Project Direction Overview

I would like to build an application base bank account management system database which tracks account balance and all transactions across various banks. A user can check all transaction information on this app such as the type of service, vendor info, location of purchase, transaction amount, transaction date and more. This more of hybrid version that can incorporate different banks account information together in one app. With different type of banking account categories to differentiate, this app will automatically sync with your banking apps to link up account information with the this “All in One Banking app”.

Users will login with password, sync with all banking apps to download up to date purchase information. The purchases are order with the newest purchase first. The purchase information will be recorded include the purchase amount, vendor name, location of purchase, date of purchase, type of purchase. Bank to bank transfer information will be included as well, but this will be classified as transfer type transaction instead of purchase. Purchase and transfer history will be store up and separated by years. If a transaction has been cancelled and the app sync with banking app to remove the transaction history.

For the project, I will be focusing creating the database for the “All in One Banking app”. This database will have to be secured for user to feel confident that confidential information is not leaked. I always keep track of my bank accounts from different bank with their own app, but “All in One Banking app” will save my login time for multiple apps with just a database. That is why I decided to go with this banking account management system.

Use Cases and Fields

App Account signup Use Case

1. User sign up with user ID and password for account creation.
2. User will have a choice to set up paid or free account.
3. The application will ask if the user wants to sync any existing bank app to extract information in the database.
4. The app account will store password, user ID, name, phone number, and email

|  |  |  |
| --- | --- | --- |
| Field | What it Stores | Why it’s Needed |
| User ID | This is the app account ID to differentiate with different users. | It is unique to each users. It cannot be duplicated, so it is a method to identify a user. |
| FirstName | First name of user. | To identify the user first name and communication purpose. |
| LastName | Last name of user. | To identify the user last name and communication purpose. |
| PhoneNumber | This stores user phone number. | It is for customer service and communication purpose. |
| Email | This stores user email info. | It is for customer service and communication purpose. |

Associative structural rule from use case 1

I identify the User account as my first entity. There is nothing much from the first use case where entity is concerned.

Bank lookup Use Case

specialization-generalization rule from use case 1

The use of free and paid user account will create a disjoint but complete list.

1. Banking information will be included in the database for user reference
2. Accounts type, account balance, BankID, Bank name and AccountID will be store here.

|  |  |  |
| --- | --- | --- |
| Field | What it Stores | Why it’s Needed |
| BankName | This stores the bank name. | This differentiates the bank name from others. |
| AccountType | This stores the bank specific account type. | This differentiates which account type user are looking at. (Savings or checking) |
| AccountBalance | This stores the user’s money balance. | This allows user to keep track of their money from various bank. |
| BankID | This stores the user’s BankID. | This allows user to identify banks’ ID. |
| AccountID | This stores the user’s bank AccountID. | This allows the identification of AccountType linked to ID |

Associative structural rule from use case 2

I identify three entities from use case 2:

Bank, Bank account type and transaction.

**Each user account has one or many banks; Each bank is used one or many user account.**

The reason behind this is because a user can open one or many banks. Each different bank can be used one or many user accounts

**Each bank may have one to many banks account type; Each bank account type may exist one or many bank.**

The reason behind this is because a bank may have one to many savings, checking bank account type. Each bank account type may exist one or many different banks.

**Each user account may have zero to many transactions; Each transaction is associated a user account.**

Every user account may initiate zero to many transactions. Each individual transaction must be link with a specific user account.

Vendor Info lookup Use Case

1. Vendor info of transaction to be used by user.
2. Vendor name, location, vendor email, and VendorID.

|  |  |  |
| --- | --- | --- |
| Field | What it Stores | Why it’s Needed |
| Vendor name | This stores the vendor name. | This allows user to know the name of a particular vendor. |
| Location | This stores the location of vendor. | This allows user to know where the vendor is located. |
| VendorEmail | This stores the email of vendor. | This allows user to contact vendor by email. |
| VendorID | This stores the VendorID link to vendor. | This allows user to identify the VendorID from transactions |

Associative structural rule from use case 3

I identify one entity here which is vendor.

**Each purchase has a vendor; Each vendor offers one to many purchases.**

Each purchase has a specific vendor. Every vendor will offer one to many purchases’ choices.

Wire Transfer Information Use Case

1. Wire transfer information for users
2. Bank name, transaction amount, date of transaction, BankIDOrigin, BankIDDestination and AccountID will be store here.

|  |  |  |
| --- | --- | --- |
| Field | What it Stores | Why it’s Needed |
| BankName | This stores the name of bank where the transaction came from or going to. | This tells user from the origin or destination of bank of a particular transaction. |
| TransferAmount | This stores the transfer amount. | This tells user the transfer amount. |
| TransferDate | This stores the transfer date. | This tells user the date of the transfer. |
| BankIDOrigin | This stores user’s BankID where wire transfer originated. | This links the BankID to bank where wire transfer originated |
| BankIDDestination | This stores user’s BankID where wire transfer destination. | This links the BankID to bank where wire transfer destinations. |
| AccountID | This stores user’s AccountID | This links the wire transfer to the banks AccountID which is associated with AccountType. |

**Each bank account type may have zero to many wire transfers; Each wire transfer is linked to a bank account type**

Every bank account type will have zero to many wire transfers. Individual wire transfer is linked to a bank account type.

Transaction Information Use Case

1. Transaction information for users individual banks
2. TransactionID, TransactionDescription, VendorID, TransactionDate, Price, and BankID will be store here.

|  |  |  |
| --- | --- | --- |
| Field | What it Stores | Why it’s Needed |
| TransactionID | This stores the TransactionID | This allows users to track the TransactionID. |
| TransactionDescription | This stores the TransactionDescription. | This allows users to check for the name of a particular transaction. |
| VendorID | This stores the VendorID. | This allows users to look up the vendor link to purchase. |
| TransactionDate | This stores the TransactionDate. | This allows users to check the transaction date. |
| Price | This stores the Price of transaction. | This allows users to check the price of transaction. |
| BankID | This stores the BankID | This allows users to look up the bank where purchase has taken place. |

**Each transaction may have one to many purchases; Each purchase must be a transaction.**

Every transaction may have one to many purchases. Each purchase will have to be a transaction.

Bank account type balance history use case

1. Balance change history for the bank account type of the user.
2. Balance\_history\_id, bank\_account\_id, old\_balance, new\_balance, change\_date.

|  |  |  |
| --- | --- | --- |
| Field | What it store | Why it’s Needed |
| balance\_history\_id | This stores the balance history id. | It keeps track of the balance change for the history table. |
| bank\_account\_id | This stores the bank account id. | This id links up with the bank account associated with the balance change. |
| old\_balance | This stores the old balance amount. | This records the previous balance amount before it gets updated. |
| new\_balance | This stores the new balance amount. | This records the new balance amount after it gets updated. |
| Change\_date | This store the date when the balance change occurs. | This records the date balance get updated. |

To keep track of history for my project. There are two potential data to have history:

* The product price change history.
* The bank account balance history.

For my project I will assume the product price history is not crucial. My banking app need to keep track of banking balance history so I will include the bank account balance history.

Balance change history is linked to bank account type entity.

