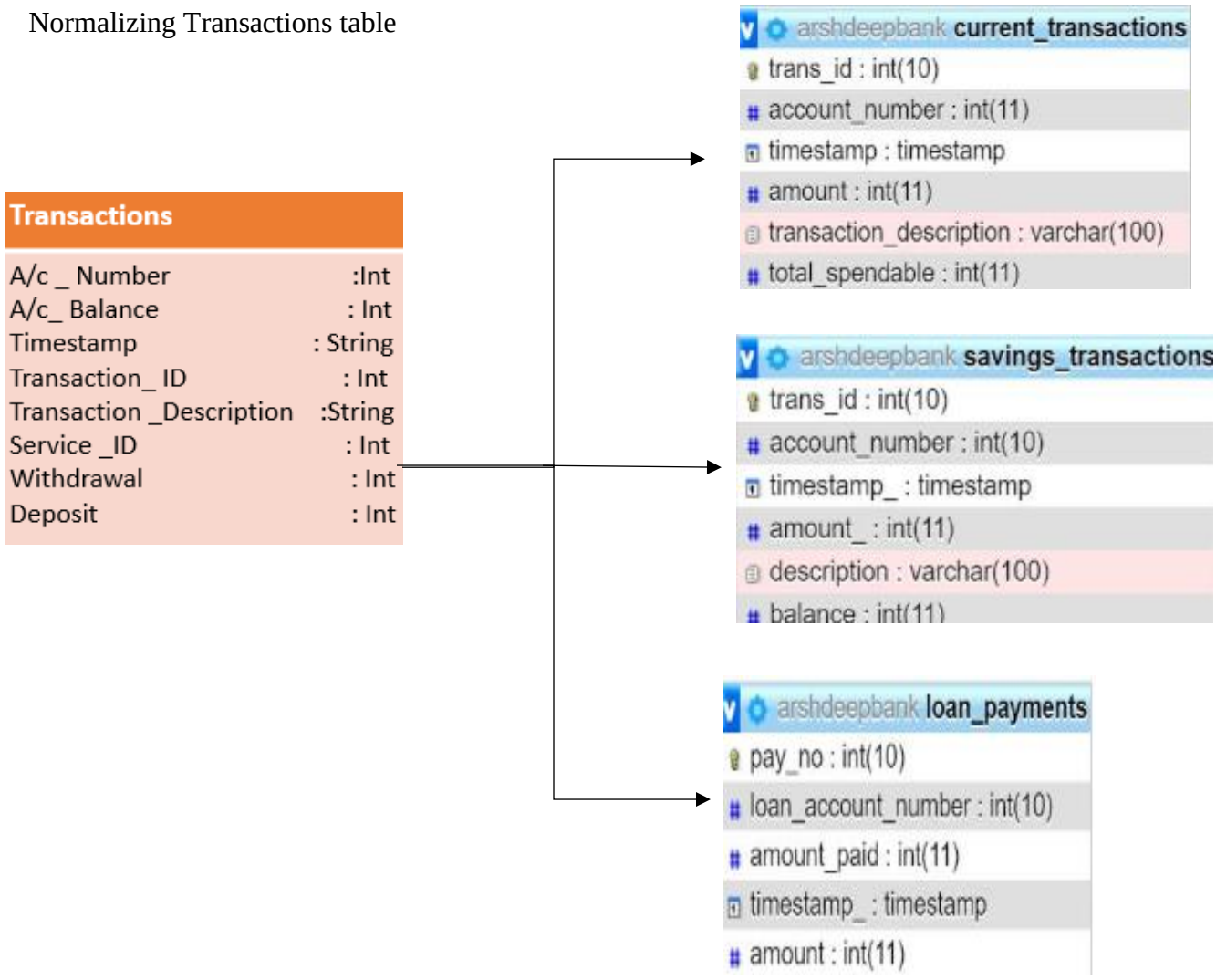
Just like the accounts table the transaction table was clubbed under one table which leads to inconsistencies and abnormalities. Hence breaking the tables into:

-Savings Transactions

-Current Transactions

-Loan payments

Normalizing Transactions table



**8) Savings_transactions**

[TID,Acc_id,Timestamp,amount,description,balance]

**Functional dependencies:**

TID-> Acc_id TID->amount

TID-> Timestamp TID-> balance

TID-> description

**Candidate Key:** TID

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Hence the table is normalized.

**9) Current_transaction**

[TID,Acc_id,Timestamp,amount,description,Total]

**Functional dependencies:**

TID-> Acc_id TID->amount

TID-> Timestamp TID-> Total

TID-> description

**Candidate Key:** TID

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Hence the table is normalized.

**10) Loan_Payments**

[Pay_no,Ln_no,amt_paid,date_timestamp,remaining_due]

**Functional dependencies:**

Pay_no->ln_no Ln_no-> amt_paid

Pay_no-> date_timestamp Ln_no->remaining_due

**Candidate Key:** Pay_no

**1NF:** It is in 1NF due to atomicity

**2NF:** It is in 2NF as it is in 1NF and there are no partial dependencies

**3NF:** It is in 3 NF as it is in 2NF and there are no transitive dependencies

**BCNF:** It is in BCNF as it is in 3NF and the LHS of the F.d's is a prime key

Hence the table is normalized.

### 4.4 Tables and Constraints

TABLE : CUSTOMER

| CUSTOMER | | |
|---|---|---|
| User_ID | int (10) | Primary key |
| Password | Varchar2(20) | Not null |
| Name | Varchar2(50) | Not null(no numerical values) |
| Email_ID | Varchar2(50) | Not null |
| Phone | Number(10) | Not null |
| Address | Varchar2(50) | Not null |
| DOB | Date | Not null |
| Created_on | Time | Not null(18 digits must) |
| Gender | Varchar2 | Not null(M,F,other) |

TABLE: PRODUCT MASTER

| PRODUCT MASTER | | |
|---|---|---|
| Product_ID | Number(5) | Primary Key |
| Acc_Type | Varchar2(10) | Not Null |
| Acc_Info | Varchar2(50) | Not Null |

TABLE: BRANCH

| BRANCH | | |
|---|---|---|
| Branch_ID | Number(5) | Primary Key |
| Address | Varchar2(50) | Not Null |
| Branch_name | Varchar2(30) | Not Null |
| Mgr_contact | Varchar2(10) | Not Null(10 digits must) |

TABLE: SAVINGS

| SAVINGS | | |
|---|---|---|
| Acc_ID | Number(10) | Primary Key |
| Cust_ID | Number(10) | Foreign Key |
| Creation_Timestamp | Timestamp | Not Null |
| Branch_ID | Number(5) | Foreign Key |
| Acc_name | Varchar2(30) | Not Null |
| Balance | Number(derived) | Not Null |

TABLE: FIXED DEPOSIT

| FIXED DEPOSIT | | |
|---|---|---|
| Ac_ID | Number(10) | Primary Key |
| Cust_ID | Number(10) | Foreign Key |
| Name | Varchar2(30) | Not Null |
| Duration | Varchar2(10) | Not Null |
| Rae_of_Interest | Varchar2(5) | Not Null |
| Created_On | Date | Not Null |
| Final_Amount | Number(20) | Not Null |

TABLE: LOAN

| LOAN | | |
|---|---|---|
| Ln_no | Number(10) | Primary Key |
| Cust_ID | Number(10) | Foreign Key |
| Created_timestamp | Timestamp | Not Null |
| Branch_ID | Number(5) | Foreign Key |
| Acc_name | Varchar2(30) | Not Null |
| Rate_of_Interest | Varchar2(5) | Not Null |
| Duration | Varchar2(10) | Not Null |
| amount | Number(10) | Not Null |

TABLE: CURRENT

| CURRENT | | |
|---|---|---|
| Acc_ID | Number(10) | Primary Key |
| Cust_ID | Number(10) | Foreign Key |
| Created_timestamp | Timestamp | Not Null |
| Branch_ID | Numbers(5) | Foreign Key |
| Acc_name | Varchar2(30) | Not null |
| Balance | Number(5) | Not Null |
| Company_name | Varchar2(10) | Not Null |
| Max_od | Number | Not Null |

TABLE: SAVINGS TRANSACTION

| SAVINGS_TRANSACTION | | |
|---|---|---|
| TID | Int(10) | Primary Key |
| Acc_ID | Int(10) | Foreign Key |
| Timestamp | Date | Not Null |
| Amount | Int(10) | Not Null |
| Description | Varchar(50) | Not Null |
| Balance | Int(10) | Not Null |

TABLE: CURRENT TRANSACTION

| CURRENT_TRANSACTION | | |
|---|---|---|
| TID | Int(10) | Primary Key |
| Acc_ID | Int(10) | Foreign Key |
| Timestamp | Date | Not Null |
| Amount | Int(10) | Not Null |
| Description | Varchar(50) | Not Null |
| Tot_spending_capacity | Varchar(50) | Not Null |

TABLE: LOAN PAYMENTS

| LOAN_PAYMENTS | | |
|---|---|---|
| Pay_no | Int(10) | Primary Key |
| Ln_No | Int(10) | Foreign Key |
| Amount_paid | Int(10) | Not Null |
| Timestamp | Date | Not Null |
| Remaining_due | Int(10) | Not Null |

# 5. IMPLEMENTATION

## 5.1 Introduction

A simple user can access their account and can deposit/withdraw money from their account. User can also transfer money from their account to any other bank account. User can see their transaction report and balance enquiry too.

## 5.2 DDL & DML Queries

1) Select all transactions on saving accounts owned by RIYA (use of inner join to join transactions accounts and customer table)

```
SELECT s.* FROM `savings_transactions` s inner join savings_accounts on savings_accounts.s_account_number=s.account_number and savings_accounts.userid in (select useri
from customers where customers.customer_name='RIYA')
```

Profiling [Edit inline] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refre

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| trans_id | account_number | timestamp_ | amount_ | description | balance |
|---|---|---|---|---|---|
| 9 | 1000000003 | 2020-10-05 19:37:11 | 8000 | 8000 opening balance | 8000 |
| 11 | 1000000003 | 2020-10-05 19:39:50 | -302 | 302 withdrawal | 7698 |

2) Display branch information for all fixed deposit accounts (use of inner join on branch and fixed deposits table)

```
SELECT b.* FROM `branch` b inner join fixed_deposits fd on fd.branch_id=b.branch_id
```

Profiling [Edit inline]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| branch_id | branch_address | branch_name | branch_mgr_contact |
|---|---|---|---|
| 11000 | A-2, First Floor, Ring Rd, Block C, South Extensio... | South delhi branch | 9381543475 |
| 10000 | S-110/111, RRR building, L&T Rd, Burma Colony, Thi... | Perungudi branch | 9845630892 |

3) Select the branch manager contact for each user. (Use of nested sub-queries)

```
SELECT branch_mgr_contact from branch where branch_id in (select branch_id from customers where userid=1000000049)
```

Profiling [Edit inlir

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

| | branch_mgr_contact |
|---|---|
| Edit Copy Delete | 7836657390 |

4) Select all loan payments on 2020-10-01(use of date function to convert timestamp to date format.)

select * from loan_payments where date(timestamp)="2020-10-01

```
select * from loan_payments where date(timestamp_)="2020-10-01"
```

