CS669-DATABASE DESIGN AND IMPLEMENTATION FOR BUSINESS

Name: Sarvesh Krishnan Rajendran
Class: MET CS669 A2

Project Direction Overview:

I would like to develop a webpage or a mobile app "Maintenance Tracker" that tracks all the maintenance related activities of any vehicle that a person owns. When someone uses this portal, they would have full access to their maintenance record of all the vehicles that they own. And have a track of when they purchased a new part for a particular vehicle and other things. If a person has 2 cars and a bicycle. They need not remember the service due dates, forthcoming due dates, amount spent in the last service, etc. People like to know the history of each car maintenance for quite some reasons that may include to know whether the vehicle is still under warranty or if a spare part that was replaced last month failed within 5 weeks and to check if that can be claimed under the warranty of the vendor of that spare part. This portal will store the data of all the maintenance records of all the vehicle that a person has, it is even helpful for people who have many vehicles of different brands and types. The portal will have all the maintenance data of all their vehicle and help them keep a track of things.

I'm interested in this data because I feel it's always going to be hard to keep track of things especially when there are a lot of things to manage. Especially vehicles if you don't keep them serviced and other things it's going to cause problem when they are required the most.

Use cases and Field:

Signing up use case:
1.The person visits the portal of "Maintenance Tracker".
2.The application asks them to create an account for the first-time users
3.The user enters their information, and the account is created in the database
Also enter the number of cars they own and each type of car.
4.The portal asks them to install plugins that's asks the person each time something is bought, so that any purchase made relevant to a car is stored in the data base

FIELD	WHAT IT STORES	WHY ITS NEEDED
Email ID	This field stores the email id of the user with the associated account	This is necessary incase people have same names and if a person have multiple account, it helps sorting the correct account with the email drop down. Also useful in sending out emails to them
FIRST NAME	Stores the first name of the account holder	This is necessary for displaying the person's name on screens and addressing them when sending them emails or other communications.
Last Name	Stores the first name of the account holder	This is necessary for displaying the person's name on screens and addressing them when sending them emails or other communications.
No. of cars owned	Stores the value for the number of vehicles owned by them	This is used to keep track of number of vehicles that each customer

Automatic purchase tracker use case:
1.The person orders any spare of things related to a car
2.The plugin detects it and asks the person which car and adds it to the relevant records

FIELD	WHAT IT STORES	WHY ITS NEEDED
Date	It's the date in which the spare was bought	Useful when the person wants to sort by purchase date
Price	Total cost of the purchase	Useful when the person wants to know how much money they spent on the purchase
Name of the product	Name of the product that was bought	Needed so that the person know what they bought
Warranty of the product	Stores the date till which the product is covered in warranty	Useful when the spare part fails within the warranty period then the product

		can be replaced for free , without paying
Bought From	Stores the name of the seller from whom the person bought	Needed to know where the person made his or her purchases from
Bought for which CAR	Stores the value for which car of the person did he buy for	Needed for future references of the person to know for which car did he buy
URL/ Store address	Stores the URL of the store from where the product was bought if the person had bought it online or the location of the store if bought offline	Needed to know where it was bought, incase for future references

Service tracker use case:

1.Person does the maintenance of his car from a place
2. The person scans the job sheet of the service into the portal
3.The portal reads from the job card and stores relevant info for the future references

FIELD	WHAT IT STORES	WHY ITS NEEDED
Name of the CAR	Stores the name of the CAR	This is needed in case the person wants to know to which car did they do the service then they can search by car name
Type Of Service Done	Stores the service type done	Stored to know the type of serve done for the car during that visit to the mechanic
Involvement of Insurance	Stores if there is an involvement of insurance or not	This helps the person to understand if the insurance has paid for the last time or whether it was paid by the person from his ow pockets
Service Summary	Stores the data of the summary of the services	This helps the person to review even after couple of months and know why did they do the service

Service Warranty	Stores if there is any warranty issued by the mechanic or not	Incase there is still some problem with the car the person will be able to check if there is any service warranty given and can go and claim any warranty
Service Cost	Stores the value of the total service cost charged by the mechanic	For the person to analyse the service for maintain the car

Purchase Lookup Use Case:

1.The person signs into the portal
2.The person selects option to look up past purchases of spare parts
3.The portal pulls a short list of recent purchase from the data base, and gives user the option to search
4.user searches based on a date range which causes a database search
5.The portal pulls matching criteria from the database
6.User selects purchase they are interested.
7.Portal displays all matching info about the selected purchase
8.Now the user can analyze the data

Service Lookup Use Case:

1.The person signs into the portal
2.The person selects for which car the data needs to be presented
3.The person selects option to look up past services.
4.The portal pulls a short list of recent service from the data base, and gives user the option to search
5.user searches based on a date range which causes a database search
6.The portal pulls matching criteria from the database
7.User selects service they are interested.
8.Portal displays all matching info about the selected purchase
9.Now the user can analyze the data about services

Structural Database Rules:

->Signing up use case:
1.The person visits the portal of "Maintenance Tracker".
2.The application asks them to create an account for the first-time users
3.The user enters their information, and the account is created in the database Also enter the number of cars they own and each type of car.
4.The portal asks them to install plugins that's asks the person each time something is bought, so that any purchase made relevant to a car is stored in the data base
We can start structural rules for account into multiple entities with relationships, but from this use case alone we don't have enough information regarding it.

->Automatic purchase tracker use case:
1.The person orders any spare of things related to a car
2.The plugin detects it and asks the person which car and adds it to the relevant records
Rules:

1. Each purchase is associated with an account ,each account maybe associated with many purchases.
2. Each purchase is bought from a online/offline store, each online/offline store has one to may purchases sold
3. Each purchase has one or more spares bought, each spares have one to many purchases.

->Service tracker use case:
1.Person does the maintenance of his car from a place
2. The person scans the job sheet of the service into the portal
3.The portal reads from the job card and stores relevant info for the future references
Rules:

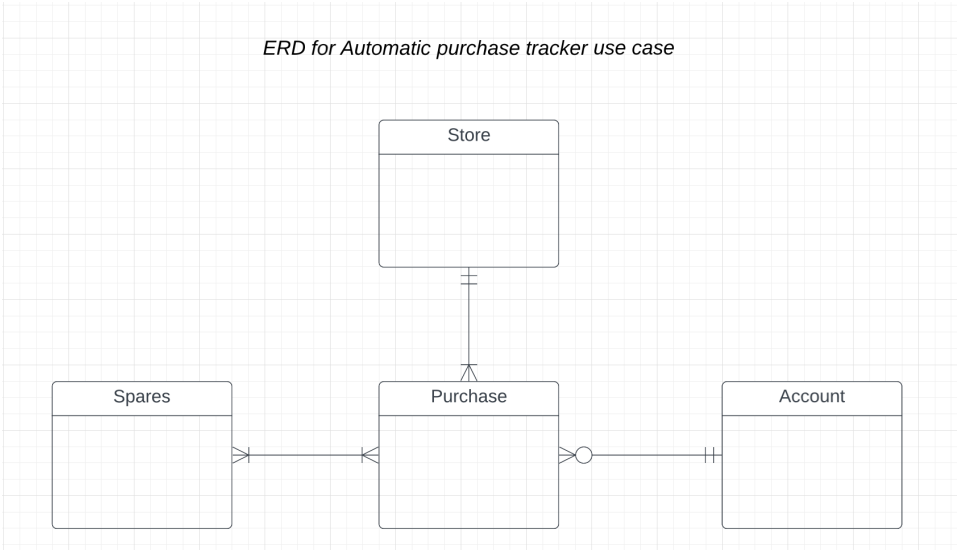
1. Each maintenance is done at a one place, each service centre has one or many services done
2. Each job sheet is associated with a particular service, each service has one job sheet.

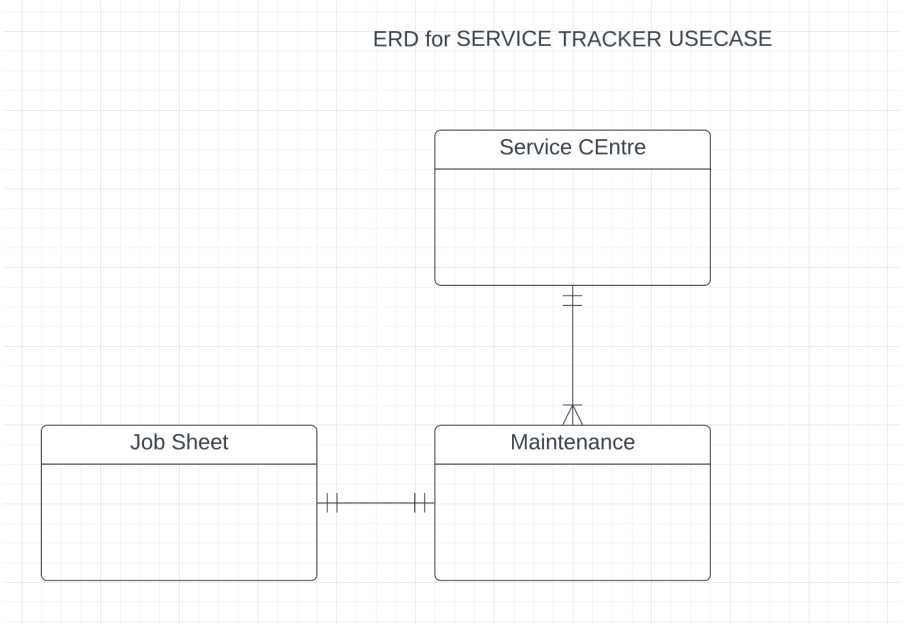
Purchase Lookup Use Case:
1.The person signs into the portal
2.The person selects option to look up past purchases of spare parts
3.The portal pulls a short list of recent purchase from the data base, and gives user the option to search
4.user searches based on a date range which causes a database search
5.The portal pulls matching criteria from the database
6.User selects purchase they are interested.
7.Portal displays all matching info about the selected purchase
8.Now the user can analyse the data

Service Lookup Use Case:
1.The person signs into the portal
2.The person selects for which car the data needs to be presented
3.The person selects option to look up past services.
4.The portal pulls a short list of recent service from the data base, and gives user the option to search
5.user searches based on a date range which causes a database search
6.The portal pulls matching criteria from the database
7.User selects service they are interested.
8.Portal displays all matching info about the selected purchase
9.Now the user can analyze the data about services

Service Lookup Use Case and Service lookup use-cases both don't have data that can be stored in the data it is basically to view the stored data

ERD DIAGRAMS:





Signing up use case(old):
1.The person visits the portal of “Maintenance Tracker”.

