DBMS - Mini Project

ONLINE BANKING SYSTEM

Submitted By:

Name-
ANSHU SANDUR

SRN – **PES2UG20CS056**

V Semester Section **- A**

Short Description and Scope of the Project

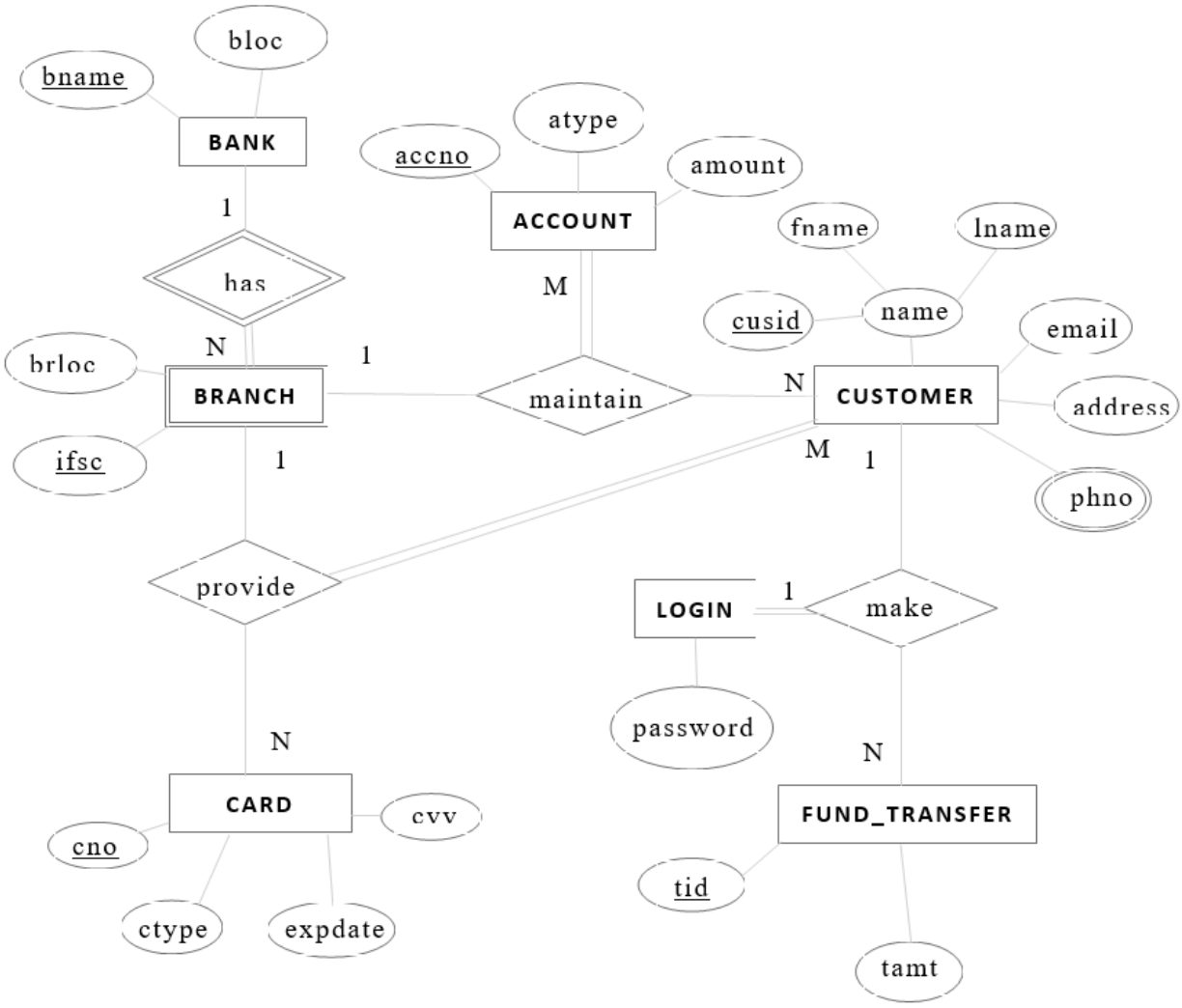
- The main objective of the Online Banking System is to manage the details of Bank, Branch, Customer, Accounts, Transaction, Card, and Balance.

- The project is totally built at administrative end and thus only the administrator is guaranteed the access.

