Report on

"Innovative Solutions for Growth: Transforming Retail Banking at City Central Bank"

Submitted in the partial fulfillment of the requirement of Post

Graduation Program in

MSC Data Science

Course: Real-time analytics

3rd Semester 2023-24

By:

Gowthami K

CU22MSD002A

Under the guidance of:

Prof. Sarath B Sribhasyam
School of Mathematics and Natural Science

Chanakya University

Bengaluru-562110

CHANAKYA UNIVERSITY

SCHOOL OF MATHEMATICS AND NATURAL SCIENCES

BENGALURU-562110

Project Report Certificate

This to Certify that

Gowthami K CU22MSD002A

Has satisfactorily completed the Project work "Innovative Solutions for Growth: Transforming Retail Banking at City Central Bank" submitted in partial fulfillment of the requirements for the Certificate of Post Graduation in MSc Data Science by Chanakya University during the academic year 2023-24.

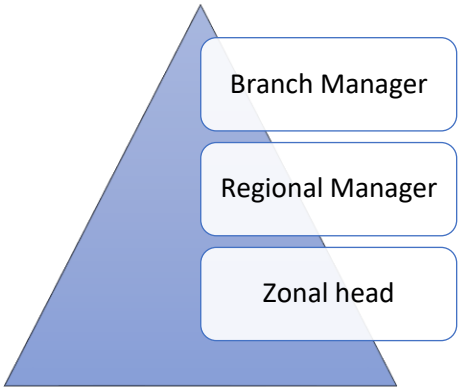
Prof. Sarath B Sribhasyam
Project guide
School of Mathematics and Natural Sciences
Chanakya University
Bengaluru-562110

Contents:

- Understand the scope
- Define key stakeholders and their Goals and problem statements
- Hierarchical relationship
- Identify Key entities and their attributes
- Define the tables in PostgreSQL
- Load Data into tables
- Develop Key Metrics

1. Understanding the Scope:
    - City Central Bank operates across 20 states in India, focusing on bringing financial services to everyday people, especially those in the lower to the middle-income range.
    - Strategically spread across four zones, aim to connect with communities on a personal level.
    - Branch managers, acting as community leaders, guide customers through the lending process, while sales managers work to build lasting relationships by encouraging deposits.
    - Each zone, collaborates to overcome challenges, ensuring that customers' financial journeys are as smooth as possible.
    - It's about making a positive impact in the lives of the people they serve.

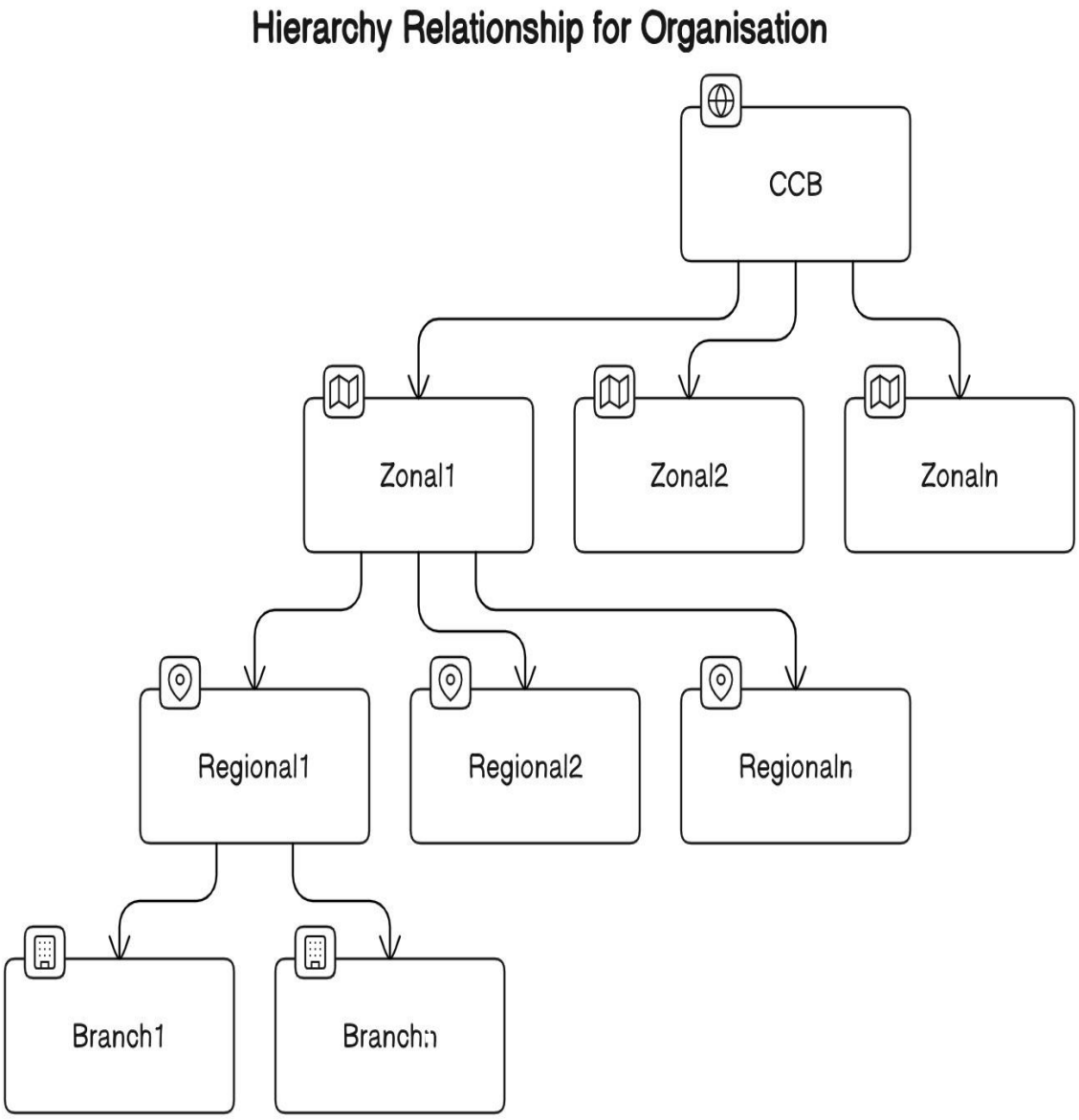
2. Define key stakeholders and their Goals and problem statements



