GTU Department of Computer Engineering
CSE414 Databases - Spring 2022
Project Report

Akif KARTAL
171044098

1) Problem Definition

The problem is to make a database management system for the **yemeksepeti** platform.

2) Solution

The project was **finished as expected** in project assignment announcement.

2.1) User Requirements

In this project we 3 users. These are customer, restaurant owner and admin of system.

1. **Customer** shall be able to sign up.
2. **Customer** shall be able to add food to his/her basket.
3. **Customer** shall be able to order food.
4. **Customer** shall be able to see status of the order and old orders.
5. **Customer** shall be able to add new address or edit them.
6. **Customer** shall be able to see all foods and restaurants.
7. **Customer** shall be able to add restaurant in his/her favorites.
8. **Customer** shall be able to load money on the digital wallet.
9. **Customer** shall be able to make review to order.
10. **Customer** shall be able to see and use his/her discount coupons.
11. **Restaurant Owner** shall be able to add new food on his/her menu or edit them.
12. **Restaurant Owner** shall be able to add new payment type for his/her restaurant.
13. **Restaurant Owner** shall be able to change working hours of restaurant.
14. **Restaurant Owner** shall be able to add new campaign for foods.
15. **Admin** shall be able to add new restaurants or edit them.
16. **Admin** shall be able to add new discount coupon to the customers
17. **Admin** shall be able to see all information in the system.

2.2) Database Tables and Relations

In this project we have **25 table**. These tables are followings.

- Customer
- Address
- AddressType
- CustomerAdress
- TelephoneNumber
- Email
- Orders
- FavouriteRestaurant
- Basket

- BasketFood
- OrderFood
- DigitalWallet
- Restaurant
- PaymentType
- PaymentRestaurant
- WorkingHours
- Food
- Category
- Review
- Campaign
- Coupon
- CouponCustomer
- FoodLog
- WalletLog
- Offer

Relations

Between these tables we have following relations;

- **4** one to one relation
- **20** one to many relation
- **6** many to many relation

Total we have **30** relations. You can see in the E-R diagram.

