FINAL PROJECT REPORT

Bank Management and Credit Card Rewarding System

Group 14

Pramoth Guhan

Akshaya Murugan

(857) 891-6677 (Tel of Student 1)

(857) 423-5726 (Tel of Student 2)

guhan.p@northeastern.edu

murugan.ak@northeastern.edu

Percentage of Effort Contributed by Student 1: 50

Percentage of Effort Contributed by Student 2: 50

Signature of Student 1: Pramoth Guhan

Signature of Student 2: Akshaya Murugan

Submission Date: 4/21/2024

EXECUTIVE SUMMARY:

Our project aims to develop an integrated Bank Management and Credit Card Rewarding System to address the challenges faced by financial institutions in managing operations and rewarding customer loyalty. By standardizing processes and leveraging data-driven insights, we seek to streamline banking operations and enhance customer engagement.

The system's core components include essential entities such as Accounts, Transactions, Credit Cards, Loans, Employees, and Customers. These entities are interconnected to facilitate seamless banking operations and personalized customer experiences.

The Credit Card Rewarding System is designed to incentivize customer engagement and loyalty by offering rewards and benefits for desired behaviors and transactions. It tracks customer interactions and transaction patterns to provide targeted reward offerings and personalized redemption experiences, strengthening the bond between the bank and its customers.

The system's flexibility allows for the implementation of various functionalities, including customer-branch relationships, loan-branch associations, employee-branch affiliations, credit card-merchant interactions, customer-reward program engagements, and reward point-redemption options.

The project aims to revolutionize banking operations and customer engagement, fostering a culture of innovation and excellence in the banking sector.

INTRODUCTION

In today's fiercely competitive banking sector, the combination of efficient operational management and compelling customer interactions is vital for sustained success. However, many financial institutions struggle with disparate systems that handle bank operations and credit card rewards independently, resulting in inefficiencies and underwhelming customer experiences.

Our project aims to develop an integrated Bank Management and Credit Card Rewarding System in order to tackle these issues head-on. By means of standardizing procedures and employing data-driven insights, our solution seeks to revolutionize the way banks manage their operations and reward client loyalty, consequently cultivating an innovative and superior culture within the sector.

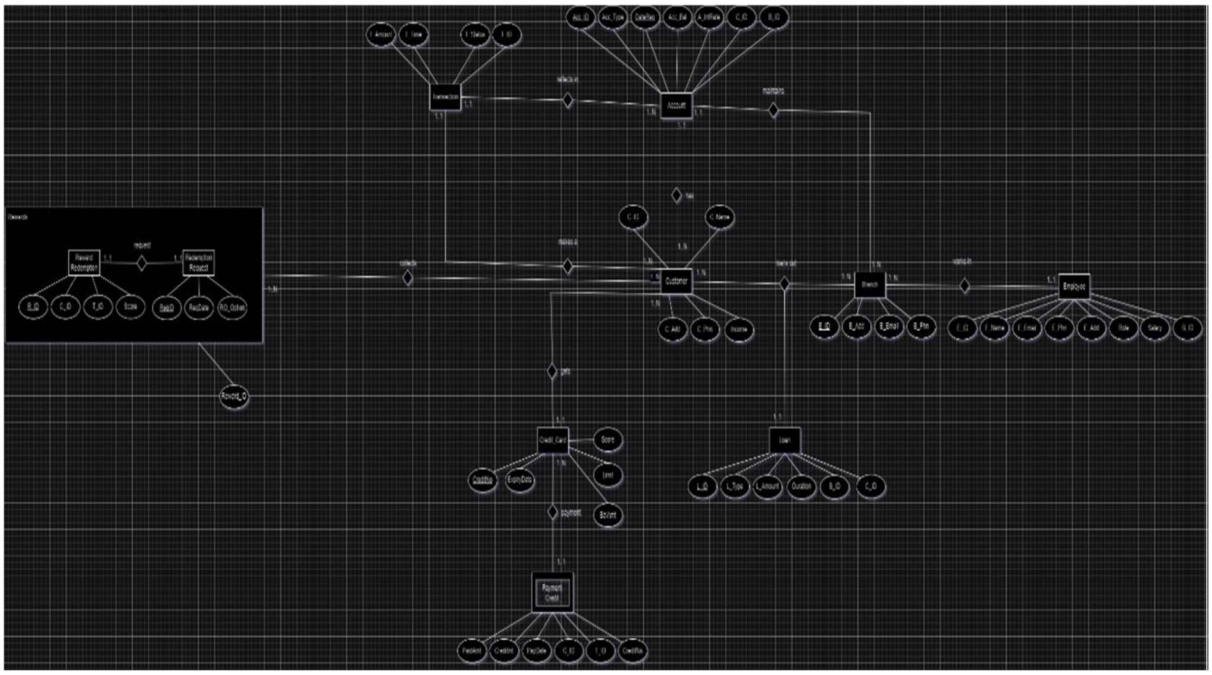
Our dedication to providing tailored experiences that are catered to the demands of each client is the foundation of our strategy. This is accomplished by utilizing thorough client profiles that are enhanced with information about account specifics, credit profiles, and transaction history. With the use of these insights, banks are able to provide specialized services that are highly responsive to the needs and tastes of each customer.

Additionally, our system leverages important entities like Accounts, Transactions, and Credit Cards to highlight the smooth coordination of financial activities. Together, these elements create a streamlined procedure that gives customers easy payment options and tailored financing solutions.

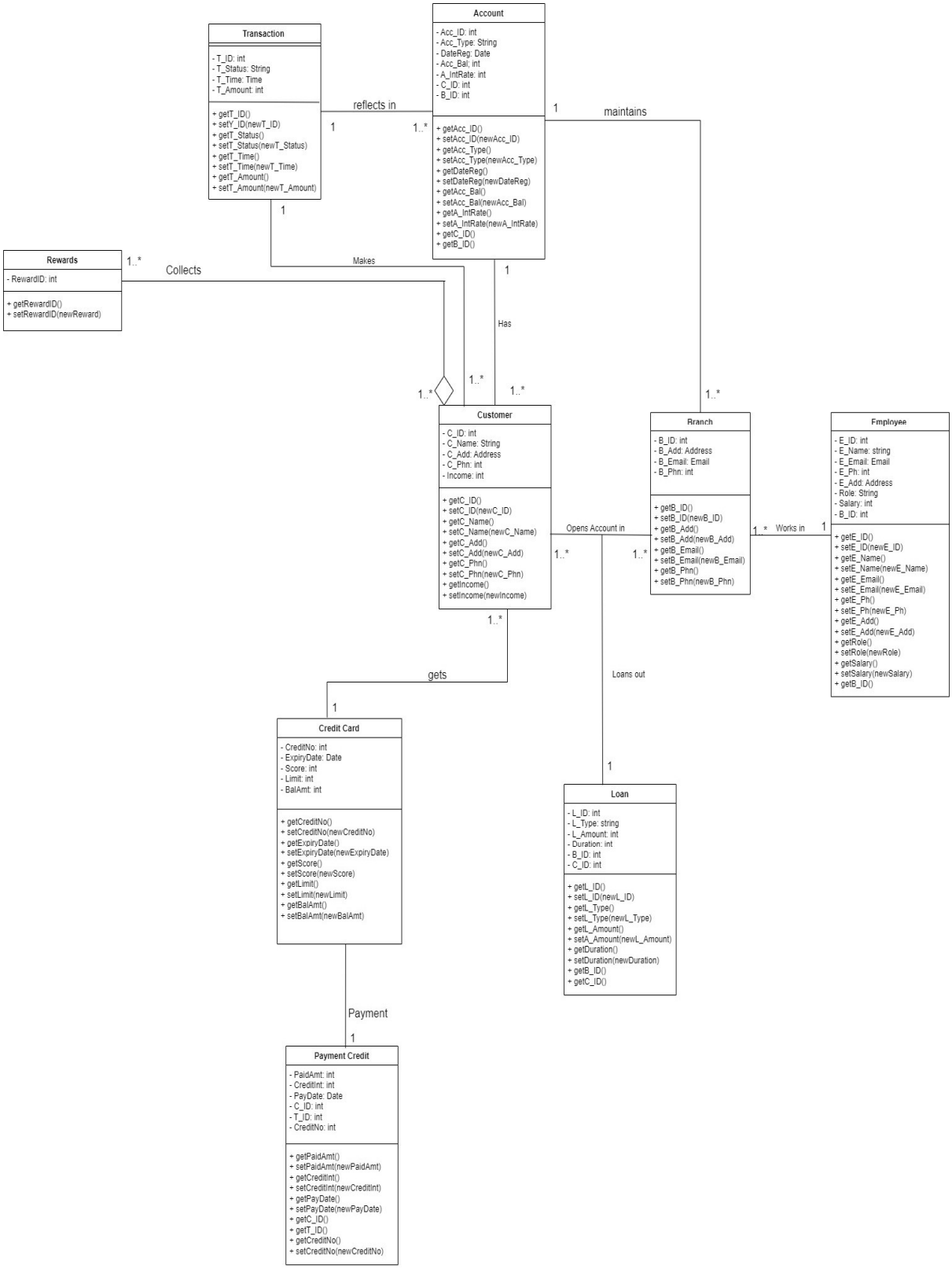
In summary, our project aims to improve efficiency, innovation, and customer pleasure to create new industry standards. It marks a substantial shift in banking operations and client involvement.

