

## MARKET OTOMASYONU

İlk haftamızda, yapmak istediğimiz market veri tabanı projemiz için, olması gereken verileri topladık. Örnek bir market firması ile görüşüp ne tarzda tablolar çıkabilir onları inceledik. Daha sonra topladığımız verileri filtreleyerek bize uygun verilerimizi oluşturduk, daha sonra verileri uygun bir şekilde ilişkilendirip taslak bir tablolar oluşturduk.

Bir müşteri gözünden projeye bakmak gerekirse; müşterinin indirim ve promosyonlar için kullanabileceği bir kartı olacak. Kullanıcı bizim arayüzümüz ile sisteme giriş yaptığında, tablolarımızda o müşterinin ID'sini sorgulayarak kart bilgilerini ve içeriğini görebilecek. Her alışveriş yaptığında ise yaptığı alışveriş ID'sine kayıt olacak. Böylece alışveriş geçmişini de görebilecek. Müşteriler tablosunun bağlı olduğu Müşteri Kart Tablosunda her bir müşteriye ait ID, Müşteriye özel indirimler, bakiyesi ve en son yaptığı alışveriş bilgileri bulunmaktadır. Eğer müşteriye özel indirim varsa bunun sorgusu yapılır ve uygulanır. Aynı şekilde personeline kendine özel Personel Kartı bulunmaktadır. Personel Kartı ise firmanın personellere özel tanımladığı indirimler ve yemek ücreti gibi bilgileri tutar. Personel Kart tablosunda personelin ID'si, indirim bilgileri ve firmanın verdiği yemek ücret bakiyesi bulunmaktadır. Bu tablodaki ID ile Personel Tablosundaki ID ilişkilendirip bilgileri Personel tablosundan almaktadır.

Markette bulunan ürünlerimizi sorgulamak için Kategori – Ürün Listesinin bulunduğu bir tablo oluşturduk. Bu tabloda ürün ID'si , ürünün hangi kategoride bulunduğunu gösteren Kategori tablosundan aldığı kategori ID'si, ürün stok durumu, alış fiyatı, satış fiyatı, markette hangi rafta bulunduğunu gösteren Raf tablosundan aldığı raf ID bilgisi, marka tablosundan aldığı marka bilgisi, ve son stok güncelleme tarihi bulunmaktadır.

Ürünleri daha kolay ulaşabilmek için Kategori tablosu oluşturduk. Bu tablo sayesinde yapacağımız arayüzde kullanıcı ürüne daha kolay bulabilecek. Bu tabloda kategori ID'si, kategori adı, bu kategoriden sorumlu personel ID'si ve Kategori stok durumu yer almaktadır.

Personelin müşteriye daha kolay yönlendirebilmesi için ürünleri raflara ayırdık. Raf tablosunda raf ID'si ve raf içeriği bulunmaktadır. Bu raf tablosunu kategori – ürün tablosu ile ilişkilendirdik. Aynı zamanda kategori – ürün tablosunu Tedarikçi tablosu ile ilişkilendirdik. Bu sayede ürün sorgularken stok durumunu da görebileceğiz. Tedarikçi tablosunda tedarikçi ID'si , Tedarik firma adı, Tedarikçinin iletişim bilgileri bulunmaktadır. Son olarak markette bulunan markaların tablosunu oluşturup her markaya ID atadık. Bu marka ID'sini diğer oluşturduğumuz tablolar ile ilişkilendirdik.

Sonraki haftalarımızda projemizi şekillendirecek olan ER diyagramımızı ve ilişkisel şemamızı aşağıda gördüğümüz gibi oluşturduk.



Bu proje, bir marketin günlük işlemlerini etkili bir şekilde yönetebilmek için geliştirilmiş bir veri tabanı yönetim sistemini içermektedir. Sistem, market personeli, müşteriler, ürünler, kategoriler, markalar, tedarikçiler ve indirimler gibi temel unsurları içerir ve bu unsurlar arasındaki ilişkileri yönetir.

Tablolar ve Özellikleri:

1. PERSONEL KART Tablosu (\_KartID, \_Bakiye, \_IndirimCeki):

- Market personelinin kimlik bilgilerini ve kart özelliklerini içerir.
- \_KartID, her personel kartını benzersiz bir şekilde tanımlar.
- \_Bakiye, personelin kartındaki para miktarını tutar.
- \_IndirimCeki, personelin kullanabileceği indirim çeki miktarını gösterir.