Creating Tables and Inserting Data

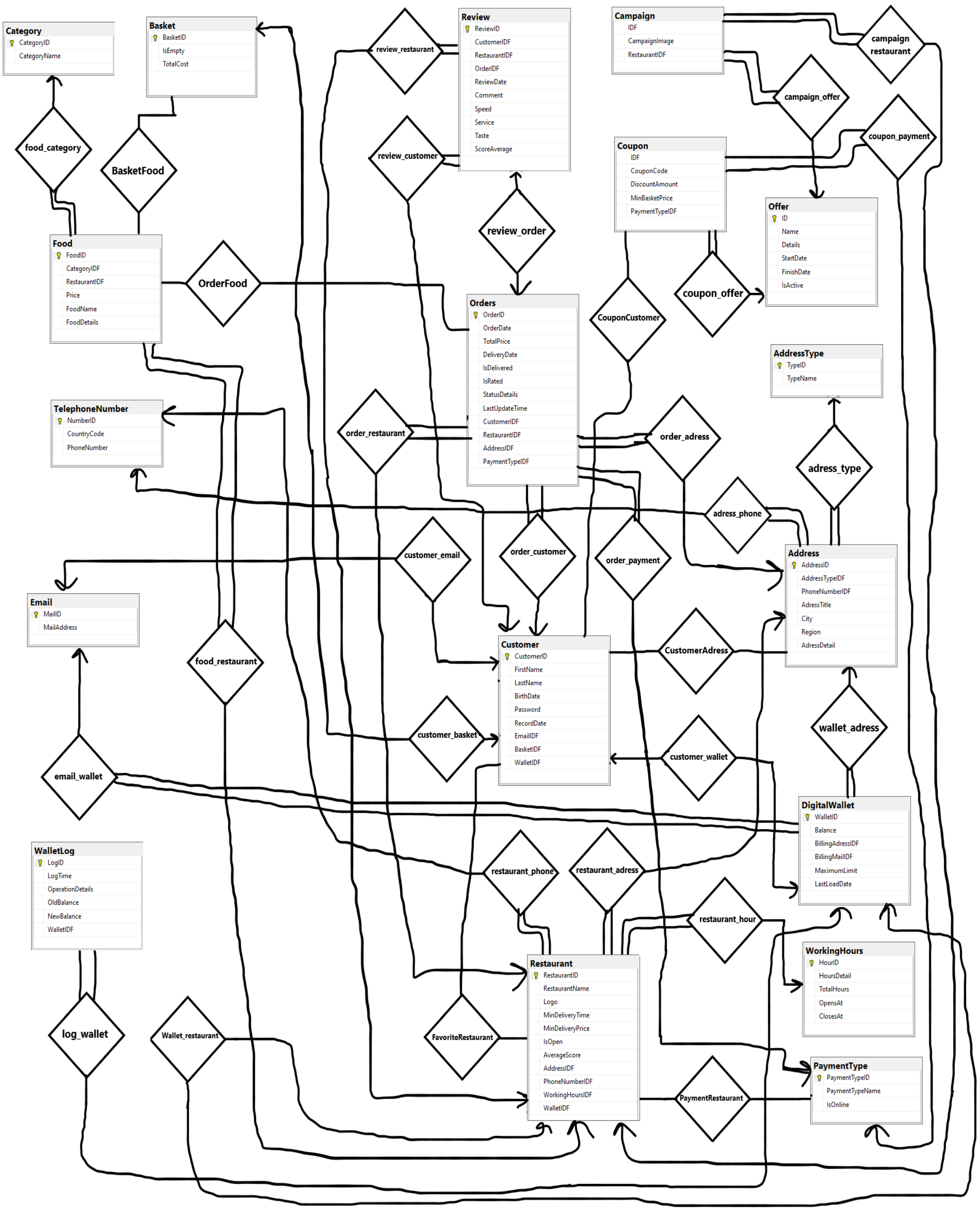
In this database everything made with SQL language. You can see in following images.

```
Create Table WalletLog (
    LogID int identity (1, 1) NOT NULL,
    LogTime datetime NULL,
    OperationDetails nvarchar(MAX) NULL ,
    OldBalance money NULL,
    NewBalance money NULL,
    WalletIDF int NULL,

    CONSTRAINT PK_WalletLog PRIMARY KEY CLUSTERED
    (
        LogID
    ),
    CONSTRAINT FK_Wallet_WalletLog FOREIGN KEY
    (
        WalletIDF
    ) REFERENCES dbo.DigitalWallet (
        WalletID
    )
)
```

```
--add phone number
insert TelephoneNumber Values('90','5555555555'),
('90','5554443333'),
('90','5437776669'),
('90','2164445555'),
('90','2163335555')
```

2.3) E-R Diagram



2.4) Normalization

As you can see from the tables I have applied normalization such that **tables are well separated** and **no data is repeated**. Also, there are no partial and transitive dependency. Lastly all my primary keys are candidate key. Therefore, **in this project I have 3NF and BCNF normalization.** For example, we have **WorkingHours** table for restaurant. This is example of 3NF and BCNF normalization.

2.5) Functional Dependencies

Customer

CustomerID ⟶ FirstName, LastName, Password, BirthDate, EmailIDF, BasketIDF, WalletIDF, RecordDate

Address

AddressID ⟶ PhoneNumberIDF, AddressTypeIDF, AdressTitle, City, Region, AdressDetail

AddressType

TypeID ⟶ TypeName

TelephoneNumber

NumberID ⟶ CountryCode, PhoneNumber

Email

MailID ⟶ MailAddress

Orders

OrderID ⟶ CustomerIDF, RestaurantIDF, AddressIDF, PaymentTypeIDF, StatusDetail, LastUpdateTime, OrderDate, TotalPrice, IsDelivered, IsRated, DeliveryDate

Basket

BasketID ⟶ IsEmpty, TotalCost

DigitalWallet

WalletID ⟶ Balance, BillingAdressIDF, BillingMailIDF, MaximumLimit, LastLoadDate

Restaurant

RestaurantID ⟶ RestaurantName, Logo, AverageScore, MinDeliveryTime, MinDeliveryPrice, IsOpen, WorkingHoursIDF, AddressIDF, PhoneNumberIDF