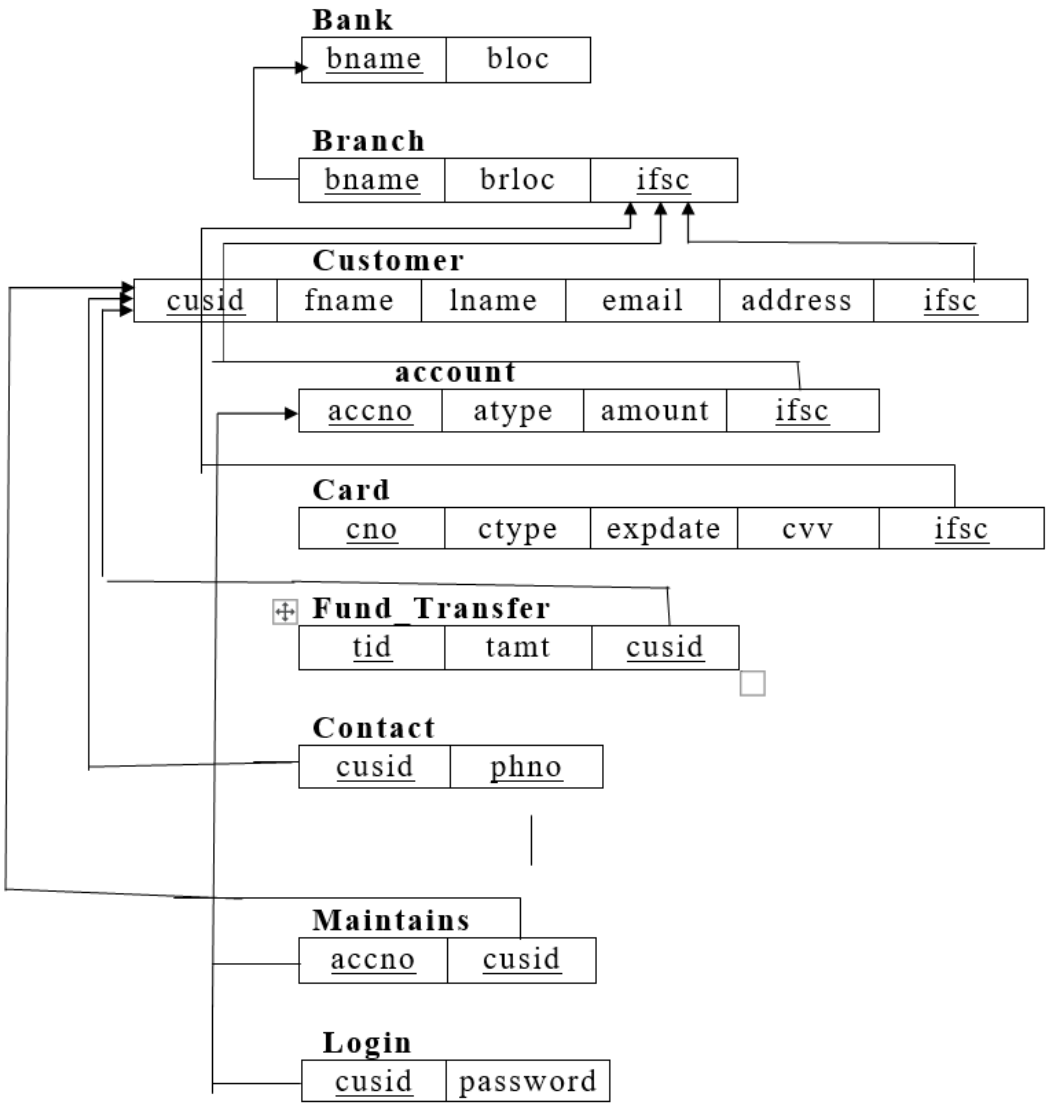
- The purpose of the project is to build an application program to reduce the manual work for managing the Bank, Branch, Accounts, Customer, Card, and Transaction.

- It tracks all the details about the Transaction, Balance.

ER Diagram

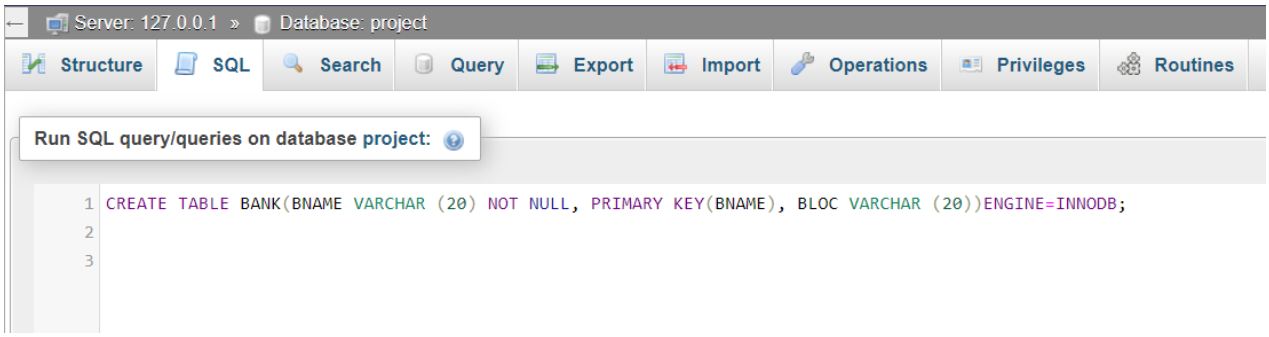

bloc
bname
BANK
1
has
N
brloc
BRANCH
ifsc
1
provide
N
CARD
cno
cvv
ctype
expdate
accno
atype
amount
ACCOUNT
M
maintain
1
N
CUSTOMER
fname
lname
cusid
name
email
address
phno
M
1
make
1
LOGIN
password
N
FUND_TRANSFER
tid
tamt