**Each bank account has one or many balance change history; Each balance change history is associated to one bank account.**

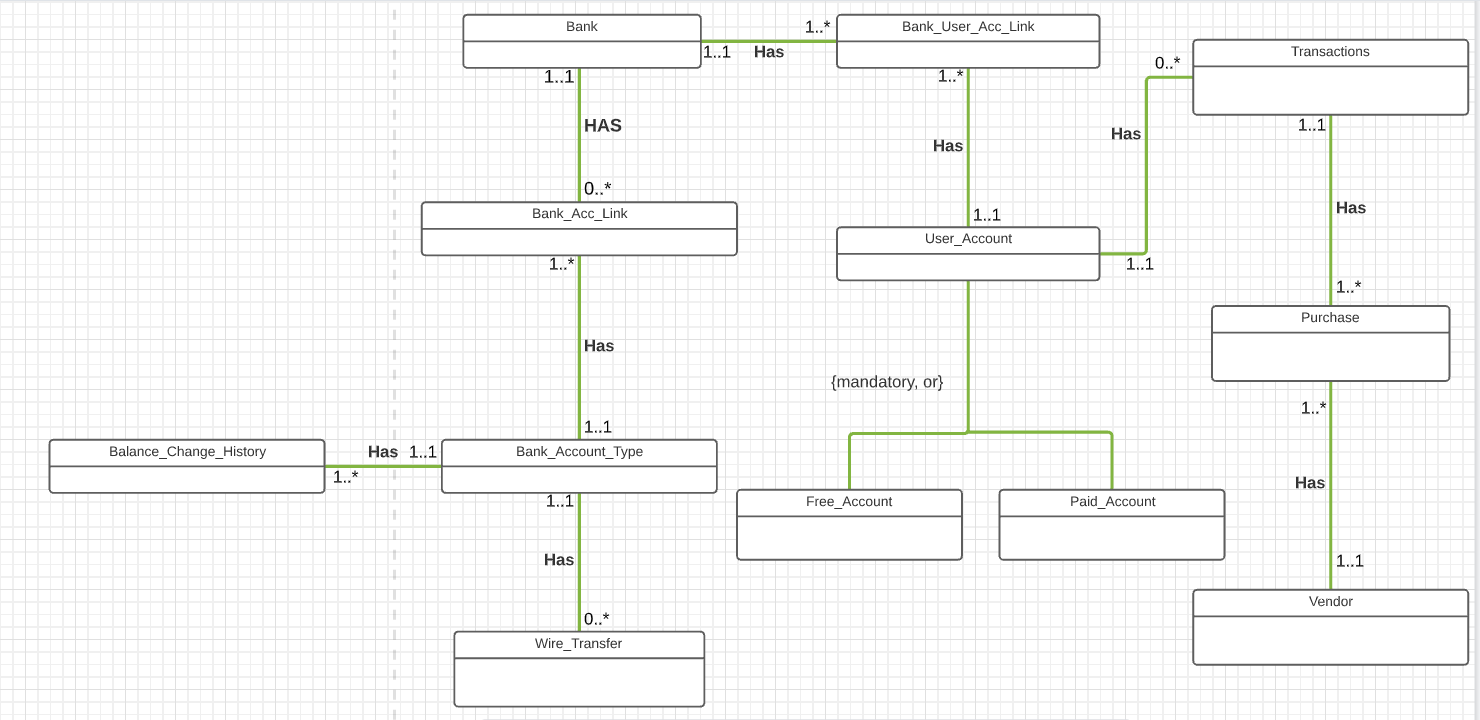
Relationship summary

All associative relationships:

1. Each user account has one or many banks; Each bank is used one or many user account.
2. Each bank may have one to many banks account type; Each bank account type may exist one or many bank.
3. Each user account may have zero to many transactions; Each transaction is associated a user account.
4. Each purchase has a vendor; Each vendor offers one to many purchases.
5. Each bank account type may have zero to many wire transfers; Each wire transfer is linked to a bank account type
6. Each transaction may have one to many purchases; Each purchase must be a transaction.
7. Each bank account has one or many balance change history; Each balance change history is associated to one bank account.

Speciliazation generalization

1. Free account and Paid account is a subtype of the user account (supertype)

 initial ERD

Mapping Associative Relationship

The associative relationship in my conceptual ERD are User account/Bank, Bank/Bank account type, Bank account type/Wire Transfer, User account/Transaction, Transaction/Purchase, and Vendor/Purchase.

Bank account type/ Wire Transfer is 1:M. Each bank account type may have zero to many wire transfers, but each wire transfer is linked to a bank account type.

User account/Transaction is 1:M. Each user can have multiple transactions, but each transaction is associated with a user.

Transaction/Purchase is 1:M. Each transaction can have multiple purchase, but each purchase is associated with a transaction.

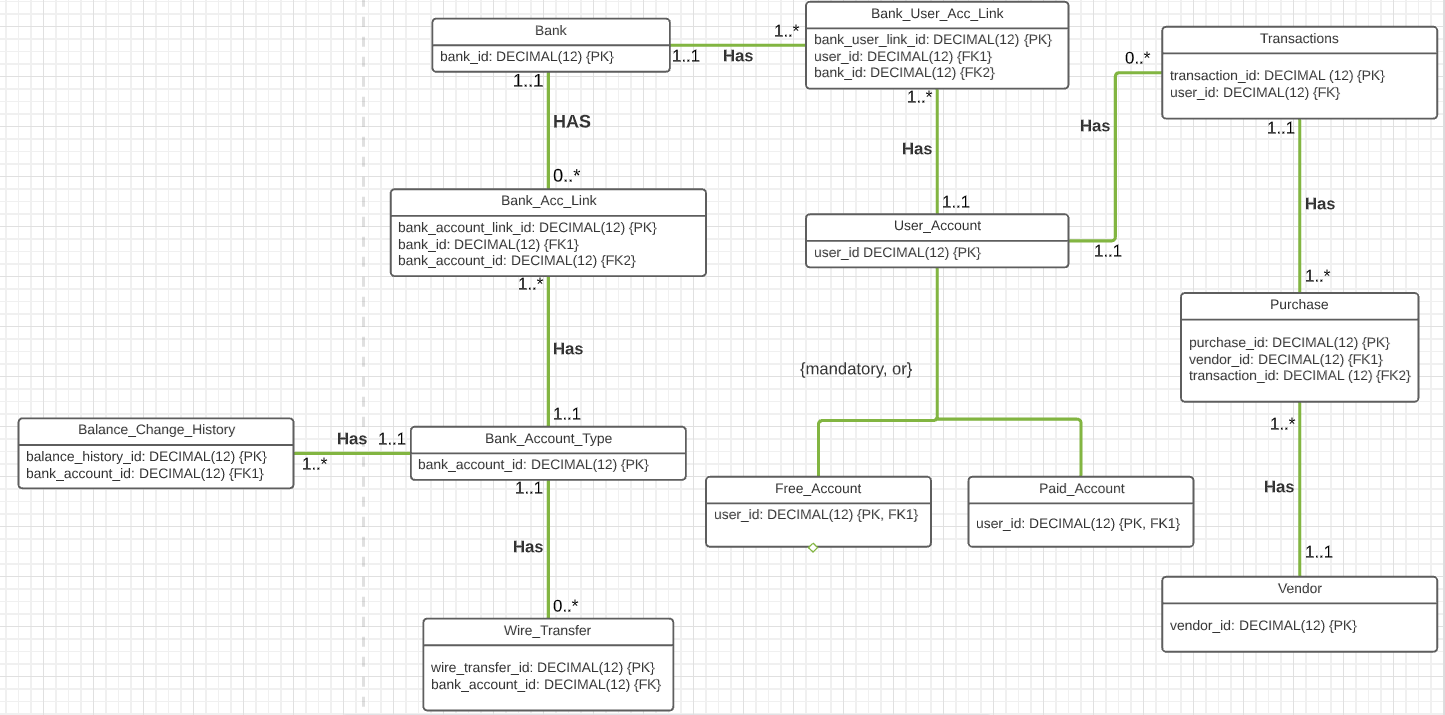
Purchase/Vendor is 1:M. Each vendor can offer multiple purchases but each purchase is associated with a vendor.