II. Conceptual Data Modeling:

1.EER Diagram:



2. UML Diagram:



III. Mapping Conceptual Model to Relational Model:

BRANCH (BRANCH ID, BRANCH ADDRESS, BRANCH EMAIL, BRANCH PHONE NUMBER)

Primary key – BRANCH ID, NOT NULL;

ACCOUNT (ACCOUNT NUMBER, DATE REGISTERED, TYPE, ACCOUNT BALANCE, *BRANCH ID*)

Primary key – ACCOUNT NUMBER, DATE REGISTERED, both NOT NULL;

Foreign key – BRANCH ID from BRANCH relation, NOT NULL;

EMPLOYEE (EMPLOYEE ID, EMPLOYEE NAME, EMPLOYEE EMAIL, SALARY, *BRANCH ID*)

Primary key – EMPLOYEE ID, NOT NULL;

Foreign key – BRANCH ID from BRANCH relation, NOT NULL;

CUSTOMER (CUSTOMER ID, CUSTOMER NAME, CUSTOMER ADDRESS, CUSTOMER PHONE NO, INCOME, *BRANCH ID*)

Primary key – CUSTOMER ID, NOT NULL;

Foreign key – BRANCH ID from BRANCH, NOT NULL;

TRANSACTION (TRANSACTION ID, TRANSACTION AMOUNT, TRANSACTION TIME, STATUS, *CUSTOMER ID*)

Primary key – TRANSACTION ID, NOT NULL

Foreign key – CUSTOMER ID from CUSTOMER relation, NOT NULL;

LOAN (LOAN ID, LOAN TYPE, LOAN AMOUNT, LOAN DURATION, *BRANCH ID, CUSTOMER ID*)

Primary key – LOAN ID, NOT NULL;

Foreign key – BRANCH ID from BRANCH, CUSTOMER ID ID from CUSTOMER, NOT NULL;

CREDIT CARD (CREDIT CARD NUMBER, EXPIRY DATE, SCORE, LIMIT, BALANCE CREDIT, *CUSTOMER ID*)

Primary key – CREDIT CARD NUMBER, NOT NULL;

Foreign key – CUSTOMER ID from CUSTOMER relation, NOT NULL;

PAYMENT (PAID AMOUNT, DATE OF PAYMENT, *CREDIT CARD NUMBER*)

Foreign key – CREDIT CARD NUMBER from CREDIT CARD, NOT NULL

REWARD (REWARD ID, REWARD AMOUNT, REQUEST DATE, REDEMPTION OPTION, *CUSTOMER ID*)

Primary key – REWARD ID, NOT NULL;

Foreign key – CUSTOMER ID from CUSTOMER relation, NOT NULL;

MERCHANT (MERCHANT ID, MERCHANT NAME, CATEGORY, *BRANCH ID, REWARD ID*)

Primary key – MERCHANT ID, NOT NULL;

Foreign key – BRANCH ID from BRANCH relation, REWARD ID from REWARD relation, NOT NULL;

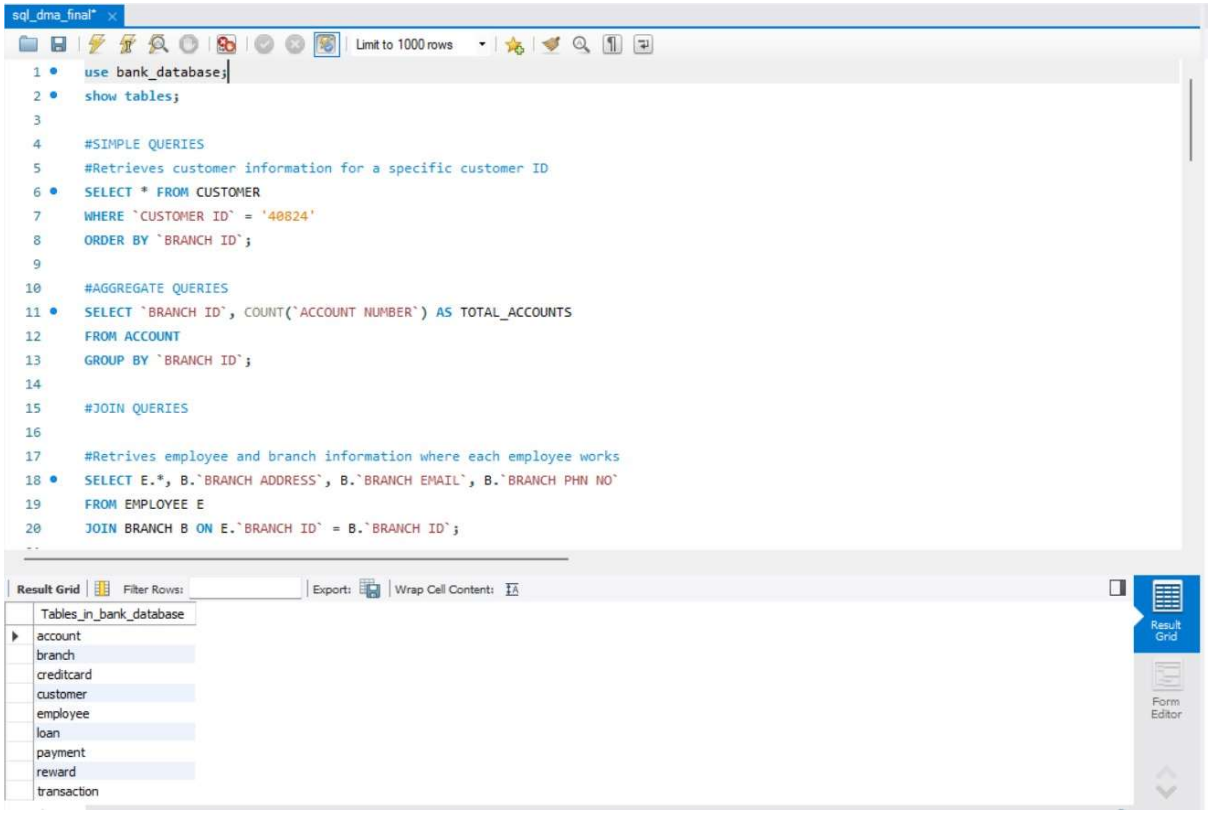
IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

use bank_database;

show tables;



