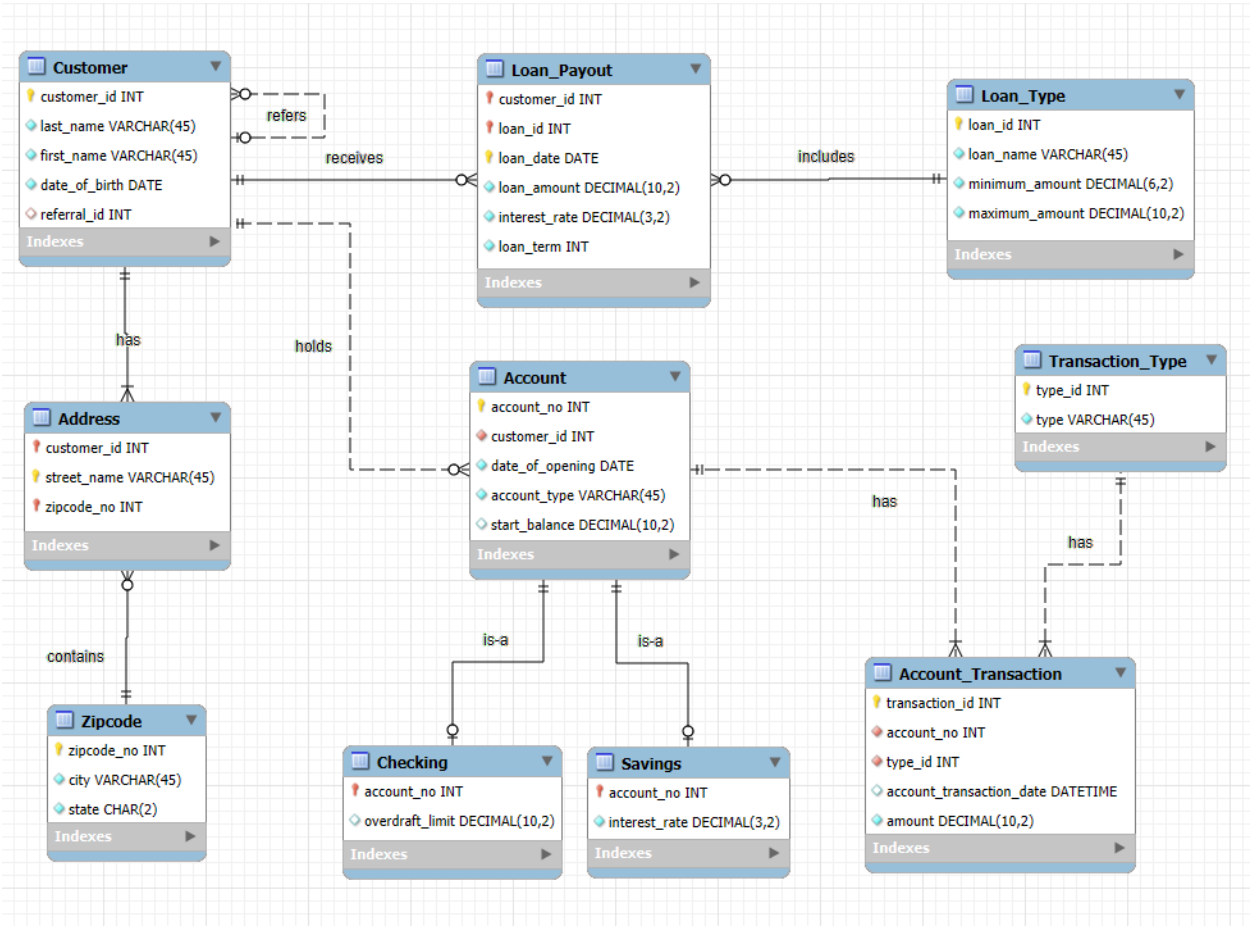
Database Design and Implementation.

My team designed and implemented a database for a Banking Institution. I completed the project using the following steps.

1. Design MySQL EER diagram based on the Logical Data Model (LDM) from Lucid Chart.

EER Diagram:



2. Once the EER model was completed, a schema was generated on MySQL using forward engineering. After creating the schema, I proceeded to the data entry process considering the dependencies when choosing the table entry order. I ensured that the foreign key (FK) values existed as primary keys (PKs) in another table so that it did not result in a foreign key constraint error.