Bank/ Bank account type has many to many relationships, so it requires a bridge link of Bank/Bank account type link.

User account/Bank has many to many relationships, so it too requires a bridge link of User account/Bank link.

Mapping Specialization-generalization relationship

The only specialization-generalization supertype is user account, the subtypes are Free account and Paid account.

The full DBMS physical ERD before adding attributes

Question on decision on what attribute I need

|  |  |
| --- | --- |
| Question | Reasoning |
| What fields do other similar  applications and databases store? | Using my own US bank and chase bank gave me some ideas for attributes for this database. |
| What fields are obvious for my entities? | There are a few obvious attributes for many of the entities I am using for my database. User account info must have name. Vendor must have email, contact number. Bank must have account types. |
| What fields are unique for my database? | My database is quite generic. So far, I will be using basic attributes. |
| What would be presented on user interface that uses my database? | Since my project is focused solely on database. There will be no data present for the end user. |

Adding attributes to you DBMS physical ERD.

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Attribute | Datatype | Reasoning |
| User\_Account | user\_first\_name | VARCHAR(255) | The first name of user with character space up to 255. |
| User\_Account | user\_last\_name | VARCHAR(255) | The last name of user with character space up to 255. |
| User\_Account | user\_email | VARCHAR(255) | The user’s email address with character space up to 255. |
| User\_Account | user\_phone\_number | DECIMAL(15) | The user’s phone number with decimal space up to 15. |
| Transactions | transaction\_description | VARCHAR(1000) | The transaction description with character space up to 1000. |
| Transactions | transaction\_date | DATE | The transaction’s date with date format. |
| Purchase | product\_name | VARCHAR(255) | The purchase’s name with character space up to 255. |
| Purchase | product\_price | DECIMAL(6,2) | This purchase price with digits up to 7 numbers. |
| Vendor | vendor\_name | VARCHAR(255) | The vendor’s name with character space up to 255. |
| Vendor | vendor\_email | VARCHAR(255) | The vendor’s email with space up to 255. |
| Vendor | vendor\_phone\_number | DECIMAL(15) | The vendor’s phone number with decimal space up to 15. |
| Bank | bank\_name | VARCHAR(255) | The bank’s name with character space up to 255. |
| Bank | bank\_phone\_number | DECIMAL(15) | The bank’s phone number with decimal space up to 15. |
| Bank\_Account\_Type | account\_name | VARCHAR(255) | The bank account type name with character space up to 255. |
| Bank\_Account\_Type | balance\_amount | DECIMAL(7,2) | The bank account type balance\_amount with up to 7 digits space. |
| Wire\_Transfer | transfer\_amount | DECIMAL(7,2) | The wire transfer amount up to 7 digits space. |
| Wire\_Transfer | to\_bank\_name | VARCHAR(255) | The wire transfer destination bank name with character space up to 255 characters. |
| Wire\_Transfer | transfer\_date | DATE | The wire transfer date with date format. |
| Paid Account | paid\_indicator | CHAR(1) | This is an indicator for the paid account. Character size of 1. |
| Free\_Account | free\_indicator | CHAR(1) | This is an indicator for the free account.  Character size of 1. |

Adding balance history table

|  |  |
| --- | --- |
| Attribute | Description |
| balance\_history\_id | This is the primary key of the history table. It is a DECIMAL(12) for storing many values. |
| bank\_account\_id | This is a foreign key to Bank\_Account\_Type table to refer to the account with balance change. Datatype match id in Bank\_Account\_Type table. |
| old\_balance | This is the old balance of the bank account type. The datatype mirrors the balance amount in the Bank\_Account\_Type. |
| new\_balance | This is the new balance of the bank account type. The datatype mirrorsthe balance amount in the Bank\_Account\_Type. |
| change\_date | This is the date the balance change occurred, with a DATE datatype. |

After careful consideration, my final normalized DBSM physical ERD is shown below.

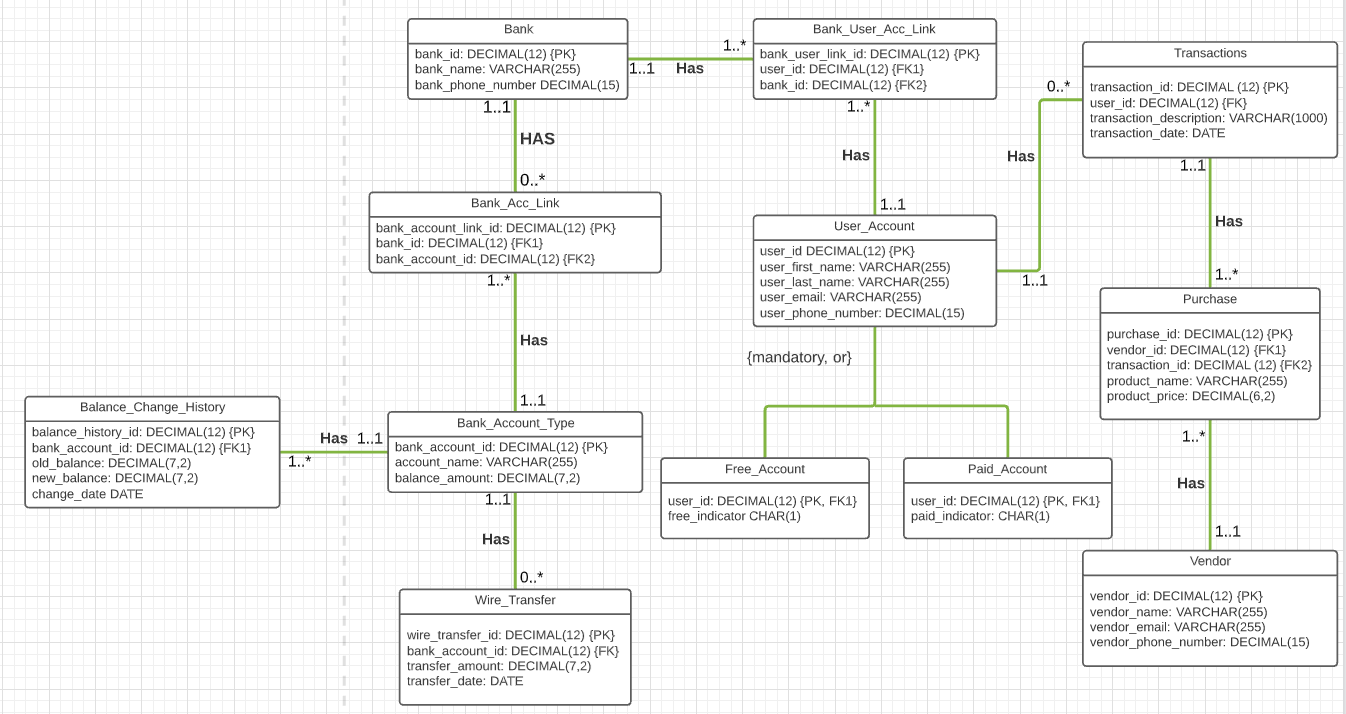
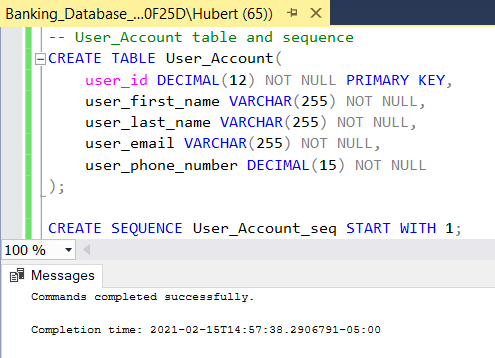
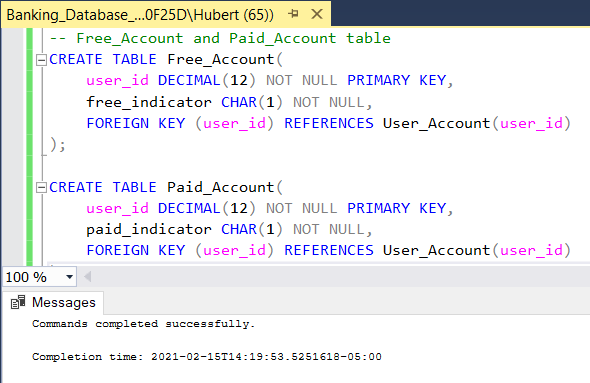
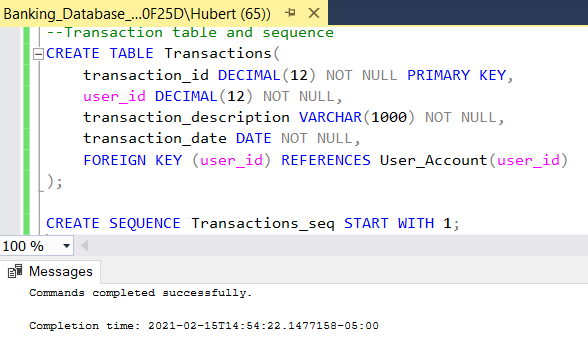
I did rename the entities for it to be valid names to be implemented into SQL server. I added two links for Bank/User account and Bank/Bank account link. Since these are these tables has M to N relationship, thus setting up links will simplify the tables linkage relationship.

