SCOPE OF PROJECT

- It carries out processes easily and quickly which are not possible manually.
- It aims to automate the transactions of banks and provide a better and faster service to customers online via the internet. It focuses on better performance and paperless banking up to a point.
- Customers can perform financial transactions like transfer funds online, pay bills, apply for loans and open a savings account among various other debit card transactions.
- It will maintain a large no. of transactions with ease and safety.
- It will manage the details of all registered customers and help them to operate their accounts online via the internet.
- It will make updating, modification and deleting of records easier.

STAKEHOLDERS

1. **BANK:** Controls/owns branches and also, manages account details of all its customers across branches.
2. **Employee**: Works for the bank(specific branch) and assists customers for loans and transaction of money(cashier) or repayment of loans(manager), along with the opening of a new account, issuing of credit/debit cards.
3. **Customer**: Owns an account in the bank and can do transactions by taking loans from the bank(borrower) or depositing money to his/her account(depositor). Also, make payments for daily use using cards of bank.

Weak Entity

1. **Account type:** It is a weak entity as it can only exist if “Account”(entity) exists in the database.
2. **Payment:** It is a weak entity as it can only exist if “Employee” and “Customer”(entity) exists in the database.
3. **Loan:** It is a weak entity as it can only exist if “Branch” and “Customer”(entity) exists in the database.

Ternary Relationship

Loan, **Customer** and **Branch** forms a Ternary relationship with a relation called “**request**”. It is a Ternary relation as:
Loan is taken by a customer,
Loan is given by particular Branch of bank &,
Customers have an account in the same (/belongs to a particular) Branch of the bank.

RELATIONAL SCHEMA

Login(<u>email_id</u>, role_of_user, username, password**)**

Bank(<u>bank_ssn</u>, Bank_name, Head_Office{ HO_email, HO_address{ State, City, Pincode } }**)**

Customer(<u>cust_id</u>, cust_dob, cust_name, phone_no ,account_no, pan_no, cust_email, address{ State, City, Pincode }**)**

Insurance(**<u>Insurance_no</u>** , type, term, premium_payment, issuing_company , cust_id, date_of_insurance**)**

Employee(<u>emp_ID</u>, emp_Salary, emp_type, emp_name, emp_DOB, {phone_no}, emp_address{ State, City, Pincode } , mgr_id**)**

Branch(<u>branch_id,</u> branch_address{ State, City, Pincode }, branch_email**)**

Loans(term, rate_of_interest, amount, type_of_loan, account_no, **loan_id)**

Account(account_no , overdraft , balance**)**

Account type(account_no , deposit_amt, transaction_limit, withdraw_limit, interest_rate)

Credit Card(cc_limit, **cc_number** , cc_cvv, cc_expirydate, account_no)

Debit Card(debit_number , cc_cvv, debit_expirydate, account_no)

Payment(payment_id, payment_amountdue, payment_duedate, account_no**)**

request(date_of_request, cust_id, branch_id, loan_id**)**

Branches(date_of_opening , bank_ssn , branch_id**)**

acc_branch(date_of_opening , branch_id , acc_no**)**

Assistance(cust_id , emp_id , date_of_assistance**)**

SQL QUERIES:

1) The number of customers by dob registered in bank.

```
SELECT  EXTRACT(YEAR FROM dob) AS year,
       EXTRACT(QUARTER FROM dob) AS quarter,
       COUNT(cust_id) AS number_of_customers
FROM Customer
GROUP BY EXTRACT(YEAR FROM dob), EXTRACT(QUARTER FROM dob)
ORDER BY EXTRACT(YEAR FROM dob) ASC, EXTRACT(QUARTER FROM dob);
```

2)

```
SELECT customer.first_name, customer.last_name
         FROM customer
         WHERE EXISTS (SELECT account_branch.date_of_opening from
         account_branch where account_branch.account_no = customer.account_no and
         customer.account_no > 600000 );
```