2. MUSTERI Tablosu (\_MusteriID, \_MusteriAdSoyad, \_MusteriTelNo, \_MusteriEmail, \_MusteriAdres, \_AlinanUrunID):

- Müşteri bilgilerini ve satın alınan ürünleri içerir.
- \_MusteriID, her müşteriye benzersiz bir şekilde tanımlar.
- \_AlinanUrunID, müşterinin satın aldığı ürünleri belirtir.

3. MUSTERIKART Tablosu (\_MusteriID, \_Indirim, \_Bakiye, \_YapilanAlisveris):

- Müşteri kart bilgilerini içerir.
- \_Indirim, müşterinin kartına tanımlanan indirim oranını gösterir.
- \_YapilanAlisveris, müşterinin toplam alışveriş miktarını tutar.

4. PERSONEL Tablosu (\_personelID, \_personelAdi, \_personelAdresi, \_personelTelNo, \_personelEmail, \_calistigiKategoriID, \_Maas):

- Market personelinin genel bilgilerini ve çalıştığı kategoriyi içerir.
- \_calistigiKategoriID, personelin çalıştığı kategoriyi belirtir.
- \_Maas, personelin maaş bilgisini içerir.

5. KATEGORI Tablosu (\_kategoriID, \_kategoriAdi, \_kategoriSorumluPersonelAdi, \_kategoriStokDurumu):

- Ürün kategorilerini içerir.
- \_kategoriID, her kategoriyi benzersiz bir şekilde tanımlar.
- \_kategoriSorumluPersonelAdi, kategoriye sorumlu olan personelin adını belirtir.
- \_kategoriStokDurumu, kategorinin genel stok durumunu gösterir.

6. RAFBILGISI Tablosu (\_RafID, \_Raficerigi):

- Rafların içeriğini ve hangi ürünlerin bulunduğunu içerir.
- \_RafID, her rafı benzersiz bir şekilde tanımlar.

7. MARKA Tablosu (\_MarkaID, \_MarkaAdi, \_MarkaStok, \_UrunID):

- Markette bulunan markaları içerir.
- \_MarkaID, her markayı benzersiz bir şekilde tanımlar.
- \_MarkaStok, markanın genel stok durumunu gösterir.

8. İNDİRİM Tablosu (\_UygulananIndirimID, \_UygulananMarkaID, \_IndirimMiktari):

- Uygulanan indirimleri ve hangi markalara uygulandığını içerir.
- \_UygulananIndirimID, her indrimi benzersiz bir şekilde tanımlar.
- \_IndirimMiktari, uygulanan indirim miktarını gösterir.

9. URUNLISTESI Tablosu (\_UrunID, \_KategoriID, \_UrunAdi, \_StokDurumu, \_AlisFiyati, \_SatisFiyati, \_RafBilgisilID):

- Markette bulunan ürünleri ve bu ürünlerin detaylarını içerir.
- \_UrunID, her ürünü benzersiz bir şekilde tanımlar.
- \_StokDurumu, ürünün stok durumunu gösterir.
- \_RafBilgisilID, ürünün hangi rafta bulunduğunu belirtir.

10. TEDARIKCI Tablosu (\_TedarikciID, \_FirmaID, \_TedarikciTelNo):

- Markete ürün tedarik eden tedarikçilerin bilgilerini içerir.
- \_TedarikciID, her tedarikçiyi benzersiz bir şekilde tanımlar.

11. CALISTIGI\_YER Tablosu (\_kategoriID, \_personelID):

- Personellerin çalıştığı kategorileri belirten ilişkisel tablo.

12. URUN\_RAF Tablosu (\_RafID, UrunID):

- Ürünlerin hangi rafta bulunduğunu belirten ilişkisel tablo.