Profiling [Edit inline] [ Edit ]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| | pay_no | loan_account_number | amount_paid | timestamp_ | amount |
|---|---|---|---|---|---|
| Edit Copy Delete | 2 | 242000001 | 18334 | 2020-10-01 21:33:30 | -1081666 |
| Edit Copy Delete | 3 | 242000000 | 10000 | 2020-10-01 21:33:30 | -1180000 |

5) Select all branch information and customer information for a user =1000000049 using join.

select * from branch inner join customers on customers.branch_id=branch.branch_id and customers.userid=1000000049

| branch_id | branch_address | branch_name | branch_mgr_contact | userid | password | customer_name | email | phone | address | date_of_birth | created_on | branch_id |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13453 | No 25, 1st East Main Road Gandhi Nagar, Katpadi, V... | Vellore branch | 7836657390 | 1000000049 | 0000 | IRENE JOHN | irene.john2019@vitstudent.ac.in | 9840849927 | HOUSE-05,gandhi nagar NAGAR,vellore,640202 | 1999-01-06 | 2020-10-05 18:16:39 | 13453 |

### 5.3 SQL Queries

1)SELECT * FROM customers inner join savings_accounts oncustomers.userid=savings_accoun ts.userid and savings_accounts.balance<=6000 (inner join to select all information on joining customers and savings acc with balance <=6000)

+ Options

| userid | password | customer_name | email | phone | address | date_of_birth | created_on | branch_id | userid | s_account_number | balance | created_on | branch_id | account_name |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000000048 | Arshdeep123 | ARSHDEEP SINGH BHATIA | arshdeepdgreat@gmail.com | 8754541603 | A4-405,adora akshaya homes,padur,chennai | 1993-02-18 | 2020-10-05 18:12:49 | 10000 | 1000000048 | 1000000001 | 5000 | 2020-10-05 18:46:08 | 10000 | savings account upi |
| 1000000053 | 0000 | JOHN | john@vit.ac.in | 7399366492 | A4-405,adora akshaya homes,padur,chennai | 1991-05-23 | 2020-10-12 22:43:45 | 10000 | 1000000053 | 1000000026 | 6000 | 2020-10-13 11:24:51 | 10000 | savings account john |

2)select loan_payments.* from loan_payments inner join loan_accounts on loan_payments.loan_ account_number=loan_accounts.loan_account_number and loan_accounts.branch_id in (select br anch_id from branch where branch_mgr_contact=9381543475)

--use of join and sub query)

| pay_no | loan_account_number | amount_paid | timestamp_ | amount |
|---|---|---|---|---|
| 1 | 242000000 | 10000 | 2020-09-01 21:33:30 | -1190000 |
| 2 | 242000001 | 18334 | 2020-10-01 21:33:30 | -1081666 |
| 3 | 242000000 | 10000 | 2020-10-01 21:33:30 | -1180000 |

3) Combined use of sub query and date function

SELECT * FROM current_account WHERE user_id in (select userid from custo- mers where year (customers.date_of_birth) = 1999)

| c_account_number | user_id | created_on | branch_id | account_name | balance | company_name | max_od |
|---|---|---|---|---|---|---|---|
| 150000000 | 1000000049 | 2020-10-05 21:02:48 | 13453 | current_acc | 10000 | l and k logistics | 10000 |

4) Illustration of right join on current accounts and branch (all branch rows are displayed irrespective of Current account)

| c_account_number | branch_id | branch_name |
|---|---|---|
| 150000000 | 13453 | Vellore branch |
| 150000001 | 10000 | Perungudi branch |
| 150000002 | 10000 | Perungudi branch |
| 150000011 | 11000 | South delhi branch |
| *NULL* | 12000 | Lal bazar branch |
| *NULL* | 13452 | Girinagar branch |

5) Illustration of left join on loan accounts and branch (query looks similar to inner join because all loan accounts mandatory to have branch id)

| loan_account_number | branch_id | branch_name |
|---|---|---|
| 242000000 | 11000 | South delhi branch |
| 242000001 | 11000 | South delhi branch |
| 242000004 | 10000 | Perungudi branch |

## 5.4 PL/SQL

### PROCEDURES (3 Questions)

Q1.CREATE A PROCEDURE THAT CHANGES THE ACCOUNT NAME TO "CURRENT_ACCOUNT" FOR A PARTICULAR INPUT FOR CURRENT ACCOUNT.

SQL Worksheet | Clear | Find | Actions | Save | Run

```
1  create or replace procedure namechange(i_user_id number,account_number number) is
2  a current_account.account_name%type;
3  begin
4  update current_account set account_name='current_account' where user_id=i_user_id and c_account_number=account_number;
5  select account_name into a from current_account where user_id=i_user_id and c_account_number=account_number;
6  DBMS_OUTPUT.PUT_LINE('New account name= '||a);
7  end;/
8  select * from current_account;
9  exec namechange(150000001, 1000000050);
```

```
Procedure created.
```

| C_ACCOUNT_NUMBER | USER_ID | CREATED_ON | BRANCH_ID | ACCOUNT_NAME | BALANCE | COMPANY_NAME | MAX_OD |
|---|---|---|---|---|---|---|---|
| 150000000 | 1000000049 | 05-OCT-20 03.32.48.000000 PM | 13453 | current_acc | 10000 | l and k logistics | 10000 |
| 150000001 | 1000000050 | 05-OCT-20 03.34.53.000000 PM | 10000 | current | 3000 | accenture | 10000 |
| 150000002 | 1000000048 | 05-OCT-20 03.35.58.000000 PM | 10000 | current account | 32000 | hcl ltd | 10000 |

Q2.CREATE A PROCEDURE THAT CHANGES THE ACCOUNT NAME TO "my fixed deposit" FOR A PARTICULAR INPUT for fixed deposit.

SQL Worksheet | Clear | Find | Actions | Save | Run

```
1  create or replace procedure fdnamechange(account_number number) is
2  a fixed_deposits.account_name%type;
3  begin
4  update fixed_deposits set account_name='my fixed deposit' where fd_account_number=account_number;
5  select account_name into a from fixed_deposits where fd_account_number=account_number;
6  DBMS_OUTPUT.PUT_LINE('New account name= '||a);
7  end;
8  /
9
```

```
Procedure created.
```

Q3: CHANGE ACCOUNT NAME TO “MY_LOAN” IN LOAN_ACCOUNTS.

SQL Worksheet  Clear  Find  Actions  Save  Run

```
1  create or replace procedure loannamechange(account_number number) is
2  a loan_accounts.account_name%type;
3  begin
4  update loan_accounts set account_name='my loan' where loan_account_number=account_number;
5  select account_name into a from loan_accounts where loan_account_number=account_number;
6  DBMS_OUTPUT.PUT_LINE('New account name= '||a);
7  end;
8  /
9
```

```
Procedure created.
```

**FUNCTIONS (3 questions)**

Q4.Count the number of customers using a function

SQL Worksheet  Clear  Find  Actions  Save  Run

```
1  create or replace function totalcustomers
2  return number is
3      total number(2):=0;
4  BEGIN
5  select count(*) into total from customers;
6  dbms_output.put_line('the total number of customers is '||total);
7  return total;
8  end;
9  /
10
11 declare
12 c number(2);
13 begin
14 c:=totalcustomers();
15 end;
16 /
```

```
Function created.
```

SQL Worksheet

Clear Find Actions Save Run

```
1  create or replace function totalcustomers
2  return number is
3      total number(2):=0;
4  BEGIN
5  select count(*) into total from customers;
6  dbms_output.put_line('the total number of customers is '||total);
7  return total;
8  end;
9  /
10
11 declare
```

```
Function created.


Statement processed.
the total number of customers is 4
```

Q5.FIND THE BRANCH ID OF A GIVEN USERID AS INPUT USING A FUNCTION.(USE SUBQUERY TO SELECT)

SQL Worksheet

Clear Find Actions Save Run

```
1  CREATE OR REPLACE FUNCTION BRANCHNAME(X IN customers.userid%TYPE)
2  RETURN VARCHAR2 IS BR_NAME BRANCH.BRANCH_NAME%TYPE;
3  BEGIN
4  SELECT BRANCH_NAME INTO BR_NAME FROM BRANCH WHERE BRANCH_ID IN(SELECT BRANCH_ID FROM CUSTOMERS WHERE USERID=X);
5  DBMS_OUTPUT.PUT_LINE('THE BRANCH NAME IS '||BR_NAME);
6  RETURN BR_NAME;
7  END;
8  /
9
10 DECLARE
11 USER_ID NUMBER;
12 VARIABLENAME VARCHAR2(50);
13 BEGIN
14 USER_ID:=1000000049;
15 VARIABLENAME:=BRANCHNAME(USER_ID);
16 END;
```

```
Function created.
```

Q6.SHOW THE TOTALSPENDABLE AMOUNT FOR A PARTICULAR USERID WHO HAS A CURRENT ACCOUNT.

SQL Worksheet

Clear Find Actions Save Run

```
1  CREATE OR REPLACE FUNCTION TOTSPENDABLE(X IN customers.userid%TYPE)
2  RETURN NUMBER IS TOT_SPEN current_transactions.total_spendable%TYPE;
3  BEGIN
4  SELECT TOTAL_SPENDABLE INTO TOT_SPEN FROM current_transactions WHERE account_number IN(SELECT c_account_number FROM
5  current_account WHERE USER_ID=X);
6  DBMS_OUTPUT.PUT_LINE('THE total spendable amount for given user is '||tot_spen);
7  return tot_spen;
8  END;/
9  DECLARE
10 USER_ID NUMBER;
11 c number;
12 BEGIN
13 USER_ID:=1000000049;
14 c:=totspendable(USER_ID);
15 END;/
```

```
Function created.


Statement processed.
THE total spendable amount for given user is 20000
```

**CURSORS (1-implicit;1-explicit total= 2 questions)**

**Implicit**

Q7:USE IMPLICIT CURSORS TO CHANGE ALL PASSWORDS FOR CUSTOMERS TO ABCD AND ALSO DISPLAY THE NUMBER OF ROWS AFFECTED.

SQL Worksheet

Clear Find Actions Save Run

```
1  DECLARE
2     total_rows number(2);
3  BEGIN
4     UPDATE customers SET password = 'abcd';
5     IF sql%notfound THEN
6        dbms_output.put_line('no customers selected');
7     ELSIF sql%found THEN
8        total_rows := sql%rowcount;
9        dbms_output.put_line( total_rows || ' customers selected ');
10    END IF;
11 END;
12 /
13
```

```
Statement processed.
4 customers selected
```

**Explicit**

Q8.DISPLAY CUSTOMER MAIL,NAME AND USERID USING CURSORS

SQL Worksheet

Clear Find Actions Save Run

```
DECLARE
   c_id customers.userid%type;
   c_name customers.customer_name%type;
   c_mail customers.email%type;
   CURSOR c_customers is
      SELECT userid, customer_name, email FROM customers;
BEGIN
   OPEN c_customers;
   LOOP
   FETCH c_customers into c_id, c_name, c_mail;
      EXIT WHEN c_customers%notfound;
      dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_mail);
   END LOOP;
   CLOSE c_customers;
END; /
```

```
Statement processed.
1000000048 ARSHDEEP SINGH BHATIA arshdeepdgreat@gmail.com
1000000049 IRENE JOHN irene.john2019@vitstudent.ac.in
1000000050 RIYA riya@vitstudent.ac.in
1000000051 KARTIK kartik@gmail.com
```

**TRIGGERS (2 questions)**

Q9.CREATE A TRIGGER THAT DISPLAYS THE OLD AND NEW PASSWORD WHENEVER A PASSWORD IN CUSTOMER TABLE IS CHANGED.