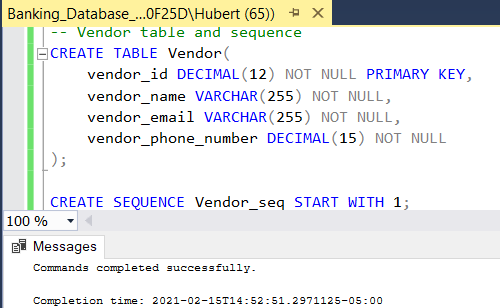
Table and sequence generation in SQL Server.

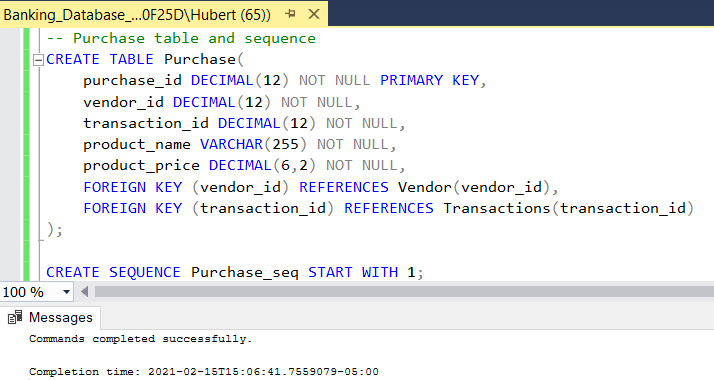
1)

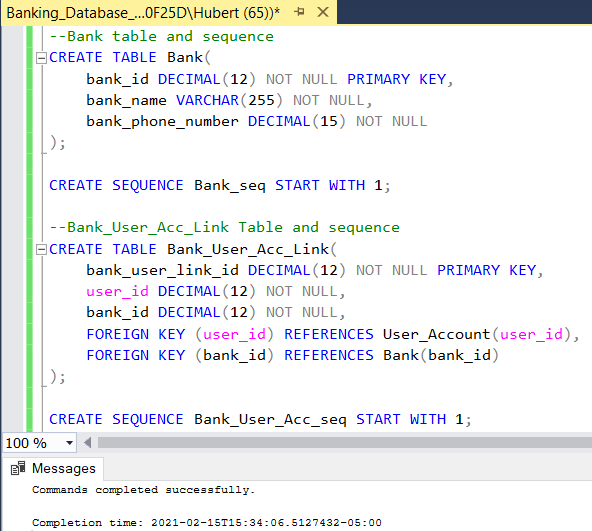
2)

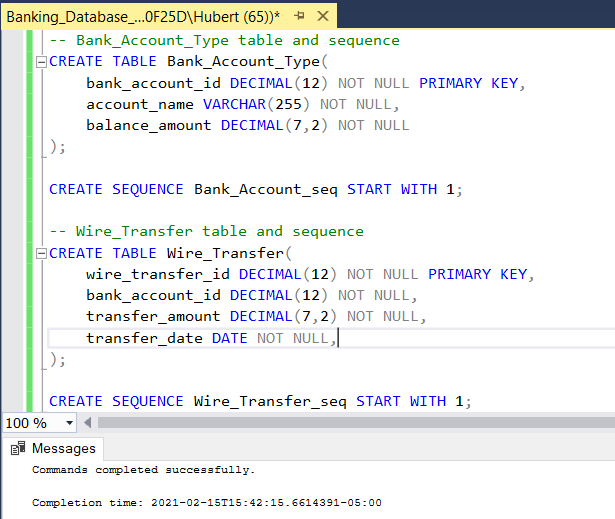
3)

4)

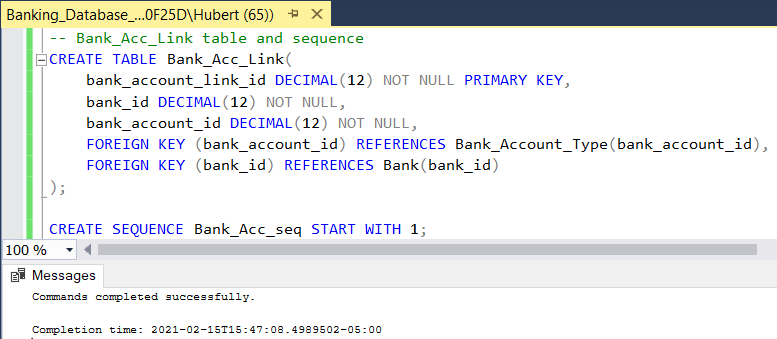


5)

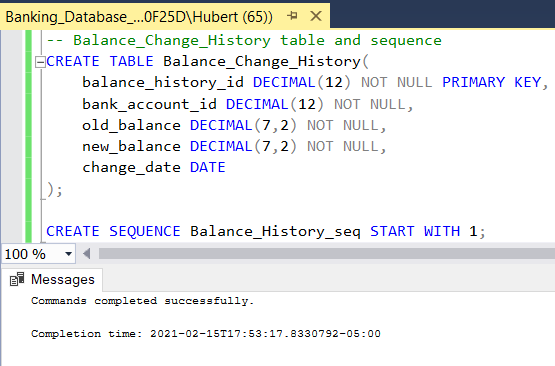
6)