İlişkiler:

- PERSONEL ve CALISTIGI\_YER tabloları arasında 1 - n bir ilişki bulunmaktadır. Bir personel birden fazla kategoride çalışabilir, ancak her kategoride bir personel sorumlu olabilir.
- URUNLISTESI ve KATEGORI tabloları arasında bir 1 - n ilişki vardır. Bir kategoriye ait birden fazla ürün olabilir, ancak her ürün bir kategoriye aittir.
- URUN\_RAF tablosu, URUNLISTESI ve RAFBILGISI tabloları arasında birleştirici bir tablo olarak kullanılarak, her ürünün hangi rafta bulunduğunu belirtir.

Bu veri tabanı, market yönetimine bir dizi araç sunarak müşteri memnuniyetini artırmayı, stok yönetimini optimize etmeyi ve personel performansını izlemeyi hedefler. İlişkisel tablolar ve veri modellemesi, veri tabanının etkili bir şekilde kullanılmasını sağlar ve iş süreçlerini daha anlamlı bir hale getirir.

Şimdi projemizde yaptığımız örnek sorgularımıza ve sql kodlarımıza diğer sayfalarda değineceğiz.

----- TRİGGERSLAR-----

--PERSONEL KART ve PERSONEL Tabloları Arasındaki Tetikleyici:

```
CREATE TRIGGER trg_personel_kart_guncelleme
ON PERSONEL_KART
AFTER UPDATE
AS
BEGIN
    UPDATE PERSONEL
    SET Maas = i.Bakiye
    FROM PERSONEL INNER JOIN inserted i ON PERSONEL.PersonelID = i.KartID;
END;
```

-----

--ÜRÜNLİSTESİ ve MARKA Tabloları Arasındaki Tetikleyici:

```
CREATE TRIGGER trg_urun_ekleme
ON URUNLISTESİ
AFTER INSERT
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM MARKA WHERE UrunID = (SELECT UrunID FROM inserted))
    BEGIN
        INSERT INTO MARKA (MarkaAdi, MarkaStok, UrunID)
        VALUES ('Belirsiz', 0, (SELECT UrunID FROM inserted));
    END
END;
```

END;

END;

-----

--ÜRÜN\_RAF ve RAFBILGISI Tabloları Arasındaki Tetikleyici:

CREATE TRIGGER trg\_urun\_raf\_guncelleme

ON URUN\_RAF

AFTER INSERT, DELETE

AS

BEGIN

UPDATE RAFBILGISI

SET Raficerigi =

CASE

WHEN EXISTS (SELECT 1 FROM inserted) THEN

CONCAT(Raficerigi, ' ', (SELECT UrunID FROM inserted))

ELSE

REPLACE(Raficerigi, CONCAT(' ', (SELECT UrunID FROM deleted)), '')

END

WHERE RafID = (SELECT RafID FROM inserted) OR RafID = (SELECT RafID FROM deleted);

END;

-----

--MÜŞTERİKART ve MÜŞTERİ Tabloları Arasındaki Tetikleyici:

CREATE TRIGGER trg\_musteri\_kart\_guncelleme

ON MUSTERIKART

AFTER UPDATE

AS

BEGIN

UPDATE MUSTERI

SET MusteriAdSoyad = CONCAT('Müşteri ', i.MusteriID)

```
FROM MUSTERI INNER JOIN inserted i ON MUSTERI.MusteriID = i.MusteriID;  
END;
```

-----

--İNDİRİM ve MARKA Tabloları Arasındaki Tetikleyici:

```
CREATE TRIGGER trg_indirim_guncelleme  
ON INDIRIM  
AFTER UPDATE  
AS  
BEGIN  
    UPDATE MARKA  
    SET MarkaStok = MarkaStok - i.IndirimMiktari  
    FROM MARKA INNER JOIN inserted i ON MARKA.MarkaID = i.UygulananMarkaID;  
END;
```

-----

```
CREATE TRIGGER trg_UrunGuncelle  
ON URUNLISTESI  
AFTER UPDATE  
AS  
BEGIN  
    UPDATE m  
    SET MarkaStok = u.StokDurumu  
    FROM MARKA m  
    INNER JOIN INSERTED i ON m.UrunID = i.UrunID  
    INNER JOIN UPDATED u ON i.UrunID = u.UrunID;  
END;
```

