

Veritabanı ve Yönetim Sistemleri

BLP105 (Son güncellenme Tarihi: 18.12.2022)

ABDULMELİK DERİNKÖK

İçindekiler

1.TEMEL KAVRAMLAR	3
2. VERİTABANI TASARIMI VE NORMALİZASYON.....	3
3.SQL SERVER KURULUMU.....	3
4.SQL SERVER MANAGEMENT STUDIO KURULUMU	13
5.SQL (STRUCTURED QUERY LANGUAGE) - YAPISAL SORGULAMA DİLİ.....	16
5.0. Veri Türleri.....	17
5.1. DQL (Data Query) – Veri Sorgulama.....	19
5.1.1 Select	19
5.2. DDL (Data Defination) – Veri Tanımlama	32
5.2.1. Create	32
5.2.2. Drop	34
5.2.3. Alter.....	35
5.2.4. Truncate.....	36
5.2.5. Rename	36
5.2.6. Comment.....	36
5.3. DML (Data Manipulation) – Veri İşleme	37
5.3.1 Insert.....	37
5.3.2 Update.....	38
5.3.3 Delete.....	39
5.3.4 Merge	40
5.3.5 Call	40
5.3.6 Explain Plan	40
5.3.7 Lock Table.....	41
5.4. DCL (Data Control) – Veri Kontrol	41
5.4.1. Grant.....	41
5.4.2. Revoke	41
5.4.3. Deny.....	41
5.5. TCL (Transaction Control)	41
5.5.1. Commit	41
5.5.2 Rollback.....	41
5.5.3. Savepoint	41
5.6. Fonksiyonlar	41
5.6.1 String (Metinsel) Fonksiyonlar.....	41

5.6.2 Sayısal Fonksiyonlar	44
5.6.3. Tarih Fonksiyonları	46
KAYNAKÇA.....	50

1.TEMEL KAVRAMLAR

...

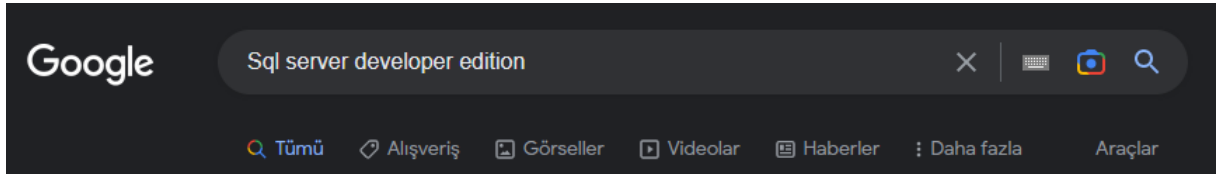
2. VERİTABANI TASARIMI VE NORMALİZASYON

...

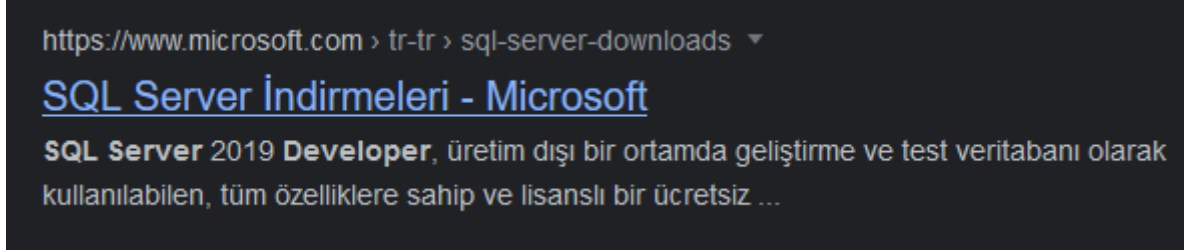
3.SQL SERVER KURULUMU

Not: Bu bölüm Kasım-2022 tarihinde oluşturulmuştur. Windows 10 işletim sistemi üzerine yüklenen SQL Server kurulumu anlatılmaktadır. Zaman içerisinde parametreler değişiklik gösterebilir.

1. "sql server developer edition" araması yapılır.



2.



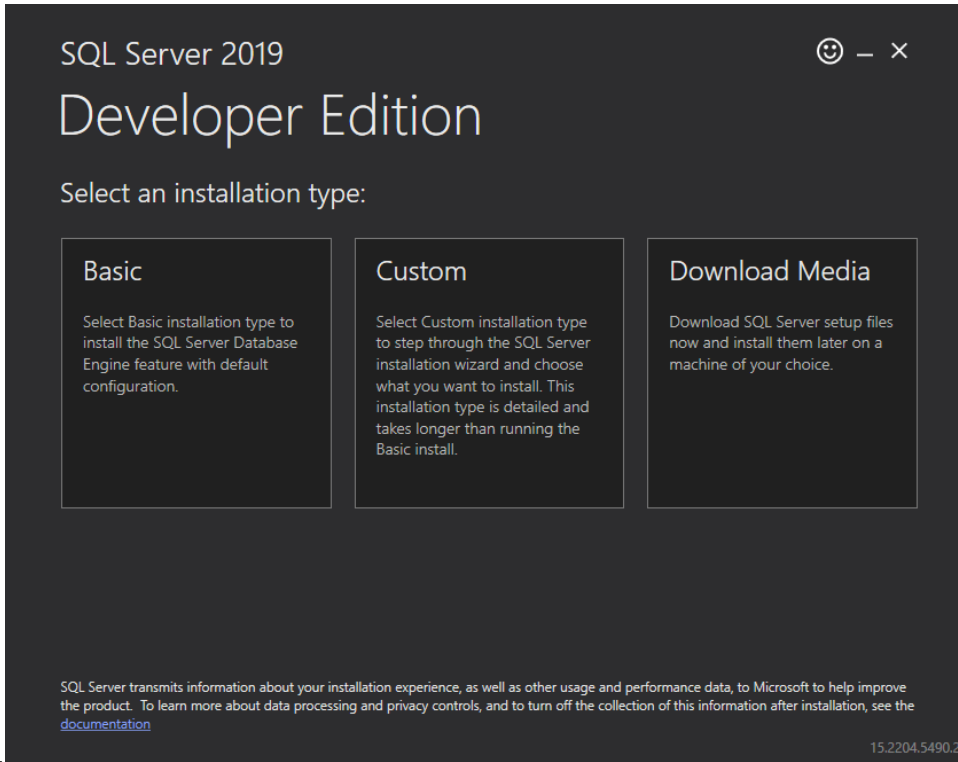
3."Developer Edition" indirilir.



