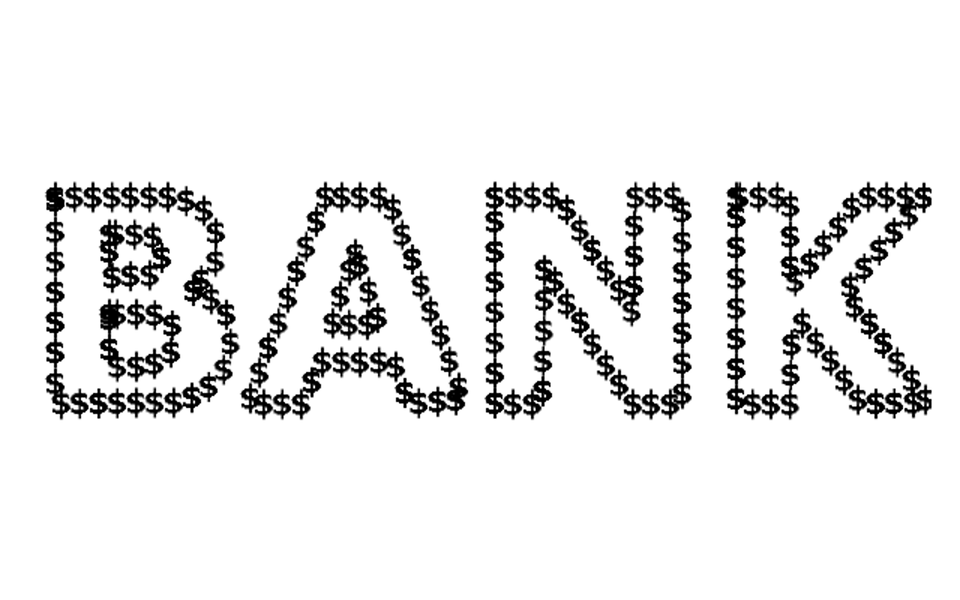
**BANK DATABASE PROJECT**

****

**KADİR HIZARCI**

**About**

We aim to create a sample bank database that is user friendly, despite being powerful. We added various concepts so the banks will be able to access necessary information about customers such as being able to check payment , account balances and credit limits, while being able to update such critical information without creating glitches.

Bank will also be able to answer questions like which customer has the highest balance, which employees working in which branch, their wages and the ability to update them when needed.

Data Analysis

The database will consist of banker, branch, customer, card, credit, debit, account, transaction, loan and payment tables, with the addition of a joint table .

Customers are identified by their customer\_ssn but also have name, surname, date\_of\_birth, gender, maiden\_name, phone\_number, credit\_limit, cityi district and age attributes.

Branches are identified by their location, denoted as branch\_city and branch\_district, as their address. They also have attributes number\_of\_personel and phone\_number.

Bankers are identified by their banker\_id. They also have name, surname, gender, phone\_number, wage as attributes.

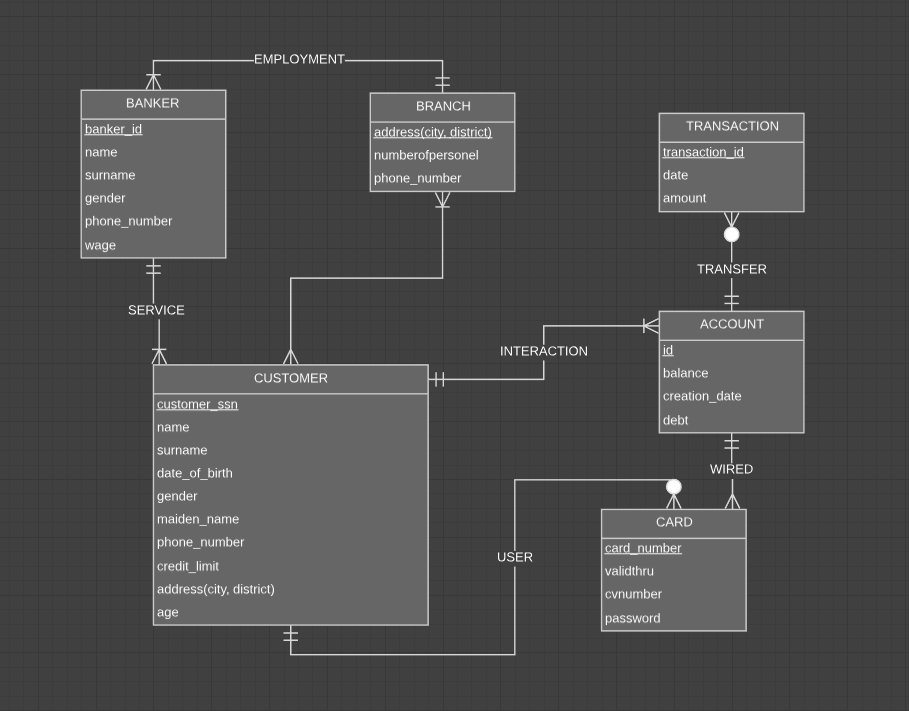
Cards are identified by their card\_number. They also have valid\_thru, cv\_number, password, customer\_ssn and account\_id as attributes.

