CS331 - HO1

Project Group 4

sds@njit.edu

kfv@njit.edu

CS-631 BANK

Kyle Villamayor

Syed Shahid

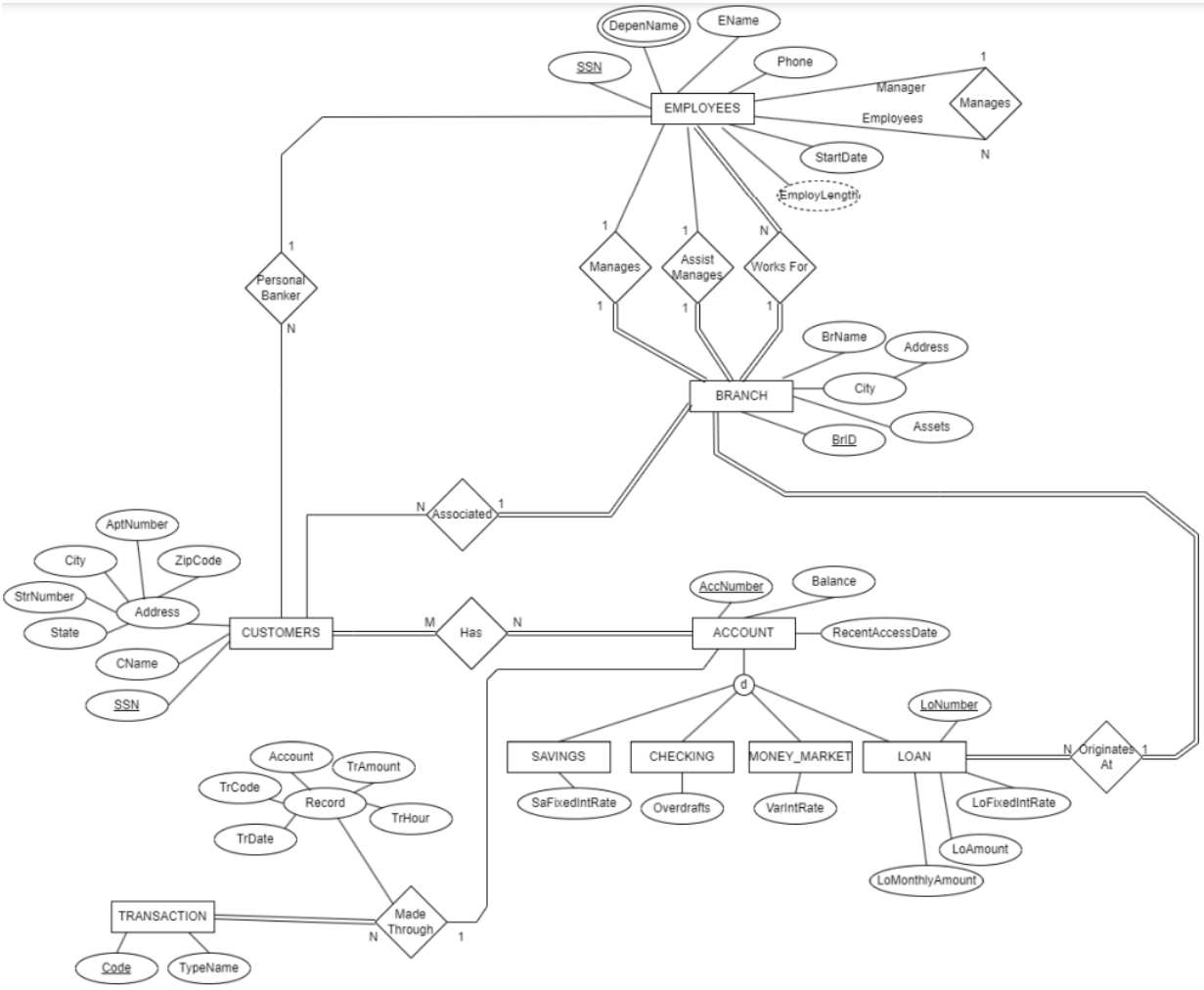
Overview / Business Requirements

The bank's database must capture essential information about branches, including unique branch IDs, names, addresses, and assets. Employees and customers are identified by social security numbers, with employees having names, phone numbers, start dates, and dependents. Customers are associated with branches and assigned personal bankers for assistance with loans and transactions. Various account types, such as savings, checking, money market, and loans, require tracking of unique account numbers, balances, and access dates. Loans originate at specific branches and involve unique loan numbers, amounts, and monthly repayment details. The database should also record transactions with unique codes, types, dates, hours, amounts, and associated accounts, distinguishing between chargeable and non-chargeable transactions. Additionally assumptions about the cardinality and participation were made about several points. For example: It is assumed that customers aren't required to take a loan, and that a loan doesn't have to be given if a customer has a bad credit history for example, or it is assumed that each manager and assistant manager is given only one branch to manage.