-- Tetikleyiciyi tekrar oluşturma

```
CREATE TRIGGER tr_urunlistesi_after_insert
```

```

ON MARKA
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE URUNLISTESI
    SET MarkaID = i.MarkaID
    FROM URUNLISTESI u
    INNER JOIN INSERTED i ON u.UrunID = i.UrunID;
END;

-- URUNLISTESI tablosuna ürün eklendiğinde çalışacak trigger
CREATE TRIGGER trg_URUNLISTESI
ON URUNLISTESI
AFTER INSERT
AS
BEGIN
    DECLARE @RafID INT;
    -- Varsayılan bir raf seçimi, örneğin RafID = 1
    SET @RafID = 1;
    -- URUN_RAF tablosuna yeni ürünü ve rafı ekleyen SQL sorgusu
    INSERT INTO URUN_RAF (RafID, UrunID)
    SELECT @RafID, UrunID FROM INSERTED;
END;

-----

---- kategori sorumlu personle verisini girme,
---- rastgele sorumlu ataması yapar
WITH RandomizedPersonel AS (
    SELECT
    KATEGORI.KategoriID,

```



```
PERSONEL.PersonelAdi,  
ROW_NUMBER() OVER (PARTITION BY KATEGORI.KategoriID ORDER BY NEWID()) AS RowNum  
FROM  
KATEGORI  
CROSS JOIN PERSONEL  
)  
UPDATE KATEGORI  
SET KategoriSorumluPersonelAdi = RandomizedPersonel.PersonelAdi  
FROM KATEGORI  
INNER JOIN RandomizedPersonel ON KATEGORI.KategoriID = RandomizedPersonel.KategoriID  
WHERE RandomizedPersonel.RowNum = 1;
```

## UPDATE ORNEK

Marka adı 'apple' olan verinin marka stok durumunu 100 olarak güncelle

-- TABLO OLUŞTUR...SEKOC\aysee (53)) SQLQuery7.sql - AY...YSEKOC\aysee (53))

```
-- SELECT * FROM MARKA
-- UPDATE MARKA
-- SET _MarkaStok = 100
-- WHERE _MarkaAdi = 'Apple';
-- SELECT * FROM MARKA
```

146 %

Results Messages

	_MarkaID	_MarkaAdi	_MarkaStok	_UrunID
1	1	Apple	1	1000
2	2	Samsung	2	2000
3	3	Huawei	3	3000
4	4	Xiaomi	4	4000
5	5	Oppo	5	5000

---

	_MarkaID	_MarkaAdi	_MarkaStok	_UrunID
1	1	Apple	100	1000
2	2	Samsung	2	2000
3	3	Huawei	3	3000
4	4	Xiaomi	4	4000
5	5	Oppo	5	5000
6	6	Vivo	6	6000
7	7	Realme	7	7000
8	8	Motorola	8	8000
9	9	LG	9	9000

## SELECT ORNEK

PERSONEL\_KART tablosundaki \_Bakiye niteliği 900 den büyük olan personellerin listesi

AYSEKOC\SQLXPRES...dbo.URUNLISTESI SQLQuery9.sql - AY...YSEKOC\aysee (53))

```
-- SELECT _KartID, _Bakiye, _IndirimCeki
-- FROM PERSONEL_KART
-- WHERE _Bakiye >900;
```

146 %

Results Messages

	_KartID	_Bakiye	_IndirimCeki
1	8	925.14	NULL
2	21	963.58	NULL
3	27	937.44	NULL
4	33	925.28	NULL
5	43	970.25	NULL
6	50	916.84	NULL
7	59	949.31	NULL
8	66	929.15	NULL
9	79	928.37	NULL
10	92	943.21	NULL
11	98	967.24	NULL
12	107	915.20	NULL
13	112	984.75	NULL
14	124	936.20	NULL
15	132	943.76	NULL
16	143	954.62	NULL
17	154	915.68	NULL
18	162	943.76	NULL
19	173	975.42	NULL
20	184	975.20	NULL
21	189	952.84	NULL
22	198	925.28	NULL

## SUBQUERY ORNEK

MARKA ve INDIRIM tablolarını iç içe birleştirerek, her marka için uygulanan en yüksek indirim miktarını döndürür.