Accounts are identified by their account\_id. They also have balance, creation\_date, customer\_ssn and debt as their attributes.

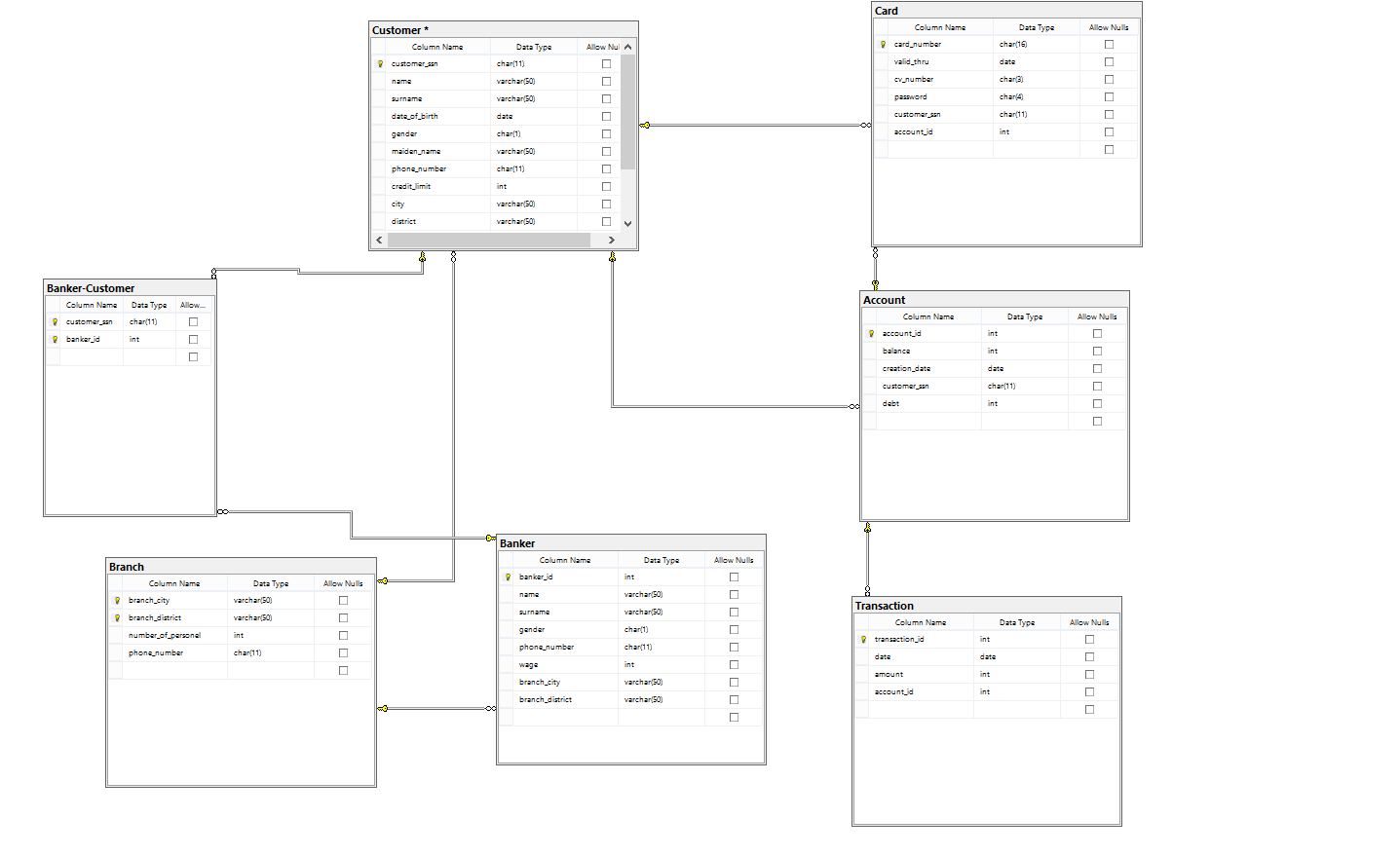
Transactions are identified by their transaction\_id. They also have date and amount as attributes.