Webpage

CS-361 Bank and Co.

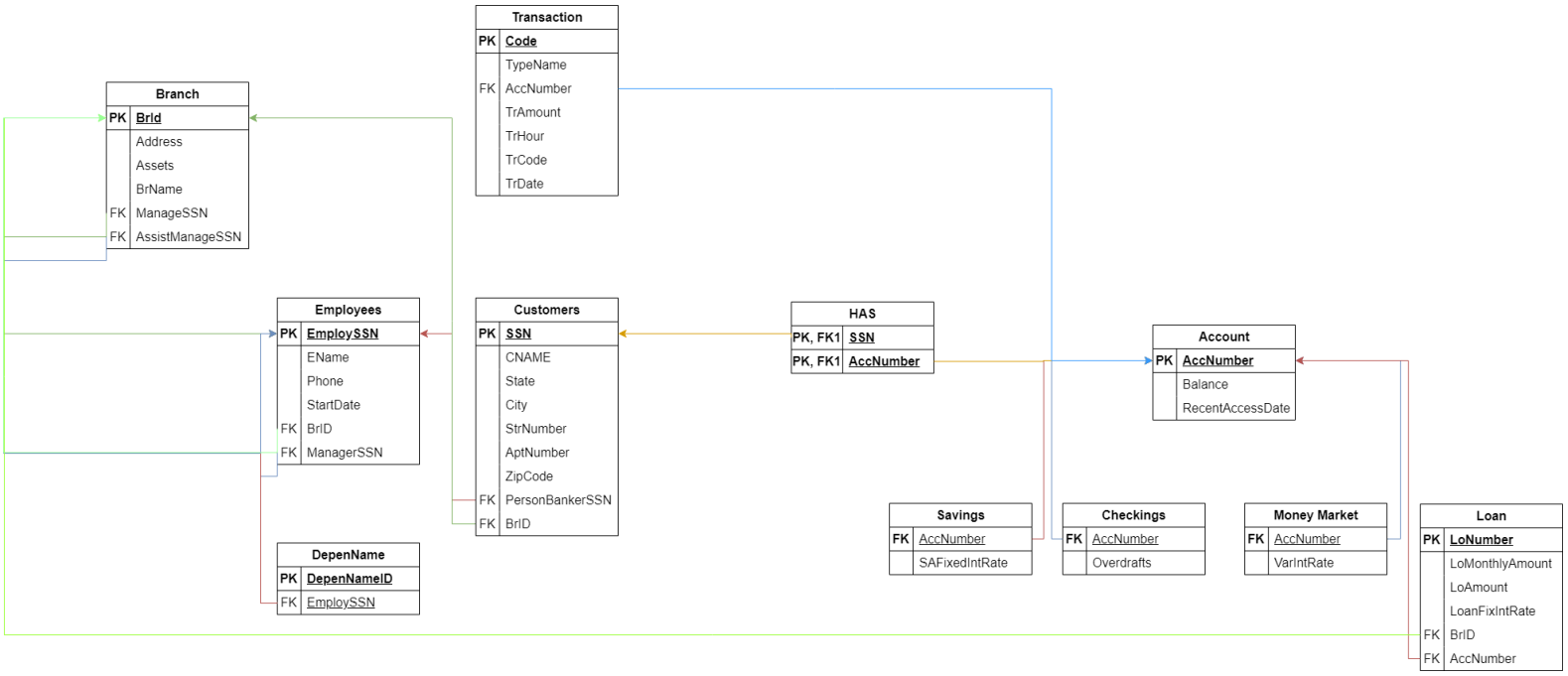
ERD Diagram



Design Rationale

- Rationale was originally denoted by assumptions made on two different aspects of this DBMS, assumptions made regarding the entities, and assumptions made regarding the relationships. The entity assumptions were easier to confirm or deny and thus left the specifications for certain relationships as the main aspect. The rationale was to find the exact participation, cardinality, and then finally which entities are involved in each relation. Only after finding these specifications did we want to move on to the entities since the initial prompt's wording can lead to numerous working and normalized, yet slightly different models.