2.The application asks them to create an account for the first-time users

3.The user enters their information, and the account is created in the database Also enter the number of cars they own and each type of car.

4.The portal asks them to install plugins that’s asks the person each time something is bought, so that any purchase made relevant to a car is stored in the data base

I will offer a free account subscription and a paid account subscription

Signing up use case(new):

1.The person visits the portal of “Maintenance Tracker”.

2. The application asks them to create a free or a paid account for first-time users

3. The user selects the type of account and enters their information, and the account is created in the database Also enter the number of cars they own and each type of car.

4. .The portal asks them to install plugins that’s asks the person each time something is bought, so that any purchase made relevant to a car is stored in the data base

We can add one more structural rules to our existing one which we did in the previous iteration

Which is ,An account is a free account or a paid account.

Automatic purchase tracker use case(Old):

1.The person orders any spare of things related to a car

2.The plugin detects it and asks the person which car and adds it to the relevant records

We can modify the use case as follows,

Automatic purchase tracker use case(new):

1.The person orders any spare of things related to a car

2.The plugin detects it and asks the person which car and adds it to the relevant records such as whether the purchase is online or face to face.

Another structural database rule is ,

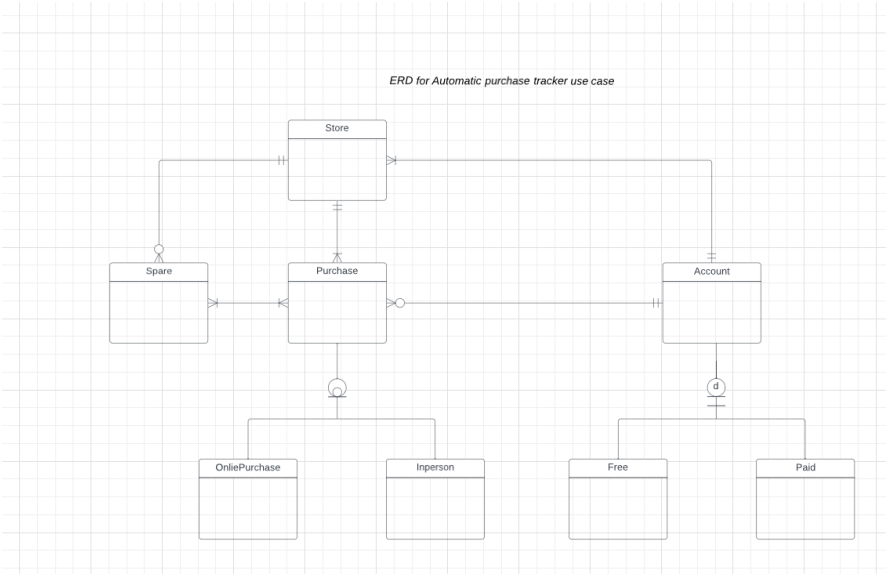
A purchase is an online purchase, a face-to-face purchase, both or none of these.

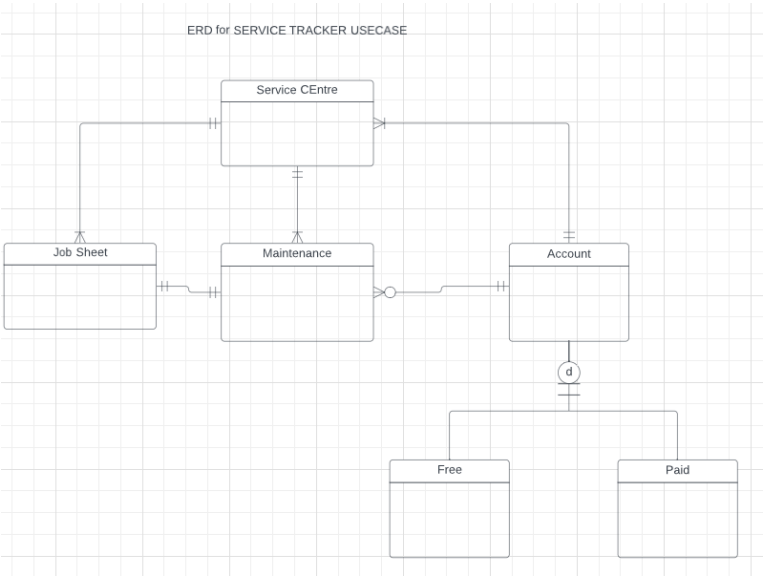
Now the updated structural database rules are mentioned below:

Rules:

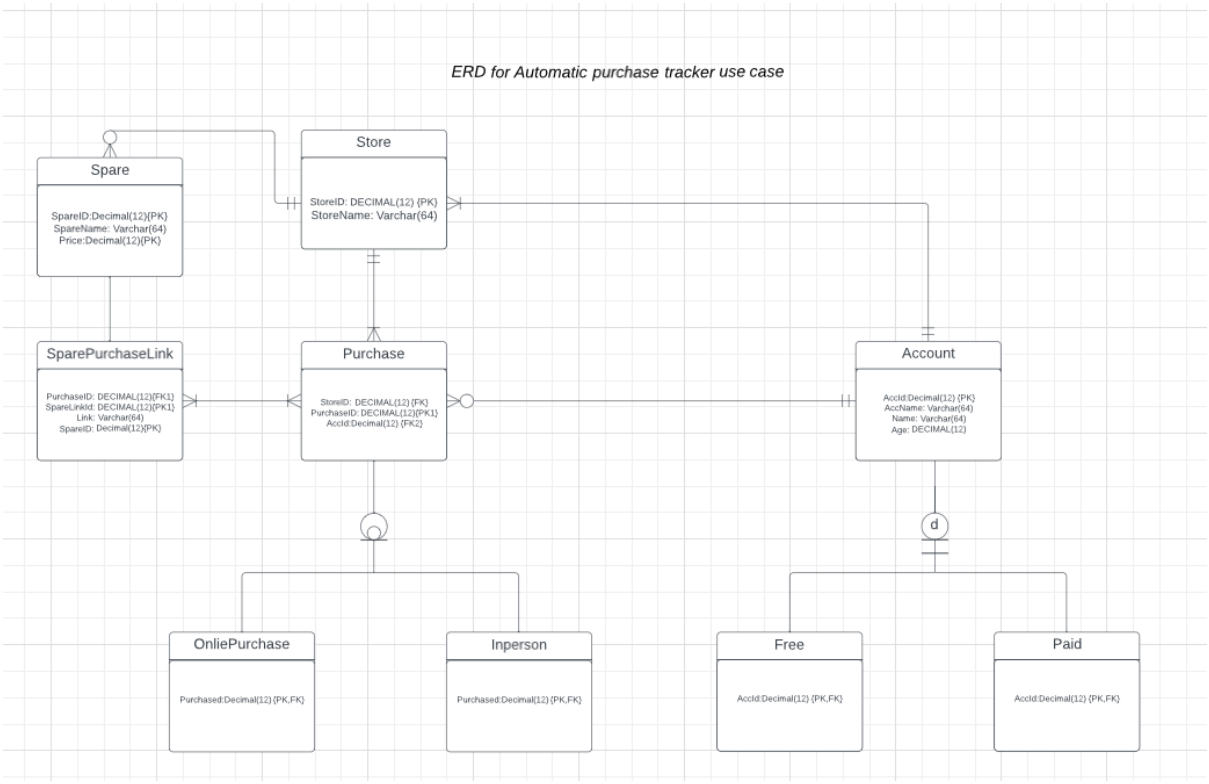
1. Each purchase is associated with an account ,each account maybe associated with many purchases.
2. Each purchase is bought from a online/offline store, each online/offline store has one to may purchases sold
3. Each purchase has one or more spares bought, each spares have one to many purchases.
4. An account is a free account or a paid account.
5. A purchase is an online purchase, a face-to-face purchase, both or none of these.