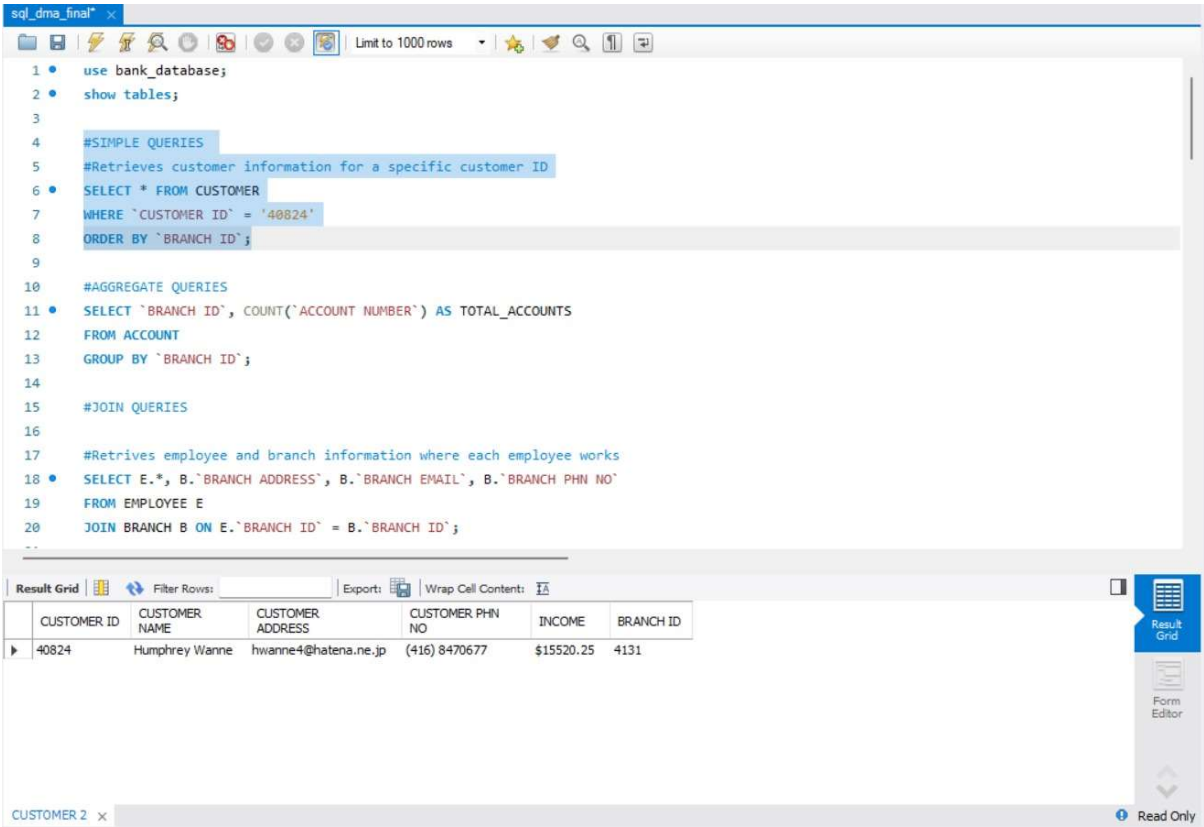
#SIMPLE QUERIES

This query retrieves customer information for a specific customer ID ('40824') and orders the results by branch ID.

SELECT * FROM CUSTOMER

WHERE CUSTOMER ID = '40824'

ORDER BY BRANCH ID;



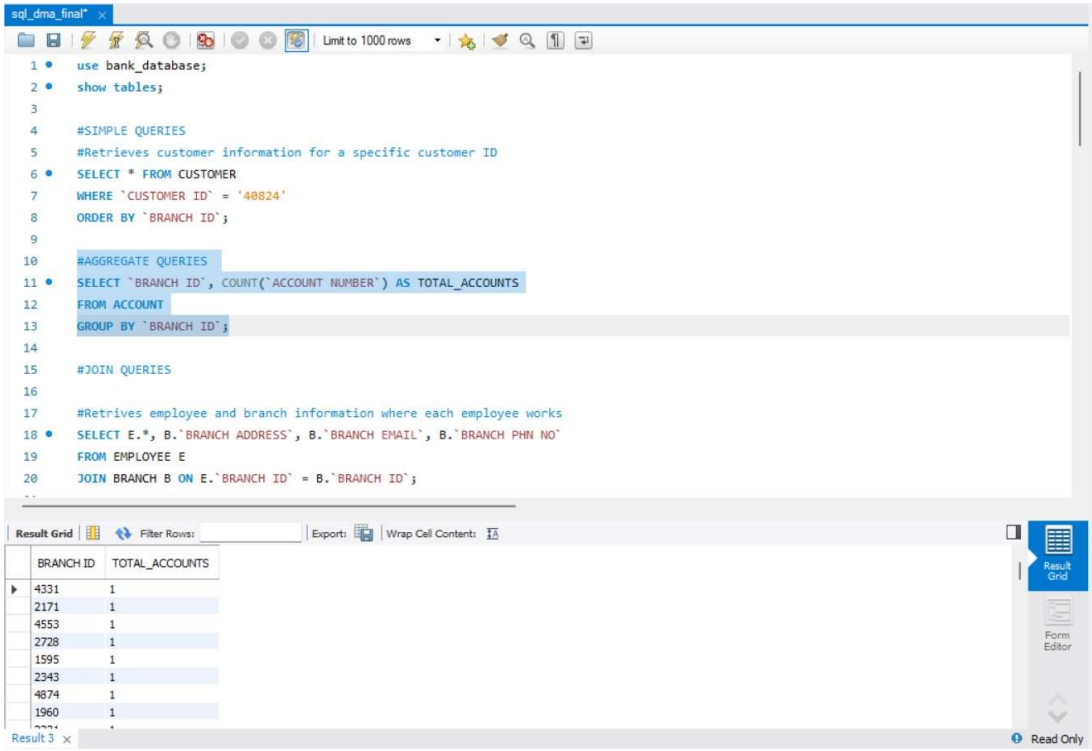
#AGGREGATE QUERIES

This query calculates the total number of accounts for each branch by counting the number of account numbers (ACCOUNT NUMBER) in the ACCOUNT table and grouping them by branch ID.

SELECT BRANCH ID, COUNT(ACCOUNT NUMBER) AS TOTAL_ACCOUNTS

FROM ACCOUNT

GROUP BY BRANCH ID;



#JOIN QUERIES

The first join query combines employee information with corresponding branch details, including address, email, and phone number, by joining the EMPLOYEE and BRANCH tables on the BRANCH ID column.

SELECT E.*, B.BRANCH ADDRESS, B.BRANCH EMAIL, B.BRANCH PHN NO

FROM EMPLOYEE E

JOIN BRANCH B ON E.BRANCH ID = B.BRANCH ID;

```
#JOIN QUERIES

#Retrives employee and branch information where each employee works
SELECT E.*, B.`BRANCH ADDRESS`, B.`BRANCH EMAIL`, B.`BRANCH PHN NO`
FROM EMPLOYEE E
JOIN BRANCH B ON E.`BRANCH ID` = B.`BRANCH ID`;
```

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

EMPLOYEE ID	EMPLOYEE NAME	EMPLOYEE EMAIL	SALARY	BRANCH ID	BRANCH ADDRESS	BRANCH EMAIL	BRANCH PHN NO
782	Jonell Poure	jpoure2i@skyrock.com	9379.48	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
843	Anna-diane Razzell	arazzell28@berkeley.edu	6973.51	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
670	Darlene Tolefree	dtolefree1y@skype.com	8188.15	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
830	Sidnee Raine	sraine1o@studiopress.com	6081.23	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
576	Rebbecca Maddrah	rmaddrah1e@yelp.com	4114.04	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
732	Krystle Nijs	knijs14@google.com	8724.6	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
392	Anestassia Monkhouse	amonkhouseu@tumblr.com	5303.52	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
731	Juieta Benck	jbenckk@jalbum.net	7579.4	2808	Room 1802	schasier0@hhs.gov	(518) 9847741

Result 4 | Read Only

The second join query merges payment information with credit card details by joining the PAYMENT and CREDITCARD tables on the CREDIT CARD NUMBER column.