3)

```
SELECT emp_id, first_name, last_name
      , salary
FROM Employee
WHERE first_name in (SELECT DISTINCT first_name
             FROM Employee
             WHERE emp_dob >  1995-01-01
                  AND emp_address_city = "Delhi" OR emp_address_city = "Agartala"
OR emp_address_city="Ranchi")
     AND salary >= (SELECT AVG(salary)
                  FROM Employee
                  WHERE mgr_ID >10)
Limit 100;
```

4)

display only the details of employees who either earn the highest salary or the lowest salary in each department from the employee table.

```
Select temp.*
FROM Employee e
join (select emp_id, first_name,emp_type,salary,
        min(salary) over (partition by emp_type) as SALARY_MIN,
        max(salary) over (partition by emp_type) as SALARY_MAX
        from employee) temp
on e.emp_id = temp.emp_id
and (e.salary = temp.SALARY_MIN or e.salary = temp.SALARY_MAX)
order by temp.emp_type, temp.salary;
```

5)

```
CREATE VIEW loans10 AS
SELECT Customer.cust_id, customer.first_name, Customer.account_no,
loans.amount as Loan_Amount
FROM (loans
        INNER JOIN customer ON
loans.account_no=customer.account_no
    )
WHERE loans.amount  > 10000 ;
```

6)

Cities which start with D, along with the number of customers from this city must be greater than equal to 2.

Opt query:

```
SELECT cust_address_city, COUNT(*) AS number_customers
FROM Customer
WHERE cust_address_city LIKE 'D%'
GROUP BY cust_address_city
HAVING count(*) >= 2;
```

7)

```
SELECT emp_branch.date_of_joining, employee.first_name, branch.branch_email
        from ((emp_branch
        inner join employee on emp_branch.emp_id = employee.emp_id)
        inner join branch on branch.branch_id = emp_branch.branch_id);
```

8)

```
SELECT first_name, last_name, cust_id AS 'Customers'
FROM Customer c
LEFT JOIN credit_card o
ON c.cust_id = o.c_id
WHERE o.c_id IS NULL
```

9)

```
SELECT COUNT(cust_id), customer.cust_address_city
FROM customer
```

```
GROUP BY customer.cust_address_city
HAVING COUNT(cust_id) > 1;
```

10)
Second Highest salary for employees.

```
Select Distinct(t1.salary) from Employee t1
Where 3 =
(
Select count(Distinct (t2.salary)) FROM Employee t2
Where t1.salary < t2.salary
)
```

General:

Mth largest salary:

```
Select Distinct(t1.salary) from Employee t1
Where M-1 =
(
Select count(Distinct (t2.salary)) FROM Employee t2
Where t1.salary < t2.salary
)
```

Indexing:

Attributes which are chosen to be the indexes are the attributes used in the WHERE ,JOIN, LIKE and the ORDER BY clause.

Account_branch.date_of_opening
Customer.dob
Employee.first_name and Employee.dob
Loans.account_no
Customer.account_no
Employee.Salary
customer.cust_address_city
Customer.c_id
Clustered {Customer.cust_address_city, Customer.c_id}
Employee.emp_type

Grants:

```
CREATE ROLE admin;
CREATE ROLE employee;
CREATE ROLE customer;

CREATE ROLE insurance;

GRANT ALL PRIVILEGES ON OnlineBankingSystem.* TO admin;
GRANT SELECT, INSERT ON OnlineBankingSystem.account TO employee;
GRANT SELECT, INSERT ON OnlineBankingSystem.customer TO employee;
GRANT SELECT, INSERT ON OnlineBankingSystem.employee TO employee;
GRANT SELECT, INSERT ON OnlineBankingSystem.loans TO employee;
GRANT SELECT, INSERT ON OnlineBankingSystem.payment TO employee;
GRANT SELECT, INSERT ON OnlineBankingSystem.request TO employee;
GRANT SELECT, INSERT ON OnlineBankingSystem.branch TO employee;
GRANT SELECT, INSERT ON OnlineBankingSystem.insurance TO employee;
GRANT SELECT, UPDATE ON OnlineBankingSystem.insurance TO insurance;
```