Relational Schema

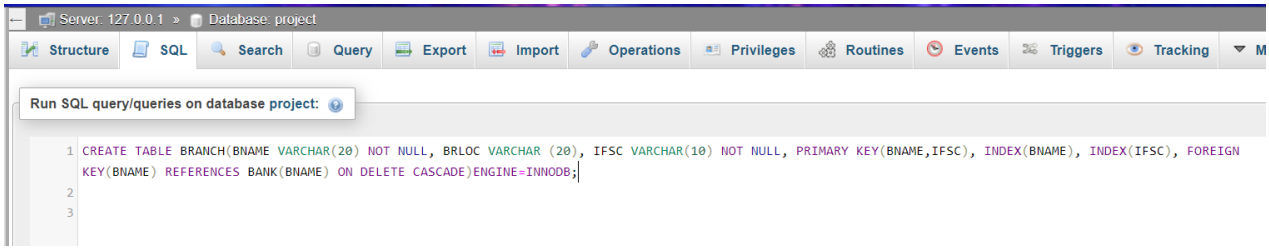


DDL statements - Building the database

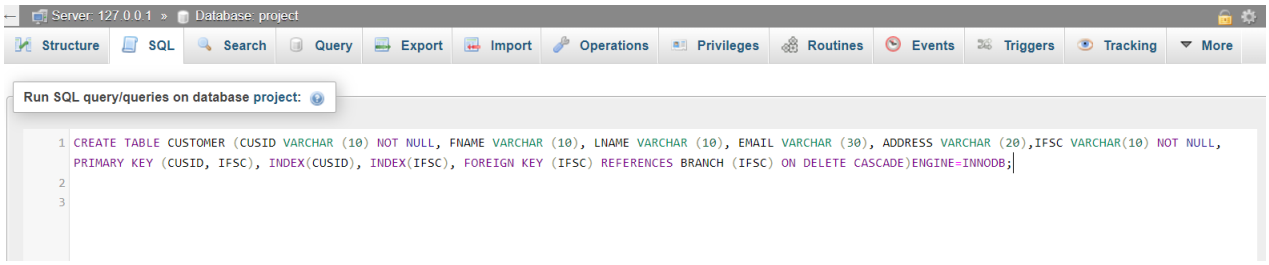
BANK TABLE :



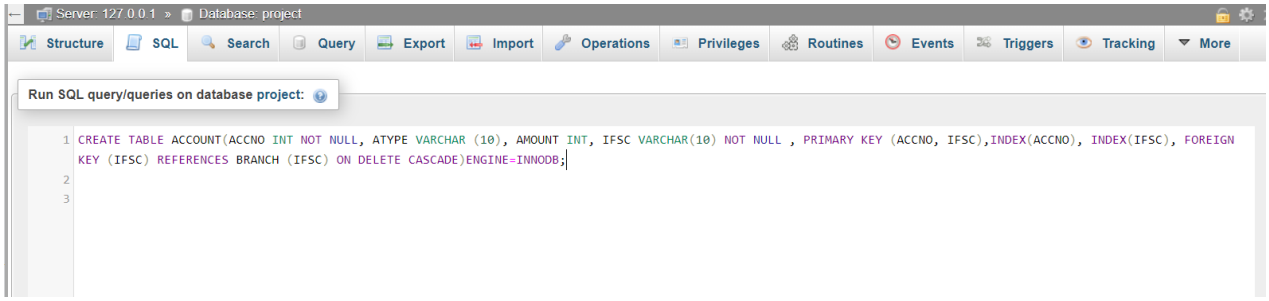
BRANCH TABLE :



CUSTOMER TABLE :



ACCOUNT TABLE :



CARD TABLE :

Server: 127.0.0.1 » Database: project

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Tracking | More

Run SQL query/queries on database project:

```
CREATE TABLE CARD(CNO INT NOT NULL, CTYPE VARCHAR(10), EXPDATE DATE, CVV VARCHAR(3), IFSC VARCHAR(10)  NOT NULL, PRIMARY KEY (IFSC), INDEX(IFSC), FOREIGN KEY(IFSC)
REFERENCES BRANCH(IFSC) ON DELETE CASCADE)ENGINE=INNODB;
```

FUND TRANSFER TABLE :

Server: 127.0.0.1 » Database: project

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Tracking | More

Run SQL query/queries on database project:

```
CREATE TABLE FUND_TRANSFER(TID INT NOT NULL, TAMT INT, IFSC VARCHAR(10) NOT NULL , PRIMARY KEY(TID, CUSID),INDEX(TID), INDEX(CUSID), FOREIGN KEY (CUSID) REFERENCES
CUSTOMER (CUSID) ON DELETE CASCADE)ENGINE=INNODB;
```

LOGIN TABLE :

Server: 127.0.0.1 » Database: project

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Tracking | More

Run SQL query/queries on database project:

```
CREATE TABLE LOGIN(CUSID VARCHAR(10) NOT NULL, PASSWORD VARCHAR (20),INDEX(CUSID), FOREIGN KEY (CUSID) REFERENCES CUSTOMER (CUSID) ON DELETE CASCADE)ENGINE=INNODB;
```

MAINTAINS TABLE :

Server: 127.0.0.1 » Database: project

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Tracking | More

Run SQL query/queries on database project:

```
CREATE TABLE MAINTAINS(ACCNO INT NOT NULL, CUSID VARCHAR(10) NOT NULL, PRIMARY KEY(ACCNO, CUSID), INDEX(ACCNO), INDEX(CUSID), FOREIGN KEY(ACCNO) REFERENCES
ACCOUNT(ACCNO) ON DELETE CASCADE, FOREIGN KEY(CUSID) REFERENCES CUSTOMER(CUSID) ON DELETE CASCADE)ENGINE=INNODB;
```

CONTACT TABLE:

Server: 127.0.0.1 » Database: project

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Tracking | More

Run SQL query/queries on database project:

```
CREATE TABLE CONTACT(CUSID VARCHAR(10) NOT NULL, PHNO INT NOT NULL, PRIMARY KEY(CUSID, PHNO), INDEX(CUSID), INDEX(PHNO), FOREIGN KEY(CUSID) REFERENCES
CUSTOMER(CUSID) ON DELETE CASCADE)ENGINE=INNODB;
```

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	account	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	bank	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	branch	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	card	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	contact	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	customer	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	fund_transfer	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	login	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	maintains	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
	9 tables	Sum	0	InnoDB	utf8mb4_general_ci	336.0 KiB	0 B

Populating the Database

1) BANK

Server: 127.0.0.1 » Database: project » Table: bank

Browse | Structure | SQL | Search | Insert | Export | Import

Run SQL query/queries on table project.bank:

```
INSERT INTO BANK (bname , bloc)  VALUES (‘SBI’, ‘DELHI’);
INSERT INTO BANK (bname , bloc) VALUES (‘CANARA BANK’, ‘BANGALORE’);
INSERT INTO BANK (bname , bloc) VALUES (‘AXIS BANK’, ‘DELHI’);
```

2) BRANCH

Run SQL query/queries on database project:

```
INSERT INTO BRANCH VALUES ("SBI", "BANGALORE", "SBIN000002");
INSERT INTO BRANCH VALUES ("CANARA BANK", "CHENNAI", "CNRB000002");
INSERT INTO BRANCH VALUES ("AXIS BANK", "HYDERBAD", "UTIB000002");
```

3)CUSTOMER

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines

Run SQL query/queries on database project:

```
INSERT INTO CUSTOMER VALUES("SBI001","ROHAN","KUMAR" ,"ROHANK@GMAIL.COM", "BANGALORE","SBIN000002");
INSERT INTO CUSTOMER VALUES ("CNR001", "MOHAN", "KUMAR" , "MOHANK@GMAIL.COM", "CHENNAI", "CNRB000002");
INSERT INTO CUSTOMER VALUES ("UTI001", "ASHOK", "KUMAR" , "ASHOKK@GMAIL.COM", "HYDERBAD",  "UTIB000002");
```

4) ACCOUNT

Run SQL query/queries on database project:

```
INSERT INTO ACCOUNT VALUES('1001', ‘FIXED’, ‘10000’, ‘SBIN000002’);
INSERT INTO ACCOUNT VALUES(‘2001’, ‘SAVINGS’, ‘20000’, ‘CNRB000002’);
INSERT INTO ACCOUNT VALUES(‘3001’, ‘SAVINGS’, ‘10000’, ‘UTIB000002’);
```

5) CARD

Run SQL query/queries on database project:

```
INSERT INTO CARD VALUES('12001', 'DEBIT', '21-12-2019', '111', 'SBIN000002');
INSERT INTO CARD VALUES('22001', 'DEBIT', '21-12-2019', '122', 'CNRB000002');
INSERT INTO CARD VALUES('32001', 'CREDIT', '21-12-2019', '133', 'UTIB000002');
```

6)FUND TRANSFER

Run SQL query/queries on table project.fund_transfer:

```
INSERT INTO FUND_TRANSFER VALUES ('67345','2000','CNR001001');
INSERT INTO FUND_TRANSFER VALUES ('89723','8000','SBI001001');
INSERT INTO FUND_TRANSFER VALUES ('98123','1000','UTI001001');
```

7) LOGIN

Run SQL query/queries on table project.login:

```
INSERT INTO LOGIN VALUES ('SBI001', '123451');
INSERT INTO LOGIN VALUES ('CNR001', '123452');
INSERT INTO LOGIN VALUES ('UTI001', '123453');
```

8) MAINTAINS

Run SQL query/queries on database project:

```
INSERT INTO  MAINTAINS VALUES ('1001', 'SBI001');
INSERT INTO MAINTAINS VALUES ('2001', 'CNR001');
INSERT INTO MAINTAINS VALUES ('3001', 'UTI001');
```

9) CONTACT

Run SQL query/queries on database project:

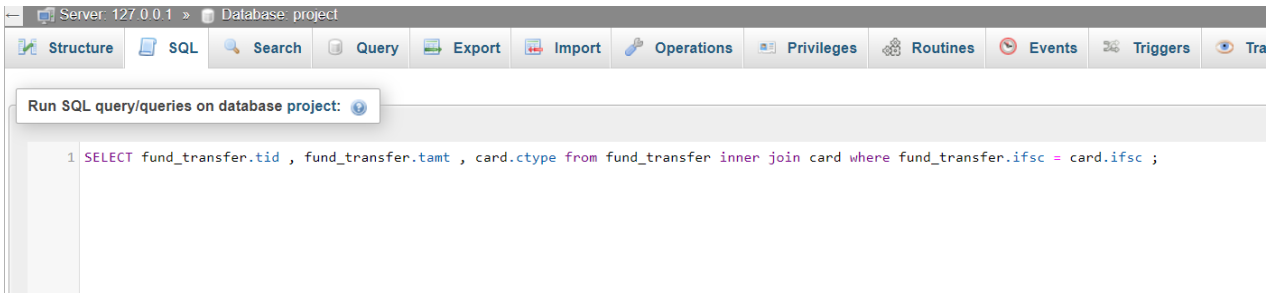
```
INSERT INTO CONTACT VALUES ('SBI001', '9972114480');
INSERT INTO CONTACT VALUES ('CNR001', '9972114481');
INSERT INTO CONTACT VALUES ('UTI001', '9972114482');
```