SELECT PC.*, CC.EXPIRY DATE, CC.SCORE, CC.LIMIT, CC.BALANCE CREDIT

FROM PAYMENT PC

INNER JOIN CREDITCARD CC ON PC.CREDIT CARD NUMBER = CC.CREDIT CARD NUMBER;

```
#Retrives payment information and corresponding credit card information
SELECT PC.*, CC.`EXPIRY DATE`, CC.`SCORE`, CC.`LIMIT`, CC.`BALANCE CREDIT`
FROM PAYMENT PC
INNER JOIN CREDITCARD CC ON PC.`CREDIT CARD NUMBER` = CC.`CREDIT CARD NUMBER`;
```

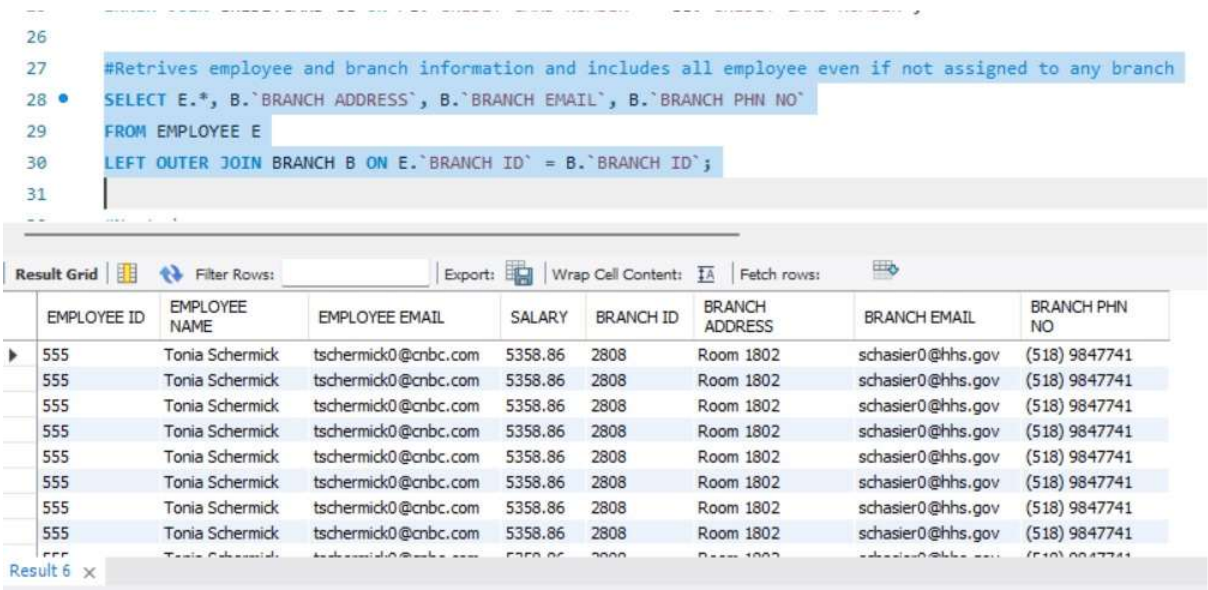
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

PAID AMOUNT	DATE OF PAYMENT	CREDIT CARD NUMBER	CUSTOMER ID	EXPIRY DATE	SCORE	LIMIT	BALANCE CREDIT
$8750.12	06/24/2022	5.11e15	83713	2027-12	445	$5784.49	$2063.35
$7003.77	04/18/2022	5.11e15	99614	2027-12	445	$5784.49	$2063.35
$2419.47	05/25/2022	5.11e15	40888	2027-12	445	$5784.49	$2063.35
$3992.10	05-08-2022	5.11e15	62839	2027-12	445	$5784.49	$2063.35
$9978.93	07/31/2023	5.11e15	15619	2027-12	445	$5784.49	$2063.35
$8182.38	06-12-2022	5.11e15	47870	2027-12	445	$5784.49	$2063.35
$4028.14	09/22/2022	5.11e15	30300	2027-12	445	$5784.49	$2063.35
$913.22	05/29/2022	5.11e15	35133	2027-12	445	$5784.49	$2063.35

Result 5

The third join query pairs employee information with branch details using a left outer join to ensure all employee records are included, even if there's no corresponding branch information.

```
SELECT E.*, B.BRANCH ADDRESS, B.BRANCH EMAIL, B.BRANCH PHN NO

FROM EMPLOYEE E

LEFT OUTER JOIN BRANCH B ON E.BRANCH ID = B.BRANCH ID;
```



```
#Retrives employee and branch information and includes all employee even if not assigned to any branch
SELECT E.*, B.`BRANCH ADDRESS`, B.`BRANCH EMAIL`, B.`BRANCH PHN NO`
FROM EMPLOYEE E
LEFT OUTER JOIN BRANCH B ON E.`BRANCH ID` = B.`BRANCH ID`;
```

EMPLOYEE ID	EMPLOYEE NAME	EMPLOYEE EMAIL	SALARY	BRANCH ID	BRANCH ADDRESS	BRANCH EMAIL	BRANCH PHN NO
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741
555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808	Room 1802	schasier0@hhs.gov	(518) 9847741

#Nested query

This nested query selects employee information where the salary exceeds the average salary of employees in the same branch. It calculates the average salary per branch in a subquery and compares each employee's salary with the average salary of their respective branch.

```
SELECT E.* FROM EMPLOYEE E

JOIN (

   SELECT BRANCH ID, AVG(SALARY) AS AVG_BRANCH_SALARY

   FROM EMPLOYEE

   GROUP BY BRANCH ID

) AS AVG_SALARY_BRANCH

ON E.BRANCH ID = AVG_SALARY_BRANCH.BRANCH ID
```

WHERE E.SALARY > AVG_SALARY_BRANCH.AVG_BRANCH_SALARY;

```
31
32  #Nested query
33
34  #Retrives employee information having salary higher than the average salary
35  SELECT E.* FROM EMPLOYEE E
36  JOIN (
37      SELECT `BRANCH ID`, AVG(`SALARY`) AS AVG_BRANCH_SALARY
38      FROM EMPLOYEE
39      GROUP BY `BRANCH ID`
40  ) AS AVG_SALARY_BRANCH
41  ON E.`BRANCH ID` = AVG_SALARY_BRANCH.`BRANCH ID`
42  WHERE E.`SALARY` > AVG_SALARY_BRANCH.AVG_BRANCH_SALARY;
43
```

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

EMPLOYEE ID	EMPLOYEE NAME	EMPLOYEE EMAIL	SALARY	BRANCH ID
506	Sharyl Yankov	syankov2@elpais.com	7048.32	2508
878	Desdemona MacParlan	dmacparlan4@bigcartel.com	9672.85	4131
595	Warde Bricknell	wbricknell5@rediff.com	7183.41	2847
108	Agosto Streak	astreak6@answers.com	6516.01	4720
669	Chaunce Clapham	cclapham8@woothemes.com	7688.28	4826
358	Crichton McAlester	cmcalestera@dion.ne.jp	8294.44	2808
669	Frederique Ninnoli	fninnolib@vk.com	9481.85	2946
177	Rufe Beaufoy	rbeaufoyc@youku.com	9464.98	2508

Result 7

#UNION

This query combines unique IDs from the BRANCH and CUSTOMER tables into a single column named ID.

```
SELECT ID
FROM (
    SELECT BRANCH ID AS ID FROM BRANCH
    UNION
    SELECT CUSTOMER ID AS ID FROM CUSTOMER
) AS IDs;
```

```
#UNION
#Retrives branch id and customer id in a single table
SELECT ID
FROM (
    SELECT `BRANCH ID` AS ID FROM BRANCH
    UNION
    SELECT `CUSTOMER ID` AS ID FROM CUSTOMER
) AS IDs;
```

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

ID
2808
2946
2508
3021
4131
2847
4720
1252
4826

Result 8

#SUBQUERIES in SELECT CLAUSE

This query retrieves the names of customers along with the latest request date recorded for each customer in the "REWARD" table.

```
SELECT CUSTOMER NAME, (
   SELECT MAX(REQUEST DATE) FROM REWARD
   WHERE REWARD.CUSTOMER ID = CUSTOMER.CUSTOMER ID
) AS LAST_REQUEST_DATE
FROM CUSTOMER;
```

```
#SUBQUERIES in SELECT CLAUSE
#Retrives customer name and recent request date for each customer
SELECT `CUSTOMER NAME`, (
    SELECT MAX(`REQUEST DATE`) FROM REWARD
    WHERE REWARD.`CUSTOMER ID` = CUSTOMER.`CUSTOMER ID`
) AS LAST_REQUEST_DATE
FROM CUSTOMER;
```

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

CUSTOMER NAME	LAST_REQUEST_DATE
Yurik Willishire	04/21/2022
Randall Guidera	07/29/2022
Rhody Dunlap	08-11-2023
Lars Normanton	04/29/2023
Humphrey Wanne	02-03-2023
Alfy Girk	01-06-2024
Lanae Bestiman	10/31/2022
Sim Houldey	10/13/2023

Result 9

#SUBQUERIES in FROM CLAUSE

This query counts the number of transactions associated with each branch, displaying the branch ID, branch address, and the total number of transactions made at each branch.