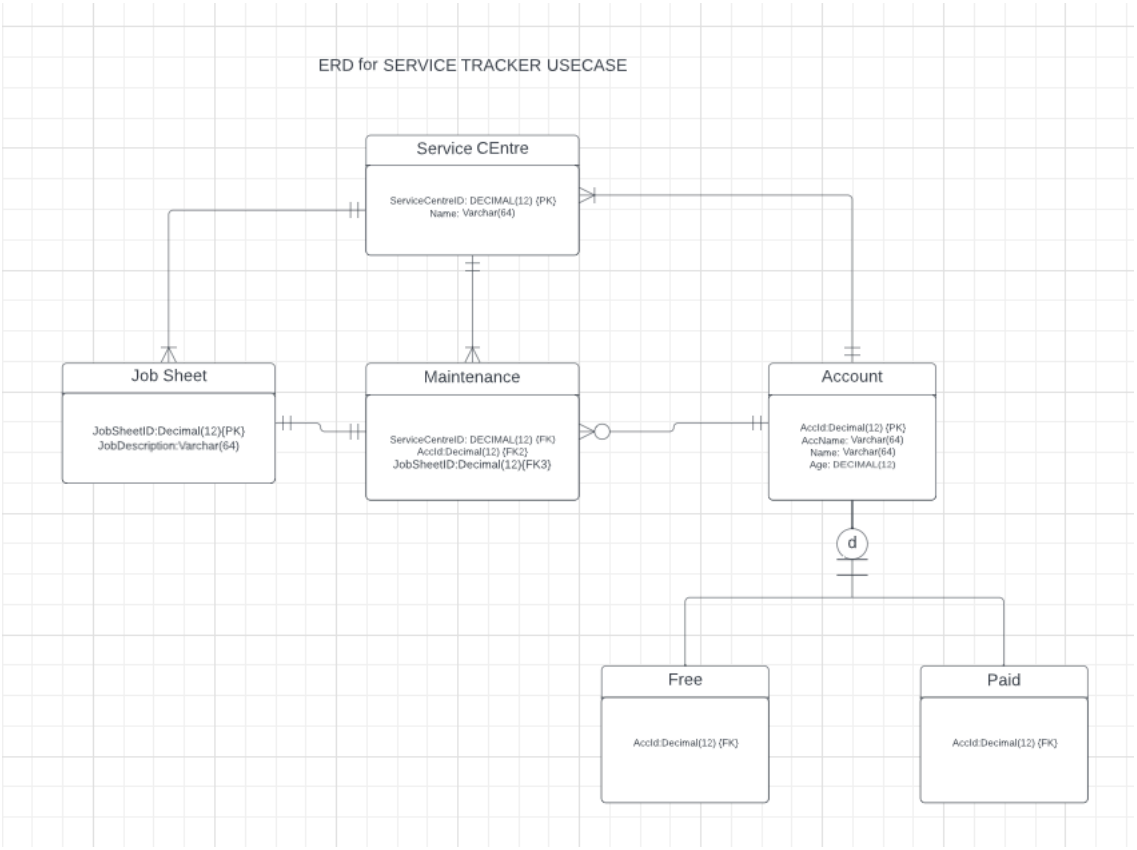
CLASSIFICATION AND ASSOCIATIVE MAPPING:





SPECIALIZATION-GENERALIZATION MAPPING:





ADDING ATTRIBUTES TO MY DBMS PHYSICAL ERD:

The below is the process of adding attributes and their datatypes table-by-table. The reasoning is also mentioned

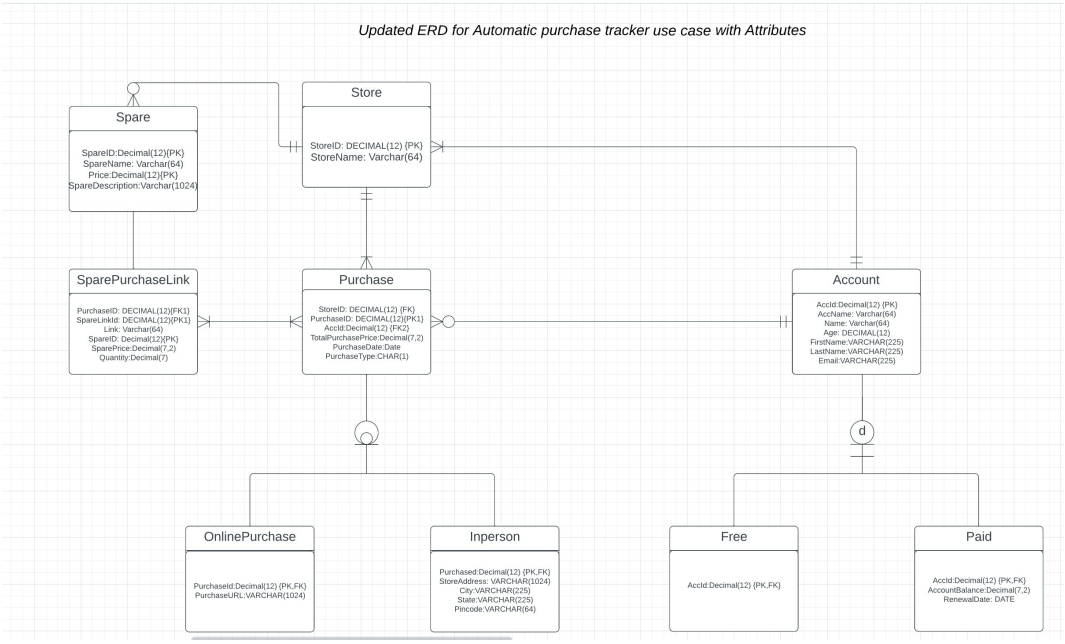
For Spare Purchase:

Table	**Attribute**	**Datatype**	**Reasoning**
Spare	Spare Description	Varchar(1024)	Every product has its own description. The length of the description can depend on the type of spare. Hence I'm giving 1024 length
SparePurchaseLink	SparePrice	Decimal(7,2)	When a person buys a spare they buy it at a price. Though the price may change over time, the price at which they bought wont change
SparePurchaseLink	Quantity	Decimal(7)	The quantity of purchase depends on the requirement of the customer.I allow for upto 7 digits as a safe boud
Purchase	Total Purchase Price	Decimal(7,2)	When the buyer buys he or she can buy many different types of items.
Purchase	Purchase Date	Date	There would be a specific purchase date for every purchase
Purchase	Purchase Type	CHAR(1)	There can be two type of purchase online or inshop
Account	First Name	VARCHAR(255)	This is the first name of the user

Account	Last Name	VARCHAR(255)	This is the last name off the user
Account	Email	VARCHAR(255)	This is the email address o the user
Paid	Account Balance	Decimal(7,2)	This is the unpaid balance , if any for the paid account.I allow for upto 7 digits, though it will likely never get near this high
Paid	Renewal Date	DATE	The date on which the account needs to be renewd for continuing the perks of members
Online Purchase	Purchase URL	VARCHAR(1024)	Every online purchase has its own link
InPerson	Store Address	VARCHAR(1024)	Every inperson purchase has its own store address
InPerson	City	VARCHAR(225)	In which city the store is located
Inperson	State	VARCHAR(225)	In which state the store located
InPerson	Pincode	VARCHAR(64)	The pincode of theaddress

For the given use cases and structural database rules I've developed so far , these attributes suffice.

The updated ERD with attributes:



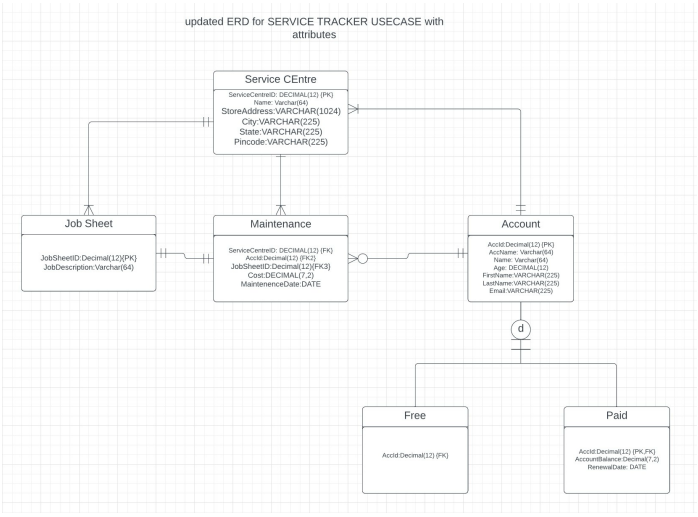
FOR Service:

Table	**Attribute**	**Datatype**	**Reasoning**
Service Centre	Store Address	VARCHAR(1024)	Every service centre has its own address
Service Centre	City	VARCHAR(225)	In which city the service centre is located
Service Centre	State	VARCHAR(225)	In which state the service centre is located
Service Centre	Pincode	VARCHAR(64)	The pincode of theaddress

Maintenance	Cost	Decimal (7,2)	The total cost of all service done
Maintenance	MaintenceDate	DATE	The date in which the maintenance is done
Account	First Name	VARCHAR(255)	This is the first name of the user