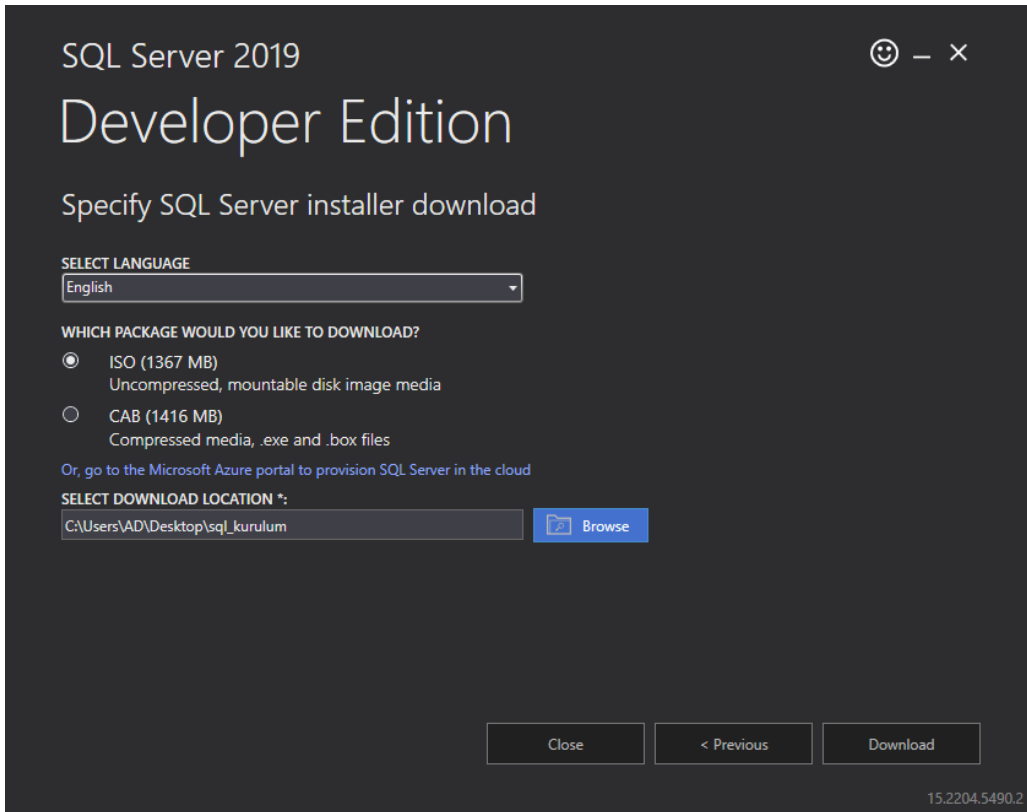
4.