Data Entry:

Zip code Table

```
SELECT * FROM Zipcode;
SELECT * FROM Transaction_Type;
SELECT * FROM Loan_Type;
SELECT * FROM Customer;
SELECT * FROM Address;
```

Result Grid | Filter Rows: | Edit: | Expor

zipcode_no	city	state
10101	New York	NY
19716	Newark	DE
60007	Chicago	IL
77001	Houston	TX
90001	Los Angeles	CA
NULL	NULL	NULL

Transaction_Type Table

```
SELECT * FROM Zipcode;
SELECT * FROM Transaction_Type;
SELECT * FROM Loan_Type;
SELECT * FROM Customer;
SELECT * FROM Address;
```

Result Grid | Filter Rows: | Edit: | Export/Impo

type_id	type
1	Deposit
2	Withdrawal
3	Bank Transfer In
4	Bank Transfer Out
5	Bill Payment
NULL	NULL

Loan_Type Table

```sql
SELECT * FROM Zipcode;
SELECT * FROM Transaction_Type;
SELECT * FROM Loan_Type;
SELECT * FROM Customer;
SELECT * FROM Address;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

loan_id	loan_name	minimum_amount	maximum_amount
11	Home Loan	5000.00	500000.00
22	Car Loan	1000.00	100000.00
33	Personal Loan	500.00	50000.00
44	Education Loan	1000.00	150000.00
55	Business Loan	2500.00	9800000.00
NULL	NULL	NULL	NULL

Customer Table

```sql
SELECT * FROM Zipcode;
SELECT * FROM Transaction_Type;
SELECT * FROM Loan_Type;
SELECT * FROM Customer;
SELECT * FROM Address;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

customer_id	last_name	first_name	date_of_birth	referral_id
1234	Uttreja	Simran	1999-08-28	NULL
2345	Adzuah	Grace	1990-11-20	1234
3456	Robert	Brice	1975-09-30	NULL
4567	Emily	Rain	1992-01-10	3456
5678	Michael	Bonny	1988-12-25	2345
NULL	NULL	NULL	NULL	NULL

Address Table

```
SELECT * FROM Zipcode;
SELECT * FROM Transaction_Type;
SELECT * FROM Loan_Type;
SELECT * FROM Customer;
SELECT * FROM Address;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

customer_id	street_name	zipcode_no
1234	123 Main St	10101
5678	202 Main St	19716
3456	789 Pine Rd	60007
4567	101 Maple Dr	77001
2345	456 Oak Ave	90001
NULL	NULL	NULL

Account Table

```
SELECT * FROM Transaction_Type;
SELECT * FROM Loan_Type;
SELECT * FROM Customer;
SELECT * FROM Address;
SELECT * FROM Account;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

account_no	customer_id	date_of_opening	account_type	start_balance
1001	1234	2023-01-01	Checking	1000.00
1002	2345	2023-02-01	Savings	5000.00
1003	3456	2023-03-01	Checking	1500.00
1004	4567	2023-04-01	Savings	3000.00
1005	5678	2023-05-01	Checking	2000.00
1006	1234	2024-01-01	Savings	2000.00
1007	2345	2024-02-01	Checking	7000.00
1008	3456	2024-03-01	Savings	1900.00
1009	4567	2024-04-01	Checking	8000.00
1010	5678	2024-05-01	Savings	7000.00
NULL	NULL	NULL	NULL	NULL

Checking Table

```
SELECT * FROM Loan_Type;
SELECT * FROM Customer;
SELECT * FROM Address;
SELECT * FROM Account;
SELECT * FROM Checking;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

account_no	overdraft_limit
1001	500.00
1003	750.00
1005	1000.00
1007	800.00
1009	1200.00
NULL	NULL

Savings Table

```
SELECT * FROM Customer;
SELECT * FROM Address;
SELECT * FROM Account;
SELECT * FROM Checking;
SELECT * FROM Savings;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

account_no	interest_rate
1002	0.08
1004	0.12
1006	0.14
1008	0.07
1010	0.06
NULL	NULL

Loan_Payout Table

```
5 •  SELECT * FROM Address;
6 •  SELECT * FROM Account;
7 •  SELECT * FROM Checking;
8 •  SELECT * FROM Savings;
9 •  SELECT * FROM Loan_Payout;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

customer_id	loan_id	loan_date	loan_amount	interest_rate	loan_term
1234	11	2023-03-15	100000.00	0.09	15
2345	22	2023-04-10	20000.00	0.08	5
3456	33	2023-05-15	5000.00	0.04	3
4567	44	2023-06-20	75000.00	0.15	10
5678	55	2023-07-25	150000.00	0.13	20
NULL	NULL	NULL	NULL	NULL	NULL

Account_Transaction Table