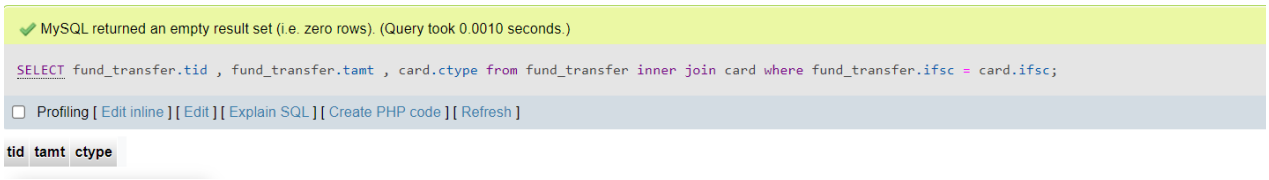
Join Queries

Showcase at least 4 join queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

1) INNER JOIN

IFSC is the common column between the tables fund_transfer and card table . The condition will be fund_transfer.ifsc = card.ifsc

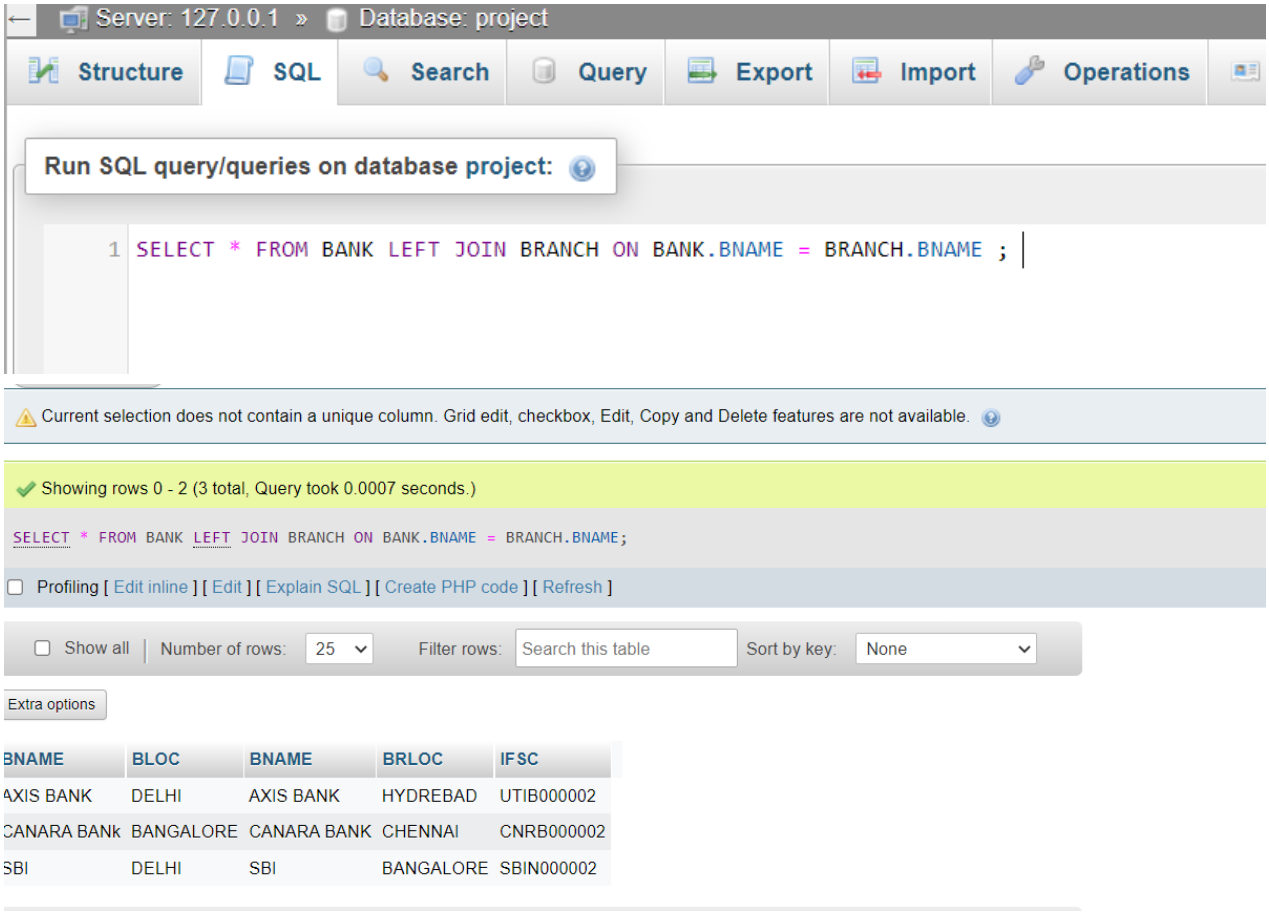




2) LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN**.**

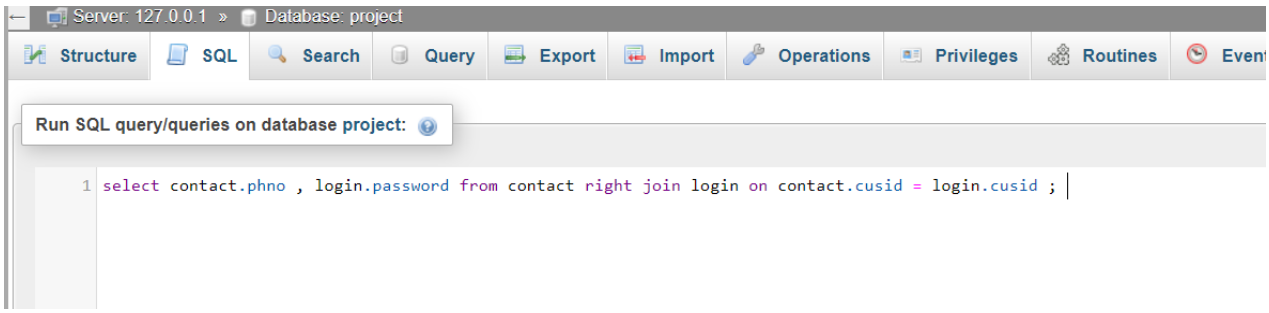
Applying left join on the tables bank (bname , bloc columns) and branch (bname , brloc and ifsc) with the condition that bank.bname = branch.bname , as branch name is the matching column between the tables bank and branch



3) RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*.

Selecting the columns phone number from contact table and password from login table and using right join by the criteria (matching column) customer Id



Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.)

```
select contact.phno , login.password from contact right join login on contact.cusid = login.cusid;
```

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

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

Extra options

phno	password