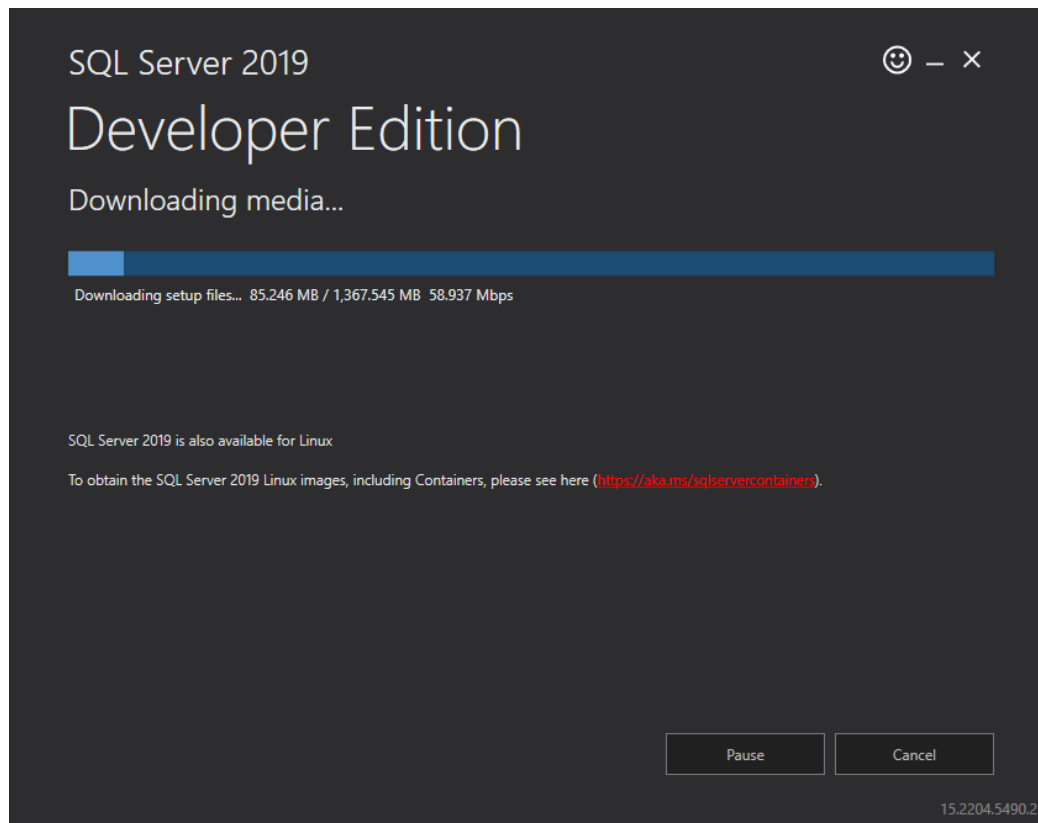


5.

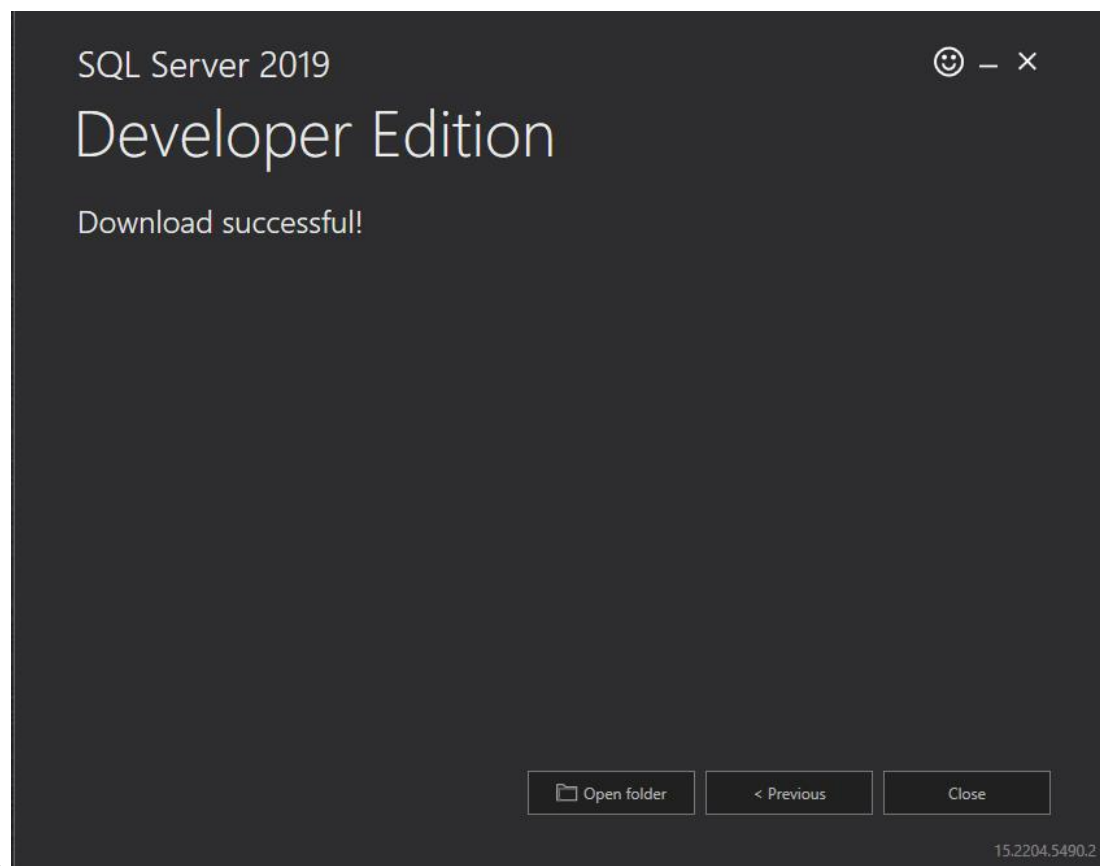


6.



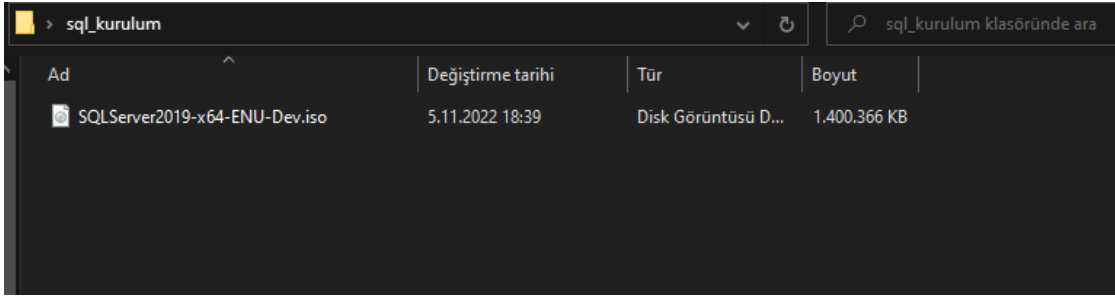


7.