```
6 •  SELECT * FROM Account;
7 •  SELECT * FROM Checking;
8 •  SELECT * FROM Savings;
9 •  SELECT * FROM Loan_Payout;
10 •  SELECT * FROM Account_Transaction;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

transaction_id	account_no	type_id	account_transaction_date	amount
1	1001	1	2023-06-01 09:00:00	500.00
2	1002	2	2023-06-02 15:30:00	200.00
3	1003	3	2023-06-03 10:15:00	100.00
4	1004	4	2023-06-04 11:45:00	300.00
5	1005	5	2023-06-05 14:00:00	500.00
6	1001	2	2024-06-01 09:30:00	1000.00
7	1002	3	2024-06-02 14:30:00	700.00
8	1003	4	2024-06-03 11:15:00	800.00
9	1004	1	2024-06-03 08:15:00	400.00
10	1005	5	2024-06-03 07:15:00	100.00
NULL	NULL	NULL	NULL	NULL

3. For the final part of the project, I constructed and implemented SQL queries to pull information out of the database. A description of the query was highlighted together with a use case scenario when that particular query could be used at the bank.

Query 1

Description: This query calculates the total transaction amounts grouped by the different transaction types arranged in descending order.

Use Case - This query helps the bank to know the total amount processed for each type of transaction in order to prioritize resources and analyze trends for operational planning.

 # Query 1

SELECT tt.type AS Transaction_Type, SUM(at.amount) AS Total_Amount

FROM Account_Transaction AS at INNER JOIN Transaction_Type AS tt ON

at.type_id = tt.type_id

GROUP BY tt.type

ORDER BY Total_Amount DESC;

Result:

```
# Query 1

SELECT tt.type AS Transaction_Type, SUM(at.amount) AS Total_Amount
FROM Account_Transaction at INNER JOIN Transaction_Type tt ON
    at.type_id = tt.type_id
GROUP BY tt.type
ORDER BY Total_Amount DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Transaction_Type	Total_Amount
Withdrawal	1200.00
Bank Transfer Out	1100.00
Deposit	900.00
Bank Transfer In	800.00
Bill Payment	600.00

Query 2

Description: This query calculates the total loan amount disbursed for each loan type for total loan amount greater than $50,000 and arranges the result in descending order.

Use Case - This query helps the bank analyze which types of loans have disbursed more than $50,000 which aids the bank in prioritizing marketing campaigns for popular loan types and to assess the bank's risk exposure in the high-value loan category.

 # Query 2

SELECT lt.loan_name AS Loan_Type, SUM(lp.loan_amount) AS Total_Loan_Amount

FROM Loan_Payout lp INNER JOIN Loan_Type lt ON

lp.loan_id = lt.loan_id

GROUP BY lt.loan_name

HAVING SUM(lp.loan_amount) > 50000

ORDER BY Total_Loan_Amount DESC;

Result:

```
# Query 2

SELECT lt.loan_name AS Loan_Type,SUM(lp.loan_amount) AS Total_Loan_Amount
FROM Loan_Payout lp INNER JOIN Loan_Type lt ON
     lp.loan_id = lt.loan_id
GROUP BY lt.loan_name
HAVING SUM(lp.loan_amount) > 50000
ORDER BY Total_Loan_Amount DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Loan_Type	Total_Loan_Amount
Business Loan	150000.00
Home Loan	100000.00
Education Loan	75000.00

Query 3

Description: The query retrieves information about customers with savings accounts, focusing on those residing in cities starting with "N" and having a starting balance above $500 arranged in descending order.

Use Case - The bank's marketing team uses this query to identify high-value customers in cities like "Newark" or "New York" for targeted promotions. This helps the bank focus its efforts on customers who may be more receptive to such opportunities.

 # Query 3

SELECT c.customer_id, c.first_name, c.last_name, a.street_name, z.city, ac.account_no, ac.start_balance

FROM Customer c INNER JOIN Address a ON

c.customer_id = a.customer_id INNER JOIN Zipcode z ON

a.zipcode_no = z.zipcode_no INNER JOIN Account ac ON

c.customer_id = ac.customer_id

WHERE z.city LIKE 'N%'

AND ac.account_type = 'Savings'

AND ac.start_balance > 500

ORDER BY ac.start_balance DESC;