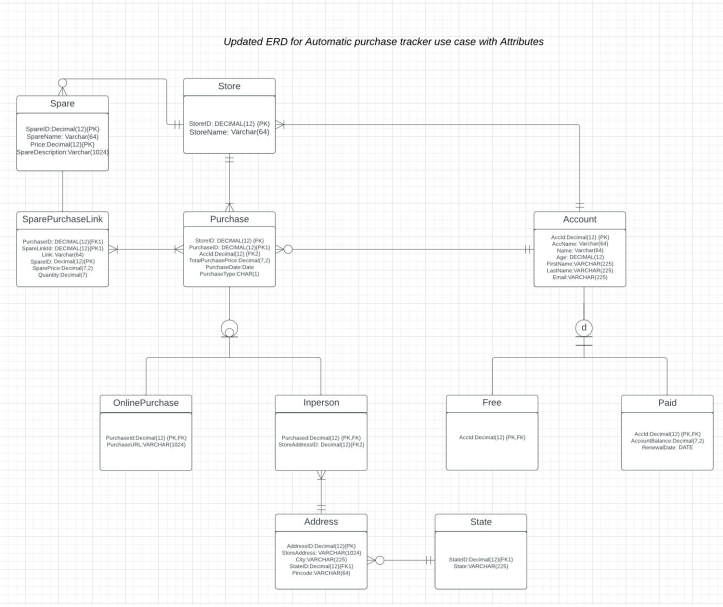
Account	Last Name	VARCHAR(255)	This is the last name off the user
Account	Email	VARCHAR(255)	This is the email address of the user
Paid	Account Balance	Decimal(7,2)	This is the unpaid balance , if any for the paid account.I allow for upto 7 digits, though it will likely never get near this high
Paid	Renewal Date	DATE	The date on which the account needs to be renewd for continuing the perks of members

The updated ERD with attributes:



NORMALIZING DBMS PHYSICAL ERDS:

I noticed only one place where there is need for normalizing in my physical erd which is the Spare purchase ERD and that is where with the address information inPerson purchase entity.If many purchases are made inPerson store the address info will repeat.

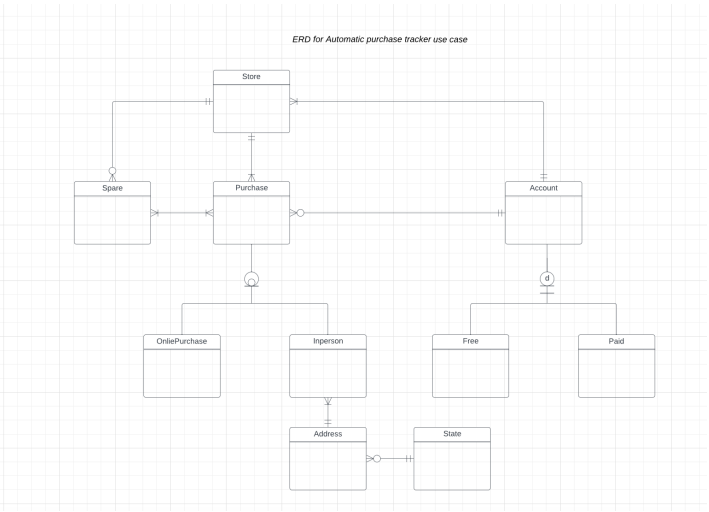


There are 2 additional entities after normalization- Address and State. By moving the primary address information into its own entity, I do not need to repeat the address every time a purchase is made. I can reference the address instead. Likewise, rather than repeating the state name every time a purchase is made, the address entity reference's the state entity instead.

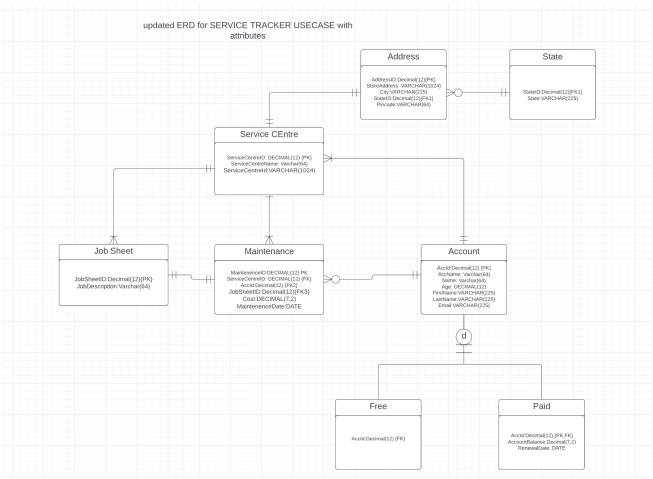
Below are my structural database rules modified to reflect the new entities.

1. Each purchase is associated with an account ,each account maybe associated with many purchases.
2. Each purchase is bought from a online/offline store, each online/offline store has one to may purchases sold
3. Each purchase has one or more spares bought, each spares have one to many purchases.
4. An account is a free account or a paid account.
5. A purchase is an online purchase, a face-to-face purchase, both or none of these.
6. Each in-person purchase is made at an address; each address is associated with one or more in-person purchase.
7. Each address has a state; each state may be associated with many address.

Modified conceptual erd for the automatic purchase tracker:



For Service Tracker:



The same reason as mentioned for tracking spare parts.
Below are my structural database rules modified to reflect the new entities:

1. Each purchase is associated with an account ,each account maybe associated with many purchases.
2. Each purchase is bought from a online/offline store, each online/offline store has one to may purchases sold
3. Each purchase has one or more spares bought, each spares have one to many purchases.
4. An account is a free account or a paid account.
5. A purchase is an online purchase, a face-to-face purchase, both or none of these.
6. Each in-person purchase is made at an address; each address is associated with one or more in-person purchase.
7. Each address has a state; each state may be associated with many address.

8. Each Service Centre has an address; each address is associated to one service centre
9. Each Service centre address has a state each state may be associated with many address

MY PROJECT SCRIPT:

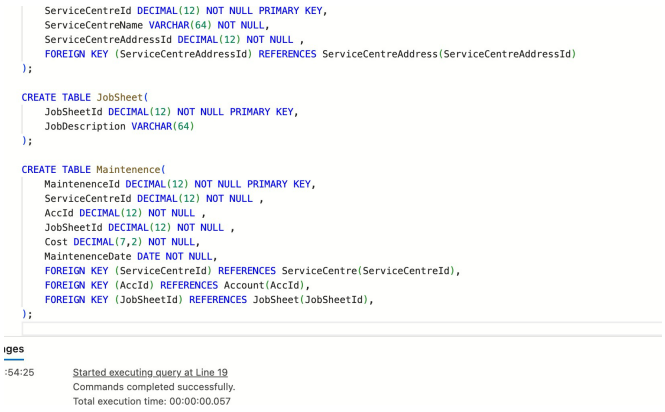
```
----------------------Creating Tables
CREATE TABLE Account(
    AccId DECIMAL(12) NOT NULL PRIMARY KEY,
    AccName VARCHAR(64) NOT NULL,
    UserName VARCHAR(64) NOT NULL,
    Age DECIMAL(12) NOT NuLL,
    FirstName VARCHAR(225) NOT NULL,
    LastName VARCHAR(225) NOT NULL,
    Email VARCHAR(225) NOT NULL
);

CREATE TABLE FreeAccount(
    AccId DECIMAL(12) NOT NULL PRIMARY KEY,
    FOREIGN KEY (AccId) REFERENCES Account(AccId)
);

CREATE TABLE PaidAccount(
    AccId DECIMAL(12) NOT NULL PRIMARY KEY,
    AccountBalance DECIMAL(7,2) NOT NULL,
    RenewalDate DATE NOT NULL,
    FOREIGN KEY (AccId) REFERENCES Account(AccId)
);

CREATE Table Store(
    StoreId DECIMAL(12) NOT NULL PRIMARY KEY,
    StoreName VARCHAR(64) NOT NULL
);

CREATE TABLE Spare(
    SpareId DECIMAL(12) NOT NULL PRIMARY KEY,
    SpareName VARCHAR(64) NOT NULL,
    Price DECIMAL(12) NOT NULL ,
    SpareDescription VARCHAR(1024) NOT NULL
);
```

```
  CREATE TABLE Purchase(
      StoreId DECIMAL(12) NOT NULL ,
      FOREIGN KEY (StoreId) REFERENCES Store(StoreId),
      PurchaseId DECIMAL(12) NOT NULL PRIMARY KEY,
      AccId DECIMAL(12) NOT NULL  ,
      FOREIGN KEY (AccId) REFERENCES Account(AccId),
      TotalPurchasePrice DECIMAL(7,2) NOT NULL,
      PurchaseDate DATE NOT NULL,
      PurchaseType CHAR(1) NOT NULL,
  );

  CREATE TABLE SparePurchaseLink(
      PurchaseId DECIMAL(12) NOT NULL ,
      SpareLinkId DECIMAL(12) NOT NULL PRIMARY KEY,
      Link VARCHAR(64) NOT NULL,
      SpareId DECIMAL(12) NOT NULL ,
      SparePrice DECIMAL(7,2) NOT NULL,
      Quantity DECIMAL(7),
      FOREIGN KEY (PurchaseId) REFERENCES Purchase(PurchaseId),
      FOREIGN KEY (SpareId) REFERENCES Spare(SpareId)
  );

  CREATE TABLE OnlinePurchase(
      PurchaseId DECIMAL(12) NOT NULL PRIMARY KEY,
      PurchaseURL VARCHAR(1024)
      FOREIGN KEY (PurchaseId) REFERENCES Purchase(PurchaseId)
  );

  CREATE TABLE StoreState(
      StoreStateId Decimal(12) NOT NULL PRIMARY KEY,
      StoreState VARCHAR(225) NOT NULL
  );
```

```
CREATE TABLE StoreAddress(
    AddressId DECIMAL(12) NOT NULL PRIMARY KEY,
    StoreAddress VARCHAR(1024) NOT NULL,
    StoreCity VARCHAR(225) NOT NULL,
    StoreStateId Decimal(12) NOT NULL,
    StorePinCode VARCHAR(64) NOT NULL,
    FOREIGN KEY (StoreStateId) REFERENCES StoreState(StoreStateId)
);

CREATE TABLE InPerson(
    PurchaseId DECIMAL(12) NOT NULL ,
    AddressId DECIMAL(12) NOT NULL ,
    FOREIGN KEY (PurchaseId) REFERENCES Purchase(PurchaseId),
    FOREIGN KEY (AddressId) REFERENCES StoreAddress(AddressId)
);

----
CREATE TABLE ServiceCentreState(
    ServiceCentreStateId DECIMAL NOT NULL PRIMARY KEY,
    ServiceCentreState VARCHAR(225) NOT NULL
);
CREATE TABLE ServiceCentreAddress(
    ServiceCentreAddressId DECIMAL(12) NOT NULL PRIMARY KEY,
    ServiceCentreAddress VARCHAR(1024) NOT NULL,
    ServiceCentreCity VARCHAR(225) NOT NULL,
    ServiceCentreStateId DECIMAL NOT NULL ,
    ServiceCentrePinCode VARCHAR(225) NOT NULL,
    FOREIGN KEY(ServiceCentreStateId) REFERENCES ServiceCentreState(ServiceCentreStateId)
);
```

```
CREATE TABLE ServiceCentre(
    ServiceCentreId DECIMAL(12) NOT NULL PRIMARY KEY,
    ServiceCentreName VARCHAR(64) NOT NULL,
    ServiceCentreAddressId DECIMAL(12) NOT NULL ,
    FOREIGN KEY (ServiceCentreAddressId) REFERENCES ServiceCentreAddress(ServiceCentreAddressId)
);

CREATE TABLE JobSheet(
    JobSheetId DECIMAL(12) NOT NULL PRIMARY KEY,
    JobDescription VARCHAR(64)
);

CREATE TABLE Maintenence(
    MaintenenceId DECIMAL(12) NOT NULL PRIMARY KEY,
    ServiceCentreId DECIMAL(12) NOT NULL ,
    AccId DECIMAL(12) NOT NULL ,
    JobSheetId DECIMAL(12) NOT NULL ,
    Cost DECIMAL(7,2) NOT NULL,
    MaintenenceDate DATE NOT NULL,
    FOREIGN KEY (ServiceCentreId) REFERENCES ServiceCentre(ServiceCentreId),
    FOREIGN KEY (AccId) REFERENCES Account(AccId),
    FOREIGN KEY (JobSheetId) REFERENCES JobSheet(JobSheetId),
);
```