PaymentType

PaymentTypeID ⟶ PaymentTypeName, IsOnline

WorkingHours

HourID ⟶ HoursDetail, TotalHour, OpensAt, ClosesAt

Food

FoodID ⟶ FoodName, FoodDetail, Price, CategoryIDF, RestaurantIDF

Category

CategoryID ⟶ CategoryName

Review

ReviewID ⟶ CustomerIDF, RestaurantIDF, Comment, ReviewDate, Speed, Service, Taste, ScoreAverage, OrderIDF

Campaign

IDF ⟶ CampaignImage, RestaurantID

Coupon

IDF ⟶ CouponCode, DiscountAmount, MinBasketPrice, PaymentTypeIDF

FoodLog

LogID ⟶ DeletedTime, FoodName, FoodDetail, Price, RestaurantName

WalletLog

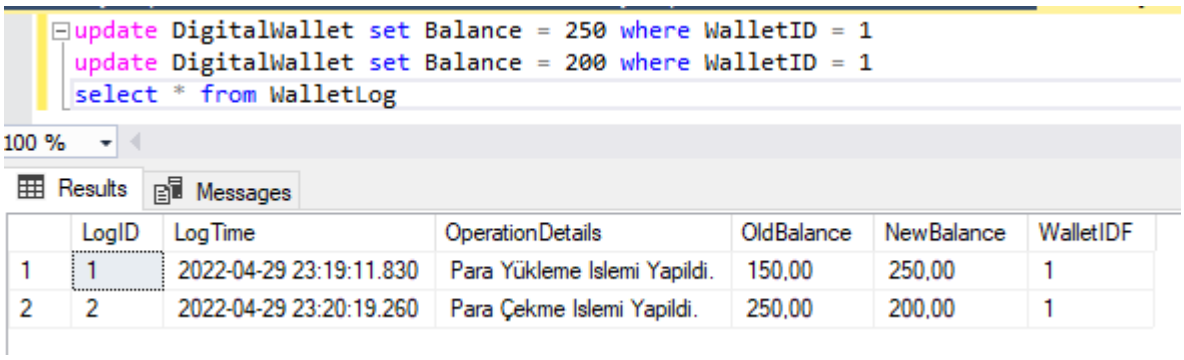
LogID ⟶ LogTime, OperationDetails, OldBalance, NewBalance, WalletIDF

Offer

ID ⟶ Name, Details, StartDate, FinishDate, IsActive

2.6) Triggers

1. **trg_LogWallet :** By using this trigger we will **insert a new log** to WalletLog after updating DigitalWallet. In this way we can see the user operations in any problem. You can see the changes in WalletLog table in following image.



```
update DigitalWallet set Balance = 250 where WalletID = 1
update DigitalWallet set Balance = 200 where WalletID = 1
select * from WalletLog
```

100 %

Results | Messages

	LogID	LogTime	OperationDetails	OldBalance	NewBalance	WalletIDF
1	1	2022-04-29 23:19:11.830	Para Yükleme Islemi Yapildi.	150,00	250,00	1
2	2	2022-04-29 23:20:19.260	Para Çekme Islemi Yapildi.	250,00	200,00	1

2. **trg_ReviewAverage:** By using this trigger after a new review(speed,service,taste) made by the customer **we can update restaurant average score**. You can see the changes in Restaurant table in following image.

```
select * from Restaurant where RestaurantID = 2
Go
insert Review values(2,2,2,GETDATE(),'Teşekkürler. Uzun zaman sonra eksiksiz bir sipariş geldi.',8,9,9)
Go
select * from Restaurant where RestaurantID = 2
```

100 %

Results | Messages

	RestaurantID	RestaurantName	Logo	MinDeliveryTime	MinDeliveryPrice	IsOpen	AverageScore	AddressIDF	PhoneNumberIDF	WorkingHoursIDF
1	2	Dominos	https://dominos.gif	40	60,00	1	5.00	4	4	2

	RestaurantID	RestaurantName	Logo	MinDeliveryTime	MinDeliveryPrice	IsOpen	AverageScore	AddressIDF	PhoneNumberIDF	WorkingHoursIDF
1	2	Dominos	https://dominos.gif	40	60,00	1	6.83	4	4	2

3. **trg_BasketPriceUpdate:** By using this trigger **we will update basket total price** after a food added to the basket. You can see the changes in Basket table in following image.

```
select * from Basket where BasketID = 1
Go
insert BasketFood(BasketIDF,FoodIDF,Quantity,Price) Values(1,2,1,80)
Go
insert BasketFood(BasketIDF,FoodIDF,Quantity,Price) Values(1,1,1,60)
Go
select * from Basket where BasketID = 1
```