SQL Worksheet

Clear Find Actions Save Run

```
create or replace trigger funds
before update on savings_accounts
for each row
declare
fund_diff number;
begin
fund_diff:=:new.balance-:old.balance;
dbms_output.put_line('old balance= '||:old.balance);
dbms_output.put_line('new balance= '||:new.balance);
dbms_output.put_line('difference in fund is '||fund_diff);
end;
/
update savings_accounts
set savings_accounts.balance=190002 where s_account_number=1000000001;
```

```
1 row(s) updated.
old balance= 5000
new balance= 190002
difference in fund is 185002
```

Q10.CREATE A TRIGGER THAT DISPLAYS A DIFFERENCE IN THE BALANCE IF A SAVINGS ACCOUNT IS UPDATED

SQL Worksheet

Clear Find Actions Save Run

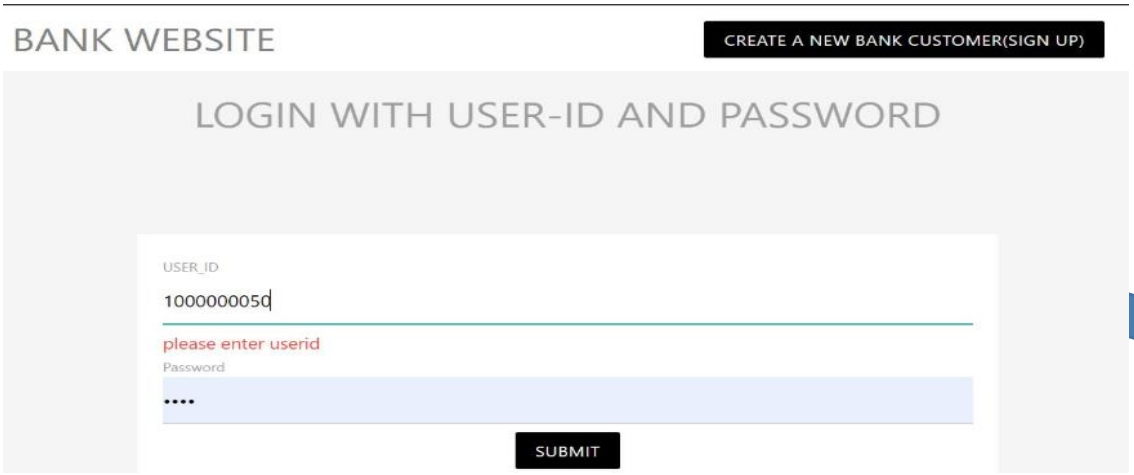
```
2  before update on customers
3  for each row
4  begin
5  dbms_output.put_line('old password= '||:old.password);
6  dbms_output.put_line('new password= '||:new.password);
7  end;
8  /
9  update customers
10 set password='ire0205' where customer_name='IRENE JOHN';
11
12
```

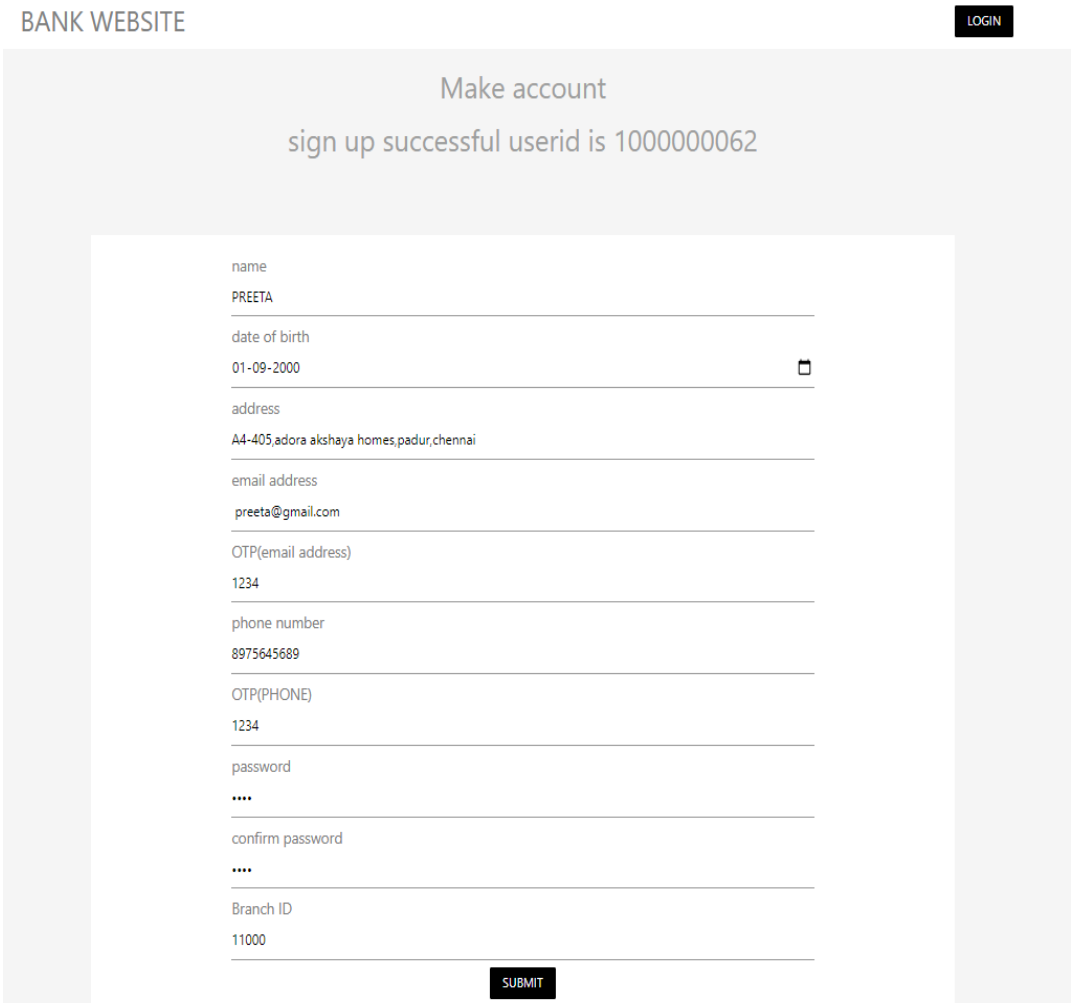
```
Trigger created.


1 row(s) updated.
old password= 0000
new password= ire0205
```

# 6. SCREENSHOTS (FRONT END-WITH EXPLANATION)

## 6.1 Login screen

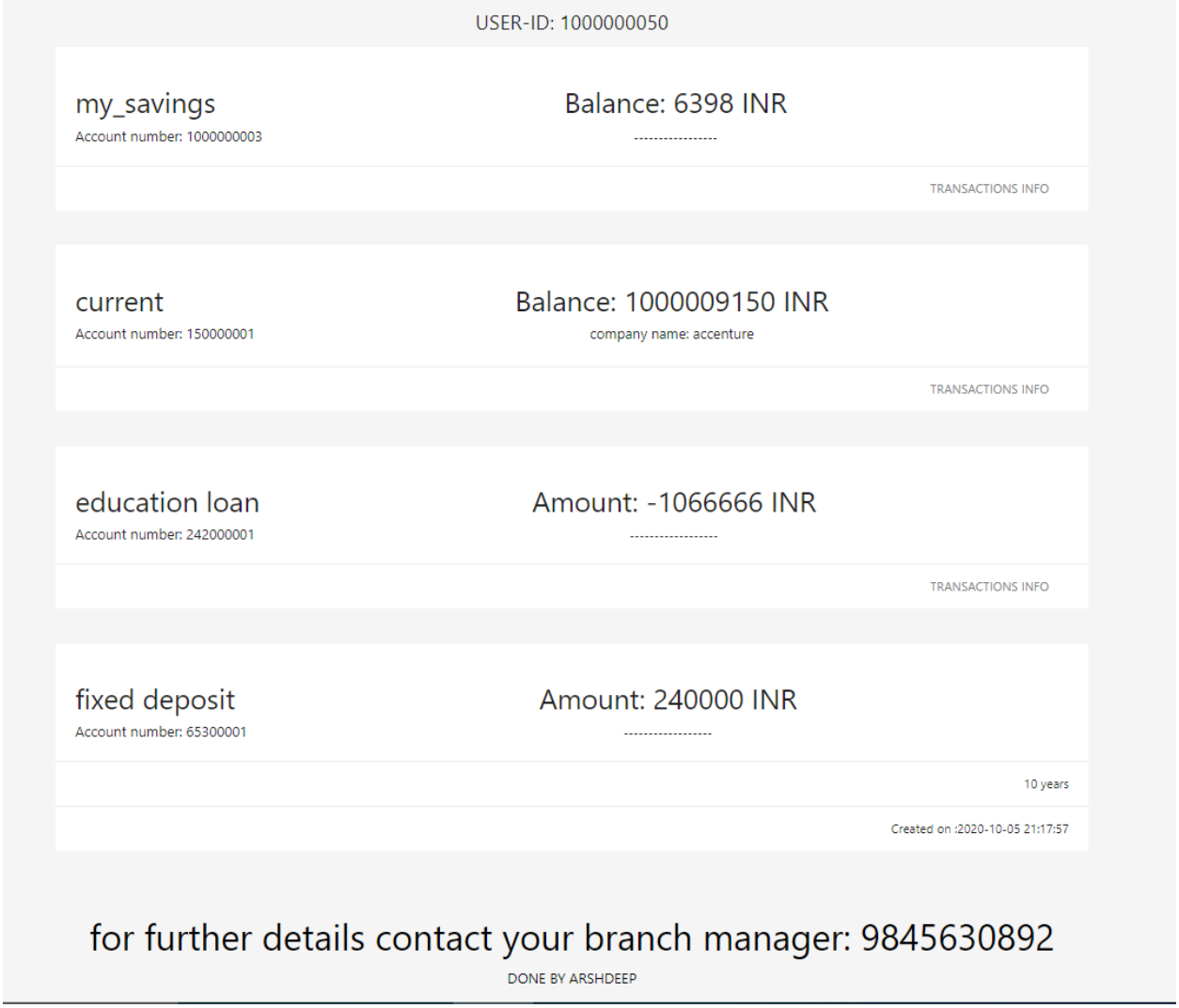



We register as a customer Preeta.