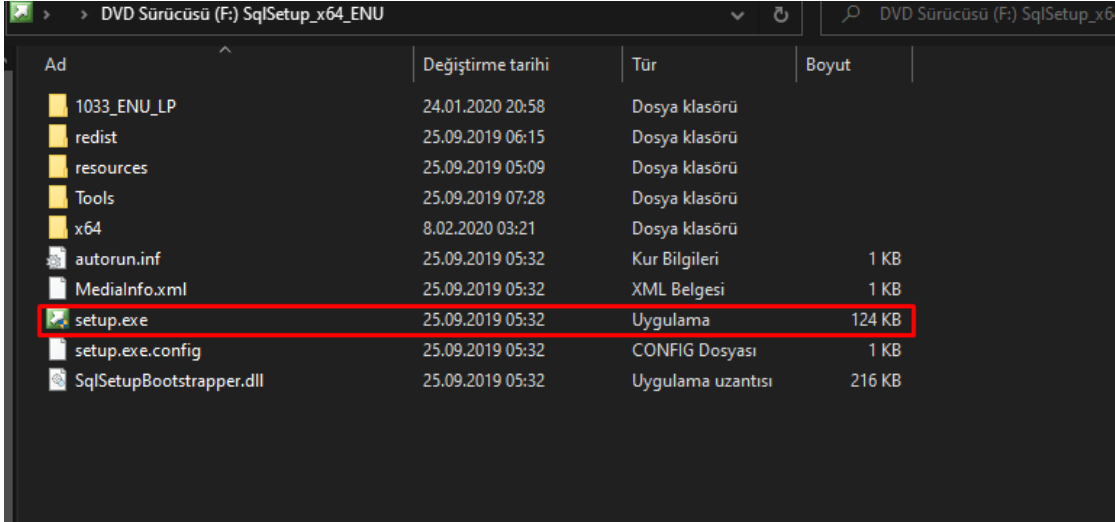


8.

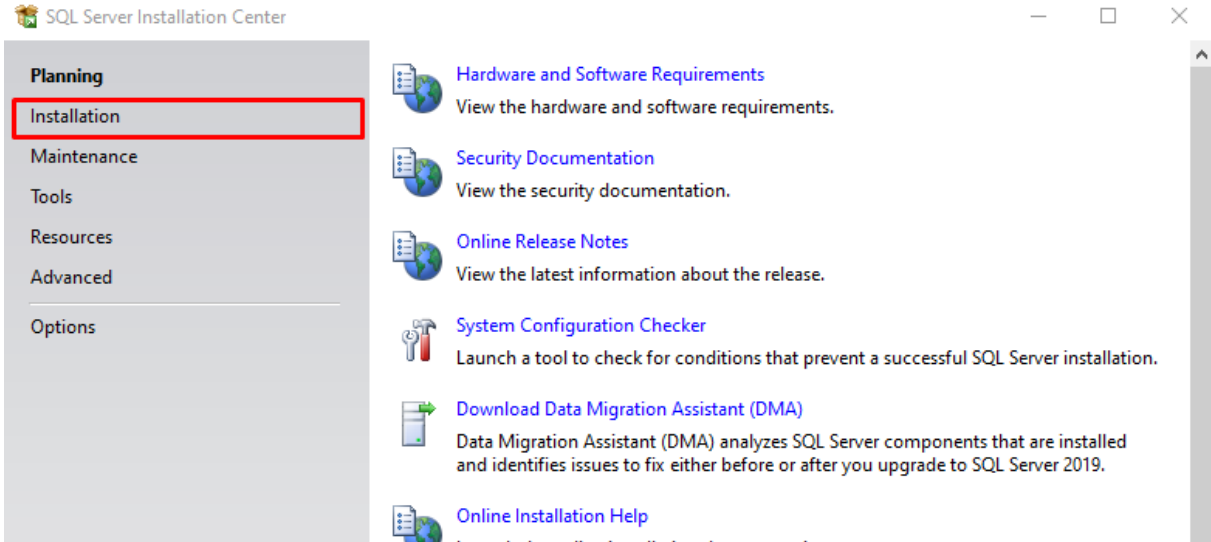
9. İndirilen ISO dosyası açılır.




10. "setup.exe" dosyası açılır.




11. "Installation" sekmesine tıklanır.




12. "New SQL Server stand-alone installation ..." seçeneğine tıklanır.

**New SQL Server stand-alone installation or add features to an existing installation**


Launch a wizard to install SQL Server 2019 in a non-clustered environment or to add features to an existing SQL Server 2019 instance.

**Install SQL Server Reporting Services**

Launch a download page that provides a link to install SQL Server Reporting Services. An internet connection is required to install SSRS.

**Install SQL Server Management Tools**

Launch a download page that provides a link to install SQL Server Management Studio, SQL Server command-line utilities (SQLCMD and BCP), SQL Server PowerShell provider, SQL Server Profiler and Database Tuning Advisor. An internet connection is required to install these tools.

**Install SQL Server Data Tools**

Launch a download page that provides a link to install SQL Server Data Tools (SSDT).

13. "Developer" sürümünün seçili olmasına dikkat edilir ve "Next" butonuna tıklanır.

SQL Server 2019 Setup

Product Key

Specify the edition of SQL Server 2019 to install.