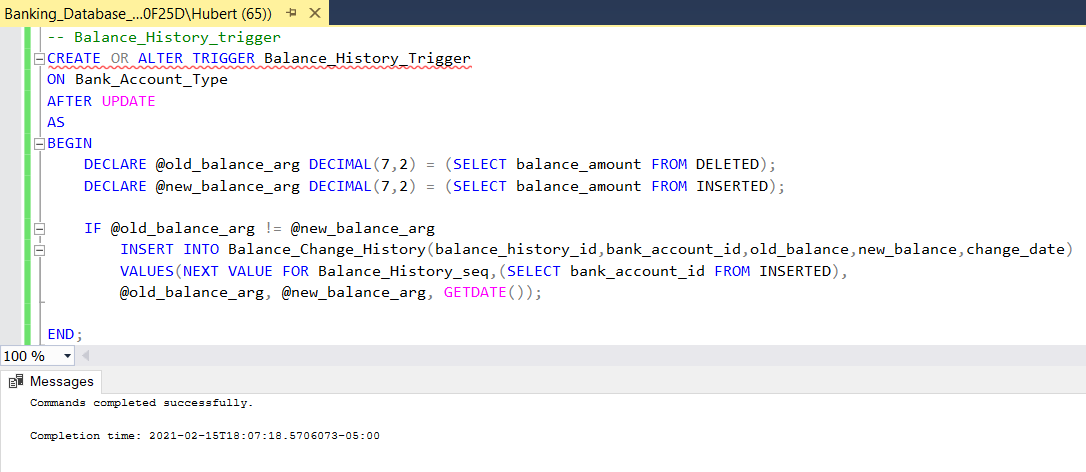
7)

8)



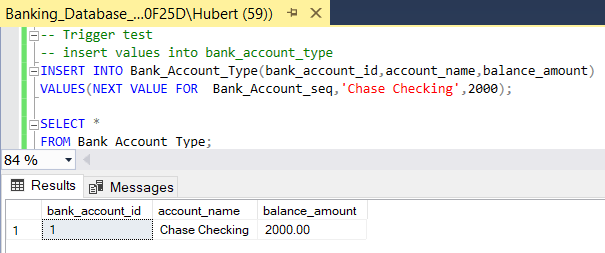
Adding the history table

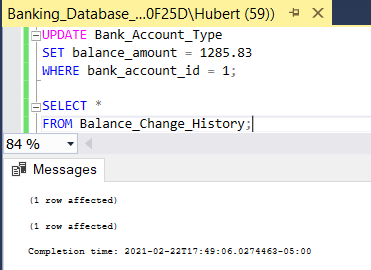
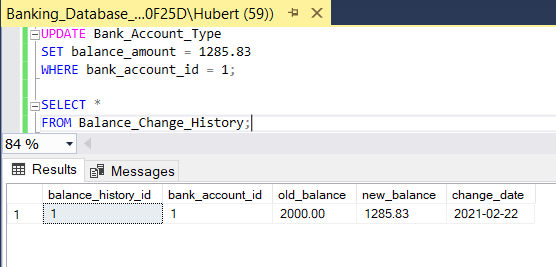


Adding trigger to add data into history table

Testing for balance history trigger

1. Insert value into Bank\_Account\_Type and displayed it.



1. Update the balance\_amount using UPDATE.
2. Display the balance history table.

For primary keys which has been indexed.

Below is the list:

* User\_Account.user\_id
* Transactions.transaction\_id
* Purchase.purchase\_id
* Vendor.vendor\_id
* Bank\_User\_Acc\_Link.bank\_user\_link\_id
* Bank.bank\_id
* Bank\_Acc\_Link.bank\_account\_link\_id
* Bank\_Account\_Type.bank\_account\_id
* Wire\_Transfer.wire\_transfer\_id

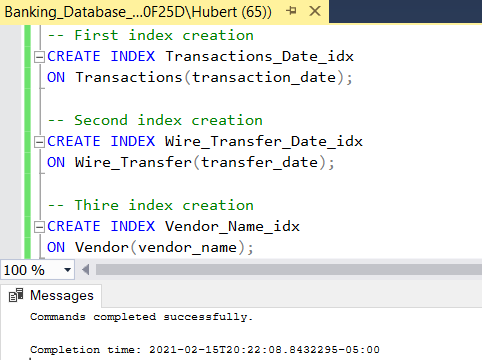
Foreign keys are explained on the next page.

|  |  |  |
| --- | --- | --- |
| Column | Unique? | Description |
| Transactions.user\_id | ot unique | The foreign key in Transactions referencing User\_Account is not unique because there can be many Transactions for the same user\_id. |
| Purchase.vendor\_id | Not unique | The foreign key in Purchase referencing Vendor is not unique because there can be many purchase from the same vendor. |
| Purchase.transaction\_id | Not unique | The foreign key in Purchase referencing Transaction is unique because there can many Purchases for the same transaction. |
| Bank\_User\_Acc\_Link.user\_id.user\_id | Not unique | The foreign key in Bank\_User\_Acc\_Link referencing User\_Account is not unique because there be many Bank\_User\_Acc\_link for the same user. |
| Bank\_User\_Acc\_Link.user\_id.bank\_id | Not unique | The foreign key in Bank\_User\_Acc\_Link referencing Bank is not unique because there be many Bank\_User\_Acc\_link for the same bank. |
| Bank\_Acc\_Link.bank\_id | Not unique | The foreign key in Bank\_Acc\_Link referencing Bank is not unique because there be many Bank\_User\_Acc\_link for the same bank. |
| Bank\_Acc\_Link.bank\_account\_id | Not unique | The foreign key in Bank\_Acc\_Link referencing Bank\_Account\_Type is not unique because there be many Bank\_User\_Acc\_link for the same bank account type. |
| Wire\_Transfer.bank\_account\_id | Not unique | The foreign key in Wire\_Transfer referencing Bank\_Account\_Type is not unique because there be many wire transfer for the same bank account type. |
| Balance\_Change\_History.bank\_account\_id | Not unique | The foreign key in Balance\_Change\_History referencing Bank\_Account\_Type is not unique because there be many balance change history for the same bank account type. |

Three query driven indexes.