**On successful sign in**

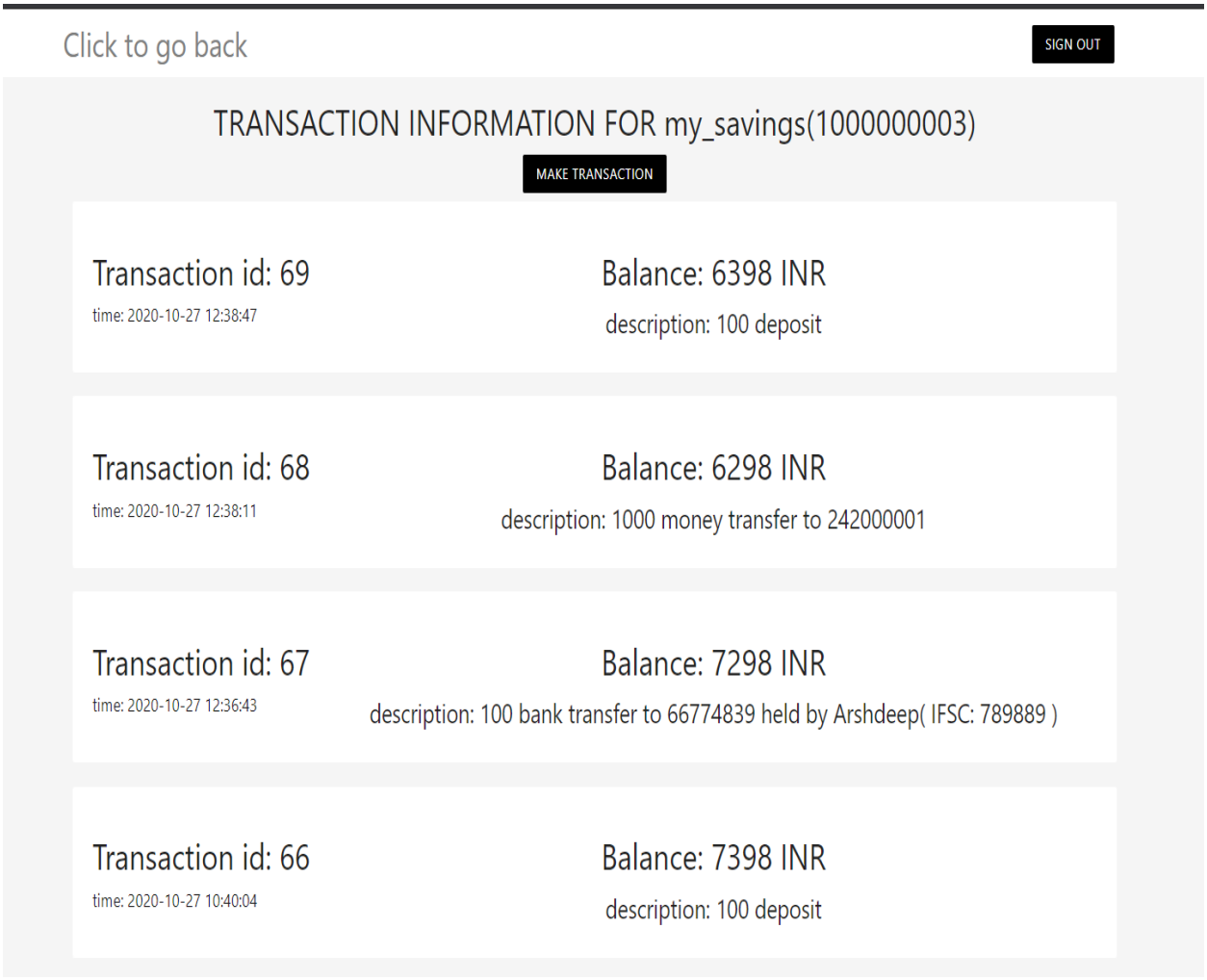
**Accounts view**



**All transactions read for “my_savings” account**

Displayed values

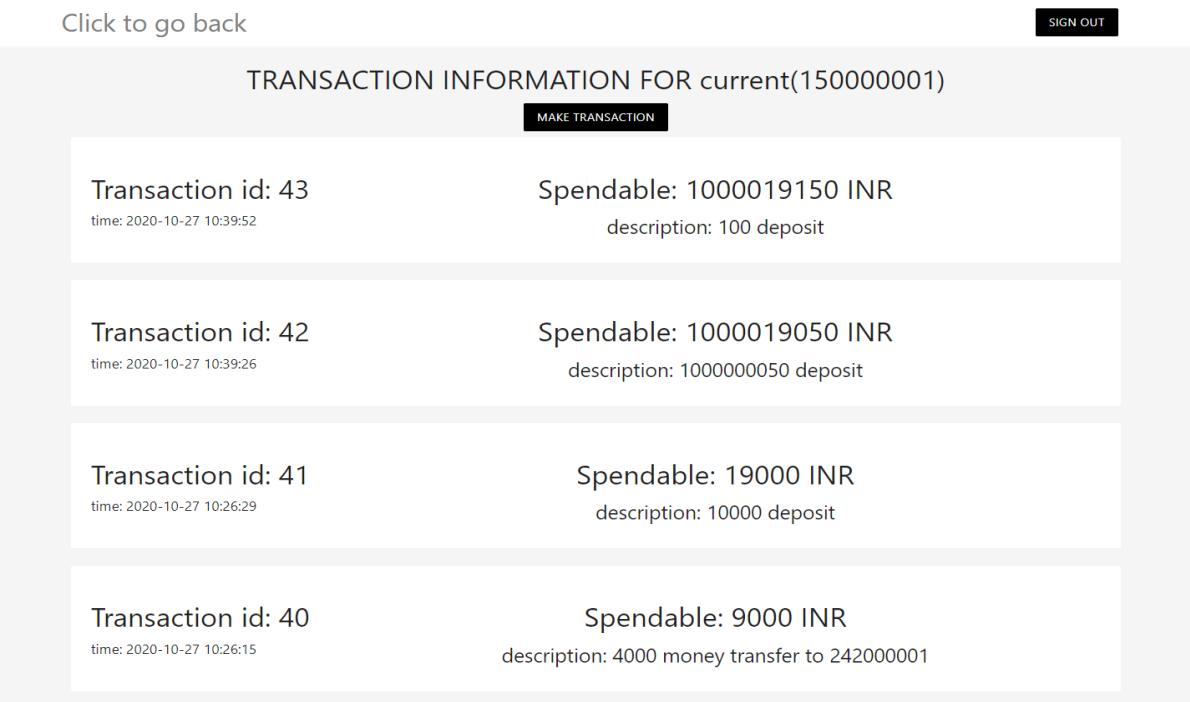
1. Transaction id
2. Balance after transaction
3. Transaction timestamp
4. Transaction description



**All transactions read for “current” account**

Displayed values

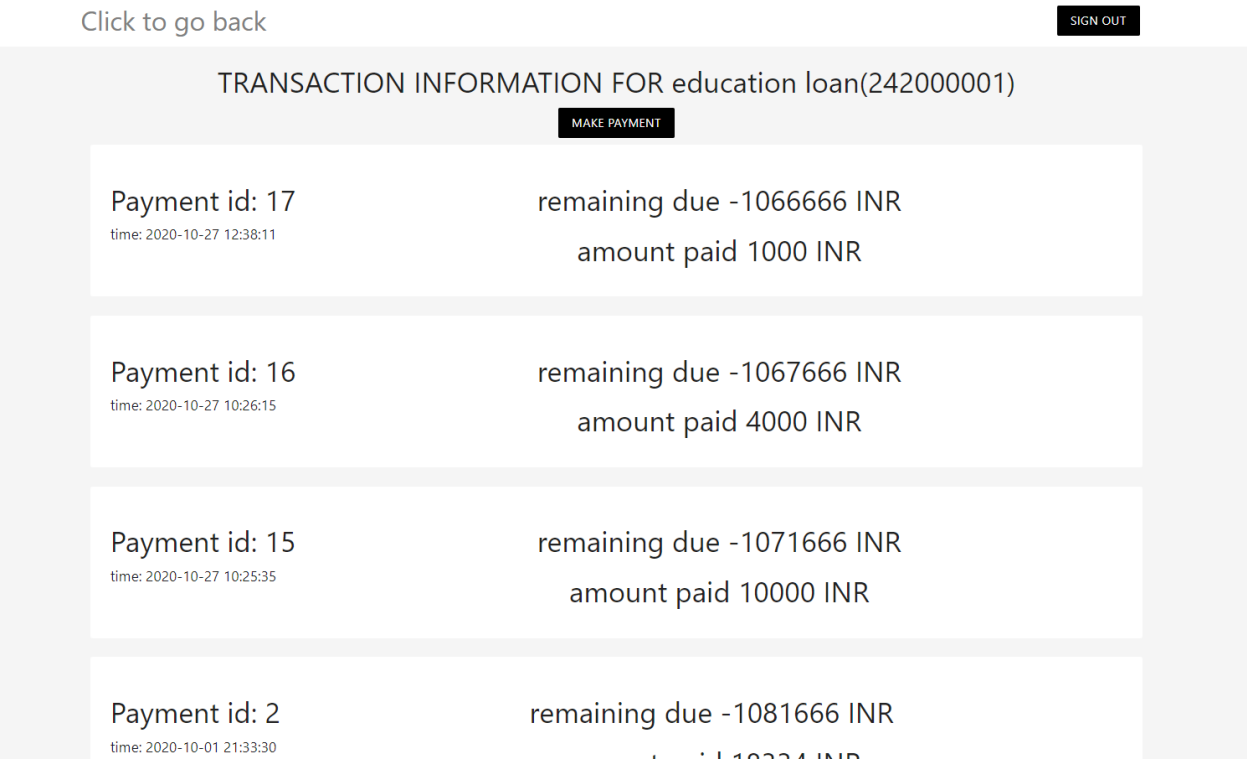
1. Transaction id
2. Balance after transaction
3. Transaction timestamp
4. Transaction description