SELECT B.BRANCH ID, B.BRANCH ADDRESS, COUNT(T.TRANSACTION ID) AS NUM_TRANSACTIONS

FROM BRANCH B

JOIN (SELECT BRANCH ID, TRANSACTION ID FROM TRANSACTION INNER JOIN CUSTOMER ON TRANSACTION.CUSTOMER ID = CUSTOMER.CUSTOMER ID)

T ON B.BRANCH ID = T.BRANCH ID

GROUP BY B.BRANCH ID, B.BRANCH ADDRESS;

#BETWEEN

This query retrieves transactions from the TRANSACTION table where the transaction amount falls between $1000.00 and $5000.00, inclusive.

SELECT * FROM TRANSACTION

WHERE TRANSACTION AMOUNT BETWEEN '$1000.00' AND '$5000.00';

```
60
61      #BETWEEN
62      #Retrives transaction amount between $1000 and $5000
63 •    SELECT * FROM TRANSACTION
64      WHERE `TRANSACTION AMOUNT` BETWEEN '$1000.00' AND '$5000.00';
65
```

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

TRANSACTION ID	TRANSACTION AMOUNT	TRANSACTION TIME	STATUS	CUSTOMER ID
8341882	$42001.28	19:15:49	SUCCESS	57849
8299034	$28211.43	21:06:46	SUCCESS	37152
9263762	$1785.49	23:07:02	FAIL	40824
9968591	$38568.93	22:29:00	SUCCESS	82764
5704525	$29437.94	04:34:23	FAIL	66412
6666708	$35284.35	17:27:05	FAIL	94474
4507010	$2991.17	14:25:15	SUCCESS	12157
6361695	$13160.88	06:19:52	FAIL	36468

RANSACTION 10 ×

#CORRELATED QUERY

This query identifies branches with a higher number of accounts compared to the average number of accounts per branch.

SELECT b.BRANCH ID, COUNT(a.ACCOUNT NUMBER) AS num_accounts

FROM BRANCH b

JOIN ACCOUNT a ON b.BRANCH ID = a.BRANCH ID

GROUP BY b.BRANCH ID

HAVING COUNT(a.ACCOUNT NUMBER) > (

```
    SELECT AVG(num_accounts)
    FROM (
        SELECT COUNT(ACCOUNT NUMBER) AS num_accounts
        FROM ACCOUNT
        GROUP BY BRANCH ID
    ) AS avg_accounts_per_bank
);
```

```
#CORRELATED QUERY
#Retrives total accounts of each branch where count is greater than average accounts of number per branch
SELECT b.`BRANCH ID`, COUNT(a.`ACCOUNT NUMBER`) AS num_accounts
FROM BRANCH b
JOIN ACCOUNT a ON b.`BRANCH ID` = a.`BRANCH ID`
GROUP BY b.`BRANCH ID`
HAVING COUNT(a.`ACCOUNT NUMBER`) > (
    SELECT AVG(num_accounts)
    FROM (
        SELECT COUNT(`ACCOUNT NUMBER`) AS num_accounts
        FROM ACCOUNT
        GROUP BY `BRANCH ID`
    ) AS avg_accounts_per_branch
);
```

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

BRANCH ID	num_accounts
4826	10

Result 11

#ANY/ALL

This query retrieves customers with incomes greater than at least one other customer at the same branch.

```
SELECT * FROM CUSTOMER c1
WHERE INCOME > ANY (
    SELECT INCOME
```

```
    FROM CUSTOMER c2
    WHERE c1.BRANCH ID = c2.BRANCH ID
    AND c1.CUSTOMER ID <> c2.CUSTOMER ID
);
```

```
80
81  #ANY
82  #Retrives income of customer is greater than atleast one customer in that branch
83  SELECT * FROM CUSTOMER c1
84  WHERE INCOME > ANY (
85      SELECT INCOME
86      FROM CUSTOMER c2
87      WHERE c1.`BRANCH ID` = c2.`BRANCH ID`
88      AND c1.`CUSTOMER ID` <> c2.`CUSTOMER ID`
89  );
```

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

CUSTOMER ID	CUSTOMER NAME	CUSTOMER ADDRESS	CUSTOMER PHN NO	INCOME	BRANCH ID
57849	Yurik Willishire	ywillishire0@blogspot.com	(162) 9390959	$36583.20	2808
42148	Randall Guidera	rguidera1@unblog.fr	(797) 8232112	$184884.93	2946
37152	Lars Normanton	lnormanton3@utexas.edu	(574) 1562580	$185846.18	3021
40824	Humphrey Wanne	hwanne4@hatena.ne.jp	(416) 8470677	$15520.25	4131
82764	Alfy Girk	agirk5@stumbleupon.com	(862) 5751839	$196591.82	2847
66412	Lanae Bestiman	lbestiman6@google.pl	(783) 2854746	$151168.59	4720
19623	Nealy Morat	nmorat8@sogou.com	(838) 6830519	$35507.63	4826
63798	Ansel Ebbing	aebbing9@omniture.com	(484) 9548764	$18203.48	4999

USTOMER 12

#EXISTS

The first query fetches customers who have redeemed a reward with the redemption option of "Coupon".

```
SELECT * FROM CUSTOMER c
WHERE EXISTS (
    SELECT 1
    FROM REWARD r
    WHERE r.CUSTOMER ID = c.CUSTOMER ID
     AND r.REDEMPTION OPTION = 'COUPON'
```

);

```
#EXISTS
#Retrives customer details who has redeemed atleast one reward as coupon
SELECT * FROM CUSTOMER c
WHERE EXISTS (
    SELECT 1
    FROM REWARD r
    WHERE r.`CUSTOMER ID` = c.`CUSTOMER ID`
      AND r.`REDEMPTION OPTION` = 'COUPON'
);
```

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

CUSTOMER ID	CUSTOMER NAME	CUSTOMER ADDRESS	CUSTOMER PHN NO	INCOME	BRANCH ID
57849	Yurik Willishire	ywillishire0@blogspot.com	(162) 9390959	$36583.20	2808
37152	Lars Normanton	lnormanton3@utexas.edu	(574) 1562580	$185846.18	3021
82764	Alfy Girk	agirk5@stumbleupon.com	(862) 5751839	$196591.82	2847
66412	Lanae Bestiman	lbestiman6@google.pl	(783) 2854746	$151168.59	4720
71486	Sim Houldey	shouldey7@jugem.jp	(903) 1123608	$114202.08	1252
63798	Ansel Ebbing	aebbing9@omniture.com	(484) 9548764	$18203.48	4999
94474	Daphna Rigmand	drigmanda@last.fm	(843) 7225740	$104431.94	2808
12157	Ashlie Matyushonok	amatyushonokb@hhs.gov	(499) 6384834	$128660.77	2946

USTOMER 13

The second query retrieves branches without any customers earning more than $100,000.

```
SELECT * FROM BRANCH b
WHERE NOT EXISTS (
   SELECT 1
   FROM CUSTOMER c
   WHERE c.BRANCH ID = b.BRANCH ID
    AND c.INCOME > 100000
);
```

```sql
#Retrives branches with no customer earning more than $100,000
SELECT * FROM BRANCH b
WHERE NOT EXISTS (
    SELECT 1
    FROM CUSTOMER c
    WHERE c.`BRANCH ID` = b.`BRANCH ID`
      AND c.`INCOME` > 100000
);
```

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

BRANCH ID	BRANCH ADDRESS	BRANCH EMAIL	BRANCH PHN NO
2808	Room 1802	schasier0@hhs.gov	(518) 9847741
2946	Room 1258	mlambswood1@wufoo.com	(167) 4279338
2508	PO Box 78161	dsollam2@theguardian.com	(935) 3073168
3021	Suite 71	gsimonini3@fda.gov	(689) 4997059
4131	14th Floor	ficzokvitz4@apache.org	(375) 3088743
2847	PO Box 89484	rrothermel5@netlog.com	(263) 4603387
4720	10th Floor	battestone6@cbc.ca	(196) 1658779
1252	Suite 53	rwhittington7@vimeo.com	(950) 1049145