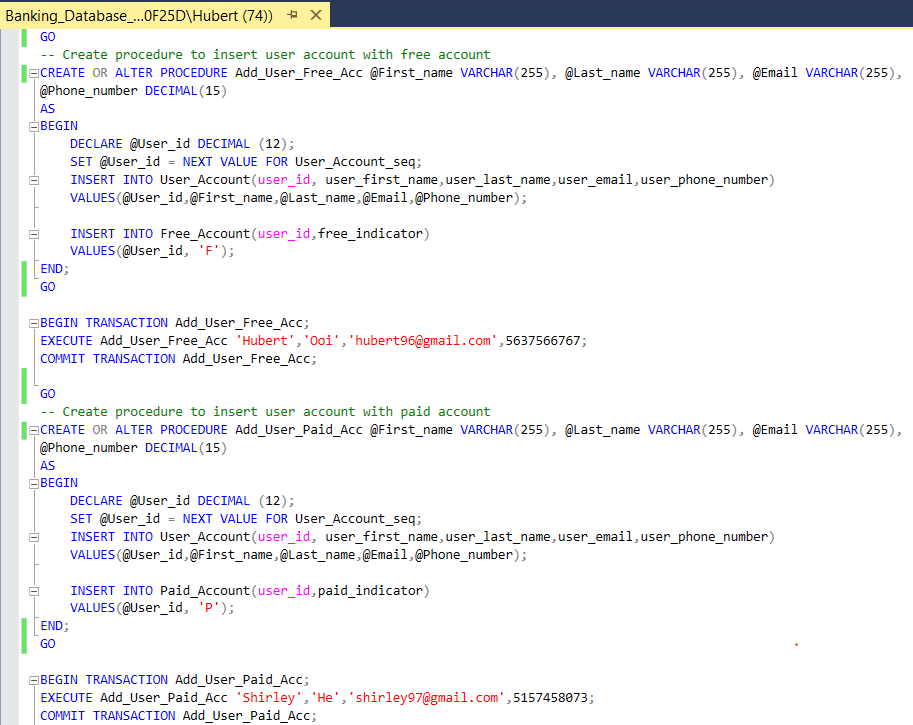
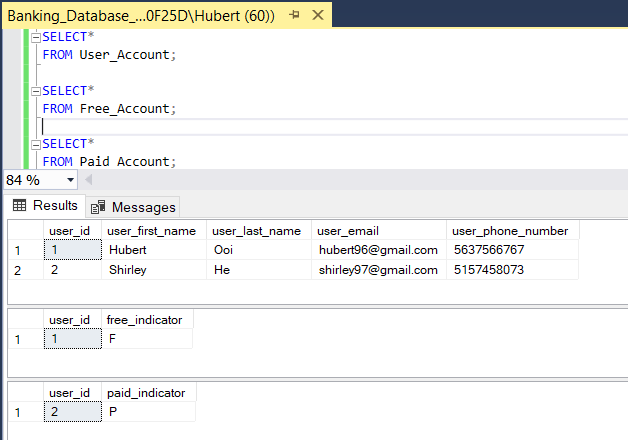
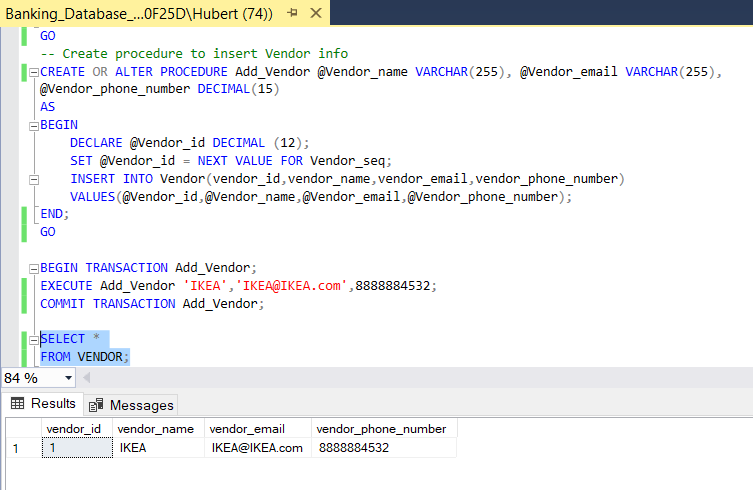
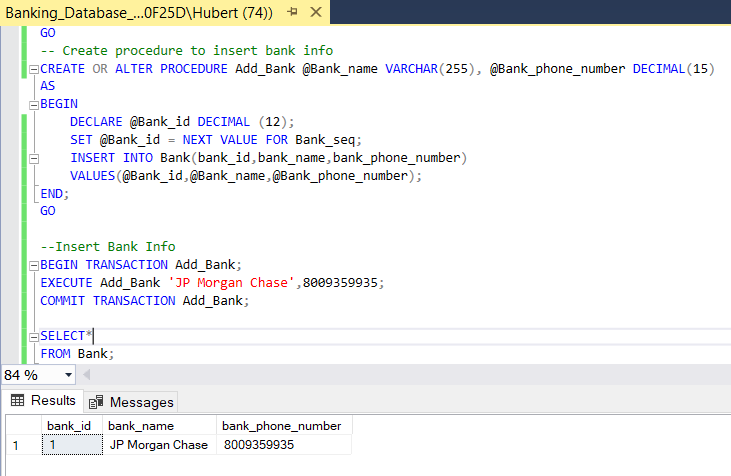
First it would be Transaction.transaction\_date. User can query limited to transaction\_date. This is not unique because different transactions can occur on the same date.

Second, it would be Wire\_Transfer.transfer\_date. User can query limited to transfer\_date. This is not unique because many wire transfer can occur on the same date.

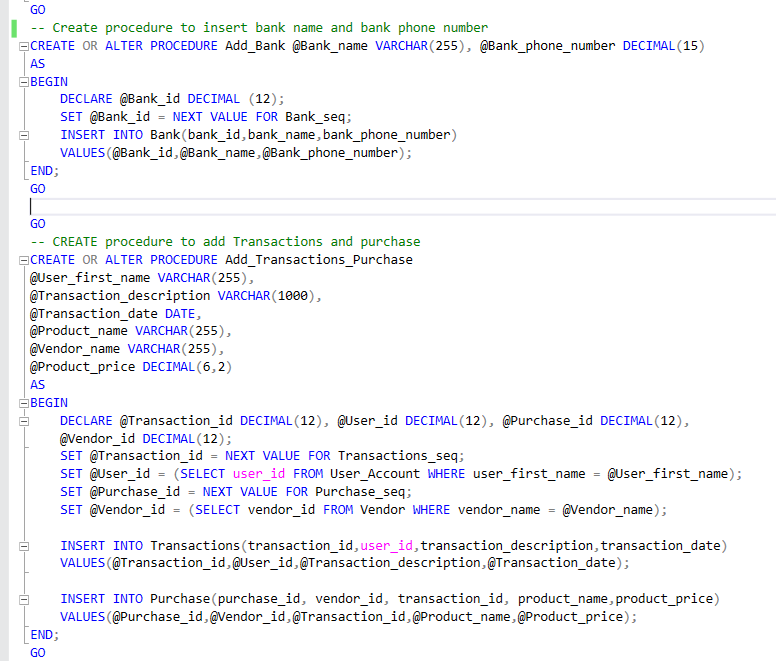
Third, it would be Vendor.vendor\_name. User can query limited to the vendor name. This is unique because each vendor will have a unique name.

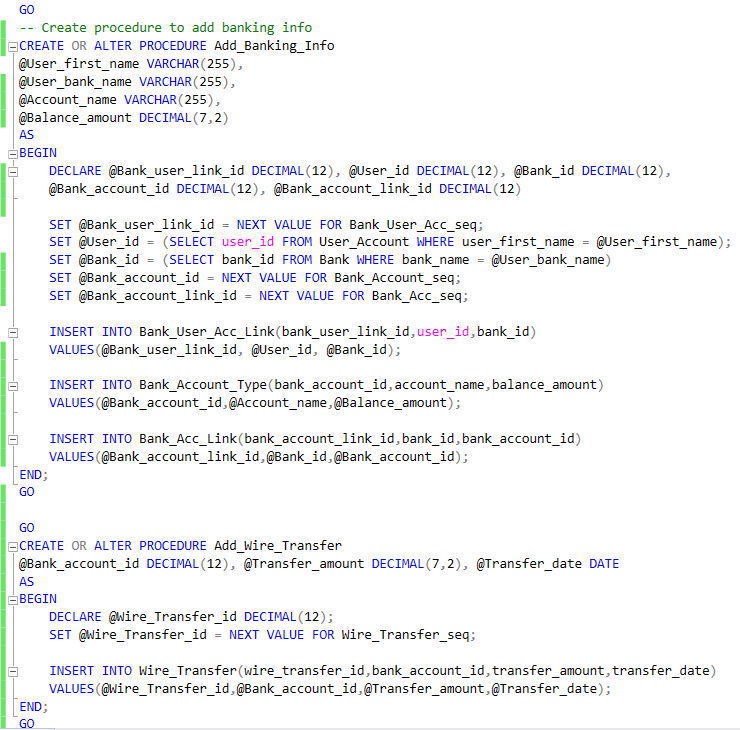
Implement store procedures and Transaction in my database