We obtained our design from a government bank, albeit we naturally were not given access to critical customer information therefore we are gonna create our own randomly.

**ER Diagram**



**Database Diagram**



**Information About the Tables & Attributes**

We created 7 tables for MS SQL. They are Banker, Customer, Branch, Transaction, Account and Banker-Customer. Details for each table are down below.

* BANKER

Banker table consists of all information about the bankers employed in the bank.

* banker\_id is a primary key and also an identifier.
* name and surname of the banker
* gender is constrained as either ‘m’ as male or ‘f’ as female.
* phone\_number strictly takes 11 characters
* wage is of int type
* branch\_city is a foreign key from BRANCH table, showing the branch the banker is working for
* branch\_district is also a foreign key for the same purpose
* CUSTOMER

Customer table holds information about bank customers.

* customer\_ssn is a primary key that is strictly 11 characters
* name and surname of the customer
* date\_of\_birth of the customer denoted in date type
* gender is constrained as either ‘m’ as male or ‘f’ as female
* phone\_number strictly takes 11 characters
* credit\_limit is of int type and may change upon using some producers that will explained later.
* city and district customer is from
* branch\_city and branch\_district are foreign keys from BRANCH table
* age is calculated with the formula (CONVERT([smallint],datediff(year,[date\_of\_birth],getdate())))
* **BRANCH**

Branch holds information about each branch of the bank.

* branch\_city and branch\_district are primary keys.
* number\_of\_personel is of int type and may update depending on the usage of some procedures
* phone\_number strictly takes 11 characters
* **CARD**

Each card belongs to a certain account and a customer

* card\_number is a primary key and strictly takes 16 characters
* valid\_thru is of date type and holds till when the card will be functional
* cv\_number strictly takes 3 characters.
* password strictly takes 3 characters
* customer\_ssn is a foreign key from CUSTOMER table.
* account\_id is a foreign key from ACCOUNT table.
* **ACCOUNT**

Information about a customer specific account.

* account\_id is a primary key and also an identifier
* balance is of int type and hold how much cash customer has
* creation\_date is of date type
* customer\_ssn is a foreign key from CUSTOMER table
* debt is of int type and max amount is constrained by a formula related to credit\_limit from CUSTOMER table.
* **TRANSACTION**
* transaction\_id is a primary key and also an identifier
* date is of date type
* amount is denoted in int and directly linked to balance attribute in ACCOUNT table.
* account\_id is a foreign key from ACCOUNT table.
* **Banker-Customer**
* customer\_ssn is a foreign key CUSTOMER table
* banker\_id is also a foreign key, from BANKER table.
* Combination of them is a primary key.

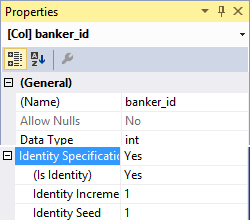
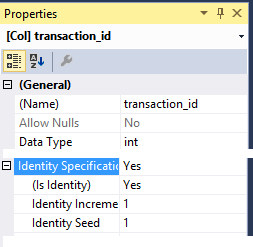
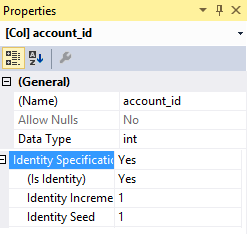
Relationship Analysis

* A banker may have one or many customers as a customer also may have one or many bankers.
* A banker can only be employed by one branch while a branch may have one or many bankers.
* A customer belongs to one and only one branch while a branch have many customers.
* A customer may have one or many accounts but an account belongs to one and only one customer.
* A customer may have none or many cards but each card belongs to one customer.
* An account may have zero or many transactions but each transaction belongs to one and only one account.
* An account is wired to one or many cards but a card can only be wired to one and only one account.

**CONSTRAINTS & INDICES & DEFAULTS**

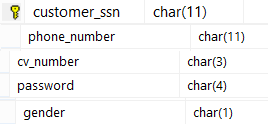
Indices

* Each indice is created for uniqueness and easier tracking purposes.



**Constraints**

* Main purpose of contraints is to keep database less error prone.