BRANCH 14

NoSQL Implementation:

1.This query retrieves an account from the "Account" collection in the database where the value of the field "ACCOUNT_NUMBER" is equal to "21-303-2973".

db.Account.find({ACCOUNT_NUMBER:"21-303-2973"});

2. This query performs an aggregation operation on the "Employee" collection. It groups all documents in the collection together (since `_id: null`), then calculates the average value of the "SALARY" field across all documents.

db.Employee.aggregate([{ $group: { _id: null, avg_salary: { $avg: "$SALARY" } } }]);

```
BankDB> db.Account.find({ACCOUNT_NUMBER:"21-303-2973"});
[
  {
    _id: ObjectId('661c38a84b3c6b6e8da7b910'),
    ACCOUNT_NUMBER: '21-303-2973',
    DATE_REGISTERED: '07/14/2023',
    ACCOUNT_TYPE: false,
    ACCOUNT_BALANCE: '$2839.71',
    BRANCH_ID: 4331
  }
]
BankDB> db.Employee.aggregate([{ $group: { _id: null, avg_salary: { $avg: "$SALARY" } } }]);
[ { _id: null, avg_salary: 7011.8014 } ]
BankDB> |
```

3. This query retrieves a single document from the "Customer" collection. Since it uses `findOne()`, it doesn't specify any criteria, so it will return any one document from the collection.

db.Customer.findOne();

```
BankDB> db.Customer.findOne();
{
  _id: ObjectId('661c3bb84b3c6b6e8da7ba42'),
  CUSTOMER_ID: 57849,
  CUSTOMER_NAME: 'Yurik Willishire',
  CUSTOMER_ADDRESS: 'ywillishire0@blogspot.com',
  CUSTOMER_PHN_NO: '(162) 9390959',
  INCOME: '$36583.20',
  BRANCH_ID: 2808
}
BankDB> |
```

4. This query retrieves documents from the "Loan" collection. It doesn't specify any filter criteria, so it retrieves all documents. It then sorts the retrieved documents first by "LOAN_AMOUNT" in descending order (`-1`), and then by "LOAN_DURATION" in ascending order (`1`).

db.Loan.find().sort({ LOAN_AMOUNT: -1, LOAN_DURATION: 1 });

```
mongosh mongodb://127.0.0.

BankDB> db.Loan.find().sort({ LOAN_AMOUNT: -1, LOAN_DURATION: 1 });
[
  {
    _id: ObjectId('661c3bea4b3c6b6e8da7bb0f'),
    LOAN_ID: 215386,
    LOAN_TYPE: 'EDUCATION',
    LOAN_AMOUNT: '$99346.56',
    LOAN_DURATION: '(877) 9565821',
    BRANCH_ID: 2802,
    CUSTOMER_ID: 37152
  },
  {
    _id: ObjectId('661c3bea4b3c6b6e8da7bb32'),
    LOAN_ID: 908965,
    LOAN_TYPE: 'EDUCATION',
    LOAN_AMOUNT: '$95312.23',
    LOAN_DURATION: '(973) 8154524',
    BRANCH_ID: 4317,
    CUSTOMER_ID: 51222
  },
  {
    _id: ObjectId('661c3bea4b3c6b6e8da7bb40'),
    LOAN_ID: 182217,
    LOAN_TYPE: 'EDUCATION',
    LOAN_AMOUNT: '$93785.62',
    LOAN_DURATION: '(339) 1584085',
    BRANCH_ID: 4661,
    CUSTOMER_ID: 22858
  },
  {
    _id: ObjectId('661c3bea4b3c6b6e8da7bb2e'),
    LOAN_ID: 567030,
    LOAN_TYPE: 'PERSONAL',
    LOAN_AMOUNT: '$92992.49',
    LOAN_DURATION: '(796) 7268052',
    BRANCH_ID: 1156,
    CUSTOMER_ID: 39848
  },
  {
    _id: ObjectId('661c3bea4b3c6b6e8da7bb5a'),
    LOAN_ID: 555256,
    LOAN_TYPE: 'EDUCATION',
    LOAN_AMOUNT: '$91926.51',
    LOAN_DURATION: '(761) 5607281',
```

5. This query performs an aggregation operation on the "Customer" collection. It first looks up related documents from the "Loan" collection based on matching "CUSTOMER_ID" fields between the two collections. It then projects (selects) specific fields from the "Customer" collection (`CUSTOMER_NAME`, `CUSTOMER_ADDRESS`), along with the array of related loans, and renames it as `customer_loans`.

db.Customer.aggregate([{ $lookup: {from: "Loan", localField: "CUSTOMER_ID", foreignField: "CUSTOMER_ID", as: "customer_loans"} }, { $project: { CUSTCUSTOMER_NAME: 1, CUSTOMER_ADDRESS: 1, customer_loans: 1 } }]);

```
mongosh mongodb://127.0.0.1 ×   mongosh mongodb://127.0.0. ×
BankDB> db.Customer.aggregate([{ $lookup: {from: "Loan", localField: "CUSTOMER_ID", foreignField: "CUSTOMER_ID", as: "customer_loans"} }, { $project: { CUST
CUSTOMER_NAME: 1, CUSTOMER_ADDRESS: 1, customer_loans: 1 } } ])
[
  {
    _id: ObjectId('661c3bb84b3c6b6e8da7ba42'),
    CUSTOMER_NAME: 'Yurik Willishire',
    CUSTOMER_ADDRESS: 'ywillishire0@blogspot.com',
    customer_loans: [
      {
        _id: ObjectId('661c3bea4b3c6b6e8da7bb0c'),
        LOAN_ID: 681419,
        LOAN_TYPE: 'EDUCATION',
        LOAN_AMOUNT: '$36891.56',
        LOAN_DURATION: '(496) 3689741',
        BRANCH_ID: 4702,
        CUSTOMER_ID: 57849
      }
    ]
  },
  {
    _id: ObjectId('661c3bb84b3c6b6e8da7ba43'),
    CUSTOMER_NAME: 'Randall Guidera',
    CUSTOMER_ADDRESS: 'rguideral@unblog.fr',
    customer_loans: [
      {
        _id: ObjectId('661c3bea4b3c6b6e8da7bb0d'),
        LOAN_ID: 845465,
        LOAN_TYPE: 'EDUCATION',
        LOAN_AMOUNT: '$85288.28',
        LOAN_DURATION: '(927) 5847949',
        BRANCH_ID: 4533,
        CUSTOMER_ID: 42148
      }
    ]
  },
  {
    _id: ObjectId('661c3bb84b3c6b6e8da7ba44'),
    CUSTOMER_NAME: 'Rhody Dunlap',
    CUSTOMER_ADDRESS: 'rdunlap2@hibu.com',
    customer_loans: [
      {
        _id: ObjectId('661c3bea4b3c6b6e8da7bb0e'),
        LOAN_ID: 770766,
        LOAN_TYPE: 'PERSONAL',
```

V. Database Access via Python

The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using sqlalchemy , followed by converting the list into a dataframe using create engine and using matplotlib to plot the graphs for the analytics.

jupyter Python Implementation Last Checkpoint: Last Thursday at 02:33 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Bank Management and Credit Card Rewarding System

MySQL Connection Implementation

In [1]:
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
```

In [2]:
```
connection = mysql.connector.connect(
        host='localhost',
        database='bank_database',
        user='root',
        password='pramothguhan12',

    )
```

In [3]:
```
if connection.is_connected():
        db_Info = connection.get_server_info()
        print(f"Connected to MySQL Server version {db_Info}")


        cursor = connection.cursor()
```

```
Connected to MySQL Server version 8.0.36
```

In [4]:
```
connection
```

Out[4]:
```
<mysql.connector.connection_cext.CMySQLConnection at 0x1bbb8212790>
```

1. This query retrieves all the records from the employee table, including all columns. It essentially returns all the information stored in the employee table.

jupyter Python Implementation Last Checkpoint: Last Thursday at 02:33 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Query Implementation

In [5]:
```
df = pd.read_sql_query("SELECT * FROM employee", connection)
df
```