OUTPUT:



```
    ServiceCentreId DECIMAL(12) NOT NULL PRIMARY KEY,
    ServiceCentreName VARCHAR(64) NOT NULL,
    ServiceCentreAddressId DECIMAL(12) NOT NULL ,
    FOREIGN KEY (ServiceCentreAddressId) REFERENCES ServiceCentreAddress(ServiceCentreAddressId)
);

CREATE TABLE JobSheet(
    JobSheetId DECIMAL(12) NOT NULL PRIMARY KEY,
    JobDescription VARCHAR(64)
);

CREATE TABLE Maintenence(
    MaintenenceId DECIMAL(12) NOT NULL PRIMARY KEY,
    ServiceCentreId DECIMAL(12) NOT NULL ,
    AccId DECIMAL(12) NOT NULL ,
    JobSheetId DECIMAL(12) NOT NULL ,
    Cost DECIMAL(7,2) NOT NULL,
    MaintenenceDate DATE NOT NULL,
    FOREIGN KEY (ServiceCentreId) REFERENCES ServiceCentre(ServiceCentreId),
    FOREIGN KEY (AccId) REFERENCES Account(AccId),
    FOREIGN KEY (JobSheetId) REFERENCES JobSheet(JobSheetId),
);
```

ages

:54:25 Started executing query at Line 19
Commands completed successfully.
Total execution time: 00:00:00.057

INDEXING:

As far as primary keys which are already indexed, here is the list

Account.AccId
Store.StoreId
Spare.SpareId
Purchase.PurchaseId
SparePurchaseLink.SpareLinkId
StoreState.StoreStateId
StoreAddress.AddressId
ServiceCentreState.ServiceCentreStateId
ServiceCentreAddress.ServiceCentreAddressId
ServiceCentre.ServiceCentreId
JobSheet.JobSheetId
Maintenance.MaintenenceId

As far as foreign key all of them need to be indexed

No.	**Column**	**Unique?**	**Description**
1	FreeAccount.AccId	No	The foreign key in FreeAccount referencing Account is not unique because there can be many Free account in Account
2	PaidAccount.AccId	No	The foreign key in PaidAccount referencing Account is not unique because there can be many Paid account in Account
3	Purchase.StoreId	No	The foreign key in Purchase referencing Store is not unique because there can be many Purchases from store.
4	Purchase.AccId	No	The foreign key in Purchase referencing

			Account is not unique because there can be many Purchases from store.
5	SparePurchaseLink.PurchaseId	No	The foreign key in SparePurchaseLink referencing Purchase is not unique because there can be many links of spares from purchase.
6	SparePurchaseLink.SpareId	No	The foreign key in SparePurchaseLink referencing Spares is not unique because there can be many links of spares.
7	OnlinePurchase.PurchaseId	No	The foreign key in OnlinePurchasing referencing Purchase is not unique because there can be many OnlinePurchase of the same Purchase
8	StoreAddress.StoreStateId	No	The foreign key in StoreAddress referencing StoreState is not unique because there can be many States of the same address
9	InPerson.PurchaseId	No	The foreign key InPerson refencing Purchase is not unique because there can be many purchases from InPerson

10	InPerson.AddressId	No	The foreign key in InPerson refencing Address is not unique because there can be many same address from the InPerson
11	ServiceCentreAddress.ServiceCentreStateId	No	The foreign key in ServiceCentreAddress referencing SerrviceCentreState is not unique because there can be many Service centre in the same Sate
12	ServiceCentre.ServiceCentreAddressId	No	The foreign key in ServiceCentre referencing SerrviceCentreAddress is not unique because there can be many address of the same service centre.
13	Maintenence.ServiceCentreId	No	The foreign in Maintenence key refencing ServiceCentre is not unique because many maintenance can occur in the same service centre
14	Maintenence.AccId	No	The foreign key in maintenence refencing Account is not unique because many maintenance can occur for the same account
15	Maintenance.JobSheetId	No	The foreign key in Maintenance

			refencing ServiceCentre is not unique because many maintenance can occur in the same service centre

3 query driven indexes that are neither foreign key and primary keys are:

1. There will be many query that limit by account balances, to see what accounts are over or under a certain balance.So I select PaidAccount.AccountBalance to be index
2. There will be many query that limit by date of purchase, by a particular year,month,week or day so I select Purchase.PurchaseDate to index.
3. There will be many query that limit by the cost of maintenance, to see how many maintenences that has been done under 1000$ or so , so I select Maintenance.Cost

The above three mentioned all are not uniqueindex.

CREATING INDEXES ON YOUR DATABASE:

```
---CREATING INDEXES IN OUR DATABASES FOR FOREIGN KEYS
--1
CREATE INDEX FreeAccountIdx
on FreeAccount(AccId);

--2
CREATE INDEX PaidAccountIdx
on PaidAccount(AccId);

--3
CREATE INDEX PurchaseStoreIdx
on Purchase(StoreId);

--4
CREATE INDEX PurchaseAccIdx
on Purchase(AccId);

--5
CREATE INDEX SparePurchaseLinkPurchaseIdx
on SparePurchaseLink(PurchaseId);

--6
CREATE INDEX SparePurchaseLinkSpareIdIdx
on SparePurchaseLink(SpareId);

--7
CREATE INDEX OnlinePurchasePurchaseIdIdx
on OnlinePurchase(PurchaseId);

--8
CREATE INDEX StoreAddressStoreStateIdIDX
on StoreAddress(StoreStateId);

--9
CREATE INDEX InPersonPurchaseIdIDX
on InPerson(PurchaseId);

--10
CREATE INDEX InPersonAddressIdIDX
on InPerson(AddressId);
```

```sql
--10
CREATE INDEX InPersonAddressIdIDX
on InPerson(AddressId);

--11
CREATE INDEX ServiceCentreAddressStateIDX
on ServiceCentreAddress(ServiceCentreStateId);

--12
CREATE INDEX ServiceCentreAddressIdIDX
on ServiceCentre(ServiceCentreAddressId);

--13
CREATE INDEX MaintenenceServiceCentreIdIDX
on Maintenence(ServiceCentreId);

--14
CREATE INDEX MaintenenceAccIdIDX
on Maintenence(AccId);

--15
CREATE INDEX MaintenanceJobSheetIdIDX
on Maintenence(JobSheetId);


-------------------------------------------
--------Creating index for query driven index
---1
CREATE INDEX PaidAccountAccountBalanceIDX
ON PaidAccount(AccountBalance);

---2
CREATE INDEX PurchaseDateIDX
ON Purchase(PurchaseDate);

--3
CREATE INDEX MaintenanceCostIDX
ON Maintenence(Cost)
```