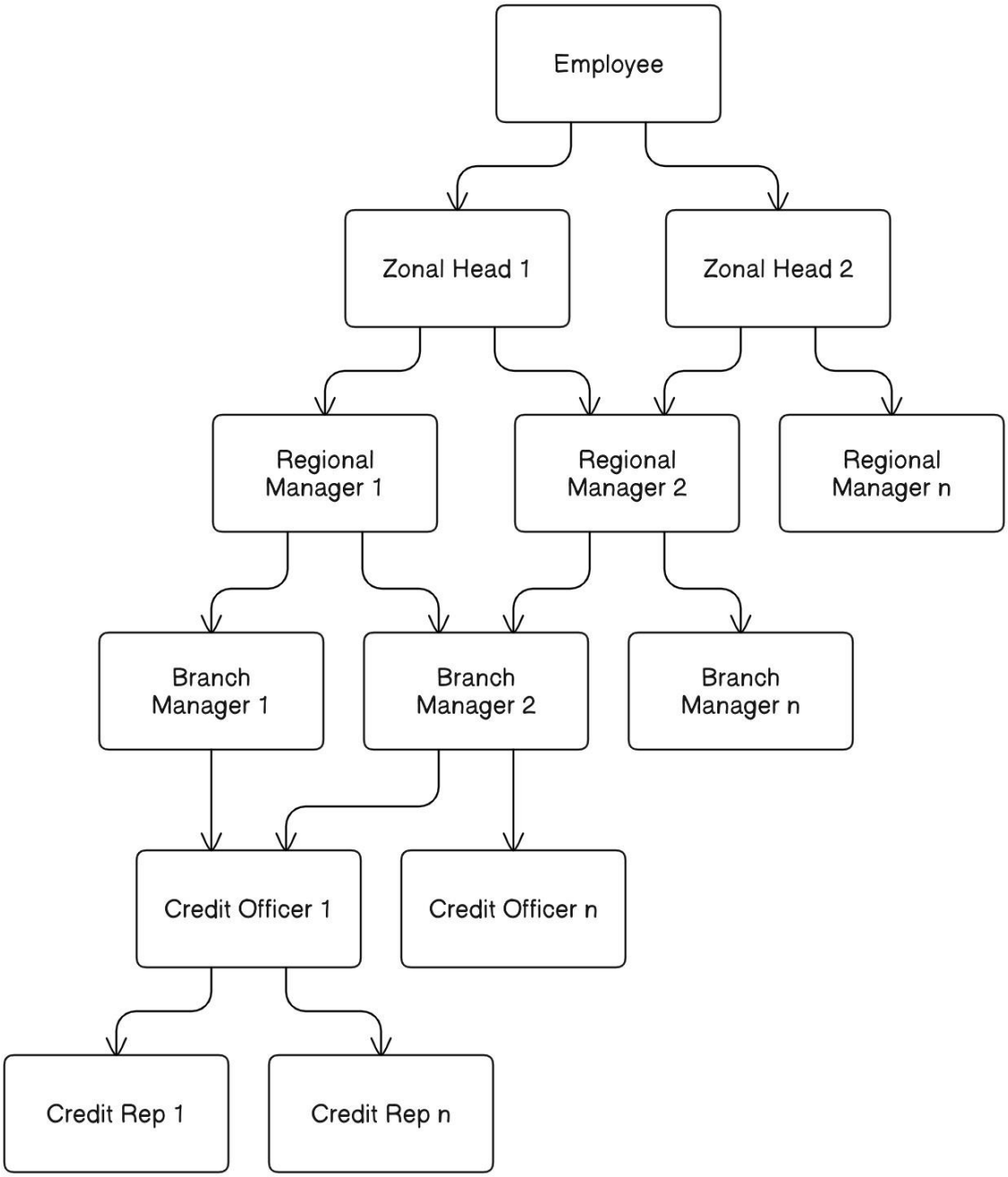
- **Branch Manager:**
  Goal:
  Efficiently oversee lending and saving business at the branch level.
  Problem Statements:
  Track daily customer interactions by Credit Rep.
  Monitor the progress of loan applications and approval times.
  Assess the quality and risk profile of customers.
  Understand daily and weekly collections.
  Monitor deposit mobilization and upcoming maturity dates.
- **Regional Manager:**
  Goal:
  Ensure accurate updates and performance monitoring of branches.
  Problem Statements:
  Obtain accurate daily data on collections and deposits.
  Analyze branch-wise performance metrics and new customer acquisition.
  Derive risk profiles for branches based on defaults and customer risk.
  Identify KPIs for quick assessment of branch intervention needs.
- **Zonal Head:**
  Goal:
  Oversee the overall performance of the zones.