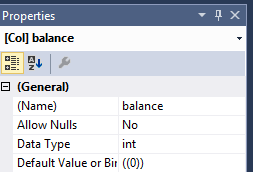
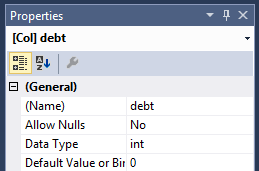
* We used char(length) types to receive inputs that are valid.



* We also used CHECK to constraint user to select either of two for their genders.

**Defaults**

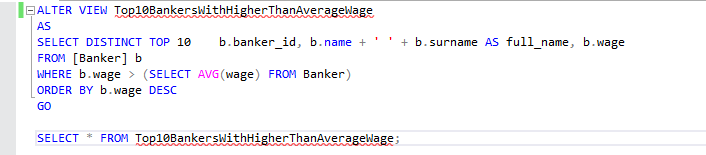
We use default values when an attribute is not mandatory to be filled initially.



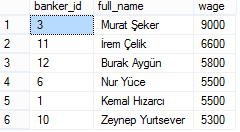
* Debt and balance fields are 0 by default

**Views**

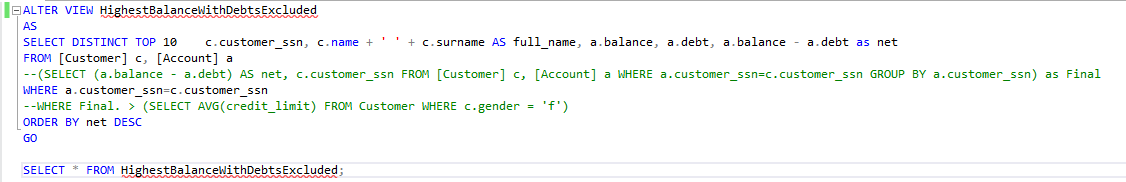
* **Top 10 Bankers With Higher Than Average Wages**



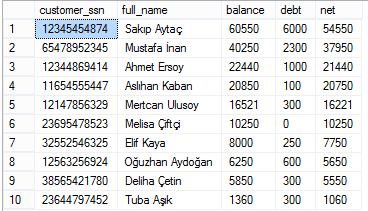
We created columns with ‘select distinct top’ keyword and merged banker name and surname into a full\_name column. Then we put a condition in where keyword so that we would only get the wages higher than average.



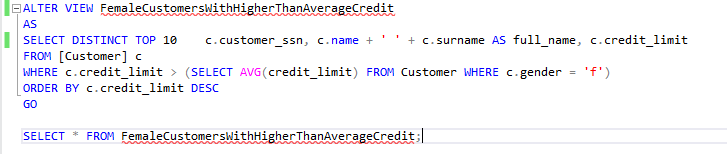
* **Top 10 Highest Account Balances With Debts Exclude**



We defined a new column named net and did a subtraction to find the net amount of money. Then we made sure the value is on sync with account names, finally ordering the view by net column.



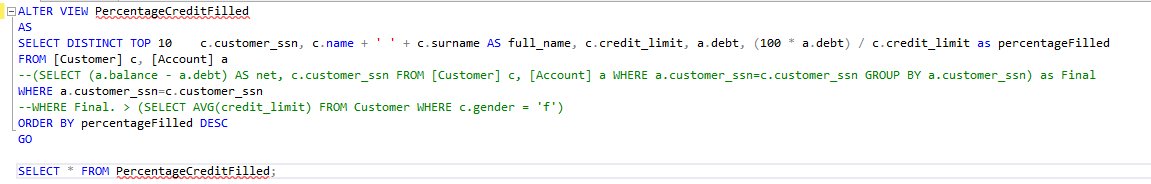
* **Top 10 Female Customers With Higher Than Average Credit Limit**



We concantenated the name and surname strings and also created a credit\_limit column. Then we used “where” keyword to limit our searches to the females only by using a nested inner query. Then we filtered them by them being greater than average, ultimately reaching our goal. Finally we ordered the results.



* **Percentage of Credit Limit Filled By Debts**



We formulated the percentage and kept the formula inside percentageFilled column. Then we made sure Account a’s customer\_ssn (foreign key from Customer table) corresponded to the one in Account table. After getting the results, we ordered the table.