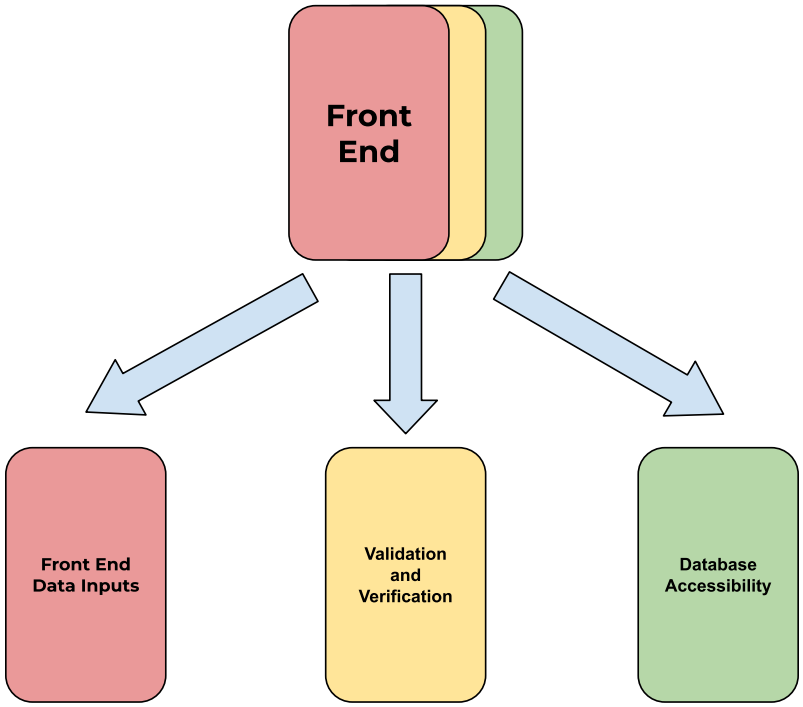
Relational Diagram



Design Rationale

- The rationale for designing our relational model was much simpler since we had our ERD diagram to build off of. Since the participation and cardinality were not denotable in the relational diagram, the only point of contention we had were which table should the transaction relational attributes be placed onto, which we decided should be the ‘Transaction’ table since the values placed there, made the most semantic sense to be there. Besides that the methodology setting up the ‘HAS’ table took a bit to understand.

Application Program Design Diagram



Design Rationale

- The idea was to set up a simple website interface to model how a large scale web interface would work. Having predefined sections for user input and then validating so that it can be easily matched with their associated database values, then verifying if the inputted information results in an employee account or user account. The idea was to set up three different pages, one log-in for both users and employee, a user account table showing the user's active funds and deposits in CS-361 Bank, and then an "admin" or employee managerial verification check to bring us to a page that can test the 4 queries against their associated English. Only managerial accounts would be allowed to run these 4 test queries to show that we must not only verify the inputted information is for an employee, but for an employee who is a manager, further showing the strength of our database system.