OUTPUT:

```sql
----------------
---CREATING INDEXES IN OUR DATABASES FOR FOREIGN KEYS
--1
CREATE INDEX FreeAccountIdx
on FreeAccount(AccId);

--2
CREATE INDEX PaidAccountIdx
on PaidAccount(AccId);

--3
CREATE INDEX PurchaseStoreIdx
on Purchase(StoreId);

--4
CREATE INDEX PurchaseAccIdx
on Purchase(AccId);

--5
CREATE INDEX SparePurchaseLinkPurchaseIdx
on SparePurchaseLink(PurchaseId);

--6
CREATE INDEX SparePurchaseLinkSpareIdIdx
on SparePurchaseLink(SpareId);
```

ges

03:47 Started executing query at Line 141
Commands completed successfully.
Total execution time: 00:00:00.049

Implementing Transactions in my Database:

1)I’m Creating two transaction that creates a free account and a paid accountFor Free Account:

```
---Transaction
---#1
DROP PROCEDURE AddFreeAccount
CREATE PROCEDURE AddFreeAccount @AccountID DECIMAL(12),@AccountUsername VARCHAR(64),@EncryptedPassword VARCHAR(255),@Age DECIMAL(12,0),
@FirstName VARCHAR(255), @LastName VARCHAR(255), @Email VARCHAR
AS
BEGIN
INSERT INTO Account(AccId,UserName,EncryptedPassword,Age,FirstName,LastName,Email,CreatedOn,AccountType)
VALUES (@AccountID ,@AccountUsername ,@EncryptedPassword ,@Age,
@FirstName , @LastName , @Email,GETDATE(),'F' )
INSERT INTO FreeAccount(AccId)
VALUES(@AccountID)
END;
GO
```

```
Messages
18:37:05    Started executing query at Line 5
            Commands completed successfully.
            Total execution time: 00:00:00.014
```

```
BEGIN TRANSACTION AddFreeAccount;
EXECUTE AddFreeAccount 1,'pittbrad','password1',25,'Brad','Pitt','brad.pitt@gmail.com';
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddFreeAccount;
EXECUTE AddFreeAccount 3,'alice_jones', 'password6', 29, 'Alice', 'Jones', 'alice.jones@email.com';
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddFreeAccount;
EXECUTE AddFreeAccount 5,'bob_miller', 'password7', 31, 'Bob', 'Miller', 'bob.miller@email.com';
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddFreeAccount;
EXECUTE AddFreeAccount 7, 'emily_white', 'password8', 26, 'Emily', 'White', 'emily.white@email.com';
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddFreeAccount;
EXECUTE AddFreeAccount 9, 'peter_miller', 'pm@pwd123', 33, 'Peter', 'Miller', 'peter.miller@example.com';
COMMIT TRANSACTION AddFreeAccount;
```

```
Messages
18:37:54    Started executing query at Line 17
            (1 row affected)
            (1 row affected)
            (1 row affected)
            (1 row affected)
            (1 row affected)
            (1 row affected)
            (1 row affected)
            (1 row affected)
            (1 row affected)
            (1 row affected)
            Total execution time: 00:00:00.056
```

For Paid Account:

```
DROP PROCEDURE AddPaidAccount
Create Procedure AddPaidAccount @AccountID DECIMAL(12),@AccountUsername VARCHAR(64),@EncryptedPassword VARCHAR(255),@Age DECIMAL(12,0),
@FirstName VARCHAR(255), @LastName VARCHAR(255), @Email VARCHAR, @RenewalDate DATE
AS
BEGIN
INSERT INTO Account(AccId,UserName,EncryptedPassword,Age,FirstName,LastName,Email,CreatedOn,AccountType)
VALUES (@AccountID ,@AccountUsername ,@EncryptedPassword ,@Age,
@FirstName , @LastName , @Email,GETDATE(),'P' )
INSERT INTO PaidAccount(AccId,AccountBalance,RenewalDate)
VALUES(@AccountId,100,GETDATE())
END;
GO
```

Messages

18:38:54 Started executing query at Line 38
Commands completed successfully.
Total execution time: 00:00:00.020

```
BEGIN TRANSACTION AddPaidAccount;
EXECUTE AddPaidAccount 2, 'jane_smith', 'js@pass456', 35, 'Jane', 'Smith', 'jane.smith@example.com', '2023-01-02' ;
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddPaidAccount;
EXECUTE AddPaidAccount 4, 'alice_brown', 'ab@pwd123', 25, 'Alice', 'Brown', 'alice.brown@example.com', '2023-01-04';
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddPaidAccount;
EXECUTE AddPaidAccount 6, 'susan_white', 'sw@pass123', 28, 'Susan', 'White', 'susan.white@example.com', '2023-01-06';
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddPaidAccount;
EXECUTE AddPaidAccount 8, 'linda_hill', 'lh@password', 31, 'Linda', 'Hill', 'linda.hill@example.com', '2023-01-08';
COMMIT TRANSACTION AddFreeAccount;

BEGIN TRANSACTION AddPaidAccount;
EXECUTE AddPaidAccount 10, 'sandra_clark', 'sc@pass456', 27, 'Sandra', 'Clark', 'sandra.clark@example.com', '2023-01-10';
COMMIT TRANSACTION AddFreeAccount;
```

Messages

18:39:31 Started executing query at Line 50
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
Total execution time: 00:00:00.030

```
SELECT * FROM Account
```

Results Messages

	AccId	UserName	EncryptedPassword	Age	FirstName	LastName	Email	CreatedOn	AccountType
1	1	pittbrad	password1	25	Brad	Pitt	b	2023-12-02	F
2	2	jane_smith	js@pass456	35	Jane	Smith	j	2023-12-02	P
3	3	alice_jones	password6	29	Alice	Jones	a	2023-12-02	F
4	4	alice_brown	ab@pwd123	25	Alice	Brown	a	2023-12-02	P
5	5	bob_miller	password7	31	Bob	Miller	b	2023-12-02	F
6	6	susan_white	sw@pass123	28	Susan	White	s	2023-12-02	P
7	7	emily_white	password8	26	Emily	White	e	2023-12-02	F
8	8	linda_hill	lh@password	31	Linda	Hill	l	2023-12-02	P
9	9	peter_miller	pm@pwd123	33	Peter	Miller	p	2023-12-02	F
10	10	sandra_clark	sc@pass456	27	Sandra	Clark	s	2023-12-02	P

2). I’m creating a transaction to enter the store name

```
74  ---#2
75  DROP PROCEDURE AddStore
76  CREATE PROCEDURE AddStore @StoreID DECIMAL(12),@StoreName VARCHAR(64)
77  AS
78  BEGIN
79  INSERT INTO Store(StoreId,StoreName)
80  VALUES (@StoreID,@StoreName)
81  END;
82  GO
83
84  BEGIN TRANSACTION AddStore
```

Messages

18:41:24 Started executing query at Line 76
Commands completed successfully.
Total execution time: 00:00:00.007

```
84  BEGIN TRANSACTION AddStore
85  EXECUTE AddStore 1, 'Spare&Spare';
86  COMMIT TRANSACTION AddStore;
87
88  BEGIN TRANSACTION AddStore
89  EXECUTE AddStore 2, 'AutoPartsHub';
90  COMMIT TRANSACTION AddStore;
91
92  BEGIN TRANSACTION AddStore
93  EXECUTE AddStore 3, 'GearGalore';
94  COMMIT TRANSACTION AddStore;
95
96  BEGIN TRANSACTION AddStore
97  EXECUTE AddStore 4, 'MegaMotors';
98  COMMIT TRANSACTION AddStore;
99
100 BEGIN TRANSACTION AddStore
101 EXECUTE AddStore 5, 'SpeedySolutions';
102 COMMIT TRANSACTION AddStore;
103
104 BEGIN TRANSACTION AddStore
105 EXECUTE AddStore 6, 'FastFix';
106 COMMIT TRANSACTION AddStore;
107
108 BEGIN TRANSACTION AddStore
109 EXECUTE AddStore 7, 'QuickCarCare';
110 COMMIT TRANSACTION AddStore;
111
112 BEGIN TRANSACTION AddStore
113 EXECUTE AddStore 8, 'RoadReady';
114 COMMIT TRANSACTION AddStore;
115
116 BEGIN TRANSACTION AddStore
117 EXECUTE AddStore 9, 'SwiftSpares';
118 COMMIT TRANSACTION AddStore;
119
120 BEGIN TRANSACTION AddStore
121 EXECUTE AddStore 10, 'RapidRepairs';
122 COMMIT TRANSACTION AddStore;
```

ssages

18:41:24 Started executing query at Line 76
Commands completed successfully.
Total execution time: 00:00:00.007

```
124
125 SELECT * FROM Store
126
```

Results Messages

	StoreId	StoreName
1	1	Spare&Spare
2	2	AutoPartsHub
3	3	GearGalore
4	4	MegaMotors
5	5	SpeedySolutions
6	6	FastFix
7	7	QuickCarCare
8	8	RoadReady
9	9	SwiftSpares
10	10	RapidRepairs

3). I have created a Transaction for Adding online purchase

```
DROP PROCEDURE AddOnlinePurchase
CREATE PROCEDURE AddOnlinePurchase @StoreId DECIMAL(12),@PurchaseId DECIMAL(12),@AccId DECIMAL(12),@TotalPurchasePrice DECIMAL(7,2),
@PurchaseURL VARCHAR(1024)
AS
BEGIN
    INSERT INTO Purchase(StoreId,PurchaseId,AccId,TotalPurchasePrice,PurchaseDate,PurchaseType)
    VALUES(@StoreId ,@PurchaseId ,@AccId ,@TotalPurchasePrice ,GETDATE(),'Online')
    INSERT INTO OnlinePurchase(PurchaseId,PurchaseURL)
    VALUES(@PurchaseId,@PurchaseURL)
END;
GO
```