Product Key

- License Terms
- Global Rules
- Microsoft Update
- Product Updates
- Install Setup Files
- Install Rules
- Feature Selection
- Feature Rules
- Feature Configuration Rules
- Ready to Install
- Installation Progress
- Complete

Validate this instance of SQL Server 2019 by entering the 25-character key from the Microsoft certificate of authenticity or product packaging. You can also specify a free edition of SQL Server: Developer, Evaluation, or Express. Evaluation has the largest set of SQL Server features, as documented in SQL Server Books Online, and is activated with a 180-day expiration. Developer edition does not have an expiration, has the same set of features found in Evaluation, but is licensed for non-production database application development only. To upgrade from one installed edition to another, run the Edition Upgrade Wizard.

☒ Specify a free edition:

Developer

☐ Enter the product key:

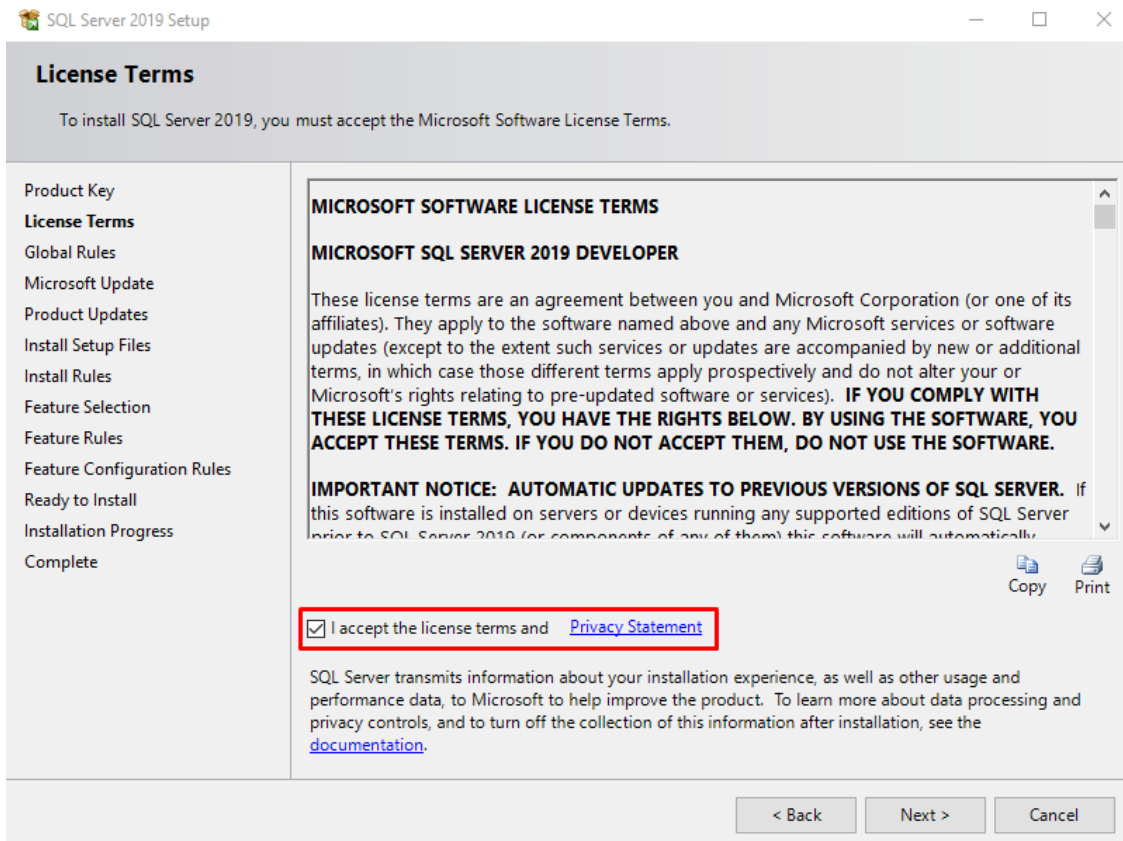
- - - -

< Back

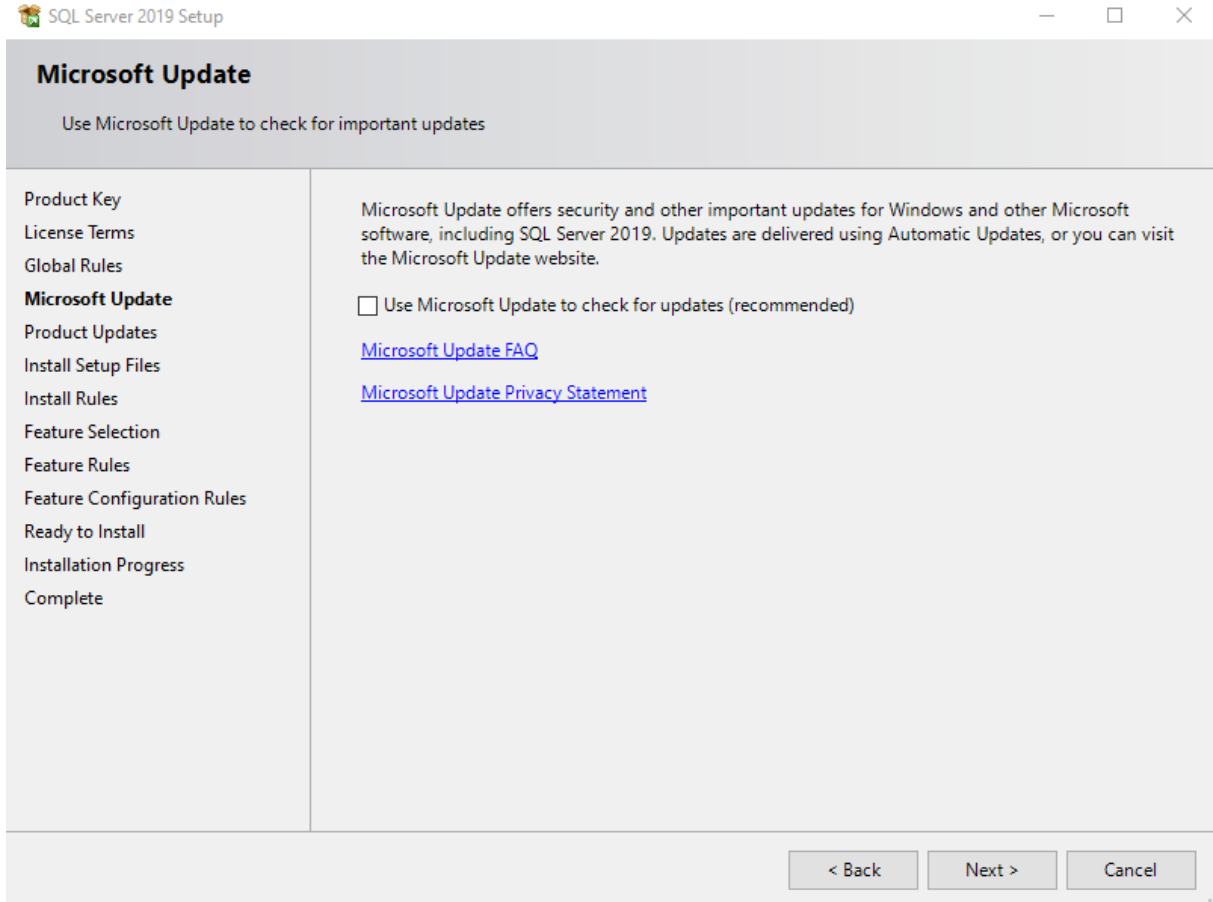
Next >

Cancel

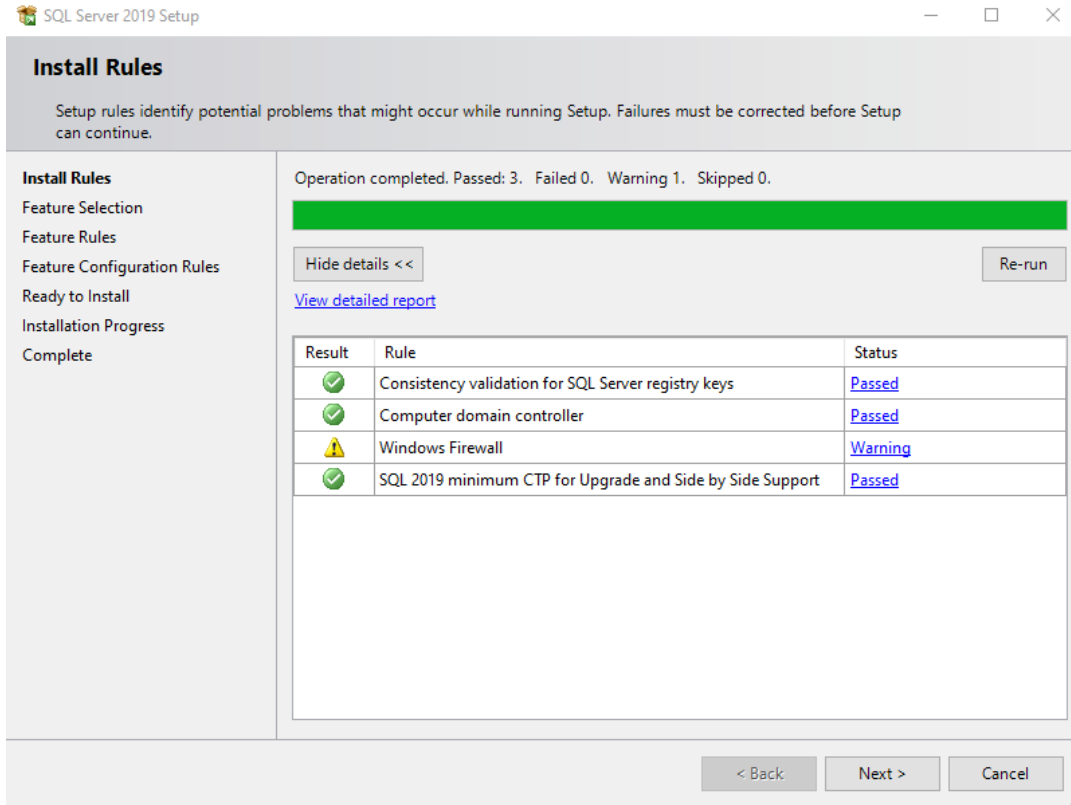
14. Lisans ve gizlilik sözleşmesi kabul edilir. "Next" butonuna tıklanır.