Primary Key, Foreign Keys, and Attributes

- Branch(**BranchId**, BranchName, BranchAddress, Assets, BranchMgrSSN, AssistMgrSSN)
    - Primary Key: BranchId
- Employees(**EmpSSN**, EmpName, Phone, StartDate, BranchId, EmpMgrSSN)
    - Primary Key: EmpSSN
    - Foreign Keys:
        - EmpBranchFK (BranchId references Branch(BranchId))
        - EmpEmpMgrFK (EmpMgrSSN references Employees(EmpSSN))
- Dependents(**DepenName, EmpSSN**)
    - Primary Key: (DepenName, EmpSSN)
    - Foreign Key: DependentEmpMultivalueFK (EmpSSN references Employees(EmpSSN))
- Customers(**CustSSN**, CustName, CustState, CustCity, StNumber, AptNumber, Zipcode, PersBankerSSN, BranchId)
    - Primary Key: CustSSN
    - Foreign Keys:
        - CustEmpBankerFK (PersBankerSSN references Employees(EmpSSN))
        - CustBranchFK (BranchId references Branch(BranchId))
- Accounts(**AccNumber**, Balance, RecAccessDate)
    - Primary Key: AccNumber
- SavingsAcc(**AccNumber**, SAFixedIntRate)
    - Primary Key: AccNumber
    - Foreign Key: SavingsAccSub (AccNumber references Accounts(AccNumber))
- CheckingsAcc(**AccNumber**, Overdrafts)
    - Primary Key: AccNumber
    - Foreign Key: CheckingsAccSub (AccNumber references Accounts(AccNumber))
- MoneyMarketAcc(**AccNumber**, VarIntRate)
    - Primary Key: AccNumber
    - Foreign Key: MoneyMarketAccSub (AccNumber references Accounts(AccNumber))
- Loan(**LoanNumber**, LoanMonthlyAm, LoanAmount, LoanFixIntRate, BranchId, AccNumber)
    - Primary Key: LoanNumber
    - Foreign Keys:
        - LoanAccSub (AccNumber references Accounts(AccNumber))
        - LoanBranchFK (BranchId references Branch(BranchId))
- HAS(**CustSSN**, AccNumber)
    - Primary Key: (CustSSN, AccNumber)
    - Foreign Keys:
        - Cust_HAS (CustSSN references Customers(CustSSN))
        - HAS_Account (AccNumber references Accounts(AccNumber))
- Transaction(**Code**, TypeName, TransCode, TransAmount, TransDate, TransHour, AccNumber)
    - Primary Key: Code
    - Foreign Key: TransactionAccountFK (AccNumber references Accounts(AccNumber))

Keys and Functional Dependencies

- Branch Table:
- Functional Dependencies:
    - BranchId -> BranchName, BranchAddress, Assets, BranchMgrSSN, AssistMgrSSN
    - BranchMgrSSN -> EmpName, Phone, StartDate (via Employees table)
    - AssistMgrSSN -> EmpName, Phone, StartDate (via Employees table)
- Employees Table:
- Functional Dependencies:
    - EmpSSN -> EmpName, Phone, StartDate, BranchId, EmpMgrSSN
    - BranchId -> BranchName, BranchAddress, Assets
- Dependents Table:
- Functional Dependencies:
    - EmpSSN -> EmpName, Phone, StartDate, BranchId, EmpMgrSSN
    - (DepenName, EmpSSN) -> DepenName
- Customers Table:
- Functional Dependencies:
    - CustSSN -> CustName, CustState, CustCity, StNumber, AptNumber, Zipcode, PersBankerSSN, BranchId
    - PersBankerSSN -> EmpName, Phone, StartDate (via Employees table)
    - BranchId -> BranchName, BranchAddress, Assets
- Accounts Table:
- Functional Dependencies:
    - AccNumber -> Balance, RecAccessDate
    - SavingsAcc, CheckingsAcc, MoneyMarketAcc Tables:
- Accounts Sub-Tables (Excluding Loan Account)
- Functional Dependencies:
    - AccNumber -> Balance, RecAccessDate
    - AccNumber -> SAFixedIntRate (SavAcc), Overdrafts (CheckAcc), VarIntRate (MMAcc)
- Loan Table:
- Functional Dependencies:
    - LoanNumber -> LoanMonthlyAm, LoanAmount, LoanFixIntRate, BranchId, AccNumber
    - BranchId -> BranchName, BranchAddress, Assets
    - AccNumber -> Balance, RecAccessDate
- HAS Table:
- Functional Dependencies:
    - CustSSN -> CustName, CustState, CustCity, StNumber, AptNumber, Zipcode, PersBankerSSN, BranchId
    - PersBankerSSN -> EmpName, Phone, StartDate (via Employees table)
    - BranchId -> BranchName, BranchAddress, Assets
    - AccNumber -> Balance, RecAccessDate
- Transaction Table:
- Functional Dependencies:
    - Code -> TypeName, TransCode, TransAmount, TransDate, TransHour, AccNumber
    - AccNumber -> Balance, RecAccessDate

Overview of 3NF Status

The functional dependencies are well-defined, resulting in a database design in Third Normal Form. Each table has a clear primary key, and non-prime attributes are dependent only on the primary key. Therefore, there are no transitive dependencies for non-prime attributes.