Result:

```
# Query 3

SELECT c.customer_id, c.first_name, c.last_name, a.street_name, z.city, ac.account_no, ac.start_balance
FROM Customer c INNER JOIN Address a ON
    c.customer_id = a.customer_id INNER JOIN Zipcode z ON
    a.zipcode_no = z.zipcode_no INNER JOIN Account ac ON
    c.customer_id = ac.customer_id
WHERE z.city LIKE 'N%'
AND ac.account_type = 'Savings'
AND ac.start_balance > 500
ORDER BY ac.start_balance DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	first_name	last_name	street_name	city	account_no	start_balance
5678	Bonny	Michael	202 Main St	Newark	1010	7000.00
1234	Simran	Uttreja	123 Main St	New York	1006	2000.00

Query 4

Description: This query retrieves a detailed transaction history for each customer ordered by transaction date.

Use Case - A bank manager can use this query to generate a comprehensive report of all transactions for auditing or compliance purposes. It also helps identify trends such as frequent withdrawals, large deposits aiding in monitoring customer behavior, detecting fraud, or offering tailored financial advice.

 # Query 4

SELECT concat(c.first_name,' ',c.last_name) AS Customer_Name, a.account_type, tt.type AS Transaction_Type, at.amount, at.account_transaction_date

FROM Account_Transaction at INNER JOIN Account a ON

at.account_no = a.account_no INNER JOIN Customer c ON

a.customer_id = c.customer_id INNER JOIN Transaction_Type tt ON

at.type_id = tt.type_id

ORDER BY at.account_transaction_date;

Result:

```
# Query 4

SELECT concat(c.first_name,' ',c.last_name) AS Customer_Name, a.account_type, tt.type AS Transaction_Type, at.amount, at.account_transaction_date
FROM Account_Transaction at INNER JOIN Account a ON
    at.account_no = a.account_no INNER JOIN Customer c ON
    a.customer_id = c.customer_id INNER JOIN Transaction_Type tt ON
    at.type_id = tt.type_id
ORDER BY at.account_transaction_date;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer_Name	account_type	Transaction_Type	amount	account_transaction_date
Simran Uttreja	Checking	Deposit	500.00	2023-06-01 09:00:00
Grace Adzuah	Savings	Withdrawal	200.00	2023-06-02 15:30:00
Brice Robert	Checking	Bank Transfer In	100.00	2023-06-03 10:15:00
Rain Emily	Savings	Bank Transfer Out	300.00	2023-06-04 11:45:00
Bonny Michael	Checking	Bill Payment	500.00	2023-06-05 14:00:00
Simran Uttreja	Checking	Withdrawal	1000.00	2024-06-01 09:30:00
Grace Adzuah	Savings	Bank Transfer In	700.00	2024-06-02 14:30:00
Bonny Michael	Checking	Bill Payment	100.00	2024-06-03 07:15:00
Rain Emily	Savings	Deposit	400.00	2024-06-03 08:15:00
Brice Robert	Checking	Bank Transfer Out	800.00	2024-06-03 11:15:00

Query 5

Description: The query retrieves details about all accounts either checking or savings and orders the result by account number.

Use Case - A bank operations team uses this query to prepare a consolidated report of all active accounts showing their specific features such as overdraft limits/interest rates. This also helps identify accounts with missing details ensuring proper alignment with customer agreements.

 # Query 5

SELECT a.account_no, a.account_type, c.overdraft_limit, s.interest_rate

FROM Account a LEFT JOIN Checking c ON

a.account_no = c.account_no LEFT JOIN Savings s ON

a.account_no = s.account_no

ORDER BY a.account_no;

Result:

```
# Query 5

SELECT a.account_no, a.account_type, c.overdraft_limit, s.interest_rate
FROM Account a LEFT JOIN Checking c ON
     a.account_no = c.account_no LEFT JOIN Savings s ON
     a.account_no = s.account_no
ORDER BY a.account_no;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

account_no	account_type	overdraft_limit	interest_rate
1001	Checking	500.00	NULL
1002	Savings	NULL	0.08
1003	Checking	750.00	NULL
1004	Savings	NULL	0.12
1005	Checking	1000.00	NULL
1006	Savings	NULL	0.14
1007	Checking	800.00	NULL
1008	Savings	NULL	0.07
1009	Checking	1200.00	NULL
1010	Savings	NULL	0.06

Query 6

Description: This query retrieves a list of Customers who were not referred by another customer of the bank and the results are ordered by last names.

Use Case - A bank's marketing team could use this query to identify customers who joined independently without a referral. These customers might be ideal targets for a new referral program encouraging them to refer in order to earn incentives.

 # Query 6

SELECT c1.customer_id,c1.last_name, c1.first_name, c1.referral_id, c2.last_name, c2.first_name

FROM Customer c1 LEFT JOIN Customer c2 ON

c1.referral_id = c2.customer_id

WHERE c1.referral_id is Null

Order by 2;

Result:

```
# Query 6

SELECT c1.customer_id,c1.last_name, c1.first_name, c1.referral_id, c2.last_name, c2.first_name
FROM Customer c1 LEFT JOIN Customer c2 ON
    c1.referral_id = c2.customer_id
WHERE c1.referral_id is Null
Order by 2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	last_name	first_name	referral_id	last_name	first_name
3456	Robert	Brice	NULL	NULL	NULL
1234	Uttreja	Simran	NULL	NULL	NULL