AYSEKOC\SQLXPRES...dbo.URUNLISTESI

SQLQuery9.sql - AY...YSEKOC\aysee (57))\* X -- TABLO OLUŞTUR...SEKOC\aysee (53))

```
SELECT M._MarkaAdi, MAX(I._IndirimMiktari) AS EnYuksekIndirim
FROM MARKA M
INNER JOIN INDIRIM I ON M._MarkaID = I._UygulananMarkaID
GROUP BY M._MarkaAdi;
```

146 %

Results Messages

	_MarkaAdi	EnYuksekIndirim
1	Alcatel 1S	12
2	Alcatel 3X	11
3	Apple iPhone 6s	11
4	Apple iPhone 7	10
5	Apple iPhone 8	8
6	Asus Zenfone 3	20
7	Asus Zenfone 4	20
8	Asus Zenfone 5	17
9	BlackBerry KEY2 LE	26
10	BlackBerry KEYone	21
11	BlackBerry KEYone Black Edition	25
12	Doogee S91 Pro	17
13	Doogee S92 Pro	16
14	Doogee S93 Pro	12
15	Honor 5X	26
16	Honor 6X	26
17	Honor 7X	13
18	HTC U11	26
19	HTC U11 Life	28
20	HTC U11 Lite	29
21	Huawei P10	25
22	Huawei P20	25
23	Huawei P9	26
24	Infinix Hot 5	20
25	Infinix Hot 6	18
26	Infinix Hot 7	19
27	Lenovo Z5	23
28	Lenovo Z5 Pro GT	16
29	Lenovo Z5s	24
30	LG G4	11
31	LG G5	12
32	LG G6	5
33	Meizu 14	20
34	Meizu 15	16

## TRANSACTION ORNEK

PERSONEL\_KART tablosundaki \_KartID'si '1' olan personelin \_Bakiye durumunu 100 arttırdık. Tablonun önceki ve sonraki durumları gösterdik.

AYSEKOC\SQLXPRES...dbo.URUNLISTESI SQLQuery9.sql - AY...YSEK

```
SELECT * FROM PERSONEL_KART
BEGIN TRAN

UPDATE PERSONEL_KART
SET _Bakiye = _Bakiye + 100
WHERE _KartID = 1;

COMMIT TRAN
SELECT * FROM PERSONEL_KART
```

146 %

Results Messages

	_KartID	_Bakiye	_IndirimCeki
1	1	542.76	NULL
2	2	187.45	NULL
3	3	831.29	75
4	4	462.18	NULL

---

	_KartID	_Bakiye	_IndirimCeki
1	1	642.76	NULL
2	2	187.45	NULL
3	3	831.29	75
4	4	462.18	NULL
5	5	694.20	21
6	6	318.67	91
7	7	109.82	NULL
8	8	925.14	NULL

## INDEX ORNEK

SQLQuery2.sql - AY...YSEKOC\aysee (64))\* -> X AYSEKOC\SQLXPRES...dbo.MUSTERIKART\*

```
CREATE INDEX IX_Marka_MarkaAdi
ON MARKA (_MarkaAdi);
```

146 %

Messages

Commands completed successfully.

Completion time: 2023-12-25T14:36:58.0368351+03:00

## INDEX SORGU ORNEK

SQLQuery2.sql - AY...YSEKOC\aysee (64))\* -> X AYSEKOC\SQLXPRES...dbo.MUSTERIKART\*

```
EXEC sp_helpindex 'MARKA';
```

146 %

Results Messages

	index_name	index_description	index_keys
1	IX_Marka_MarkaAdi	nonclustered located on PRIMARY	_MarkaAdi
2	PK_MARKA	clustered, unique, primary key located on PRIMARY	_MarkaID

## STORED PROCEDURE ORNEK

belirli bir müşteriye ait alışveriş miktarını günceller, indirim uygular ve bakiyeyi ayarlar

```
SQLQuery2.sql - AY...YSEKOC\aysee (64)) *  AYSEKOC\SQLXP...dbo.MUSTERIKART*  SQLQuery1.sql - AY...YSEKOC\aysee (70)) *

-- Örnek Stored Procedure
CREATE PROCEDURE SP_UPDATE_MUSTERIKART
    @MusteriID INT,
    @IndirimMiktari DECIMAL(5, 2),
    @AlisverisMiktari INT
AS
BEGIN
    -- Alışveriş miktarını güncelle
    UPDATE MUSTERIKART
    SET _YapilanAlisveris = _YapilanAlisveris + @AlisverisMiktari;

    -- Indirim uygula
    UPDATE MUSTERIKART
    SET _Indirim = @IndirimMiktari
    WHERE _MusteriID = @MusteriID;

    -- Bakiyeyi güncelle
    UPDATE MUSTERIKART
    SET _Bakiye = _Bakiye - @IndirimMiktari
    WHERE _MusteriID = @MusteriID;

    PRINT 'MUSTERIKART Bilgileri Güncellendi.';
END;
```