2147483647	123451
2147483647	123452
2147483647	123453

4) FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.

Selecting all the columns from contact using full join by the criteria (matching column) customer Id

Run SQL query/queries on database project:

```
1 select * from contact full join login  on contact.cusid = login.cusid ;
```

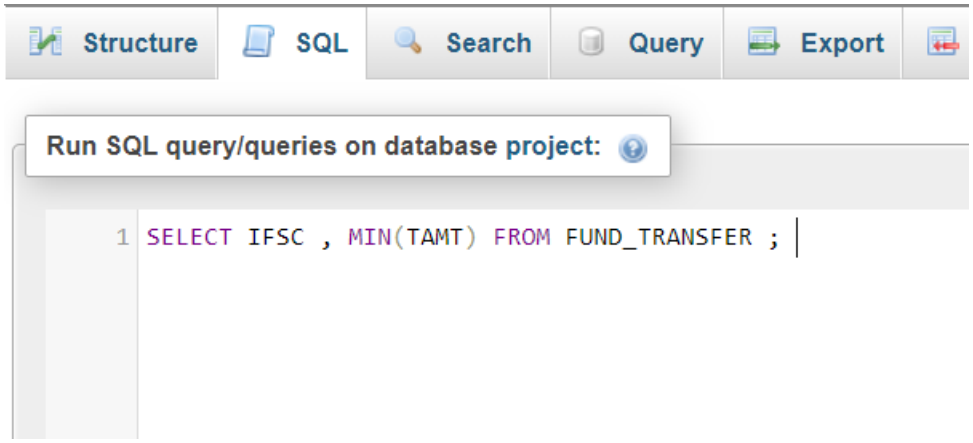
CUSID	phno	password
CNR001	2147483647	123451
SBI001	2147483647	123452
UTI001	2147483647	123452

Aggregate Functions

Showcase at least 4 Aggregate function queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

1) MIN

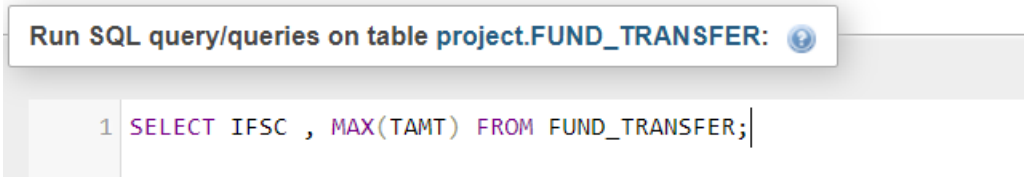
Displaying minimum transfer amount and ifsc code for fund transfer table



	IFSC	MIN(TAMT)
Edit Copy Delete	CNR001001	1000

2) MAX

Displaying maximum transfer amount and ifsc code for fund transfer table



Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)

```
SELECT IFSC , MAX(TAMT) FROM FUND_TRANSFER;
```

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

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

Extra options

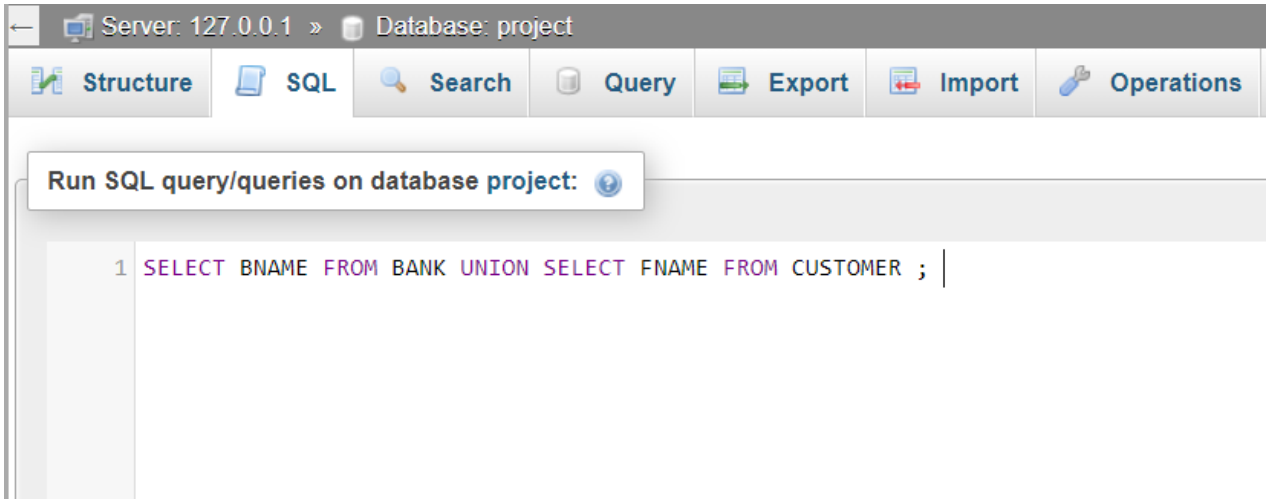
←T→	IFSC	MAX(TAMT)
Edit Copy Delete	CNR001001	8000