  Problem Statements:
  Evaluate zone-wise performance metrics, risk profiles, and KPIs.
  Identify branches that require intervention or additional support.

3. Hierarchical Relationship for Organisation (Unit) and Employee:



Employee Hierarchy



4. Identify Key entities and their attributes

Key Entities and Their Attributes:

1. **Entities**:
   - Customer
   - Employee
   - Loan
   - Unit
2. **Attributes**:

Customer:
- CustomerID (Primary Key)
- FirstName
- LastName
- ContactInfo
- Address
- CreditRating
- UnitID (Foreign Key)

Employee:
- EmployeeID (Primary Key)
- Emp_name
- RoleID (Foreign Key)
- UnitID (Foreign Key)
- ContactInfo
- Supervisor ID (Foreign key to EmpID)

Role:
- RoleID (Primary Key)
- RoleName

Supervisor Table

- Supervisor ID
- Supervisor Name

UnitTable:
- Unit_ID (Primary Key)
- Unit_name
- Unit_type
- ParentID (Foreign Key to unitID)

CustomerTransaction:
- TransactionID (Primary Key)
- CustomerID (Foreign Key)
- TransactionDate
- Amount
- TransactionType
- UnitID (Foreign Key)
- EmployeeID (Foreign Key)
- LoanID

LoanApplications:

- LoanApplicationID (Primary Key)
- CustomerID (Foreign Key)
- UnitID (Foreign Key)
- ApplicationDate
- ApprovalDate
- LoanAmount
- EmpID