146 %

Messages

Commands completed successfully.

Completion time: 2023-12-25T14:06:54.4672428+03:00

## STORED PROCEDURE KODU CALISMA SORGUSU

```
SQLQuery2.sql - AY...YSEKOC\aysee (64)) *  AYSEKOC\SQLXP...dbo.MUSTERIKART*  SQLQuery1

-- -- Bakiyeyi güncelle
-- UPDATE MUSTERIKART
-- SET _Bakiye = _Bakiye - @IndirimMiktari
-- WHERE _MusteriID = @MusteriID;

-- PRINT 'MUSTERIKART Bilgileri Güncellendi.';
--END;
DECLARE @MusteriID INT = 1;
DECLARE @IndirimMiktari DECIMAL(5, 2) = 10.00;
DECLARE @AlisverisMiktari INT = 3;

EXEC SP_UPDATE_MUSTERIKART
    @MusteriID,
    @IndirimMiktari,
    @AlisverisMiktari;
```

146 %

Messages

(200 rows affected)

(1 row affected)

(1 row affected)

MUSTERIKART Bilgileri Güncellendi.

Completion time: 2023-12-25T14:10:30.8187377+03:00

## FUNCTION ORNEK

Bu SQL fonksiyonu, MusteriID'si 1 olan müşteriye ait müşteri kartındaki bakiyeyi hesaplamak için kullanılır.

```
SQLQuery2.sql - AY...YSEKOC\aysee (64))* X AYSEKOC\SQLXP...dbo.MUSTERIKART* SQLQuery1.sql - AY...YSEKOC\aysee (70))* -- TABLO OLUŞTUR...SEKOC\aysee (52))
-- DECLARE @MusteriID INT = 1; -- @MusteriID değişkenini tanımla ve bir değer atayarak initialize et
-- DECLARE @Bakiye DECIMAL(10, 2);

-- SET @Bakiye = dbo.FN_CALCULATE_TOTAL_BAKIYE(@MusteriID); -- Parametreyi doğru şekilde kullan
-- Sonucu kullan
PRINT 'Toplam Bakiye: ' + CONVERT(VARCHAR, @Bakiye);

146 %
Messages
Toplam Bakiye: 100.00

Completion time: 2023-12-25T13:42:28.7875303+03:00
```

## TRIGGER ORNEK

### TRIGGER OLUSTURMA

```
SQLQuery2.sql - AY...YSEKOC\aysee (64))* X AYSEKOC\SQLXP...dbo.MUSTERIKART* SQLQuery1.sql - AY...YSEKOC\aysee (70))* -- TABLO OLUŞTUR...SEKOC\aysee
-- Örnek Trigger
-- CREATE TRIGGER TR_AfterInsertUpdate_Marka
-- ON MARKA
-- AFTER INSERT, UPDATE
-- AS
-- BEGIN
--     -- Eklenen veya güncellenen kayıtları kontrol etmek için "inserted" kullanılır
--     IF (SELECT COUNT(*) FROM inserted WHERE _MarkaStok IS NOT NULL) > 0
--     BEGIN
--         -- Eğer _MarkaStok güncellenirse, belirli işlemleri gerçekleştir
--         PRINT 'Marka Stok Güncellendi. İşlem Yapılıyor...';

--         -- Örneğin, bir log tablosuna ekleme yapabiliriz
--         INSERT INTO MarkaLog (MarkaID, IslemTipi, IslemTarihi)
--         SELECT m._MarkaID, 'UPDATE', GETDATE()
--         FROM MARKA m
--         INNER JOIN inserted i ON m._MarkaID = i._MarkaID;
--     END
--     ELSE
--     BEGIN
--         -- Eğer başka bir alan güncellenirse, burada gerekli işlemleri yapabilirsiniz
--         PRINT 'Marka Bilgileri Güncellendi. İşlem Yapılıyor...';
--     END
-- END;

146 %
Messages
Commands completed successfully.

Completion time: 2023-12-25T13:48:28.2503873+03:00
```