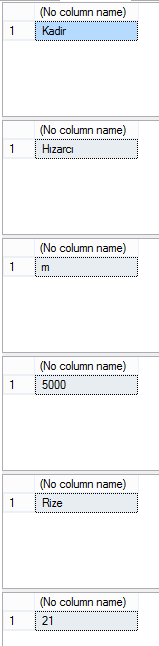


**Procedures**

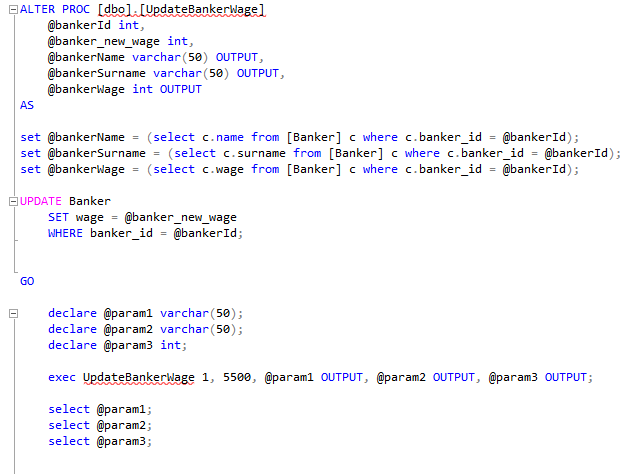
* **Get Customer Information by SSN**



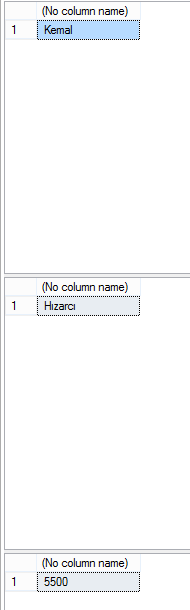
As an input, we constrained function to take 11 characters and use that value to get information from the other columns to the corresponding customer\_ssn. Then we declared parameters so that we can have the output.



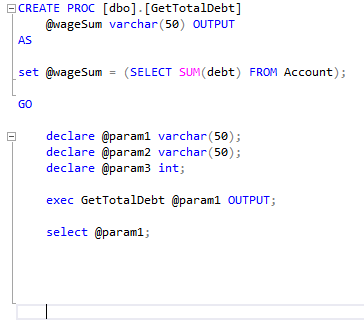
* **Update Banker Wage**



We created a function to have two inputs. First one receiving the banker’s id, second one receiving its new wage. After searching for the rest of the information about that certain banker and fetch them for output, function also updates the new wage to the database. Parameters declared show information about the banker and their new wage.



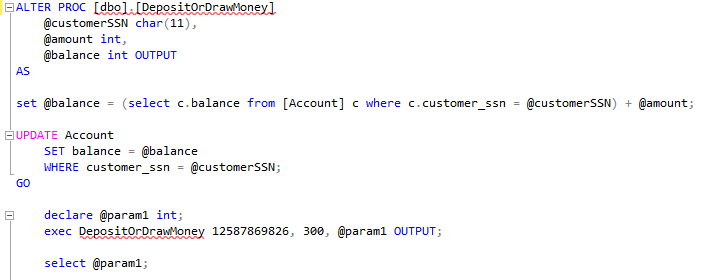
* **Get Total Debt To the Bank**



This procedure declaes wageSum as a variable and fetches all of debt columns in the database before putting it inside sum function to find the total debt. Requires no input.

C:\Users\Guest\AppData\Local\Microsoft\Windows\INetCache\Content.Word\proc 3 - 2.png

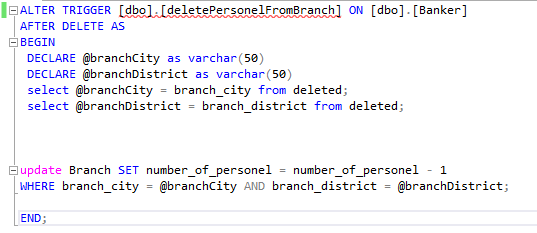
* **Deposit Or Draw Money From Account**



This procedure takes customer’s SSN as input and then fetches information about all of the accounts of that customer. If the input is positive, money gets deposited in the first account the customer has in the database by date order. If it’s negative it draws money from the customer’s account. Finally it updates account balance and returns it as a result.



* **Update Branch Number of Banker Data When a Record Is Deleted From Banker Table Trigger**



This trigger listens for a delete operation in Banker table and receives in which branch the deleted banker worked in. Since both city and district are primary keys and multiple branches can exists inside the same city, we declared two variables to collect both city and district info using “from deleted” keyword. After finding a match, we prompted our database to decrement number\_of\_personel by 1 for that branch.