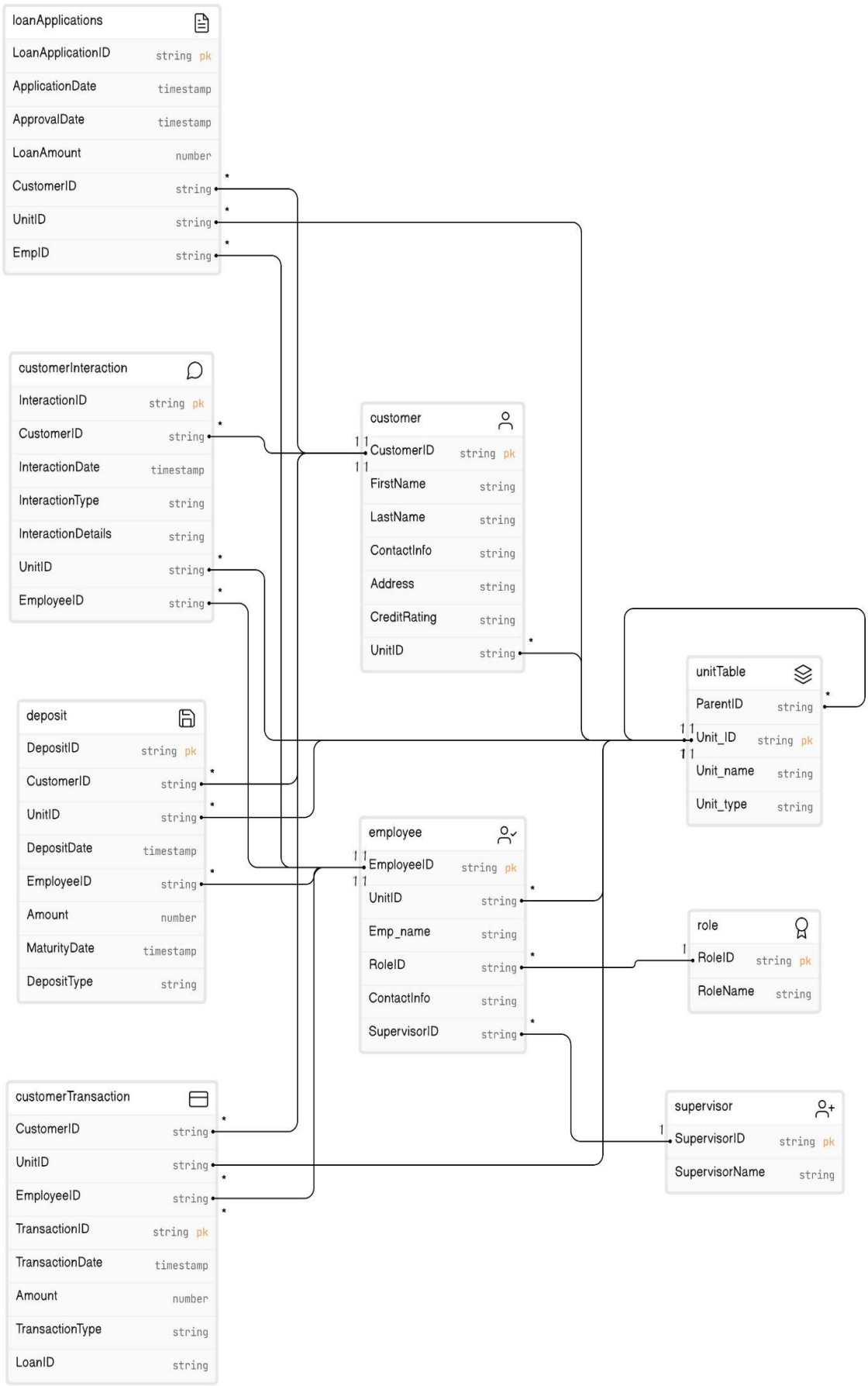
Customer Interaction Table:

- Attributes:
    - InteractionID (Primary Key)
    - CustomerID (Foreign Key)
    - InteractionDate
    - UnitID(Foreign Key)
    - EmployeeID (Foreign Key)
    - InteractionType
    - InteractionDetails

Deposit Table:

- Attributes:
    - DepositID (Primary Key)
    - CustomerID (Foreign Key)
    - DepositDate
    - Amount
    - UnitID (Foreign Key)
    - EmployeeID (Foreign Key)
    - MaturityDate
    - DepositType

Entity Relationship Diagram



6.Problem Statement Approach:

1. Track daily customer interactions by Credit Rep :

Approach:

To effectively track customer interaction by Credit Rep establish and holding data through channels such as email, phone calls, and in-person meeting.

The table contains information about Customer, Employee table contains Credit Rep details and Customer interaction table contains interaction type, date and time. Using these tables we can retrieve information about daily customer interaction by Credit Rep

To retrieve data we can use four clauses: 1. Select clause 2. From Clause 3. Where clause 4. Group by clause

1. Selecting the name or ID of Credit Rep and Count(Interaction ID) the no. of interaction for each credit Rep
2. Left Join: Joins the employee table with Interaction table based on EmpID.
3. WHERE clause: The DATE() function is used to extract the date portion of the Interaction_Date field from the Interactions table. CURDATE () returns the current date.
4. GROUP BY clause:
   Groups the results by EmpID, Emp_Name and Role_ID. This ensures that the count of interactions is calculated for each unique Credit Rep.