3 examples of use cases

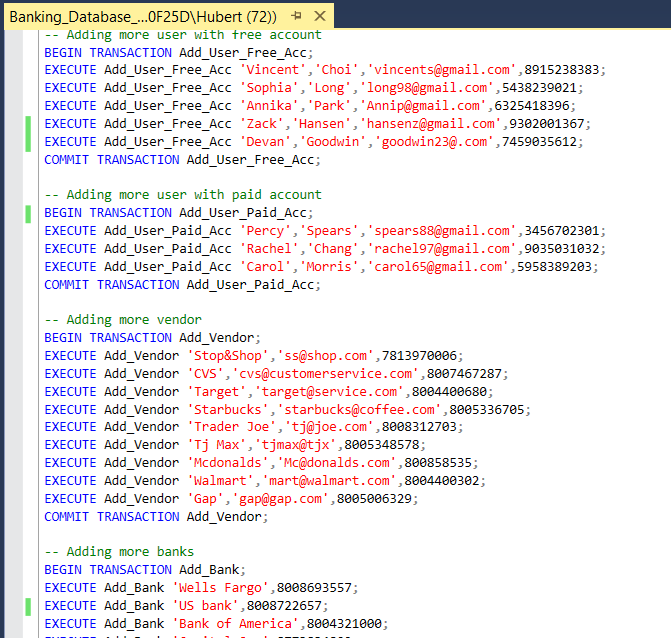
1. Store procedures and transactions for adding user account info for free and paid account
2. Store procedures and transactions for adding vendor info to Vendor table.
3. Store procedures and transactions for adding bank info to Bank table.

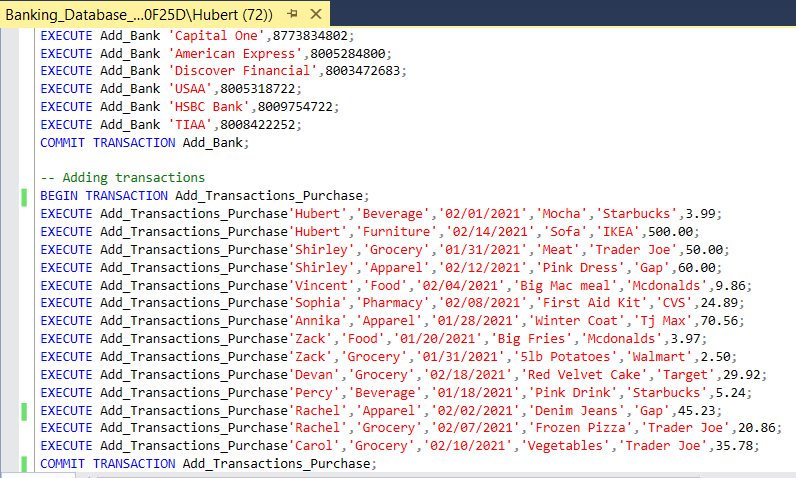
A few more store procedures to store data

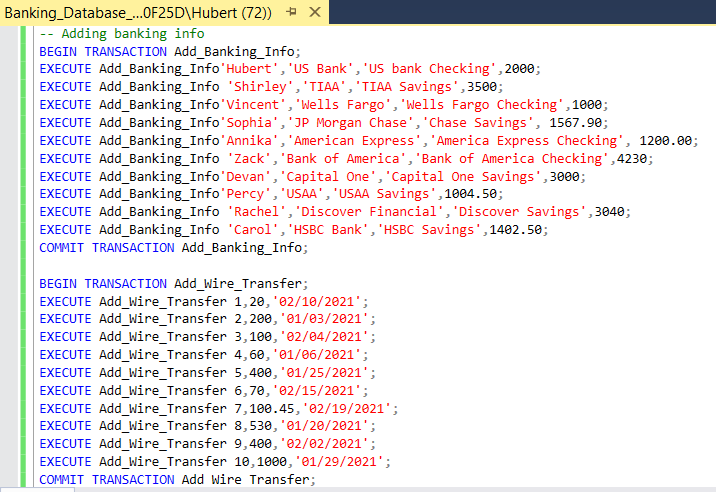


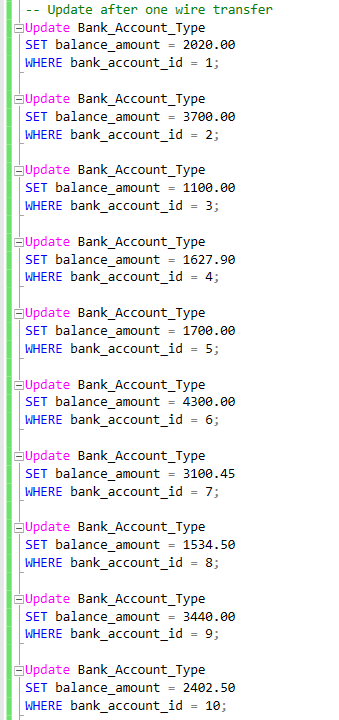


Adding data to my tables



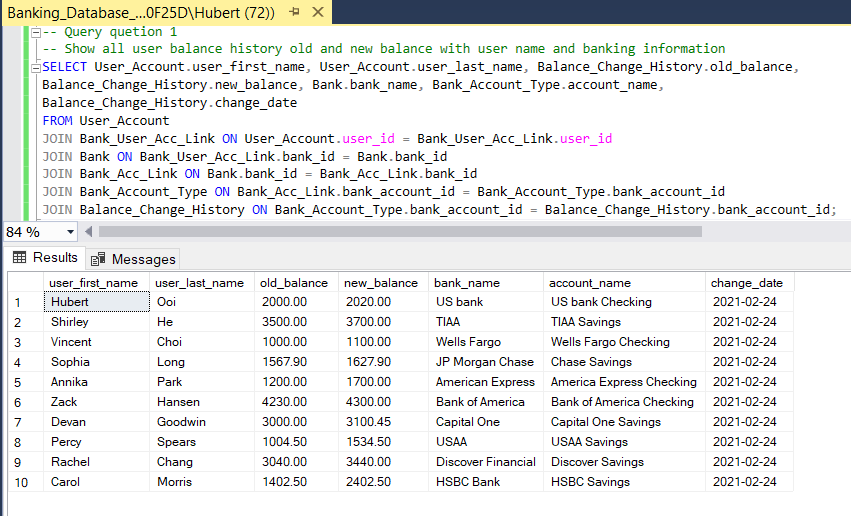




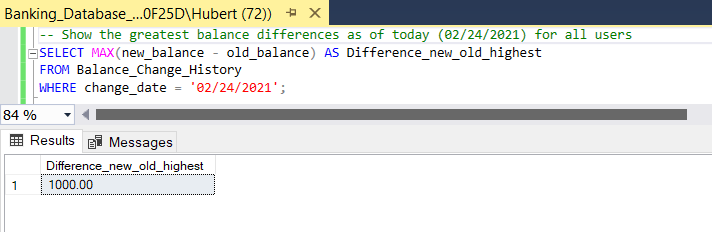
Update Balance\_Account\_Type with trigger to filled up balance history table

Querying question 1

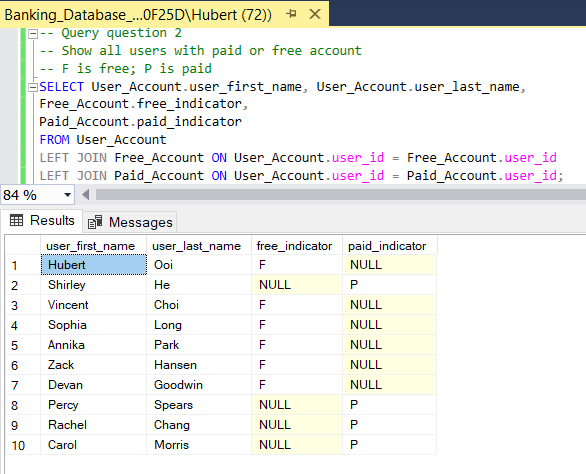
Part one: Show all users balance history with old and new balances along with its banking information as of 02/24/2021.