100 %

Results | Messages

	BasketID	IsEmpty	TotalCost
1	1	1	0,00

	BasketID	IsEmpty	TotalCost
1	1	0	140,00

4. **trg_LogFood:** By using this trigger, **after deleting a food we can insert information** about that food in FoodLog table to check later. You can see the changes in FoodLog table in following image.

```
select * from Food where FoodID = 4
Go
select * from FoodLog
Go
delete from Food where FoodID = 4
Go
select * from FoodLog
```

100 %

Results | Messages

	FoodID	CategoryIDF	RestaurantIDF	Price	FoodName	FoodDetails
1	4	2	1	60,00	Chicken Royale Menü	Chicken Royale + Büyük Boy Patates + Kutu Içecek

	LogID	DeletedTime	FoodName	FoodDetail	Price	RestaurantName

	LogID	DeletedTime	FoodName	FoodDetail	Price	RestaurantName
1	1	2022-04-29 23:54:23.363	Chicken Royale Menü	Chicken Royale + Büyük Boy Patates + Kutu Içecek	60,00	Burger King

5. **trg_OrderFood :** By using this trigger, when a **customer give the order(insert),** the foods in the **basket will be removed and they will added to order table**. You can see the changes in Basket table in following image.

```
select * from Basket where BasketID = 1
Go
insert Orders Values(GETDATE(),140,NULL,0,'Hazırlanıyor',GETDATE(),1,1,1,1)
Go
select * from Basket where BasketID = 1
```

100 %

Results | Messages

	BasketID	IsEmpty	TotalCost
1	1	0	140,00

	BasketID	IsEmpty	TotalCost
1	1	1	0,00

2.7) Views

1. vm_Customer: By using this view **we can see the customer information detailed way(joined).** Normally customer information separated into different tables. With this view we will join them. See the results in following image.

```
-- original customer table vs customer view
select * from Customer
Go
select * from vm_Customer   --password was hidden form user
```

100 %

Results | Messages

	CustomerID	FirstName	LastName	BirthDate	Password	RecordDate	EmailIDF	BasketIDF	WalletIDF
1	1	Akif	Kartal	1997-07-25	test123	2022-04-28 23:09:42.180	1	1	1
2	2	Ayse	Çinar	1992-09-13	test234	2022-04-28 23:09:42.180	2	2	2
3	3	Deniz	Kaya	1985-11-11	test356	2022-04-28 23:09:42.180	3	3	3

	Adı	Soyadı	DoğumTarihi	MailAdresi	SepetTutarı	CüzdanMaximumLimit	KayıtTarihi	SiparişSayısı	ToplamYorumSayısı	KuponSayısı	FavoriRestoranSayısı
1	Akif	Kartal	1997	test1@gmail.com	0,00	3000,00	2022-04-28 23:09:42.180	2	1	0	2
2	Ayse	Çinar	1992	test2@gmail.com	0,00	3000,00	2022-04-28 23:09:42.180	1	2	0	2
3	Deniz	Kaya	1985	test3@gmail.com	0,00	3000,00	2022-04-28 23:09:42.180	1	1	0	2

2. vm_Order : By using this view we can see the order information detailed way. Normally order information separated into different tables. With this view we will join them and add additional information. See the results in following image.

```
-- original order table vs order view
select * from Orders
Go
select * from vm_Order
```

100 %

Results | Messages

	OrderID	OrderDate	TotalPrice	DeliveryDate	IsDelivered	IsRated	StatusDetails	LastUpdateTime	CustomerIDF	RestaurantIDF	AddressIDF	PaymentTypeIDF
1	1	2022-03-19 18:47:39.370	80,00	NULL	0	1	Hazirlaniyor	2022-03-19 18:50:39.370	1	1	1	1
2	2	2022-04-19 13:55:39.370	60,00	2022-04-19 14:15:39.370	1	1	Teslim Edildi	2022-03-19 18:50:39.370	2	2	2	2
3	3	2022-05-19 14:35:39.370	120,00	2022-05-19 14:55:39.370	1	1	Teslim Edildi	2022-03-19 18:50:39.370	3	3	3	3
4	4	2022-04-30 00:10:40.550	140,00	NULL	0	0	Hazirlaniyor	2022-04-30 00:10:40.550	1	1	1	1