2. Monitor the progress of loan applications approval times and approved Status

Approach:

To effectively monitor the progress of loan applications and approval times

In this schema, the LoanApplications table stores basic information about each loan application, including an ApplicationID (unique identifier), ApplicantName, ApplicationDate, LoanAmount, and Status (which could be valued like "Pending", "Approved", "Rejected", etc.).

WHERE Status = 'Approved': This condition filters the rows in the LoanApplications table to include only those with a Status of 'Approved'.

3. Assessing the quality and risk profile of customers is a crucial task for businesses, particularly those involved in financial services such as lending or banking. This process involves evaluating various factors to determine the likelihood of a customer defaulting on payments or causing financial harm to the business.

Approach:

We can calculate various metrics and indicators that help in evaluating customer creditworthiness and default risk. For example:

Calculate the average credit score for each customer.

Analyze payment history to identify patterns of late payments or defaults.

Assess the stability of income by analyzing historical income data.

Segment customers into different risk categories , such as:

Low risk: Customers with high credit scores and stable income.

Medium risk: Customers with moderate credit scores and manageable debt levels.

High risk: Customers with low credit scores, history of late payments, or high debt-to-income ratios.

7. SQL Queries:

1.Calculate daily customer interactions and application stages:

```
postgres=# SELECT DATE(interactiondate) AS date,
      COUNT(*) AS daily_interactions,
      SUM(CASE WHEN interactiontype = 'meeting' THEN 1 ELSE 0 END) AS
under_review,
      SUM(CASE WHEN interactiontype = 'mail' THEN 1 ELSE 0 END) AS
approved,
      SUM(CASE WHEN interactiontype = 'Call' THEN 1 ELSE 0 END) AS
rejected
FROM customerinteraction
GROUP BY DATE(interactiondate);
   date    | daily_interactions | under_review | approved | rejected
-----------+--------------------+--------------+----------+----------
 2024-04-03 |                  1 |            0 |        0 |        1
 2024-03-31 |                  2 |            0 |        0 |        1
 2024-04-02 |                  2 |            0 |        0 |        0
 2024-04-01 |                  2 |            0 |        0 |        1
 2024-03-30 |                  2 |            0 |        0 |        0
 2024-03-29 |                  1 |            0 |        0 |        1
```

2.Assess the time taken for loan approval:

```
SELECT AVG(EXTRACT(DAY FROM (approvaldate, applicationdate))) AS
avg_approval_time
FROM loanapplications
WHERE approvaldate IS NOT NULL;
 avg_approval_time
--------------------
 7.0000000000000000
```

The result of the query indicates that the average approval time for the loan applications meeting the specified criteria is approximately 7 days.

3. customers grouped by their credit ratings:

```
Postgres=# SELECT creditrating,
      COUNT(*) AS customer_count
FROM customer1
GROUP BY creditrating;
 creditrating | customer_count
--------------+----------------
 730          |              1
 780          |              1
 750          |              1
 760          |              1
 800          |              1
 820          |              1
 720          |              1
 810          |              1
```

```
 830         |            1
 700         |            1
(10 rows)
```

The result displays each unique credit rating along with the count of customers having that credit rating. For example, there is 1 customer with a credit rating of 730, 1 customer with a credit rating of 780, and so on, as shown in the output.

4. Track daily and weekly collections:

```
postgres=# -- Daily collections
SELECT DATE(collection_date) AS date,
       SUM(amount) AS daily_collections
FROM collections
GROUP BY DATE(collection_date);
  date    | daily_collections
------------+-------------------
 2024-01-04 |          1800.00
 2024-01-10 |          2400.00
 2024-01-05 |          2200.00
 2024-01-09 |          2300.00
 2024-01-06 |          1700.00
 2024-01-07 |          1900.00
 2024-01-02 |          1500.00
 2024-01-08 |          2100.00
 2024-01-01 |          1000.00
 2024-01-03 |          2000.00
```

5.
```
SELECT EXTRACT(YEAR FROM collection_date) AS year,
       EXTRACT(WEEK FROM collection_date) AS week,
       SUM(amount) AS weekly_collections
FROM collections
GROUP BY EXTRACT(YEAR FROM collection_date), EXTRACT(WEEK
FROM collection_date);
 year | week | weekly_collections
------+------+--------------------
 2024 |    2 |           6800.00
 2024 |    1 |          12100.00
```