Final Comments

This project overall was an interactive experience into designing databases that can add or subtract entities without any produced anomalies or contradictions in the rest of the database system. The SQL statements to actually create the database system in a digital environment definitely took the majority of the time for this project, but provided thoughtful insight into how exactly the database models are implemented digitally. If this project had to be redone, we would have definitely looked into how current online banking systems or even just large companies have designed their DBMS to handle the large swaths of data entering the aforementioned systems before designing our original diagram.

Sample Relational Data

```
SELECT *
FROM Accounts;
/
```

Script Output | Query Result | SQL | All Rows Fetched: 20 in 0.043 sec

	ACCNUMBER	BALANCE	RECACCESSDATE
1	1	1205.16	2023-06-01
2	2	2300.45	2023-06-10
3	3	1500.78	2023-06-20
4	4	3200.9	2023-07-05
5	5	1800.25	2023-07-15
6	6	2700.6	2023-07-25
7	7	2100.95	2023-08-08
8	8	4200.3	2023-08-18
9	9	3100.65	2023-08-28
10	10	2800.4	2023-09-05
11	11	1900.75	2023-09-12
12	12	3500.1	2023-09-20
13	13	2600.35	2023-10-02
14	14	1800.7	2023-10-10
15	15	3300.05	2023-10-18
16	16	2400.3	2023-06-02
17	17	1600.65	2023-06-12
18	18	2900.9	2023-06-22
19	19	2000.25	2023-07-06
20	20	3700.5	2023-07-16

```
SELECT *
FROM has;
/
```

ript Output | Query Result | SQL | All Rows Fetc

	CUSTSSN	ACCNUMBER
1	307531658	1
2	307531658	13
3	307531658	14
4	456789012	2
5	789012345	3
6	789012345	16
7	789012345	17
8	789012345	18
9	123321456	18
10	123456789	4
11	987654321	5
12	111223344	5
13	444555666	5
14	222333444	6
15	222333444	15
16	666777888	15
17	666777888	7
18	123321456	8
19	123321456	19
20	333555777	19
21	567432789	9
22	287654321	10
23	555777888	10
24	555777888	11
25	111333555	11
26	111333555	20
27	333555777	12

```
SELECT *
FROM branch;
/
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.063 seconds

	BRANCHID	BRANCHNAME	BRANCHADDRESS	ASSETS	BRANCHMGRSSN	ASSISTMGRSSN
1	5678	BranchOne	B1Add	B1Assets	111111111	101010101
2	4321	BranchTwo	B2Add	B2Assets	222222222	202020202
3	8765	BranchThree	B3Add	B3Assets	333333333	303030303
4	6014	BranchFour	B4Add	B4Assets	444444444	404040404
5	3474	BranchFive	B5Add	B5Assets	555555555	505050505

```
SELECT *
FROM customers;
/
```

Script Output | Query Result

SQL | All Rows Fetched: 15 in 0.023 seconds

	CUSTSSN	CUSTNAME	CUSTSTATE	CUSTCITY	STNUMBER	APTNUMBER	ZIPCODE	PERSBANKERSSN	BRANCHID
1	307531658	Cust1	NJ	Bayonne	34	2	7002	456789012	5678
2	456789012	Cust2	NY	New York	45	(null)	10001	101010101	5678
3	789012345	Cust3	CA	Los Angeles	123	5	90001	101010101	5678
4	123456789	Cust4	TX	Houston	789	(null)	77002	789012345	4321
5	987654321	Cust5	FL	Miami	876	8	33101	789012345	4321
6	111223344	Cust6	IL	Chicago	567	(null)	60601	890123456	4321
7	444555666	Cust7	CO	Denver	789	3	80202	(null)	8765
8	222333444	Cust8	OR	Portland	987	(null)	97201	(null)	8765
9	666777888	Cust9	PA	Philadelphia	876	15	19102	(null)	8765
10	123321456	Cust10	MI	Detroit	234	7	48201	(null)	6014
11	567432789	Cust11	MN	Minneapolis	876	(null)	55401	667788990	6014
12	287654321	Cust12	NV	Las Vegas	543	20	89101	(null)	6014
13	555777888	Cust13	UT	Salt Lake City	123	(null)	84101	404040404	6014
14	111333555	Cust14	OH	Columbus	876	30	43201	100112233	3474
15	333555777	Cust15	WI	Milwaukee	987	(null)	53201	100112233	3474