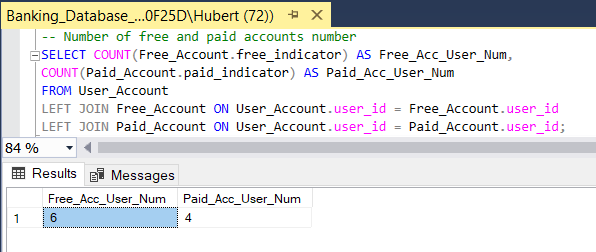
Part two: Show the user balance history with the greatest differences between old and new balances as of (2/24/2021)



Querying question 2

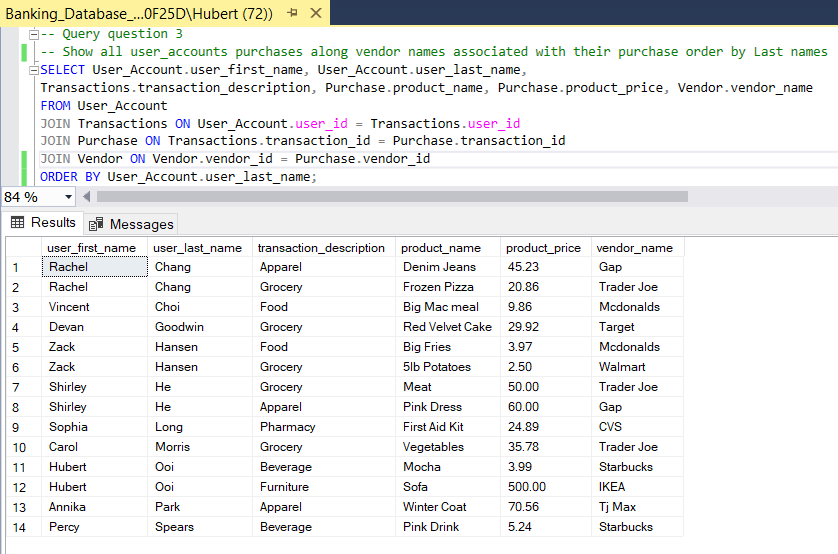
Part one: Show all users with paid or free account

Part two: Number of free vs paid accounts

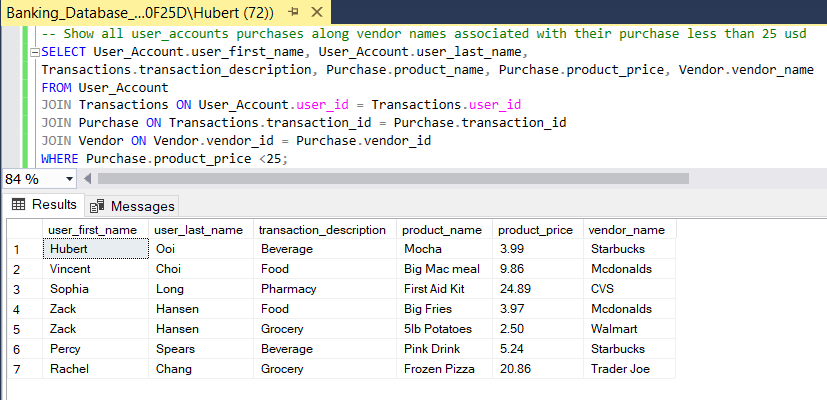


Querying question 3

Part one: Show all user accounts purchase information with vendor names, product name, and prices order by user last name.

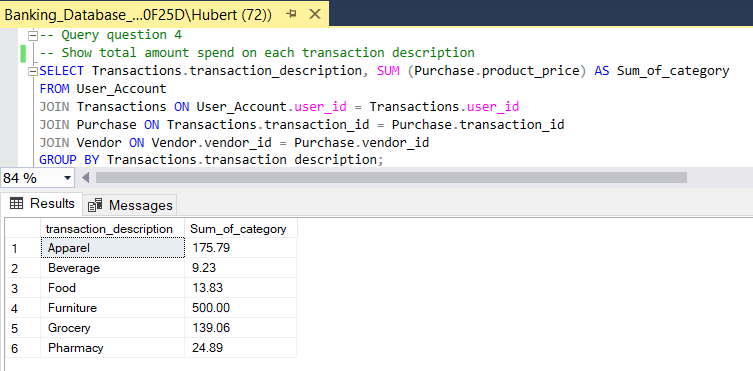


Part two: Show all user accounts purchases along vendor names, associated with their purchase less than 25 USD

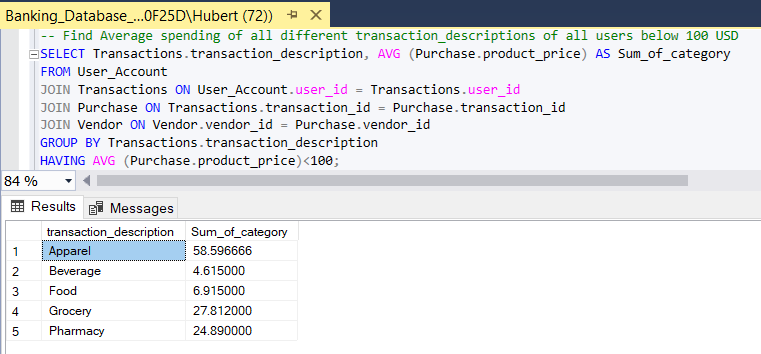


Querying question 4

Part one: Show the total sum of each transaction descriptions/categories



Part two: Find average of all different transaction descriptions/categories from all users less than 100 USD



Data Visualization 1

Category spending from user data

In this pie chart, it shows that users spend a significant amount of money for furniture. The data shows our users use the least money on beverages. Furniture prices might be one of the biggest expenditures for our users. Through this visualization is easy to distinguish what category spending more obviously to user trying to make sense their own spending habits. That is the reason when I chose pie chart as my choice of representing my data from my banking database

Data Visualization 2

Category average spending from user data

This pie chart shows the average spend on different categories with it average out from the user number of purchases with each category. It is still obvious that furniture take the top spot of biggest expenditure. The more data we have in our database, the better we will get the spending habits of all our users. This can help our database to capture how our user spend on average. This pie charts can help users decide their budget well.

Summary

For the final iteration of my project, I have learned a lot of the process of brainstorming an idea of database till I implement the database and populate it with data. Using SQL to extract needed data for certain type of analysis is one of the greatest lessons I get from this project-based learning experience. My database is not perfect, but I will further refine my project on my own for future uses and create more complex query as I learned more of it. I am grateful for my facilitator’s weekly feedback for me to refine my database as each iteration. Thank you for the valuable comments for me to learn what I am not doing correctly.