	SiparişNumarası	SiparişTarihi	Fiyat	YemekAdedi	SiparişiVerenKişi	SiparişAdresi	RestorantAdı	ÖdemeTipi	TeslimEdildimi	SonBilgiGirişZamanı	SiparişYorumu
1	1	2022-03-19 18:47:39.370	80,00	1	Akif Kartal	Yunus Emre Mah. Malazgirt Cd. No:31 D:3 Sancakte...	Burger King	MetropolCard	0	2022-03-19 18:50:39.370	Burger hastasiyim köt
2	2	2022-04-19 13:55:39.370	60,00	1	Ayse Çinar	Cumhuriyet, 2254. Sk. No:2, 41400 Gebze/Kocaeli	Dominos	Online Credit/Debit Card	1	2022-03-19 18:50:39.370	Tesekkürler. Uzun za
3	3	2022-05-19 14:35:39.370	120,00	1	Deniz Kaya	Test mah. test cad no.1 ankara/çankaya	KFC	Multinet	1	2022-03-19 18:50:39.370	Burger hastasiyim köt
4	4	2022-04-30 00:10:40.550	140,00	2	Akif Kartal	Yunus Emre Mah. Malazgirt Cd. No:31 D:3 Sancakte...	Burger King	MetropolCard	0	2022-04-30 00:10:40.550	NULL

3. vm_Max40Food: By using this view we can see the foods whose price less than 40 TL. See the results in following image.

```
-- original food table vs max40food view
select * from Food
Go
select * from vm_Max40Food
```

100 %

Results | Messages

	FoodID	CategoryIDF	RestaurantIDF	Price	FoodName	FoodDetails
1	1	2	1	60,00	Whopper Menü	Whopper Menü + Büyük Boy Patates + Kutu Içecek
2	2	3	2	80,00	Seçilmis Menü (Orta Boy Pizza)	Orta Boy Pizza + Seçeceginiz Yan Ürün + Coca-Col...
3	3	2	3	60,00	KFC TAVUK BURGER KUTU	KFC Tavuk Burger, 2 Kanat, 4 Hot Shots, Orta Boy...
4	5	11	1	20,00	Patates Kizartması	Büyük Boy Patates Kizartması

	YemekAdı	YemekDetayı	Fiyatı	YemekTürü	RestoranAdı	ServisSüresi
1	Patates Kizartması	Büyük Boy Patates Kizartması	20,00	Ekstra Yiyecek	Burger King	30 dk.

4. vm_Restaurant: By using this view we can see the restaurant information detailed way. Normally restaurant information separated into different tables. With this view we will join them and add additional information. See the results in following image.

```
-- original restaurant table vs restaurant view
select * from Restaurant
Go
select * from vm_Restaurant
```

100 %

Results | Messages

	RestaurantID	RestaurantName	Logo	MinDeliveryTime	MinDeliveryPrice	IsOpen	AverageScore	AddressIDF	PhoneNumberIDF	WorkingHoursIDF
1	1	Burger King	https://burgerking.gif	30	50,00	1	9.33	4	4	1
2	2	Dominos	https://dominos.gif	40	60,00	1	6.83	4	4	2
3	3	KFC	https://kfc.gif	50	40,00	1	8.00	4	4	3

	RestoranAdı	RestoranPuanı	ServisSüresi	MinTutar	RestoranAçıkmı	ÇalışmaSaatleri	TelefonNumarası	Adresi	ToplamYemekSayısı	ToplamYorumSayısı	FavoriSeçilmeSayısı	ToplamÖdeme
1	Burger King	9.33	30 dk.	50,00	1	08:00 - 17:00 arasi	90 2164445555	Cumhuriyet Mah. Sile Otoyolu Adnan Menderes Cad. ...	2	1	2	1
2	Dominos	6.83	40 dk.	60,00	1	09:00 - 23:00 arasi	90 2164445555	Cumhuriyet Mah. Sile Otoyolu Adnan Menderes Cad. ...	1	1	3	3
3	KFC	8.00	50 dk.	40,00	1	11:00 - 00:00 arasi	90 2164445555	Cumhuriyet Mah. Sile Otoyolu Adnan Menderes Cad. ...	1	1	1	2