**All payments on loan account**

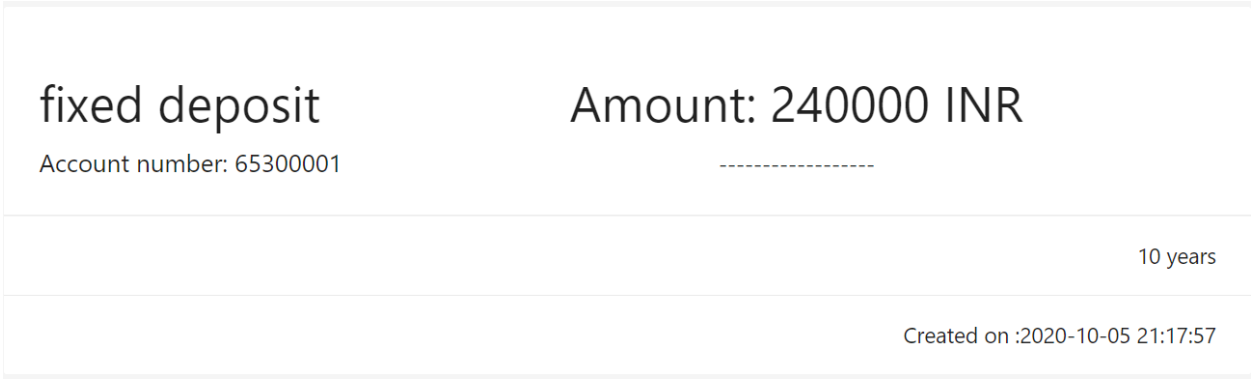
Displayed values

1. Payment id
2. Remaining Due after transaction
3. Transaction timestamp
4. Amount paid



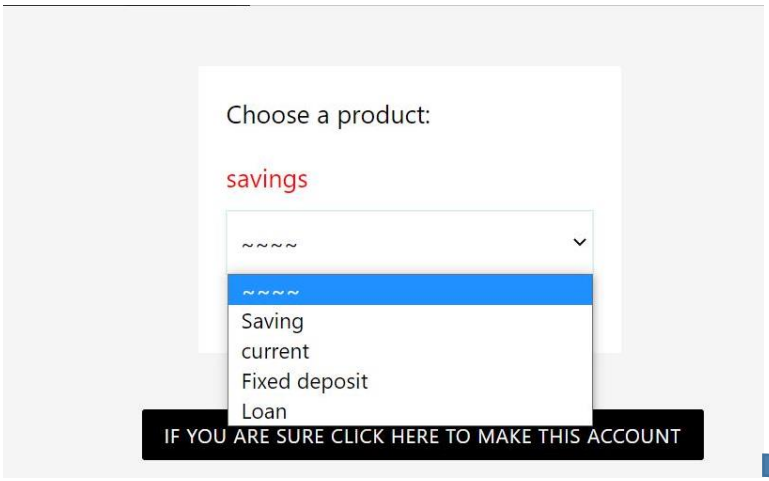
**Fixed Deposits**

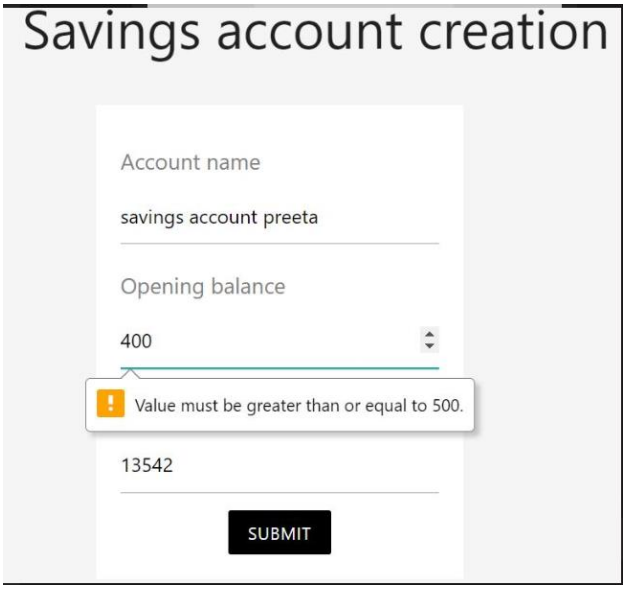
1. amount on maturity
2. account number
3. duration of FD
4. creation timestamp



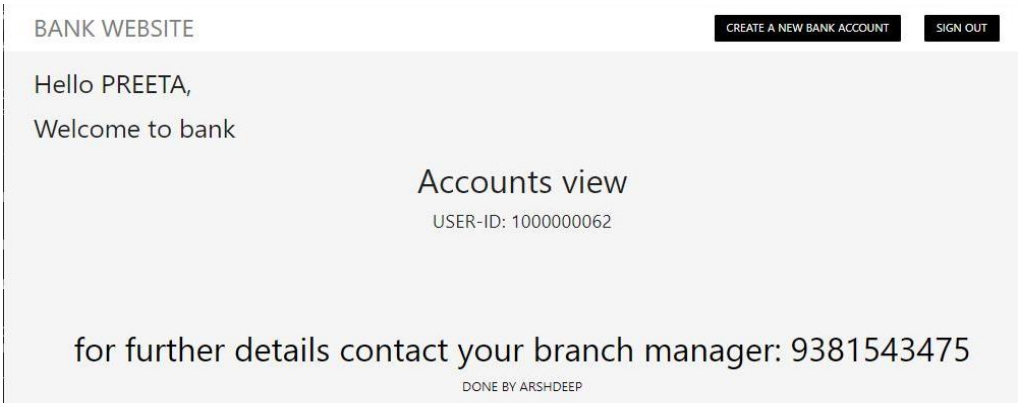
## 6.2 Savings Account creation

We create a Savings Account.