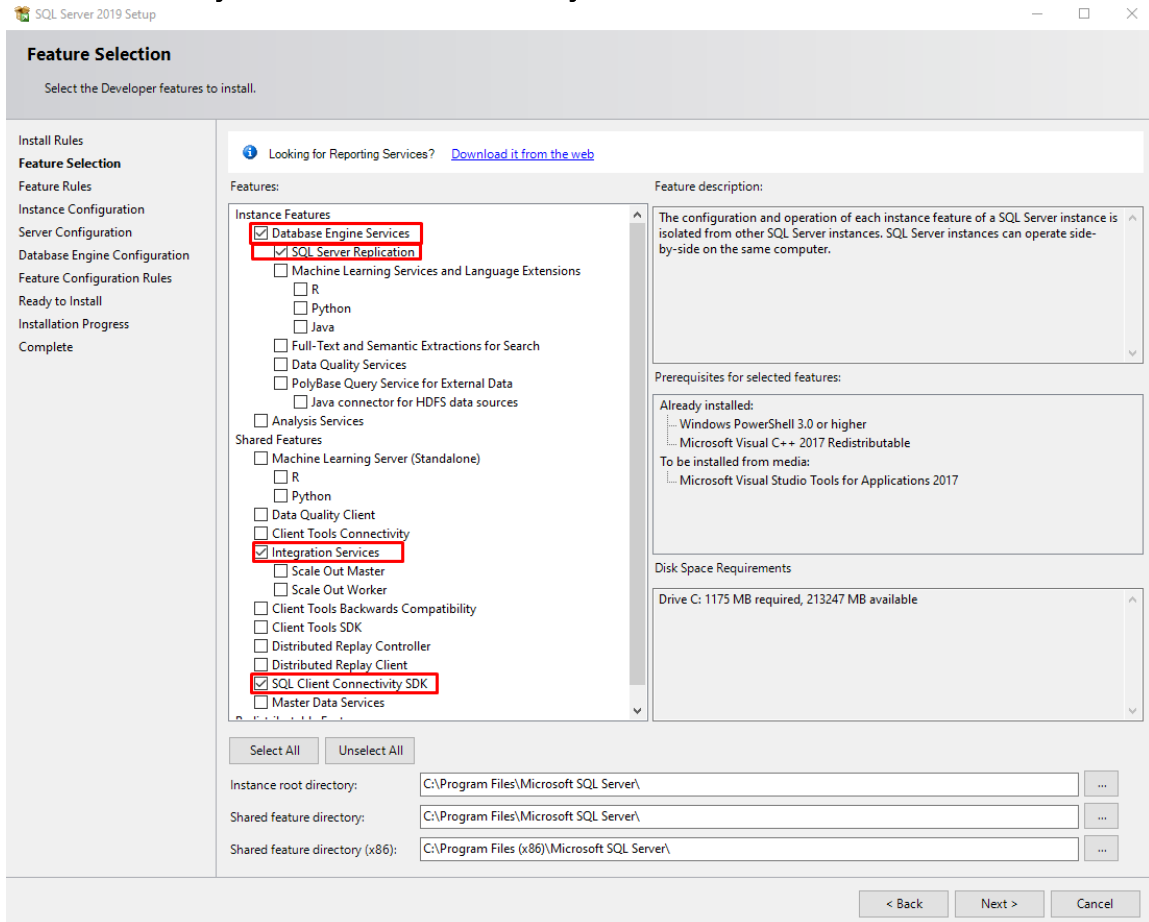
15. Otomatik güncelleme seçeneği belirlenir ve "Next" butonuna tıklanır.



16. "Next" butonuna tıklanır.



17. Görselde işaretli olan özellikler seçilir ve "Next" butonuna tıklanır.



18. "Instance ID" değeri girilir ve "Next" butonuna tıklanır.

SQL Server 2019 Setup

Instance Configuration

Specify the name and instance ID for the instance of SQL Server. Instance ID becomes part of the installation path.

Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Ready to Install
Installation Progress
Complete

☐ Default instance
☒ Named instance:

Instance ID:

SQL Server directory: C:\Program Files\Microsoft SQL Server\MSSQL15.SUNUCU_ADI\

Installed instances:

Instance Name	Instance ID	Features	Edition	Version
---------------	-------------	----------	---------	---------

< Back Next > Cancel

19. "Next" butonuna tıklanır.

SQL Server 2019 Setup

Server Configuration

Specify the service accounts and collation configuration.

Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Ready to Install
Installation Progress
Complete

Service Accounts Collation

Microsoft recommends that you use a separate account for each SQL Server service.

Service	Account Name	Password	Startup Type
SQL Server Agent	NT Service\SQLAgent\$SUNUC_ADI		Manual
SQL Server Database Engine	NT Service\MSSQL\$SUNUC_ADI		Automatic
SQL Server Integration Services 15.0	NT Service\MsDtsServer150		Automatic
SQL Server Browser	NT AUTHORITY\LOCAL SERVICE		Disabled