5. vm_Comment: By using this view we can see the review information detailed way. Normally review information separated into different tables. With this view we will join them and add additional information. See the results in following image.

```
-- original review table vs review view
select * from Review
Go
select * from vm_Comment
```

100 %

Results | Messages

	ReviewID	CustomerIDF	RestaurantIDF	OrderIDF	ReviewDate	Comment	Speed	Service	Taste	ScoreAverage
1	1	1	1	1	2022-03-19 18:47:39.370	Burger hastasiyim kötü yapani görmedim. Piyasada...	9	9	10	9.33
2	2	2	2	2	2022-03-19 18:42:39.370	Tesekkürler. Uzun zaman sonra eksiksiz bir siparis...	5	5	5	5.00
3	3	3	3	3	2022-04-11 15:49:39.370	Burger hastasiyim kötü yapani görmedim. Piyasada...	8	8	8	8.00

	Yorum	YorumuYapanKişi	YorumYapılanRestoran	YorumTarihi	HızPuanı	ServisPuanı	LezzetPuanı	OrtalamaPuan	SiparişTarihi
1	Burger hastasiyim kötü yapani görmedim. Piyasada...	Akif Kartal	Burger King	2022-03-19 18:47:39.370	9	9	10	9.33	2022-03-19 18:47:39.370
2	Tesekkürler. Uzun zaman sonra eksiksiz bir siparis...	Ayse Çinar	Dominos	2022-03-19 18:42:39.370	5	5	5	5.00	2022-04-19 13:55:39.370
3	Burger hastasiyim kötü yapani görmedim. Piyasada...	Deniz Kaya	KFC	2022-04-11 15:49:39.370	8	8	8	8.00	2022-05-19 14:35:39.370

2.8) Left, Right and Full Outer Join

Since Customer and Email tables are relational tables, we can join them by using following queries.

- `select * from Customer c Left Outer Join Email e on c.EmailIDF = e.MailID`
- `select * from Customer c Right Outer Join Email e on c.EmailIDF = e.MailID`
- `select * from Customer c Full Outer Join Email e on c.EmailIDF = e.MailID`

2.9) Transactions

1. OrderPayment

When a customer gives an order, if the customer's balance is enough to give the order we transfer the order price from the customer's digital wallet to the restaurant's digital wallet(account).

```
If @amount > @senderBalance
Begin
    print 'Bakiye Yetersiz!'
    RollBack
End
Else
Begin
    Update DigitalWallet set Balance = Balance - @amount where WalletID = @senderWalletId
    Update DigitalWallet set Balance = Balance + @amount, LastLoadDate = GETDATE() where WalletID = @receiverWalletId
    Commit
End
```

2. insertCustomer

While inserting a customer if the given email, wallet, or basket is already used we will not insert the customer, otherwise customer will be inserted.