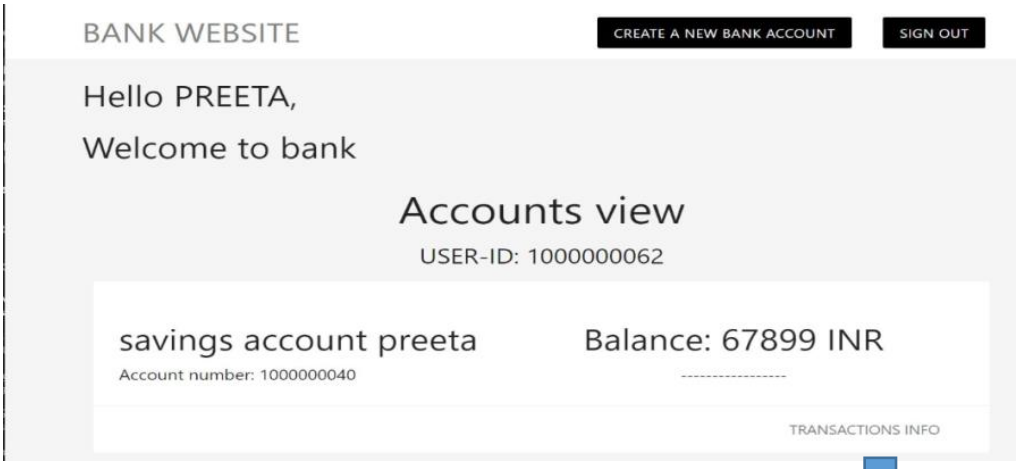


**Before Savings Account Is Created**

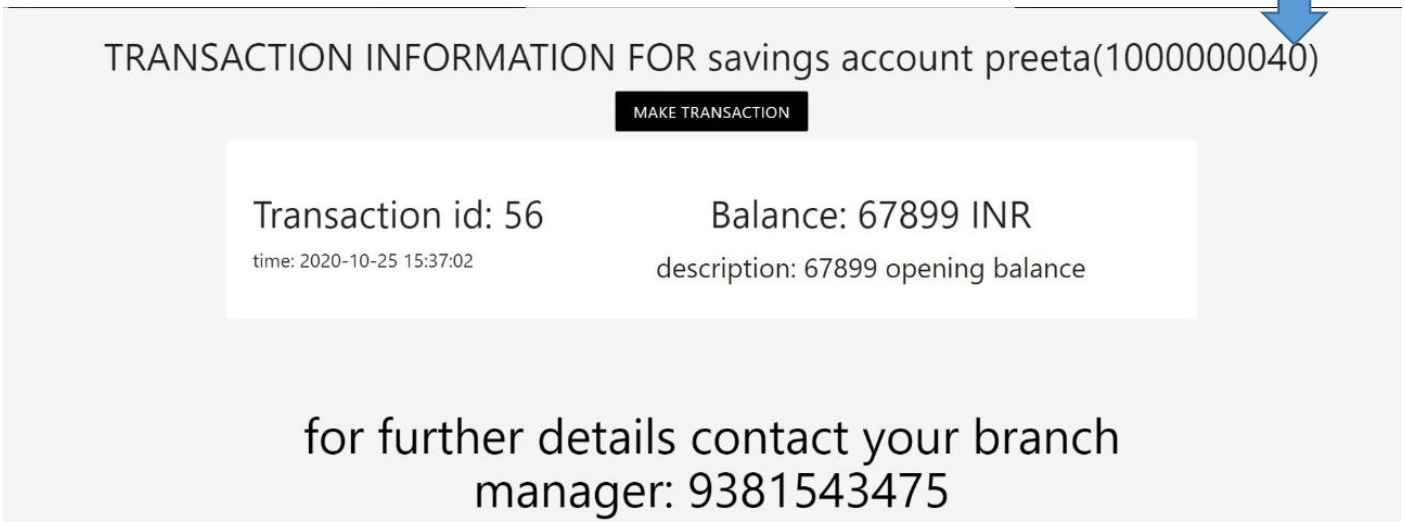


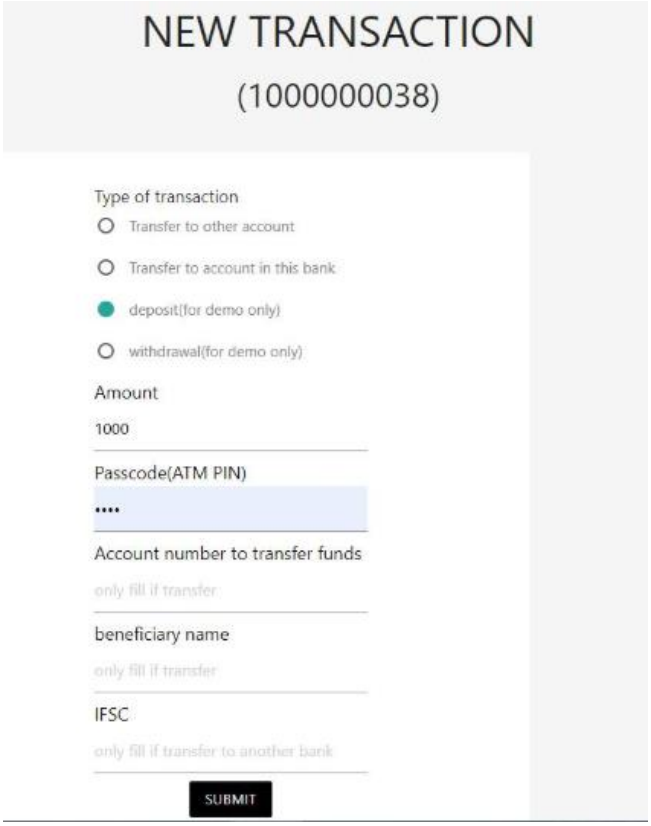

**After Savings Account Is Created**




## 6.3 Transactions

"Transaction Info" Opens To



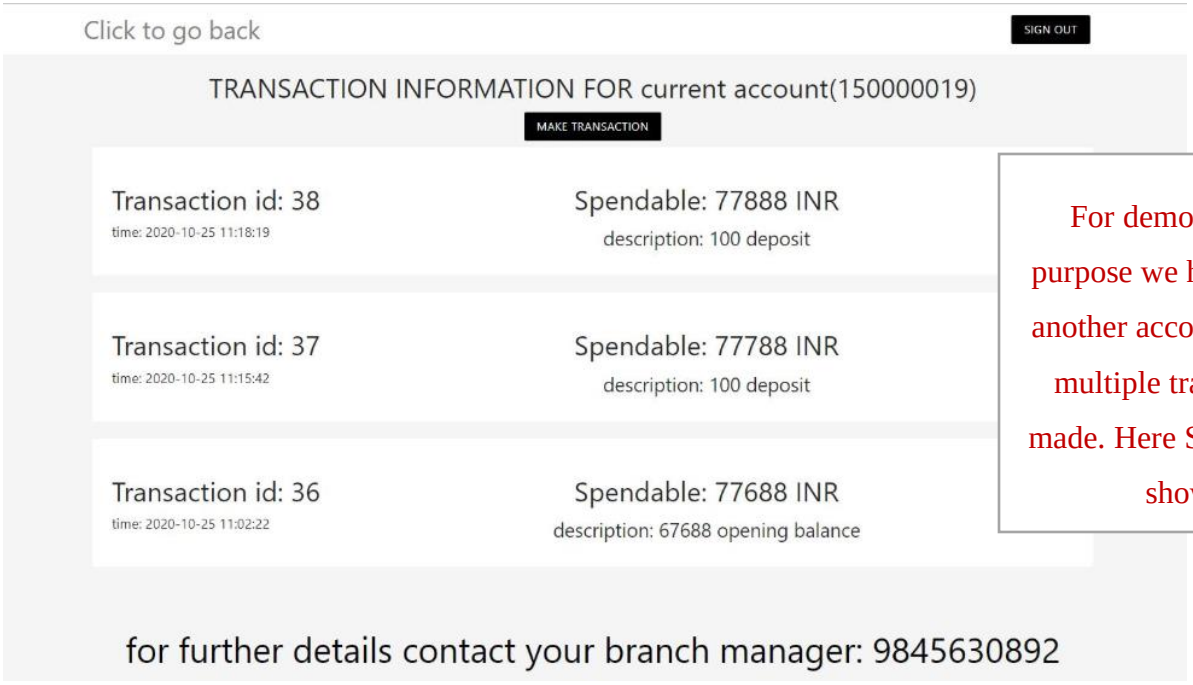



Make Transaction (Refer previous screen shot) opens to…

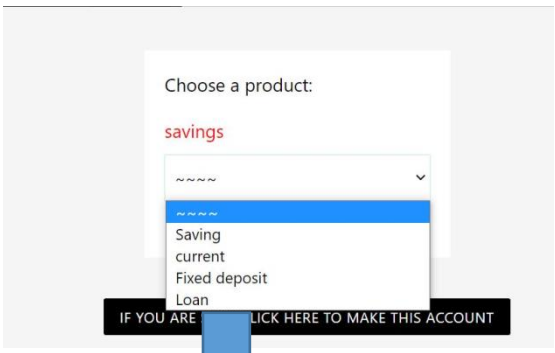
Here there are 4 different types:

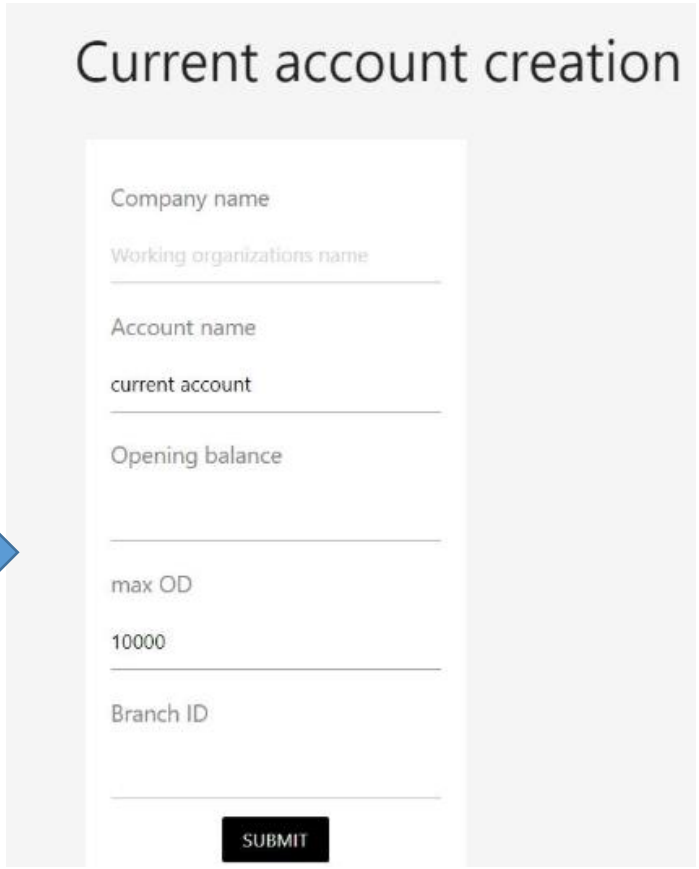
Deposit and Withdrawal works only when the user manually performs it in the bank, Here it is shown for demonstration purpose

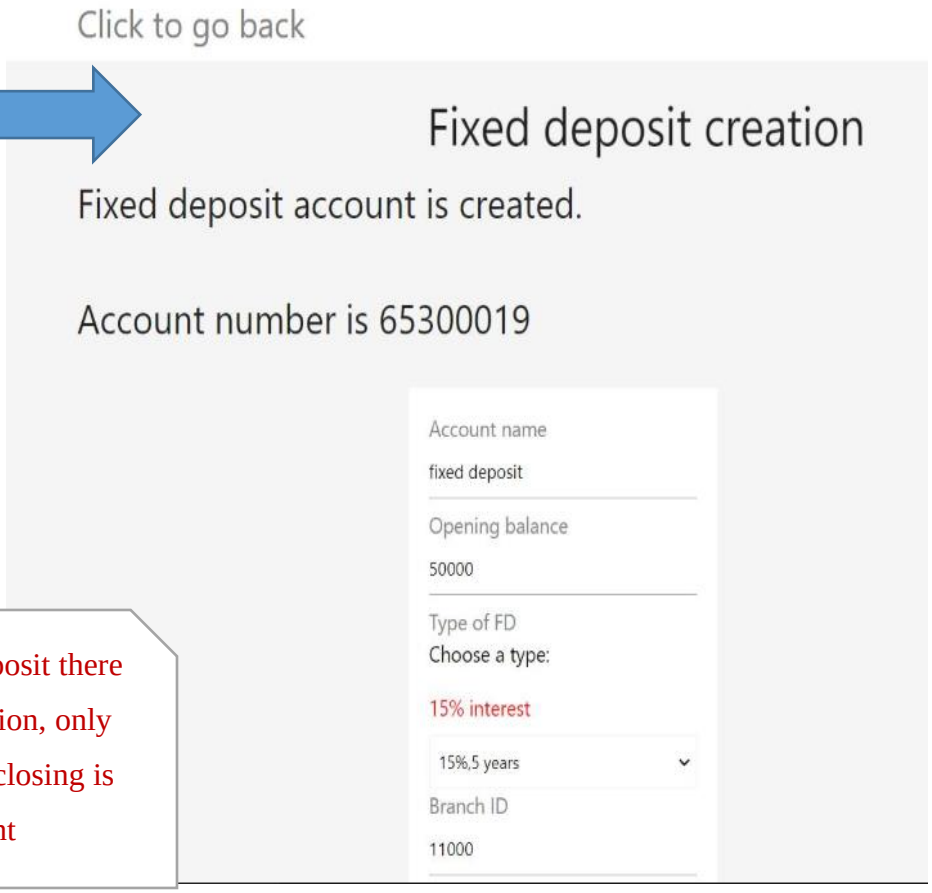
Transaction updated in database and frontend



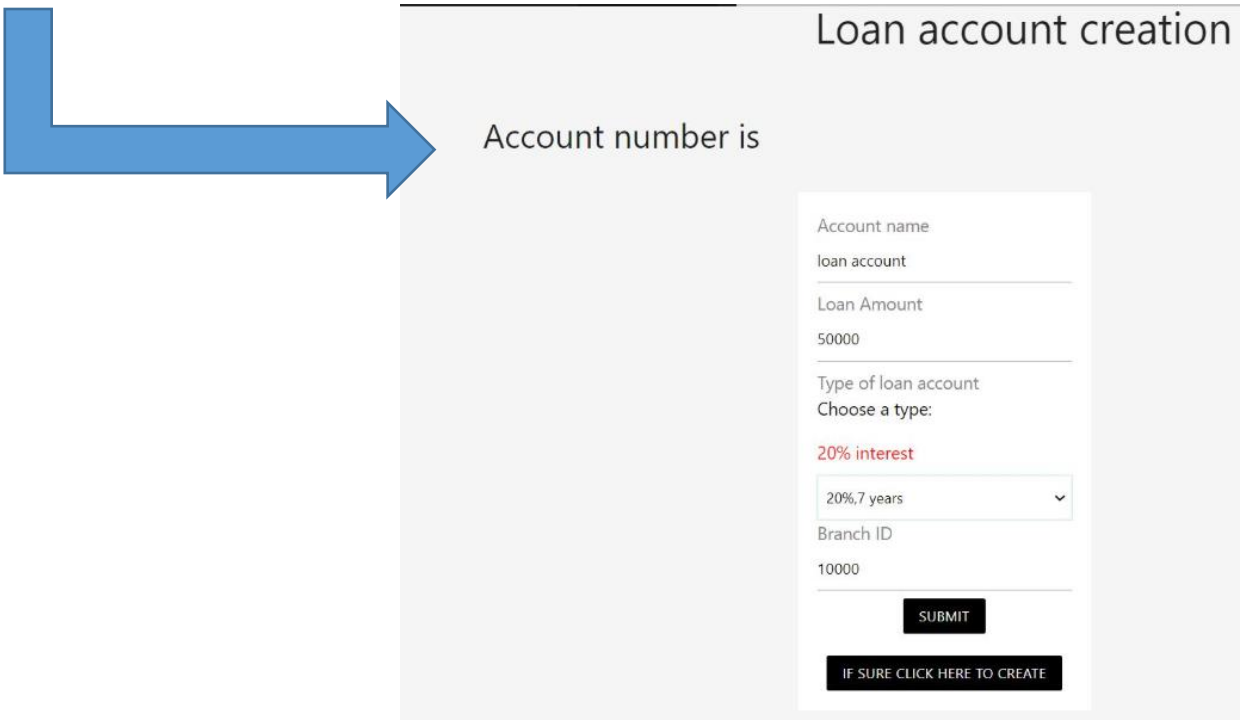

For demonstration purpose we have chosen another account to show multiple transactions made. Here Spendable is shown