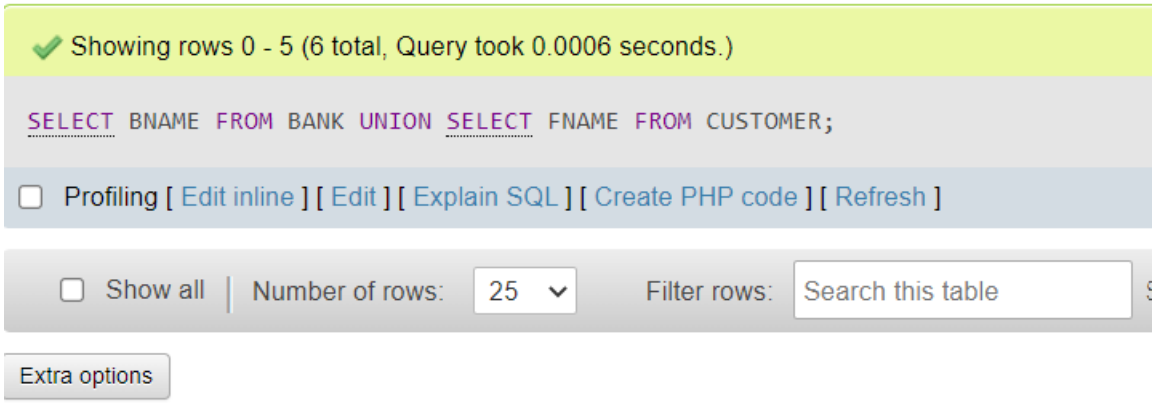
Set Operations

Showcase at least 4 Set Operations queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

UNION –

Using UNION Set Operation to combine the result of the columns bname from bank table and fname from customer table





BNAME
AXIS BANK
CANARA BANk
SBI
MOHAN
ROHAN
ASHOK

INTERSECT –
Using intersect Set Operation on the login and maintains table to display all the column results

L query/queries on database project:

```
SELECT * FROM LOGIN INTERSECT SELECT * FROM MAINTAINS ;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

```
SELECT * FROM LOGIN INTERSECT SELECT * FROM MAINTAINS;
```

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

CUSID	PASSWORD

Query results operations

Create view

Functions and Procedures

Create a Function and Procedure. State the objective of the function / Procedure. Run and display the results.

```
Delimiter $$
CREATE PROCEDURE counting()
BEGIN
Select count(accno),ifsc from account group by ifsc;

END
$$
CALL counting()
$$
```

//**this function is grouping the ifsc code from account table by counting the number of unique account numbers and displaying the count of accno as well**

Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0044 seconds.)