```
SELECT *
FROM dependents;
/
```

Script Output | Query Result | SQL | All Rows Fetched

	DEPENNAME	EMPSSN
1	Dep1Emp3	345678901
2	Dep2Emp3	345678901
3	Dep3Emp3	345678901
4	Dep1Emp6	202020202
5	Dep1Emp7	789012345
6	Dep1Emp13	444444444
7	Dep1Emp19	100112233
8	Dep2Emp19	100112233

```
SELECT *
FROM checkingsacc;
/
```

Script Output | Query Result | SQL | All Rows Fetched

	ACCNUMBER	OVERDRAFTS
1	6	6.3451
2	7	8.2156
3	8	5.1023
4	9	6.21
5	10	8.875

```
SELECT *
FROM employees;
/
```

Script Output | Query Result | SQL | All Rows Fetched: 20 in 0.016 seconds

	EMPSSN	EMPNAME	PHONE	STARTDATE	BRANCHID	EMPMGRSSN
1	111111111	Emp1	5551234567	2003-04-15	5678	(null)
2	101010101	Emp2	5552345678	2003-07-22	5678	(null)
3	345678901	Emp3	5553456789	2003-11-08	5678	(null)
4	456789012	Emp4	5554567890	2004-02-14	5678	345678901
5	222222222	Emp5	5555678901	2004-05-20	4321	(null)
6	202020202	Emp6	5556789012	2004-09-30	4321	(null)
7	789012345	Emp7	5557890123	2005-01-12	4321	(null)
8	890123456	Emp8	5558901234	2005-05-28	4321	789012345
9	901234567	Emp9	5559012345	2005-09-18	4321	789012345
10	333333333	Emp10	5551122334	2006-01-22	8765	(null)
11	303030303	Emp11	5552233445	2006-05-05	8765	333333333
12	334455667	Emp12	5553344556	2006-10-11	8765	(null)
13	444444444	Emp13	5554455667	2007-01-02	6014	(null)
14	404040404	Emp14	5555566778	2007-03-19	6014	(null)
15	667788990	Emp15	5556677889	2007-07-07	6014	444444444
16	778899001	Emp16	5557788990	2007-09-14	6014	444444444
17	555555555	Emp17	5558899001	2007-11-28	3474	(null)
18	505050505	Emp18	5559900112	2007-02-03	3474	(null)
19	100112233	Emp19	5551001123	2007-04-09	3474	(null)
20	101122334	Emp20	5551011224	2007-06-26	3474	(null)

```
SELECT *
FROM moneymarketacc;
/
```

Script Output | Query Result | SQL | All Rows Fetched

	ACCNUMBER	VARINTRATE
1	11	2.55
2	12	1.95
3	13	4.8087
4	14	4.95
5	15	5.1023

```
SELECT *
FROM loan;
/
```

cript Output | Query Result

SQL | All Rows Fetched: 5 in 0.036 seconds

	LOANNUMBER	LOANMONTHLYAM	LOANAMOUNT	LOANFIXINTRATE	BRANCHID	ACCNUMBER
1	1	12540.61	5026.83	5.3578	5678	16
2	2	9500.75	3800.3	4.8023	4321	17
3	3	14230.4	5692.16	6.215	8765	18
4	4	11780.9	4712.36	5.0023	4321	19
5	5	13690.25	5476.1	5.75	6014	20

```
SELECT *
FROM Transaction;
/
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.043 seconds

	CODE	TYPENAME	TRANSCODE	TRANSAMOUNT	TRANSDATE	TRANSHOUR	ACCNUMBER
1	WD1	Withdraw	WD	145.2	2023-07-13	4	3
2	CD1	Customer Deposit	CD	200.5	2023-08-05	9	8
3	WD2	Withdraw	WD	75.8	2023-09-18	12	14
4	Z1	Zelle	Z	300.25	2023-10-02	15	19
5	CD2	Customer Deposit	CD	120.1	2023-10-15	18	5

```
SELECT *
FROM savingsacc;
/
```

Script Output | Query Result

SQL | All Rows Fetched: 5

	ACCNUMBER	SAFIXEDINTRATE
1	1	4.5068
2	2	3.7892
3	3	5.2315
4	4	4.0023
5	5	3.2567