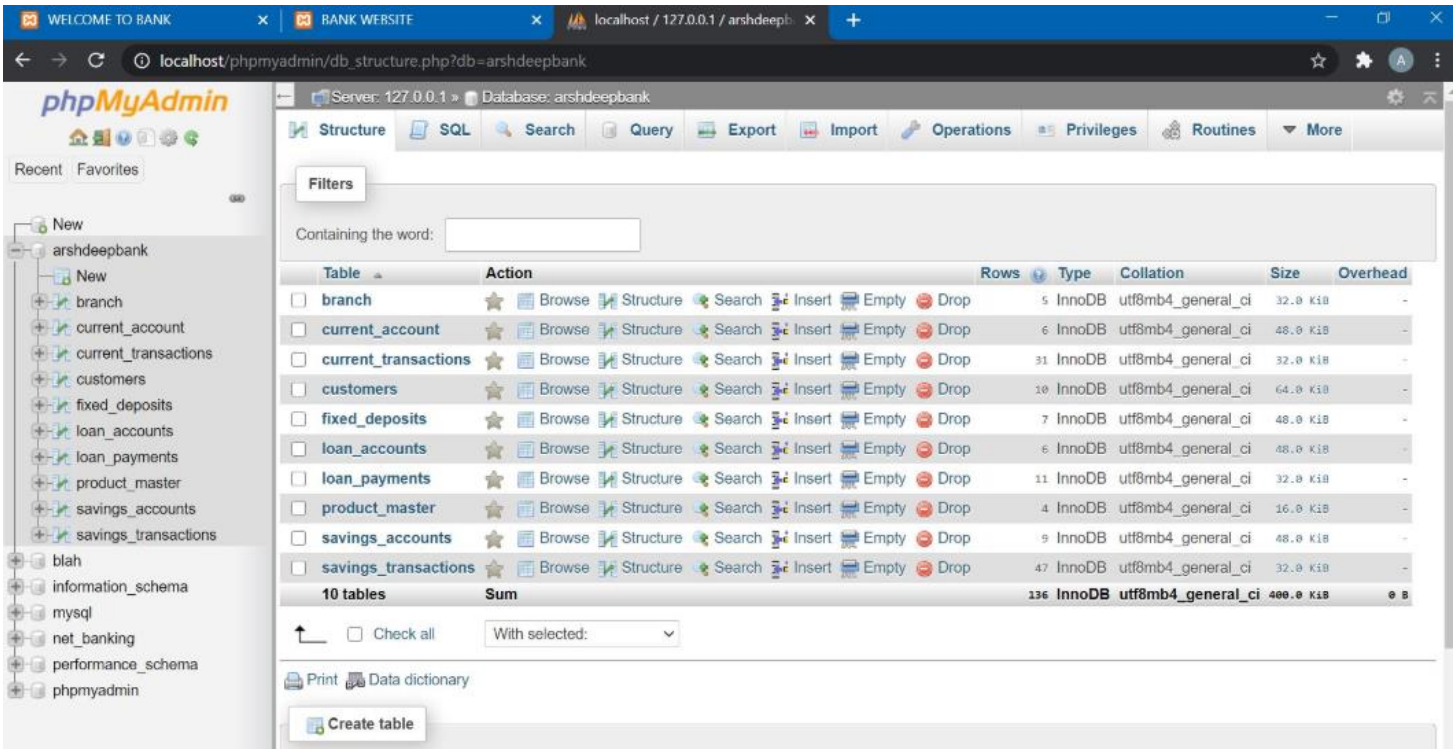




For Fixed Deposit there is no transaction, only opening and closing is present


Loan account creation
Account number is
Account name
loan account
Loan Amount
50000
Type of loan account
Choose a type:
20% interest
20%,7 years
Branch ID
10000
SUBMIT
IF SURE CLICK HERE TO CREATE

## Back-end: database view




1. **Branch table**

**Structure**

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | branch_id | int(5) | | | No | *None* | | AUTO_INCREMENT | Change Drop More |
| 2 | branch_address | text | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 3 | branch_name | varchar(30) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| 4 | branch_mgr_contact | bigint(10) | | | No | *None* | | | Change Drop More |

**Snapshot of data**

| branch_id | branch_address | branch_name | branch_mgr_contact |
|---|---|---|---|
| 10000 | S-110/111, RRR building, L&T Rd, Burma Colony, Thi... | Perungudi branch | 9845630892 |
| 11000 | A-2, First Floor, Ring Rd, Block C, South Extensio... | South delhi branch | 9381543475 |
| 12000 | 3/1, 4th Floor, RN Mukherjee Rd, Dal Housie, Lal B... | Lal bazar branch | 8757391603 |
| 13452 | 588, 2nd Main Rd, 2nd Phase, Hosakerehalli Layout,... | Girinagar branch | 6875484849 |
| 13453 | No 25, 1st East Main Road Gandhi Nagar, Katpadi, V... | Vellore branch | 7836657390 |

## 2. Customers table structure

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | **userid** | int(10) | | | No | *None* | | AUTO_INCREMENT |
| 2 | **password** | varchar(25) | utf8mb4_general_ci | | No | *None* | | |
| 3 | **customer_name** | varchar(30) | utf8mb4_general_ci | | No | *None* | | |
| 4 | **email** | varchar(50) | utf8mb4_general_ci | | No | *None* | | |
| 5 | **phone** | varchar(10) | utf8mb4_general_ci | | No | *None* | | |
| 6 | **address** | varchar(100) | utf8mb4_general_ci | | No | *None* | | |
| 7 | **date_of_birth** | date | | | No | *None* | | |
| 8 | **created_on** | timestamp | | | No | current_timestamp() | | ON UPDATE CURRENT_TIMESTAMP() |
| 9 | **branch_id** | int(10) | | | No | 0 | | |

## Foreign key constraints

| | | Database | Table | Column |
|---|---|---|---|---|
| branchfk | branch_id | arshdeepbank | branch | branch_id |
| | + Add column | | | |

ON DELETE RESTRICT

ON UPDATE RESTRICT

## Snapshot of data

| userid | password | customer_name | email | phone | address |
|---|---|---|---|---|---|
| 1000000048 | Arshdeep123 | ARSHDEEP SINGH BHATIA | arshdeepdgreat@gmail.com | 8754541603 | A4-405,adora akshaya homes,padur,chennai |
| 1000000049 | 0000 | IRENE JOHN | irene.john2019@vitstudent.ac.in | 9840849927 | HOUSE-05,gandhi nagar NAGAR,vellore,640202 |
| 1000000050 | 0000 | RIYA | riya@vitstudent.ac.in | 8974652672 | house 5,palm homes,Gandhi Nagar, Adyar, Chennai, T... |
| 1000000051 | 0000 | KARTIK | kartik@gmail.com | 8926647489 | house 205,Mandakini Apartment,Pocket 2, Sector 2 D... |
| 1000000053 | 0000 | JOHN | john@vit.ac.in | 7399366492 | A4-405,adora akshaya homes,padur,chennai |
| 1000000054 | 0000 | TEJINDER KAUR BHATIA | gbtk@yahoo.com | 9840037871 | A4-405,adora akshaya homes,padur,chennai |
| 1000000055 | 0000 | SWASTIK DAS | swastik.d@yahoo.com | 9840037856 | HOUSE 10,Palm homes,Adyar,chennai,600408 |

3. **Product master**

| Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|
| product_id | int(4) | | | No | *None* | | AUTO_INCREMENT |
| product_name | varchar(30) | utf8mb4_general_ci | | No | *None* | | |
| description | text | utf8mb4_general_ci | | No | *None* | | |

## Data snapshot

| product_id | product_name | description |
|---|---|---|
| 100 | fixed deposit | Fixed deposits will be offered with a given rate o... |
| 101 | savings account | savings account provides an account for the holder... |
| 120 | Loan account | home loan,car loan,education loan can be provision... |
| 151 | current account | current account can be provisioned for employees o... |

4. **Current accounts**

| Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|
| c_account_number | int(10) | | | No | *None* | | AUTO_INCREMENT |
| user_id | int(10) | | | No | *None* | | |
| created_on | timestamp | | | No | current_timestamp() | | |
| branch_id | int(11) | | | No | *None* | | |
| account_name | varchar(50) | utf8mb4_general_ci | | No | *None* | | |
| balance | int(11) | | | No | *None* | | |
| company_name | varchar(30) | utf8mb4_general_ci | | No | *None* | | |
| max_od | int(11) | | | No | *None* | | |

## Foreign key Constraints

| Constraint properties | Column | Foreign key constraint (INNODB) | | |
|---|---|---|---|---|
| | | Database | Table | Column |
| c_userid | user_id | arshdeepbank | customers | userid |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |
| ca_branch | branch_id | arshdeepbank | branch | branch_id |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |

## Data snapshot

| c_account_number | user_id | created_on | branch_id | account_name | balance | company_name | max_od |
|---|---|---|---|---|---|---|---|
| 150000000 | 1000000049 | 2020-10-05 21:02:48 | 13453 | current_acc | 10000 | l and k logistics | 10000 |
| 150000001 | 1000000050 | 2020-10-05 21:04:53 | 10000 | current | 1000009150 | accenture | 10000 |
| 150000002 | 1000000048 | 2020-10-05 21:05:58 | 10000 | current account | 32000 | hcl ltd | 10000 |
| 150000011 | 1000000053 | 2020-10-13 11:25:21 | 11000 | current account john | 10100 | accenture | 10000 |
| 150000014 | 1000000060 | 2020-10-24 23:26:39 | 13452 | current account | 9690 | FIITJEE | 5000 |
| 150000019 | 1000000055 | 2020-10-25 11:02:22 | 13453 | current account | 67888 | tata consultancy services | 10000 |

## 5. Current transactions

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | trans_id | int(10) | | | No | None | | AUTO_INCREMENT |
| 2 | account_number | int(11) | | | No | None | | |
| 3 | timestamp | timestamp | | | No | current_timestamp() | | |
| 4 | amount | int(11) | | | No | None | | |
| 5 | transaction_description | varchar(100) | utf8mb4_general_ci | | No | None | | |
| 6 | total_spendable | int(11) | | | No | None | | |

## Constraints

| Constraint properties | Column | Foreign key constraint (INNODB) | | |
|---|---|---|---|---|
| | | Database | Table | Column |
| curfk | account_numbe | arshdeepbank | current_account | c_account_numl |
| ON DELETE CASCADE | + Add column | | | |
| ON UPDATE RESTRICT | | | | |

## Snapshot of data

| trans_id | account_number | timestamp | amount | transaction_description | total_spendable |
|---|---|---|---|---|---|
| 1 | 150000000 | 2020-10-05 21:08:16 | 10000 | 10000 opening balance | 20000 |
| 2 | 150000001 | 2020-10-05 21:08:53 | 10000 | 10000 opening balance | 20000 |
| 3 | 150000002 | 2020-10-05 21:09:48 | 20000 | 20000 opening balance | 30000 |
| 4 | 150000001 | 2020-10-05 21:11:19 | -7000 | 7000 withdrawal | 13000 |
| 5 | 150000002 | 2020-10-05 21:12:15 | 12000 | 12000 deposit | 42000 |
| 8 | 150000011 | 2020-10-13 11:25:21 | 10000 | 10000 opening balance | 20000 |
| 11 | 150000014 | 2020-10-24 23:26:39 | 10000 | 10000 opening balance | 15000 |
| 13 | 150000014 | 2020-10-24 23:54:58 | 1000 | 1000 deposit | 16000 |
| 14 | 150000014 | 2020-10-24 23:56:27 | 100 | 100 deposit | 16100 |
| 15 | 150000014 | 2020-10-24 23:56:47 | -300 | 300 withdrawal | 15800 |
| 16 | 150000014 | 2020-10-24 23:58:37 | -100 | 100 bank transfer to 887248992 held by Gurbir Sing... | 15700 |

## 6. Fixed Deposits

### Structure

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | fd_account_number | int(10) | | | No | *None* | | AUTO_INCREMENT |
| 2 | user_id | int(10) | | | No | *None* | | |
| 3 | account_name | varchar(30) | utf8mb4_general_ci | | No | *None* | | |
| 4 | bond_duration | int(2) | | | No | *None* | | |
| 5 | rate_of_intrest | int(1) | | | No | *None* | | |
| 6 | created_on | timestamp | | | No | current_timestamp() | | |
| 7 | final_amount | int(11) | | | No | *None* | | |
| 8 | branch_id | int(5) | | | No | *None* | | |