```
CREATE PROCEDURE counting() BEGIN Select count(accno),ifsc from account group by ifsc; END;
```

[Edit inline] [Edit] [Create PHP code]

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 2 (3 total, Query took 0.0034 seconds.)

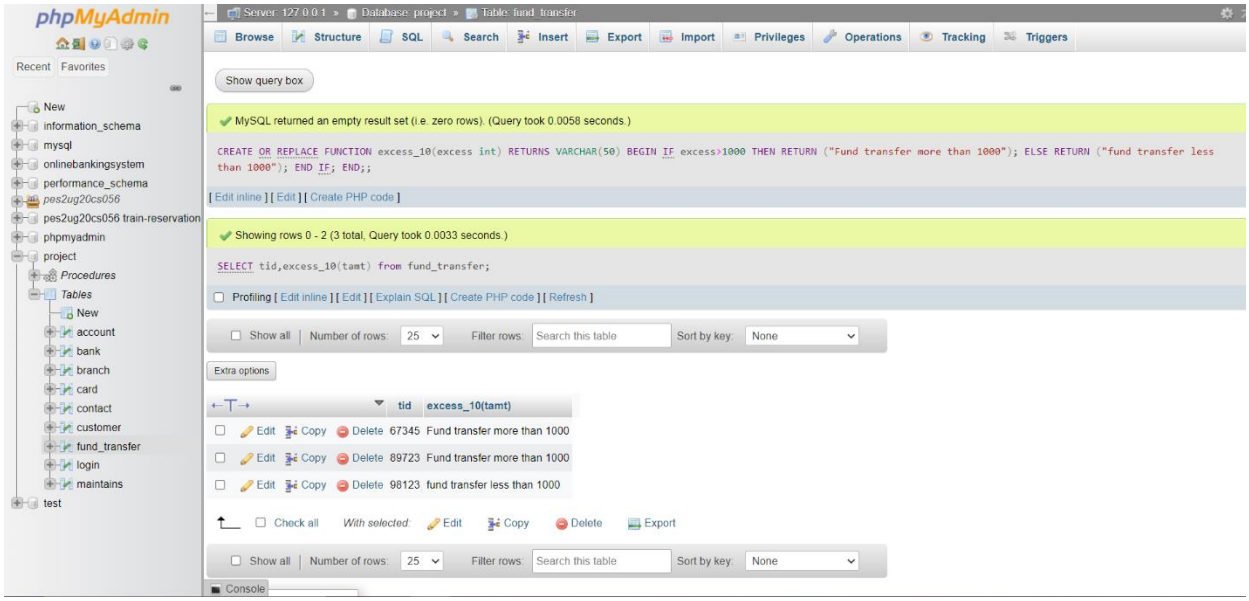
```
CALL counting();
```

[Edit inline] [Edit] [Create PHP code]

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

Extra options

count(accno)	ifsc
1	CNRB000002
1	SBIN000002
1	UTIB000002



FUNTION IS CHECKING WHETHER THE FUND TRANSFER IS LESS THAN OR GREATER THAN 1000 AND RETURNINFG TRANSFER ID OF THE CUSTOMER .

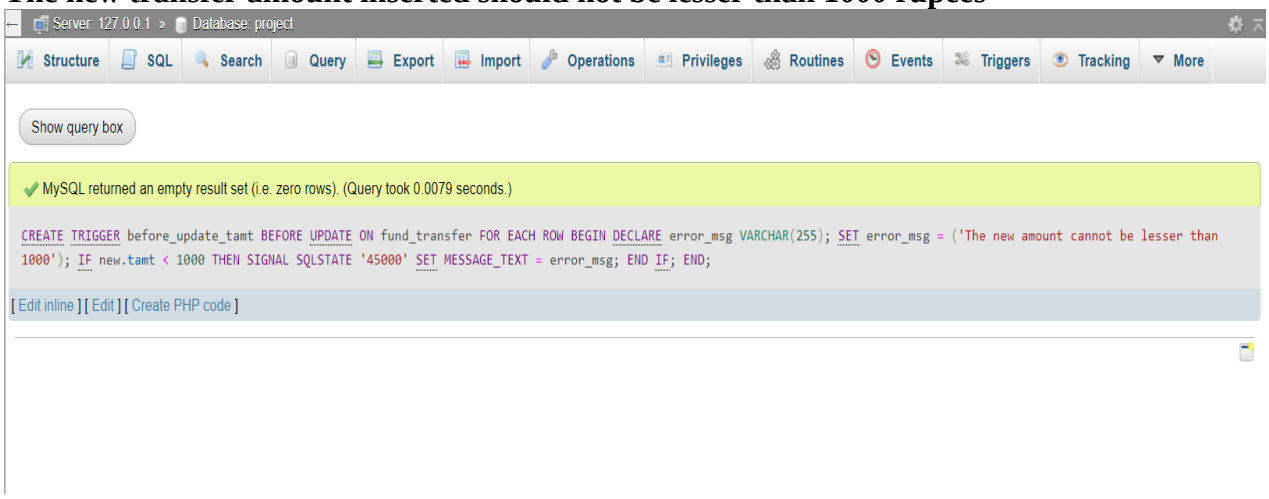
Triggers and Cursors

Create a Trigger and a Cursor. State the objective. Run and display the results.

TRIGGER

A trigger is a special kind of stored procedure that automatically execute when an event occurs in the database server

The new transfer amount inserted should not be lesser than 1000 rupees



Error

SQL query: Copy

```
UPDATE fund_transfer

set tamt=20

where tid = 67345;
```

MySQL said:

```
#1644 - The new amount cannot be lesser than 1000
```

CURSOR –

```
DELIMITER $$
--
-- Procedures
--
CREATE DEFINER=`root`@`localhost` PROCEDURE `getcategories` ()   BEGIN
   DECLARE tid,tamt TEXT;
   DECLARE exit_loop BOOLEAN DEFAULT FALSE;
   DECLARE category_cursor CURSOR FOR SELECT tid,tamt FROM fund_transfer;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop=TRUE;

   OPEN category_cursor;
   category_loop: LOOP
                FETCH FROM category_cursor INTO tid, tamt;
         IF exit_loop THEN
                LEAVE category_loop;
         END IF;
         IF tamt>='1000' THEN
                SELECT tid;
         END IF;
   END LOOP category_loop;
   CLOSE category_cursor;
END$$
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0022 seconds.)

```
-- -- Procedures -- CREATE DEFINER=`root`@`localhost` PROCEDURE `getcategories` () BEGIN DECLARE tid,tamt TEXT; DECLARE exit_loop BOOLEAN DEFAULT FALSE; DECLARE
category_cursor CURSOR FOR SELECT tid,tamt FROM fund_transfer; DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop=TRUE; OPEN category_cursor; category_loop: LOOP FETCH FROM
category_cursor INTO tid,tamt; IF exit_loop THEN LEAVE category_loop; END IF; IF tamt>='1000' THEN SELECT tid; END IF; END LOOP category_loop; CLOSE category_cursor; END;
```

[Edit inline] [Edit] [Create PHP code]

Developing a Frontend

The frontend should support

1. Addition, Modification and Deletion of records from any chosen table
2. There should be an window to accept and run any SQL statement and display the result