```
C:\Users\pramo\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engin
e/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLA
lchemy
  warnings.warn(
```

Out[5]:

	EMPLOYEE ID	EMPLOYEE NAME	EMPLOYEE EMAIL	SALARY	BRANCH ID
0	555	Tonia Schermick	tschermick0@cnbc.com	5358.86	2808
1	839	Derrick Bettenay	dbettenay1@netvibes.com	7313.88	2946
2	506	Sharyl Yankov	syankov2@elpais.com	7048.32	2508
3	714	Marcella Allbones	malbones3@bizjournals.com	7148.82	3021
4	878	Desdemona MacParlan	dmacparlan4@bigcartel.com	9672.85	4131
...	...	...	...	...	...
95	300	Colas Handrik	chandrik2n@auda.org.au	4514.90	2847
96	124	Harv Allgood	hallgood2o@mediafire.com	6186.83	4720
97	406	Claudio Hunting	chunting2p@opensource.org	6566.09	1252
98	100	Sharyl Meenehan	smeenehan2q@addthis.com	7877.04	4826
99	701	Rawley Forestel	rforestel2r@foxnews.com	8518.41	4999

100 rows × 5 columns

2. This query retrieves all the records from the reward table, including all columns. It essentially returns all the information stored in the reward table.

```
In [6]: df = pd.read_sql_query("SELECT * FROM reward;", connection)
        df.head(10)
```

```
C:\Users\pramo\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engi
ne/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using S
QLAlchemy
  warnings.warn(
```

Out[6]:

	REWARD ID	REWARD AMOUNT	REQUEST DATE	REDEMPTION OPTION	CUSTOMER ID
0	66067474	$1172.47	04/21/2022	COUPON	57849
1	15901910	$2667.44	07/29/2022	WEBSITE	42148
2	61884675	$3612.69	08-11-2023	APP	99941
3	48276456	$4060.61	04/29/2023	COUPON	37152
4	58233430	$2795.17	02-03-2023	APP	40824
5	51236128	$5175.59	01-06-2024	COUPON	82764
6	62919492	$603.09	10/31/2022	COUPON	66412
7	87440465	$3596.64	10/13/2023	COUPON	71486

3. This query retrieves the CUSTOMER ID, INCOME, and BRANCH ID columns from the CUSTOMER table. The data is ordered by the INCOME column in descending order (DESC), and only the top 10 records are returned (LIMIT 10).

```
In [7]: df = pd.read_sql_query("SELECT `CUSTOMER ID`, `INCOME`, `BRANCH ID` FROM `CUSTOMER` ORDER BY `INCOME` DESC LIMIT 10;", connectior
        df.head(10)
```

```
C:\Users\pramo\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engin
e/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLA
lchemy
  warnings.warn(
```

Out[7]:

	CUSTOMER ID	INCOME	BRANCH ID
0	99808	$93011.26	4999
1	28273	$91975.45	2508
2	31971	$88840.39	2946
3	48478	$86338.92	4131
4	10271	$85718.19	4999
5	27260	$7856.16	2847
6	22858	$76483.65	2508
7	47870	$69730.82	2946
8	24816	$67659.02	4826
9	47555	$67461.52	4999

4. This query retrieves the BRANCH ID and the count of CUSTOMER ID for each branch from the CUSTOMER table. It groups the data by BRANCH ID and calculates the count of CUSTOMER ID for each branch.

In [8]:
```
df = pd.read_sql_query("SELECT `BRANCH ID`, COUNT(`CUSTOMER ID`) AS NUMBER_OF_CUSTOMERS FROM CUSTOMER GROUP BY `BRANCH ID`;", cor
df.head(10)
```

```
C:\Users\pramo\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engin
e/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLA
lchemy
  warnings.warn(
```

Out[8]:

	BRANCH ID	NUMBER_OF_CUSTOMERS
0	2808	10
1	2946	10
2	2508	10
3	3021	10
4	4131	10
5	2847	10
6	4720	10
7	1252	10
8	4826	10
9	4999	10

5. This query retrieves all columns and records from the TRANSACTION table where the status column is equal to 'success'.

In [9]:
```
df = pd.read_sql_query("SELECT * FROM TRANSACTION WHERE status = 'success';", connection)
df.head(10)
```

```
C:\Users\pramo\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engin
e/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLA
lchemy
  warnings.warn(
```

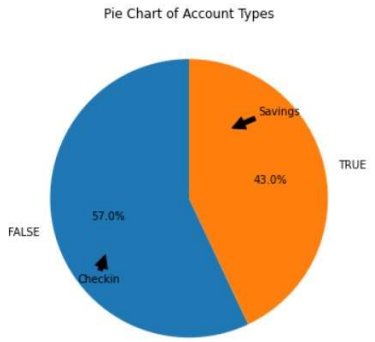
Out[9]:

	TRANSACTION ID	TRANSACTION AMOUNT	TRANSACTION TIME	STATUS	CUSTOMER ID
0	8341882	$42001.28	19:15:49	SUCCESS	57849
1	8299034	$28211.43	21:06:46	SUCCESS	37152
2	9968591	$38568.93	22:29:00	SUCCESS	82764
3	3247352	$56198.97	20:43:55	SUCCESS	63798
4	4507010	$2991.17	14:25:15	SUCCESS	12157
5	1796926	$35433.24	16:45:02	SUCCESS	64932
6	4426801	$56370.28	00:11:14	SUCCESS	74289
7	4915518	$22432.00	20:33:22	SUCCESS	97280
8	6304158	$79451.55	05:08:00	SUCCESS	66641
9	5219665	$90261.02	00:36:12	SUCCESS	31971

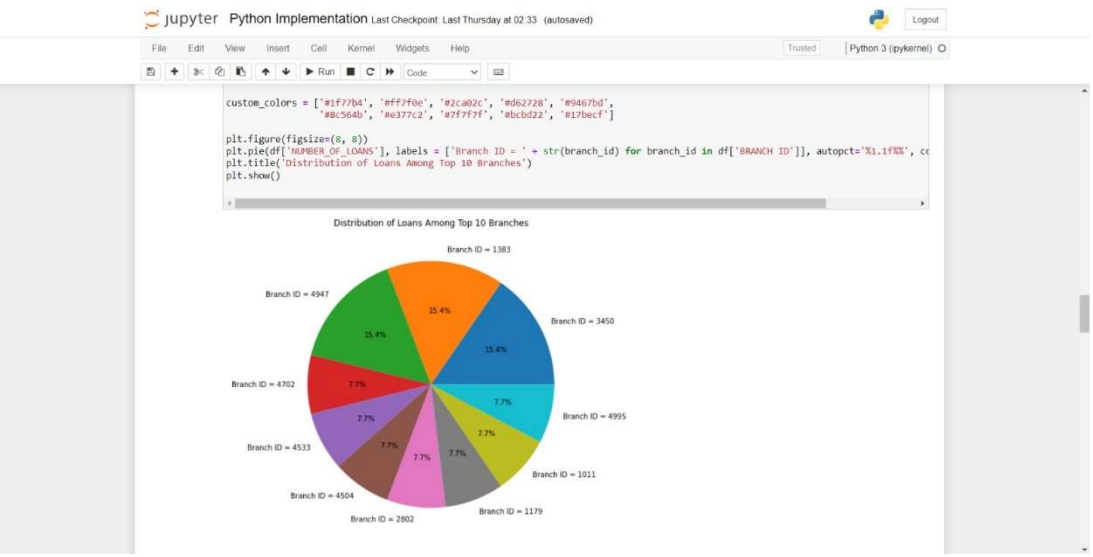
6. Pie Chart of Account Types:

Pie chart displaying the distribution of account types with percentage labels.

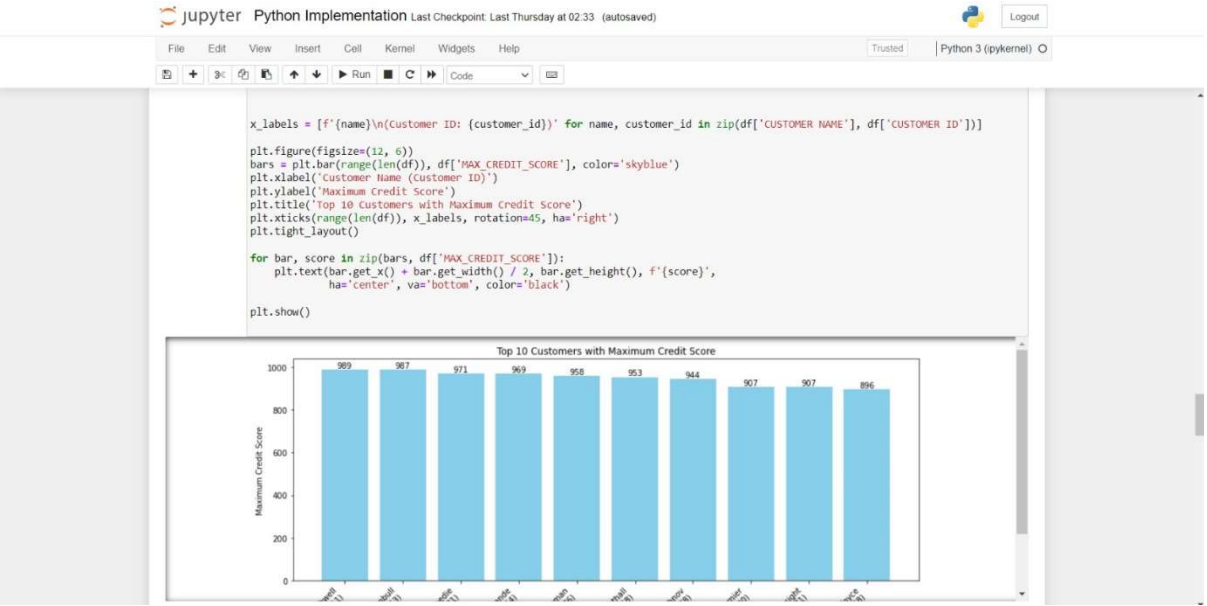
```
In [14]: # Pie chart for Account Types
         plt.figure(figsize=(8,6))
         account_types = account_df['ACCOUNT TYPE'].value_counts()
         account_types.plot.pie(autopct='%1.1f%%', startangle=90)
         plt.ylabel('')
         plt.title('Pie Chart of Account Types')