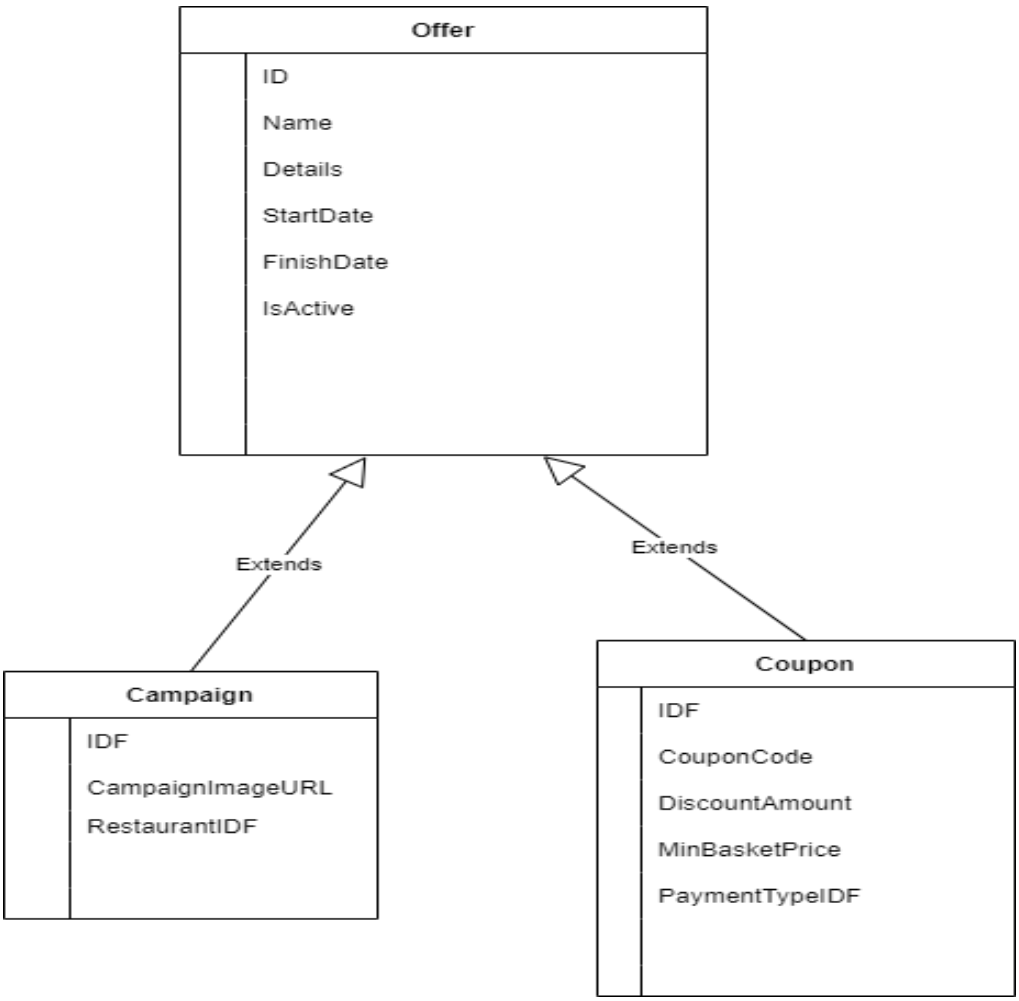
3. insertReview

A new review cannot be inserted for an order that has been reviewed before. Otherwise, it will be added.

```
If @tempId > 0
Begin
    print 'Daha önce review yapılmış bir sipariş için tekrar review yapılamaz!'
    RollBack
End
Else
```

2.10) Inheritance (Specialization)

Between Campaign and Coupon tables there is an inheritance because both are related to discount, and they have common attributes. Therefore, they inherited from Offer table.



As you can see Name, Details, StartDate, FinishDate and IsActive attributes are **common** between Campaign and Coupon tables and IDF's are foreign key. We will access the information by using this foreign key.

2.11 Extra Details

1. Using Store Procedure

In order to implement transaction and insertion operation efficient way**, I have used store procedures**. You can see implementation and usage in following images.

```
create procedure sp_insertCustomerTransaction
(
    @FirstName nvarchar (50),
    @LastName nvarchar (50),
    @BirthDate date NULL ,
    @Password nvarchar (MAX) NULL ,
    @EmailIDF int NULL ,
    @BasketIDF int NULL ,
    @WalletIDF int NULL
)
as
Begin Transaction insertCustomer
declare @tempWalletId int = 0 , @tempEmailId int = 0, @tempBasketId int = 0
Select @tempWalletId = WalletIDF from Customer where WalletIDF = @WalletIDF
Select @tempWalletId = WalletIDF from Restaurant where WalletIDF = @WalletIDF
Select @tempBasketId = BasketIDF from Customer where BasketIDF = @BasketIDF
Select @tempEmailId = EmailIDF from Customer where EmailIDF = @EmailIDF
If @tempBasketId > 0 or @tempEmailId > 0 or @tempWalletId > 0
Begin
    print 'Daha önce kullanılmış email, hesap cüzdanı ve sepet bilgileri ile yeni müşteri oluşturulamaz!'
    RollBack
End
Else
Begin
    insert Customer Values(@FirstName,@LastName,@BirthDate,@Password,getdate(),@EmailIDF,@BasketIDF,@WalletIDF)
    Commit
End
```

```
exec sp_insertCustomerTransaction 'Ahmet','Yılmaz','1998-07-27','test1',5,5,10
```