Messages

18:45:50 Started executing query at Line 130
Commands completed successfully.
Total execution time: 00:00:00.018

```
BEGIN TRANSACTION AddOnlinePurchase;
EXECUTE AddOnlinePurchase 1,1,1,350,'spare&spare.com';
COMMIT TRANSACTION AddOnlinePurchase;

BEGIN TRANSACTION AddOnlinePurchase;
EXECUTE AddOnlinePurchase 3, 3, 5, 35, 'autopartshub.com';
COMMIT TRANSACTION AddOnlinePurchase;

BEGIN TRANSACTION AddOnlinePurchase;
EXECUTE AddOnlinePurchase 5, 5, 9, 45,'geargalore.com';
COMMIT TRANSACTION AddOnlinePurchase;

BEGIN TRANSACTION AddOnlinePurchase;
EXECUTE AddOnlinePurchase 7, 7, 2, 75,'megamotorsparts.com';
COMMIT TRANSACTION AddOnlinePurchase;

BEGIN TRANSACTION AddOnlinePurchase;
EXECUTE AddOnlinePurchase 9, 9, 4, 55,'speedysolutions.co';
COMMIT TRANSACTION AddOnlinePurchase;

BEGIN TRANSACTION AddOnlinePurchase;
EXECUTE AddOnlinePurchase 7, 2, 6, 75,'megamotorsparts.com';
COMMIT TRANSACTION AddOnlinePurchase;

BEGIN TRANSACTION AddOnlinePurchase;
EXECUTE AddOnlinePurchase 9, 6, 8, 55,'speedysolutions.co';
COMMIT TRANSACTION AddOnlinePurchase;
```

Messages

18:46:28 Started executing query at Line 141
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)

```
SELECT * FROM Purchase
SELECT * FROM OnlinePurchase
```

Results Messages

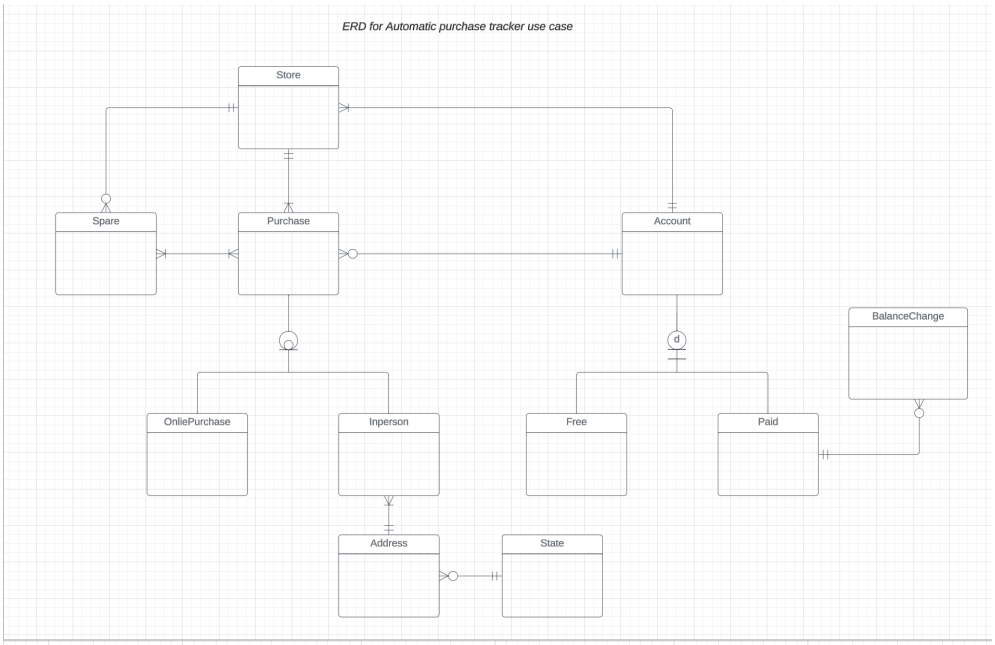
	StoreId	PurchaseId	AccId	TotalPurchasePrice	PurchaseDate	PurchaseType
1	1	1	1	350.00	2023-12-02	Online
2	7	2	6	75.00	2023-12-02	Online
3	3	3	5	35.00	2023-12-02	Online
4	5	5	9	45.00	2023-12-02	Online
5	9	6	8	55.00	2023-12-02	Online
6	7	7	2	75.00	2023-12-02	Online
7	9	9	4	55.00	2023-12-02	Online

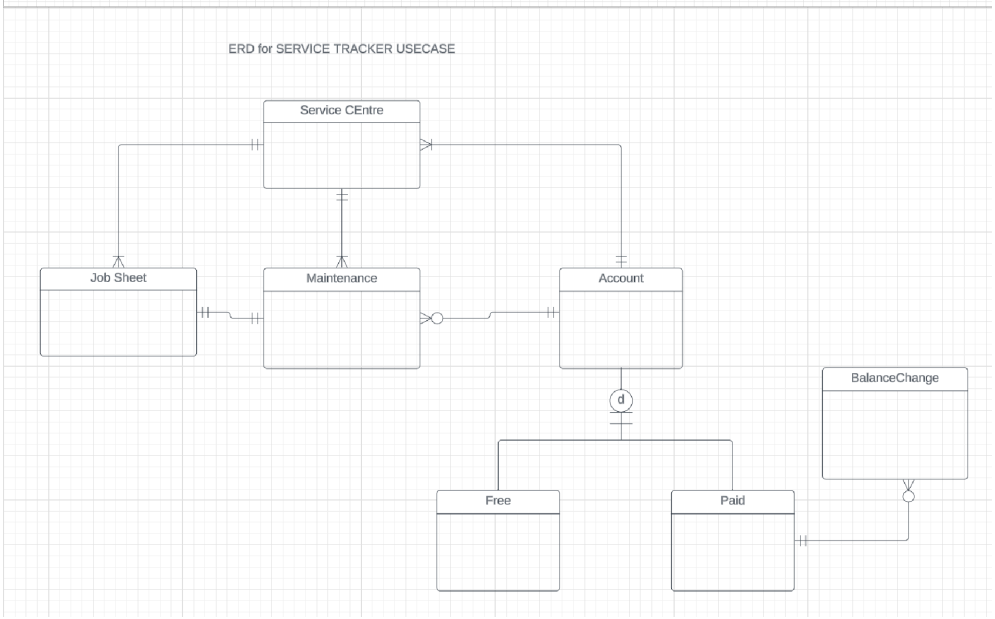
	PurchaseId	PurchaseURL
1	1	spare&spare.com
2	2	megamotorsparts.com
3	3	autopartshub.com
4	5	geargalore.com
5	6	speedysolutions.co
6	7	megamotorsparts.com
7	9	speedysolutions.co

Implementing History Table in my Database:

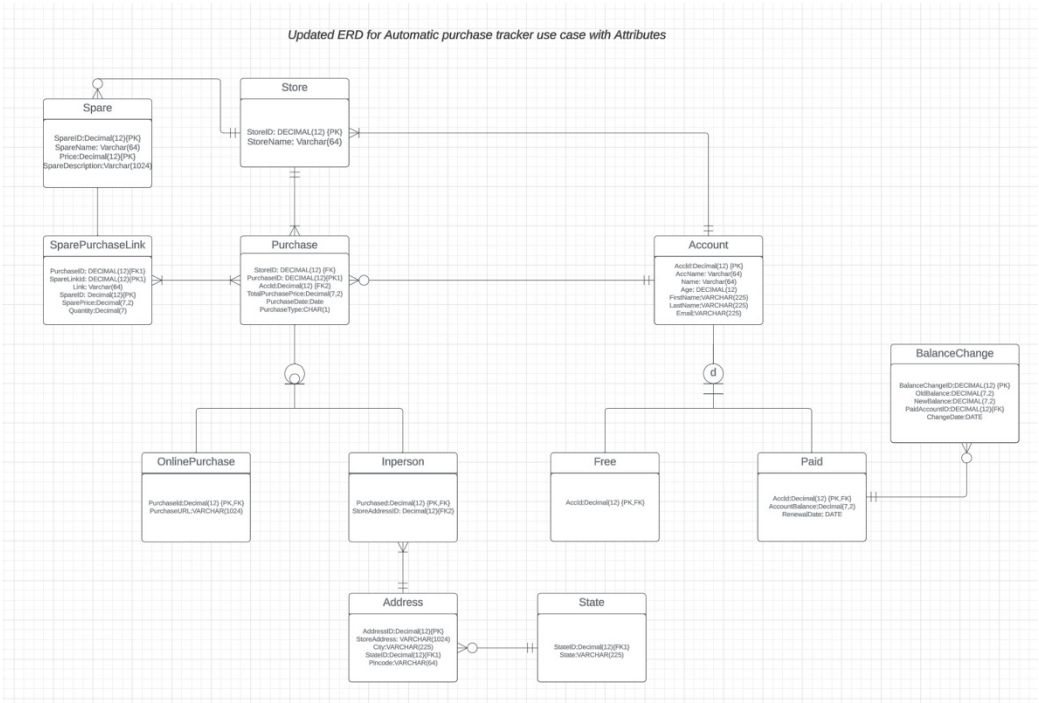
In reviewing my DBMS physical ERD, I found out that the historical record of a person'sbalance in the Paid Account table. Such a history would help keep a track of the account balances.