         plt.annotate('Savings', xy=(0.3, 0.5), xytext=(0.5, 0.6),
                      arrowprops=dict(facecolor='black', shrink=0.05))
         plt.annotate('Checkin', xy=(-0.6, -0.4), xytext=(-0.8, -0.6),
                      arrowprops=dict(facecolor='black', shrink=0.05))

         plt.show()
```



7. The SQL query retrieves the number of loans (NUMBER_OF_LOANS) for each branch (BRANCH ID) from the LOAN table. It counts the number of loans for each branch, groups the data by BRANCH ID, orders the result by NUMBER_OF_LOANS in descending order, and limits the output to the top 10 branches.

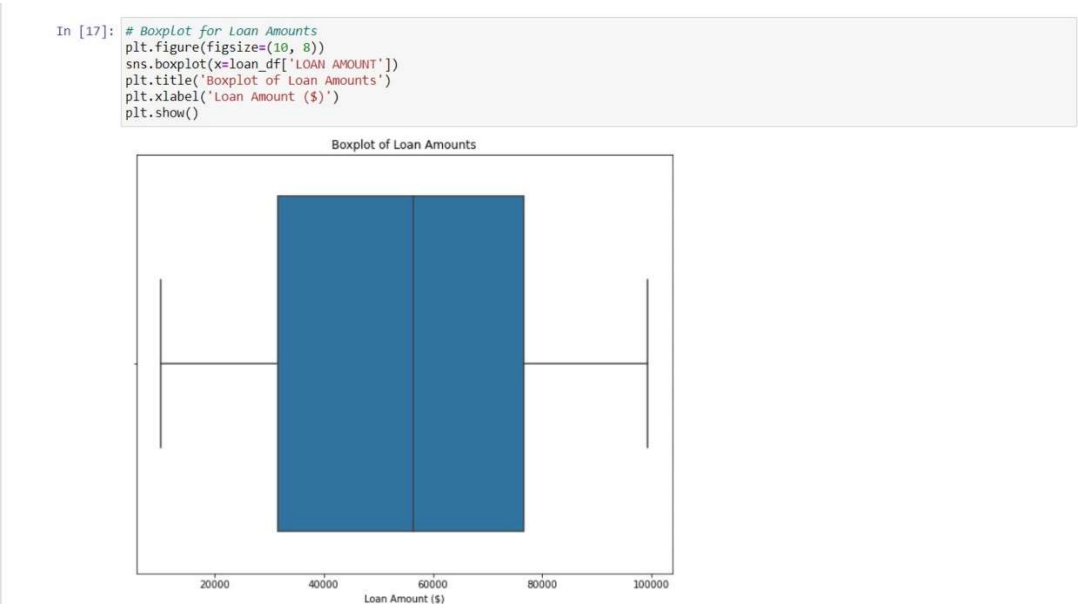


8. This SQL query retrieves the CUSTOMER NAME from the CUSTOMER table, the CUSTOMER ID and the maximum credit score (MAX_CREDIT_SCORE) from the CREDITCARD table. It joins the CREDITCARD and CUSTOMER tables on the CUSTOMER ID column. The data is then grouped by CUSTOMER ID and CUSTOMER NAME, and the maximum credit score for each customer is calculated. The result is sorted by MAX_CREDIT_SCORE in descending order, and only the top 10 records are returned.



9. Boxplot of Loan Amounts:

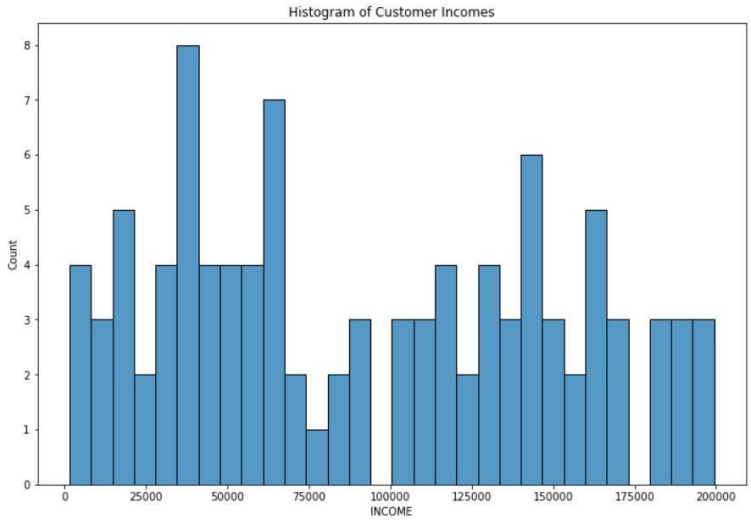
Boxplot illustrating the distribution of loan amounts with quartile information.



10. Histogram of Customer Incomes:

Histogram showing the distribution of customer incomes.

```
In [15]: # Histogram for Customer Incomes
         plt.figure(figsize=(12, 8))
         sns.histplot(customer_df['INCOME'], bins=30, kde=False)
         plt.title('Histogram of Customer Incomes')
         plt.show()
```



CONCLUSION

In conclusion, our project aimed to address significant challenges within the banking sector by developing an integrated Bank Management and Credit Card Rewarding System. Through the implementation of standardized processes and leveraging data-driven insights, our objective was to streamline operations while enhancing customer satisfaction.

The system's fundamental components, including Accounts, Transactions, Credit Cards, Loans, Employees, and Customers, were meticulously integrated to ensure seamless operation and personalized customer experiences.

A notable feature we implemented is the Credit Card Rewarding System, designed to incentivize customer engagement through tailored rewards and benefits. By analyzing customer interactions and transaction patterns, we sought to foster stronger bonds between customers and the bank.

Furthermore, our system's adaptability allows for the incorporation of various functionalities, such as managing customer-branch relationships and tracking credit card-merchant interactions, ensuring versatility to meet diverse banking needs.

In summary, our project represents a significant advancement in banking efficiency and customer satisfaction. We believe it sets a new standard for excellence and innovation within the industry, demonstrating our commitment to delivering superior banking experiences.