6.. Deposit Mobilization by Date:

```
SELECT depositdate AS date,
       SUM(amount) AS daily_deposit_mobilization
FROM deposit
WHERE deposittype = 'Savings Deposit'
GROUP BY depositdate;
    date    | daily_deposit_mobilization
------------+----------------------------
 2024-03-20 |                    5000.00
 2024-03-22 |                    7000.00
 2024-03-24 |                    9000.00
 2024-03-26 |                   11000.00
```

```
 2024-03-28 |                13000.00
```

7.Upcoming Maturities:

```
postgres=# SELECT maturitydate,
      SUM(amount) AS total_maturity_amount
FROM deposit
WHERE maturitydate >= CURRENT_DATE
GROUP BY maturitydate;
 maturitydate | total_maturity_amount
--------------+----------------------
 2025-03-21   |              6000.00
 2025-03-23   |              8000.00
 2025-03-29   |             14000.00
 2025-03-20   |              5000.00
 2025-03-26   |             11000.00
 2025-03-28   |             13000.00
 2025-03-22   |              7000.00
 2025-03-27   |             12000.00
 2025-03-25   |             10000.00
 2025-03-24   |              9000.00
```

8. postgres=# CREATE MATERIALIZED VIEW daily_deposit_mobilization_view AS

```
SELECT depositdate AS date,
      SUM(amount) AS daily_deposit_mobilization
FROM deposit
WHERE deposittype = 'Fixed Deposit'
GROUP BY depositdate;
```

SELECT 5

9. postgres=# REFRESH MATERIALIZED VIEW daily_deposit_mobilization_view;

```
REFRESH MATERIALIZED VIEW
postgres=# SELECT * FROM daily_deposit_mobilization_view;
    date    | daily_deposit_mobilization
------------+---------------------------
 2024-03-21 |                6000.00
 2024-03-23 |                8000.00
 2024-03-25 |               10000.00
 2024-03-27 |               12000.00
 2024-03-29 |               14000.00
INSERT INTO deposit (depositid, customerid, depositdate, amount, unitid,
employeeid, maturitydate, deposittype)
VALUES
   (14, 1004, '2024-03-24', 8500.00, 6, 4, '2025-03-24', 'Fixed Deposit');
```

10 Postgres=# REFRESH MATERIALIZED VIEW daily_deposit_mobilization_view;

```
REFRESH MATERIALIZED VIEW
postgres=# SELECT * FROM daily_deposit_mobilization_view;
```

date	daily_deposit_mobilization
2024-03-21	12000.00
2024-03-23	8000.00
2024-03-24	8500.00
2024-03-25	10000.00
2024-03-27	12000.00
2024-03-29	14000.00

8. Develop Key Metrics

- Customer Engagement Rate:

Calculate the number of customers met by Credit Reps daily.
Divide this number by the total number of targeted customers.
Multiply the result by 100 to get the percentage.
Formula: (Number of customers met / Total targeted customers) * 100

- Loan Approval Time:

Calculate the total time taken by Credit Officers to approve loans from the application stage.
Divide this total time by the number of loan applications processed.
This will give you the average time taken per loan approval.
Formula: Total time taken for loan approvals / Number of loan applications processed

- Collection Efficiency:

Calculate the total loan installments collected daily.
Divide this by the total expected loan installments for the day.
Multiply the result by 100 to get the percentage.
Formula: (Total loan installments collected / Total expected loan installments) * 100

- Deposit Growth Rate:

Calculate the difference between the total deposits collected in the current week and the total deposits collected in the previous week.
Divide this difference by the total deposits collected in the previous week.
Multiply the result by 100 to get the percentage growth rate.
Formula: ((Total deposits collected in current week - Total deposits collected in previous week) / Total deposits collected in previous week) * 100

- Loan Portfolio Quality:

Calculate the total value of non-performing loans (loans in default) and the total value of all loans disbursed.
Divide the total value of non-performing loans by the total value of all loans disbursed.
Multiply the result by 100 to get the percentage.
Formula: (Total value of non-performing loans / Total value of all loans disbursed) * 100

- Customer Satisfaction Score:

Collect feedback from customers regarding loan processing and services provided.
Assign numerical scores to different aspects of customer satisfaction.
Calculate the average score across all aspects to get the overall customer satisfaction score.