Results | Messages

	CustomerID	FirstName	LastName	BirthDate	Password	RecordDate	EmailIDF	BasketIDF	WalletIDF
1	1	Akif	Kartal	1997-07-25	test123	2022-05-29 14:40:49.623	1	1	1
2	2	Ayse	Çınar	1992-09-13	test234	2022-05-29 14:40:49.623	2	2	2
3	3	Deniz	Kaya	1985-11-11	test356	2022-05-29 14:40:49.623	3	3	3
4	4	Selin	Özdemir	1999-07-27	test1234	2022-05-29 17:46:46.793	4	4	7
5	5	Ahmet	Yilmaz	1998-07-27	test1	2022-05-30 01:09:37.477	5	5	10

2. Using Inner Joins

In views, I have used inner joins you can see in following image.

```
-- detaylı siparis bilgisi
Create View vm_Order
as
Select o.OrderID as SiparişNumarası,o.OrderDate as SiparişTarihi,o.TotalPrice as Fiyat,
(Select Count(*) from OrderFood o_f where o_f.OrderIDF = o.OrderID) as YemekAdedi,
c.FirstName+' '+c.LastName as SiparişiVerenKişi,a.AdressDetail as SiparişAdresi,
r.RestaurantName as RestorantAdı,p.PaymentTypeName as ÖdemeTipi,o.IsDelivered as TeslimEdildimi,
o.LastUpdateTime as SonBilgiGirişZamanı, (CASE
        WHEN o.IsRated is null THEN 'Yorum Yapılmadı!'
        ELSE (select Comment from Review s where s.OrderIDF = o.OrderID) end) as SiparişYorumu
from Orders o
inner join Customer c on c.CustomerID = o.CustomerIDF
inner join Restaurant r on r.RestaurantID = o.RestaurantIDF
inner join Address a on a.AddressID= o.AddressIDF
inner join PaymentType p on p.PaymentTypeID = o.PaymentTypeIDF
```

3. Using Check Constraint in Create Table query

In Create Table step**, I have used check constraint.** You can see in following image.

```
Create Table Customer (
    CustomerID int identity (1, 1) NOT NULL,
    FirstName nvarchar (50),
    LastName nvarchar (50),
    BirthDate date NULL ,
    Password nvarchar (MAX) NULL ,
    RecordDate datetime NULL ,
    EmailIDF int NULL ,
    BasketIDF int NULL ,
    WalletIDF int NULL ,
    CONSTRAINT PK_Customer PRIMARY KEY CLUSTERED
    (
        CustomerID
    ),
    CONSTRAINT FK_Customers_Emails FOREIGN KEY
    (
        EmailIDF
    ) REFERENCES dbo.Email (
        MailID
    ),
    CONSTRAINT FK_Customers_Wallets FOREIGN KEY
    (
        WalletIDF
    ) REFERENCES dbo.DigitalWallet (
        WalletID
    ),
    CONSTRAINT FK_Customers_Baskets FOREIGN KEY
    (
        BasketIDF
    ) REFERENCES dbo.Basket (
        BasketID
    ),
    CONSTRAINT CK_Birthdate CHECK (BirthDate < getdate())
)
Go
```

4. Using Case-When-Then, Declare and Automatic Attribute

In triggers and create table queries I have used SQL language case-when-then, declare and cast operation. You can see in following images.

```
-- wallet log triggeri
Create Trigger trg_LogWallet
on DigitalWallet
after update
as
declare @newBalance money,@oldBalance money, @walletID int
Select @newBalance = Balance,@walletID = WalletID from inserted
Select @oldBalance = Balance from deleted
declare @operation nvarchar(max) = (CASE
        WHEN @newBalance < @oldBalance THEN 'Para Çekme İşlemi Yapıldı.'
        WHEN @newBalance > @oldBalance THEN 'Para Yükleme İşlemi Yapıldı.'
        ELSE 'Bakiye Değişmedi' end)
Insert WalletLog values(GETDATE(),@operation,@oldBalance,@newBalance,@walletID)
```

```
Create Table Review (
    ReviewID int identity (1, 1) NOT NULL,
    CustomerIDF int,
    RestaurantIDF int,
    OrderIDF int,
    ReviewDate datetime,
    Comment nvarchar (MAX),
    Speed float,
    Service float,
    Taste float,
    ScoreAverage as CAST((Speed + Service +Taste) / 3 AS DECIMAL(10,2)),
```

2.12 User Interface (GUI)