### Foreign keys

| Constraint properties | Column | Foreign key constraint (INNODB) | | |
|---|---|---|---|---|
| | | Database | Table | Column |
| fd_branch | branch_id | arshdeepbank | branch | branch_id |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |
| fd_userid | user_id | arshdeepbank | customers | userid |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |

### Snapshot of data

| fd_account_number | user_id | account_name | bond_duration | rate_of_intrest | created_on | final_amount | branch_id |
|---|---|---|---|---|---|---|---|
| 65300000 | 1000000051 | fixed deposit(7yr) | 7 | 15 | 2020-10-05 21:16:46 | 115000 | 11000 |
| 65300001 | 1000000050 | fixed deposit | 10 | 20 | 2020-10-05 21:17:57 | 240000 | 10000 |
| 65300012 | 1000000053 | fixed deposit | 7 | 17 | 2020-10-15 10:28:18 | 81900 | 10000 |
| 65300015 | 1000000060 | fixed deposit | 5 | 15 | 2020-10-24 23:25:56 | 57500 | 10000 |
| 65300018 | 1000000055 | fixed deposit | 5 | 15 | 2020-10-25 11:08:17 | 57500 | 13453 |

## 7. Loan accounts

### Structure

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | loan_account_number | int(10) | | | No | *None* | | AUTO_INCREMENT |
| 2 | created_on | timestamp | | | No | current_timestamp() | | |
| 3 | user_id | int(10) | | | No | *None* | | |
| 4 | branch_id | int(11) | | | No | *None* | | |
| 5 | account_name | varchar(50) | utf8mb4_general_ci | | No | *None* | | |
| 6 | amount | int(11) | | | No | *None* | | |
| 7 | rate_of _intrest | int(2) | | | No | *None* | | |
| 8 | duration_of_loan | int(2) | | | No | *None* | | |

### Foreign key

| Constraint properties | Column | Foreign key constraint (INNODB) | | |
|---|---|---|---|---|
| | | Database | Table | Column |
| loan_branch | branch_id | arshdeepbank | branch | branch_id |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |
| loan_userid | user_id | arshdeepbank | customers | userid |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |

### Snapshot of data

| loan_account_number | created_on | user_id | branch_id | account_name | amount | rate_of _intrest | duration_of_loan |
|---|---|---|---|---|---|---|---|
| 242000000 | 2020-08-05 21:24:37 | 1000000051 | 11000 | home loan | -1180000 | 20 | 7 |
| 242000001 | 2020-08-05 21:24:37 | 1000000050 | 11000 | education loan | -1066666 | 15 | 5 |
| 242000004 | 2020-10-15 11:36:32 | 1000000053 | 10000 | loan account | -57400 | 15 | 5 |
| 242000006 | 2020-10-24 23:23:55 | 1000000060 | 10000 | loan account | -48800 | 20 | 7 |
| 242000007 | 2020-10-25 11:11:45 | 1000000055 | 13453 | loan account | -53240 | 15 | 5 |
| 242000008 | 2020-10-25 15:51:54 | 1000000055 | 10000 | loan account | -60000 | 20 | 7 |

8. **Loan Payments**

## Structure

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | pay_no | int(10) | | | No | *None* | | AUTO_INCREMENT |
| 2 | loan_account_number | int(10) | | | No | *None* | | |
| 3 | amount_paid | int(11) | | | No | *None* | | |
| 4 | timestamp_ | timestamp | | | No | current_timestamp() | | |
| 5 | amount | int(11) | | | No | *None* | | |

## Foreign key

| Constraint properties | Column | Foreign key constraint (INNODB) | | |
|---|---|---|---|---|
| | | Database | Table | Column |
| loanfk | loan_account_nı | arshdeepbank | loan_accounts | loan_account_nı |
| ON DELETE CASCADE | + Add column | | | |
| ON UPDATE RESTRICT | | | | |

## Snapshot of data

| pay_no | loan_account_number | amount_paid | timestamp_ | amount |
|---|---|---|---|---|
| 1 | 242000000 | 10000 | 2020-09-01 21:33:30 | -1190000 |
| 2 | 242000001 | 18334 | 2020-10-01 21:33:30 | -1081666 |
| 3 | 242000000 | 10000 | 2020-10-01 21:33:30 | -1180000 |
| 5 | 242000004 | 100 | 2020-10-24 12:52:11 | -57400 |
| 6 | 242000006 | 10000 | 2020-10-24 23:24:42 | -50000 |
| 7 | 242000006 | 100 | 2020-10-25 00:01:54 | -49900 |
| 8 | 242000006 | 1000 | 2020-10-25 00:25:03 | -48900 |
| 9 | 242000006 | 100 | 2020-10-25 00:43:16 | -48800 |
| 10 | 242000007 | 60 | 2020-10-25 15:08:38 | -57440 |
| 11 | 242000007 | 100 | 2020-10-25 15:10:40 | -57340 |
| 12 | 242000007 | 3000 | 2020-10-25 15:45:55 | -54340 |
| 13 | 242000007 | 1000 | 2020-10-26 18:26:20 | -53340 |

9. **Savings Accounts**

## Structure

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | **userid** | int(10) | | | No | *None* | | |
| 2 | **s_account_number** | int(10) | | | No | *None* | | AUTO_INCREMENT |
| 3 | **balance** | int(10) | | | No | *None* | | |
| 4 | **created_on** | timestamp | | | No | current_timestamp() | | |
| 5 | **branch_id** | int(5) | | | No | *None* | | |
| 6 | **account_name** | varchar(30) | utf8mb4_general_ci | | No | savings_account | | |

## Foreign keys

| Constraint properties | Column | Foreign key constraint (INNODB) | | |
|---|---|---|---|---|
| | | Database | Table | Column |
| s_branch | branch_id | arshdeepbank | branch | branch_id |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |
| s_userid | userid | arshdeepbank | customers | userid |
| ON DELETE RESTRICT | + Add column | | | |
| ON UPDATE RESTRICT | | | | |

## Snapshot of data

| userid | s_account_number | balance | created_on | branch_id | account_name |
|---|---|---|---|---|---|
| 1000000048 | 1000000000 | 10000 | 2020-10-01 17:39:09 | 10000 | savings account |
| 1000000048 | 1000000001 | 5000 | 2020-10-05 18:46:08 | 10000 | savings account upi |
| 1000000049 | 1000000002 | 50000 | 2020-10-05 18:46:08 | 13453 | savings debit_card |
| 1000000050 | 1000000003 | 6398 | 2020-10-05 19:17:26 | 10000 | my_savings |
| 1000000051 | 1000000004 | 15000 | 2020-10-05 19:18:35 | 11000 | savings_account |
| 1000000053 | 1000000026 | 7300 | 2020-10-13 11:24:51 | 10000 | savings account john |
| 1000000060 | 1000000029 | 43210 | 2020-10-24 23:23:05 | 10000 | savings account |
| 1000000055 | 1000000038 | 67789 | 2020-10-25 10:56:10 | 13452 | savings account |
| 1000000062 | 1000000040 | 67899 | 2020-10-25 15:37:02 | 13452 | savings account preeta |

10. **Savings Transactions**

## Structure

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | **trans_id** | int(10) | | | No | *None* | | AUTO_INCREMENT |
| 2 | **account_number** | int(10) | | | No | *None* | | |
| 3 | **timestamp_** | timestamp | | | Yes | current_timestamp() | | |
| 4 | **amount_** | int(11) | | | No | *None* | | |
| 5 | **description** | varchar(100) | utf8mb4_general_ci | | No | *None* | | |
| 6 | **balance** | int(11) | | | No | *None* | | |

## Foreign keys

| Constraint properties | Column | Foreign key constraint (INNODB) | | |
|---|---|---|---|---|
| | | Database | Table | Column |
| savfk | account_numbe | arshdeepbank | savings_accoun | s_account_numl |
| ON DELETE CASCADE | + Add column | | | |
| ON UPDATE RESTRICT | | | | |

## Snapshot of data

| trans_id | account_number | timestamp_ | amount_ | description | balance |
|---|---|---|---|---|---|
| 1 | 1000000000 | 2020-10-02 19:23:23 | 100 | 100 deposit | 10100 |
| 2 | 1000000000 | 2020-10-03 19:23:23 | -150 | 150 withdrawal | 9950 |
| 3 | 1000000000 | 2020-10-05 19:23:23 | -320 | 320 withdrawal | 9630 |
| 4 | 1000000000 | 2020-10-10 19:23:23 | 370 | 370 deposit | 10000 |
| 5 | 1000000001 | 2020-10-05 19:28:59 | 10000 | 10000 opening balance | 10000 |
| 6 | 1000000002 | 2020-10-05 19:30:35 | 30000 | 30000 opening balance | 30000 |
| 7 | 1000000001 | 2020-10-05 19:31:31 | -5000 | 5000 withdrawal | 5000 |
| 8 | 1000000002 | 2020-10-05 19:33:35 | 20000 | 20000 deposit | 50000 |
| 9 | 1000000003 | 2020-10-05 19:37:11 | 8000 | 8000 opening balance | 8000 |
| 10 | 1000000004 | 2020-10-05 19:38:30 | 15000 | 15000 opening balance | 15000 |
| 11 | 1000000003 | 2020-10-05 19:39:50 | -302 | 302 withdrawal | 7698 |
| 12 | 1000000000 | 2020-10-01 17:21:50 | 10000 | 10000 opening balance | 10000 |

# 7. CONCLUSION AND FUTURE WORK

## 7.1 Conclusion

Thus, we successfully created a Banking Management System and created a website by making use of Html, CSS, PHP, MYSQL in XAMMP SERVER, SQL. We also normalized our database in order to maintain consistency in our database. We also created Registration and Login system using PHP and connected that to our database to check the user's activity in real time. We also added trigger to, MYSQL in XAMMP SERVER database which will be used to store an activity such as insertion, updating, and other functions. Online banking is an innovative tool that is fast becoming a necessity. It is a successful strategic weapon for banks to remain profitable in a volatile and competitive marketplace of today. If proper training should be given to customer by the bank employs to open an account will be beneficial secondly the website should be made friendlier from where the first time customers can directly make and access their accounts.

## 7.2 Future Work

In order to improve our website further we will add Google API services to make the sign in with Google option enable so that login system can become faster. Also, also add Mailing system to this project using Node Mailer module of NodeJS. We can also deal through internet by creating web pages and a banking website for internet dealing. To attract Account Holder's we can offer various offers during festivals months. We can also deal in various types of Banking Transactions .To have more and more customer satisfaction we will emphasize more and more on our dealings.

## REFERENCES:


[1] C. J. Date, A. Kannan and S. Swamynathan, An Introduction to Database Systems, Pearson Education, Eighth Edition, 2009.

[2] Abraham Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts, McGraw-Hill Education (Asia), Fifth Edition, 2006.

[3] Shio Kumar Singh, Database Systems Concepts, Designs and Application, Pearson Education, Second Edition, 2011

[4] Matthew MacDonald, "Creating a Website - The Missing Manual", 3rd ed, 2011, O'Reilly. (A good introductory book on HTML/CSS. A new version is expected in July 2015.)

[5] PHP documentation