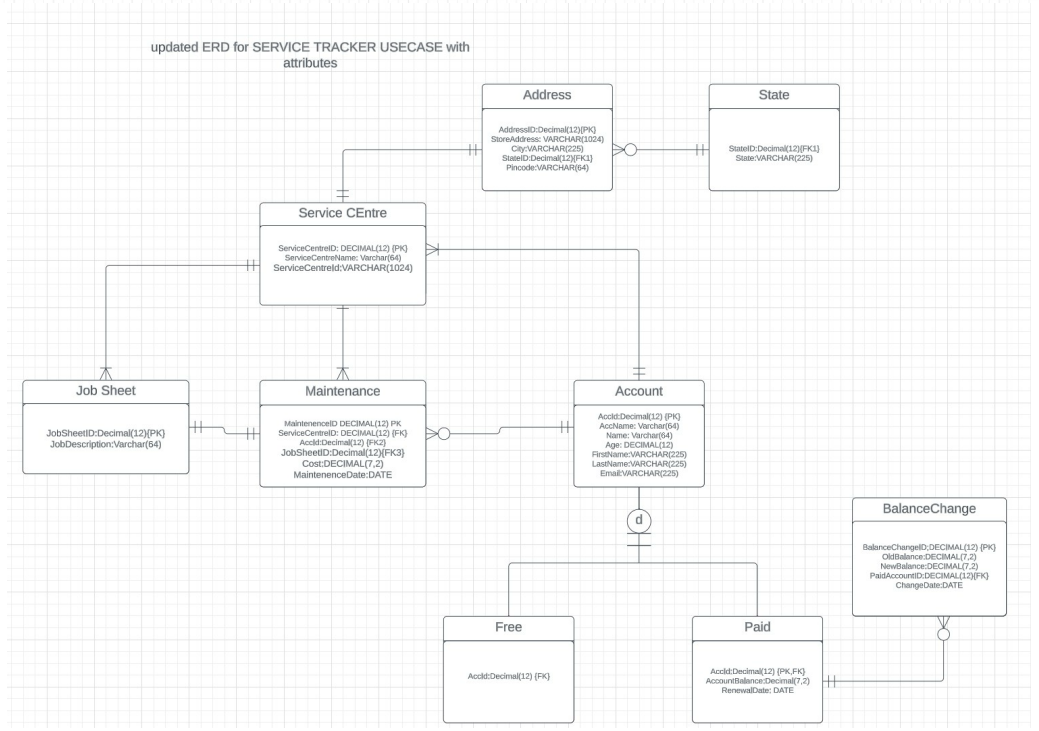
Hence, the Updated Conceptual ERD is below:





The Updated Physical ERD is Below:





Attribute	Description
BalanceChangeID	The primary key of the history table.It is a DECIMAL(12) to allow for many values
OldBalance	This is the balance of the account before the change.The datatype mirrors the balance datatype in the paid account table
NewBalance	This is the balance of the account aftwr the change.Datatype mirrors the balance datatype in the PaidAccount Table
PaidAccountID	This is a foreign key to the PaidAccount table,a ref to the acct ha had the change in balance
ChangeDate	This is the date the change occurred, with a DATE datatype

```
178  DROP TABLE BalanceChange
179  CREATE TABLE BalanceChange(
180      BalanceChangeID Decimal(12) NOT NULL PRIMARY KEY,
181      OldBalance Decimal(7,2) NOT NULL,
182      NewBalance DECIMAL(7,2) NOT NULL,
183      PaidAccoundId Decimal(12) NOT NULL,
184      ChangeDatE DATE NOT NULL,
185      FOREIGN KEY(PaidAccoundId) REFERENCES PaidAccount(AccId)
186  );
187
```

Messages

18:58:34 Started executing query at Line 179
Commands completed successfully.
Total execution time: 00:00:00.037

```
188  CREATE TRIGGER BalanceChangeTrigger
189  On PaidAccount
190  After UPDATE
191  AS
192  BEGIN
193  DECLARE @OldBalance DECIMAL(7,2) = (SELECT AccountBalance FROM DELETED)
194  DECLARE @NewBalance DECIMAL(7,2) = (SELECT AccountBalance FROM INSERTED)
195
196  IF (@OldBalance<> @NewBalance)
197  INSERT INTO BalanceChange(BalanceChangeID,OldBalance,NewBalance,PaidAccoundId,ChangeDatE)
198  VALUES(ISNULL((SELECT MAX(BalanceChangeID)+1 FROM BalanceChange),1),@OldBalance,@NewBalance,
199  (Select AccID FROM INSERTED),GETDATE());
200
201  END;
202
```

Messages

18:59:15 Started executing query at Line 188
Commands completed successfully.
Total execution time: 00:00:00.016

```
203
204  UPDATE PaidAccount
205  Set AccountBalance=40
206  WHERE AccId=4;
207
208  UPDATE PaidAccount
209  SET AccountBalance=25
210  WHERE AccId=2;
211
212  SELECT*FROM BalanceChange
213
214  SELECT*FROM PaidAccount
215
```

Results Messages

	BalanceChangeID	OldBalance	NewBalance	PaidAccoundId	ChangeDatE
1	1	100.00	40.00	4	2023-12-03
2	2	100.00	25.00	2	2023-12-03

	AccId	AccountBalance	RenewalDate
1	2	25.00	2023-12-02
2	4	40.00	2023-12-02
3	6	100.00	2023-12-02
4	8	100.00	2023-12-02
5	10	100.00	2023-12-02

Creating Questions and Queries:

```sql
--Qn1
--How many paid accounts created ,now have a balance between $10-$50 and $50-$100
SELECT CASE
        WHEN PaidAccount.AccountBalance>=10 AND PaidAccount.AccountBalance<50 THEN '$10-$50'
        WHEN PaidAccount.AccountBalance>=50 AND PaidAccount.AccountBalance<=100 THEN'$50-$100'
        ELSE '$100'
    END AS Category,
    COUNT(*) as NumberWithBalance
FROM Account
JOIN PaidAccount ON PaidAccount.AccId=Account.AccId
AND PaidAccount.AccountBalance>=10
GROUP BY CASE
    WHEN PaidAccount.AccountBalance>=10 AND PaidAccount.AccountBalance<50 THEN '$10-$50'
        WHEN PaidAccount.AccountBalance>=50 AND PaidAccount.AccountBalance<=100 THEN'$50-$100'
        ELSE '$100'
    END
```

Results | Messages

	Category	NumberWithBalance
1	$10-$50	2
2	$50-$100	3

```sql
-----
--QN2
-- Retrieve the total purchase amount made by each account, ordered by the highest total purchase amount.
SELECT
    Account.AccId,
    Account.UserName,
    SUM(Purchase.TotalPurchasePrice) AS TotalPurchaseAmount
FROM
    Account
JOIN
    Purchase ON Account.AccId = Purchase.AccId
GROUP BY
    Account.AccId, Account.UserName
ORDER BY
    TotalPurchaseAmount DESC;
```

Results | Messages

	AccId	UserName	TotalPurchaseAmount
1	1	pittbrad	350.00
2	2	jane_smith	75.00
3	6	susan_white	75.00
4	8	linda_hill	55.00
5	4	alice_brown	55.00
6	9	peter_miller	45.00
7	5	bob_miller	35.00

```sql
--Qn 3
--Retrieve the details of online purchases, including the account information and the total purchase price, for accounts with a balance greater than $50.
SELECT
    Account.AccId,
    Account.UserName,
    OnlinePurchase.PurchaseURL,
    Purchase.TotalPurchasePrice
FROM
    Account
JOIN
    PaidAccount ON Account.AccId = PaidAccount.AccId
JOIN
    Purchase ON Account.AccId = Purchase.AccId
JOIN
    OnlinePurchase ON Purchase.PurchaseId = OnlinePurchase.PurchaseId
WHERE
    PaidAccount.AccountBalance > 50;
```

Results | Messages

	AccId	UserName	PurchaseURL	TotalPurchasePrice
1	6	susan_white	megamotorsparts.com	75.00
2	8	linda_hill	speedysolutions.co	55.00

Summary And Reflections:

The database that I took is for a portal “Maintenance Tracker” which records and store all spare parts purchase and servicing done to the cars of a person. When a person purchases aspare for a car and find that the spare is not good or fails in about a week or two, he or she can be able to view the purchase history of this quite easily. The same applies for servicing also if the consumer feels that the servicing done by one mechanic last time wasn’t better, heor she would be able to pull the data like the service summary and see if whether it was the service that was done last time was the problem or whether it is something else

The Structural database rules and conceptual ERD for my database design contain the important entities as well as relationships between them. The same the physical erd does butit contains the keys such as primary and foreign keys.

The SQL script that contains all table creation that follow the specification from the DBMS physical ERD exactly. Important indexes have been created to help speed up access to my database and are also available in an index script. Stored Procedures have been created and executed transactionally to populate some of my database with data. Some qns related to mydatabase has been identified and implemented with SQL queries. A history balance table hasbeen created as well as a query which tracks changes to balance for a specific month.

As I have a look at the database now and my accomplishments, I can say its been a long process and I can see how the database has been developed step by step. I can still see there is more to develop in my database, but feel it’s a solid foundation to move forward with.