## TRIGGER SORGULAMA KODU

SQLQuery2.sql - AY...YSEKOC\aysee (64)) \* X AYSEKOC\SQLXP...dbo.MUSTERIKART\* SQLQuery1.sql - AY...YSEKOC\aysee (70)) \* -- TABLO OLUŞTUR...SEKOC\aysee (52))

```
SELECT *
FROM sys.triggers
WHERE name = 'TR_AfterInsertUpdate_Marka';
```

146 %

Results Messages

	name	object_id	parent_class	parent_class_desc	parent_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_disabled	is_not_for_replication	is_instead_of_trigger
1	TR_AfterInsertUpdate_Marka	1845581613	1	OBJECT_OR_COLUMN	1557580587	TR	SQL_TRIGGER	2023-12-25 13:48:28.247	2023-12-25 13:48:28.247	0	0	0	0

## VIEW ORNEK

SQLQuery2.sql - AY...YSEKOC\aysee (64)) \* X AYSEKOC\SQLXP...dbo.MUSTERIKART\* SQLQuery1.sql - AY...YSEKOC\aysee (70)) \* -- TABLO OLUŞTUR...SEKOC\aysee (52))

```
CREATE VIEW VW_MUSTERIKART_BILGILERI
AS
SELECT
    _MusteriID,
    _Indirim,
    _Bakiye,
    _YapilanAlisveris,
    'Musteri_' + CAST(_MusteriID AS VARCHAR) AS MusteriAdi -- Örnek bir türetilmiş sütun
FROM
    MUSTERIKART;
```

146 %

Messages

Commands completed successfully.

Completion time: 2023-12-25T14:42:27.2529558+03:00

## VIEW SORGULAMA ORNEK

SQLQuery2.sql - AY...YSEKOC\aysee (64)) \* X AYSEKOC\SQLXP...dbo.MUSTERIKART\* SQLQuery1.sql - AY...YSEKOC\aysee (70)) \* -- TABLO OLUŞTUR...SEKOC\aysee (52))

```
--CREATE VIEW VW_MUSTERIKART_BILGILERI
--AS
--SELECT
--    _MusteriID,
--    _Indirim,
--    _Bakiye,
--    _YapilanAlisveris,
--    'Musteri_' + CAST(_MusteriID AS VARCHAR) AS MusteriAdi -- Örnek bir türetilmiş sütun
--FROM
--    MUSTERIKART;
SELECT * FROM VW_MUSTERIKART_BILGILERI;
```

146 %

Results Messages

	_MusteriID	_Indirim	_Bakiye	_YapilanAlisveris	MusteriAdi
1	1	10	90	503	Musteri_1
2	2	0	200	1003	Musteri_2
3	3	0	300	1503	Musteri_3
4	4	0	400	2003	Musteri_4
5	5	0	500	2503	Musteri_5
6	6	0	600	3003	Musteri_6
7	7	0	700	3503	Musteri_7
8	8	0	800	4003	Musteri_8
9	9	0	900	4503	Musteri_9
10	10	0	1000	5003	Musteri_10
11	11	0	1100	5503	Musteri_11
12	12	0	1200	6003	Musteri_12
13	13	0	1300	6503	Musteri_13
14	14	0	1400	7003	Musteri_14
15	15	0	1500	7503	Musteri_15
16	16	0	1600	8003	Musteri_16
17	17	0	1700	8503	Musteri_17
18	18	0	1800	9003	Musteri_18
19	19	0	1900	9503	Musteri_19
20	20	0	2000	10003	Musteri_20
21	21	0	2100	10503	Musteri_21
22	22	0	2200	11003	Musteri_22
23	23	0	2300	11503	Musteri_23
24	24	0	2400	12003	Musteri_24
25	25	0	2500	12503	Musteri_25
26	26	0	2600	13003	Musteri_26