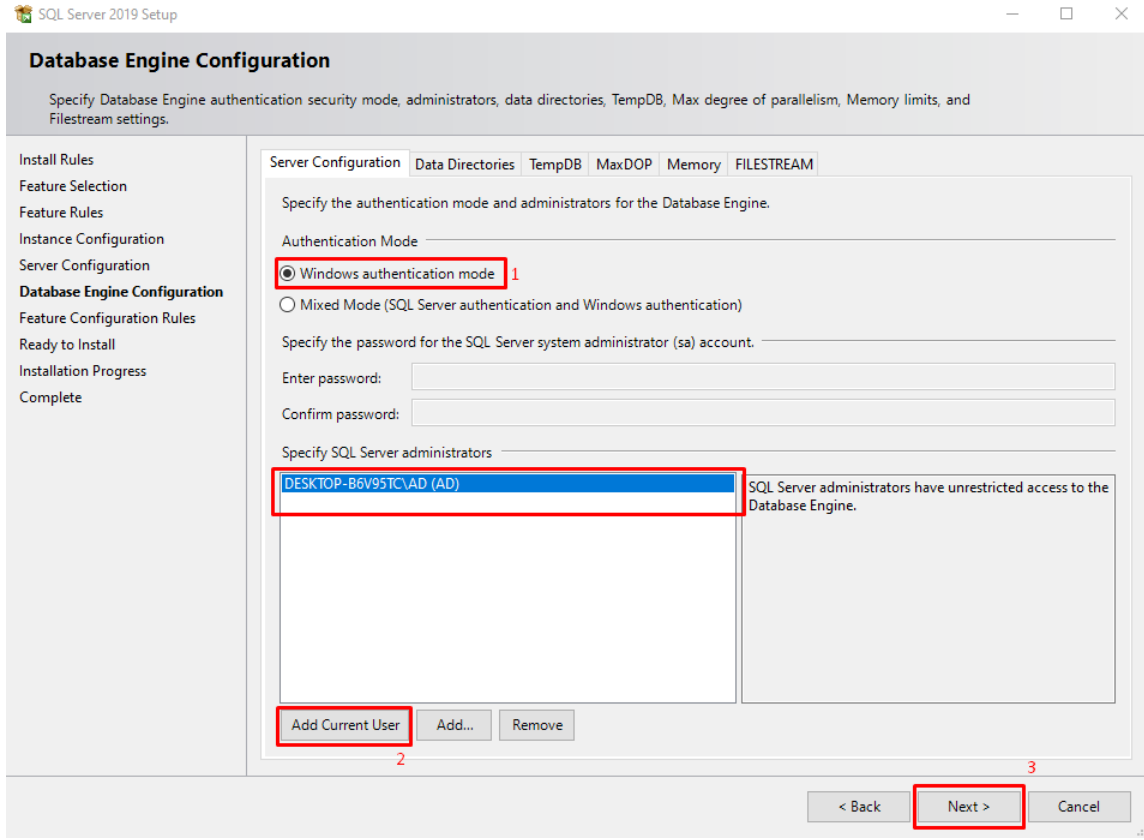
☐ Grant Perform Volume Maintenance Task privilege to SQL Server Database Engine Service

This privilege enables instant file initialization by avoiding zeroing of data pages. This may lead to information disclosure by allowing deleted content to be accessed.

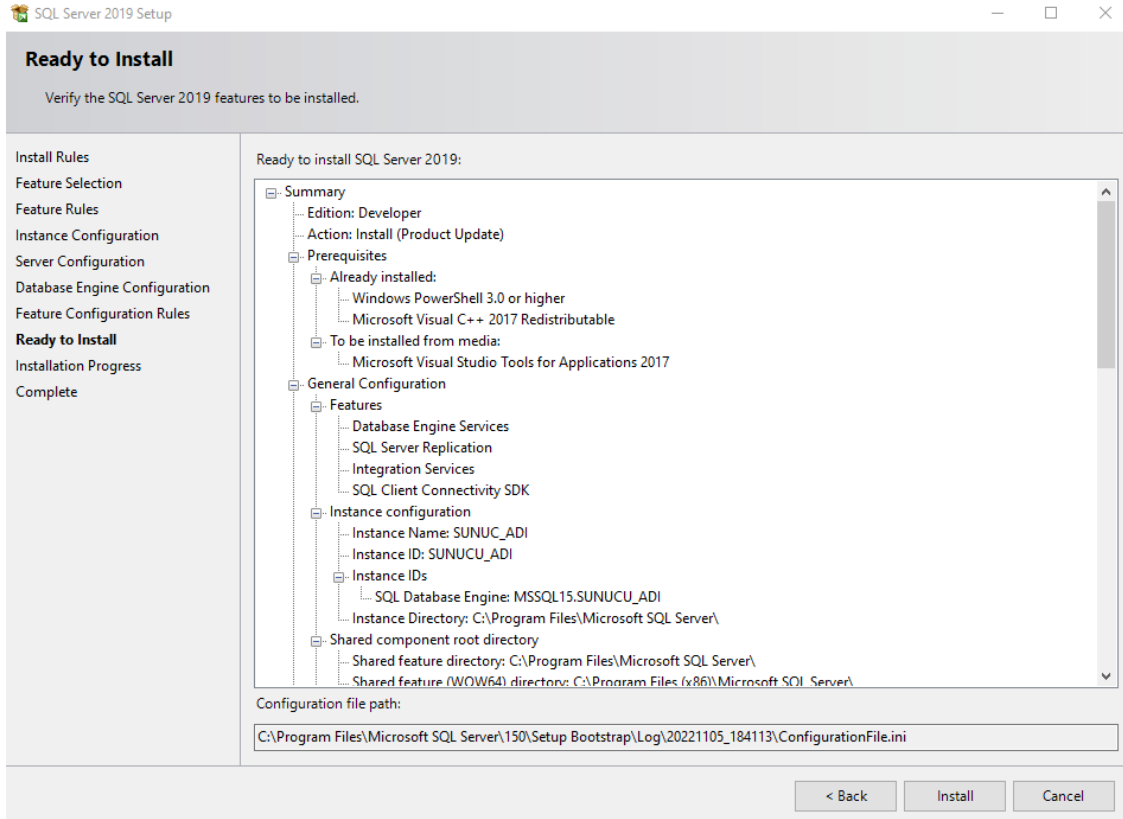
[Click here for details](#)

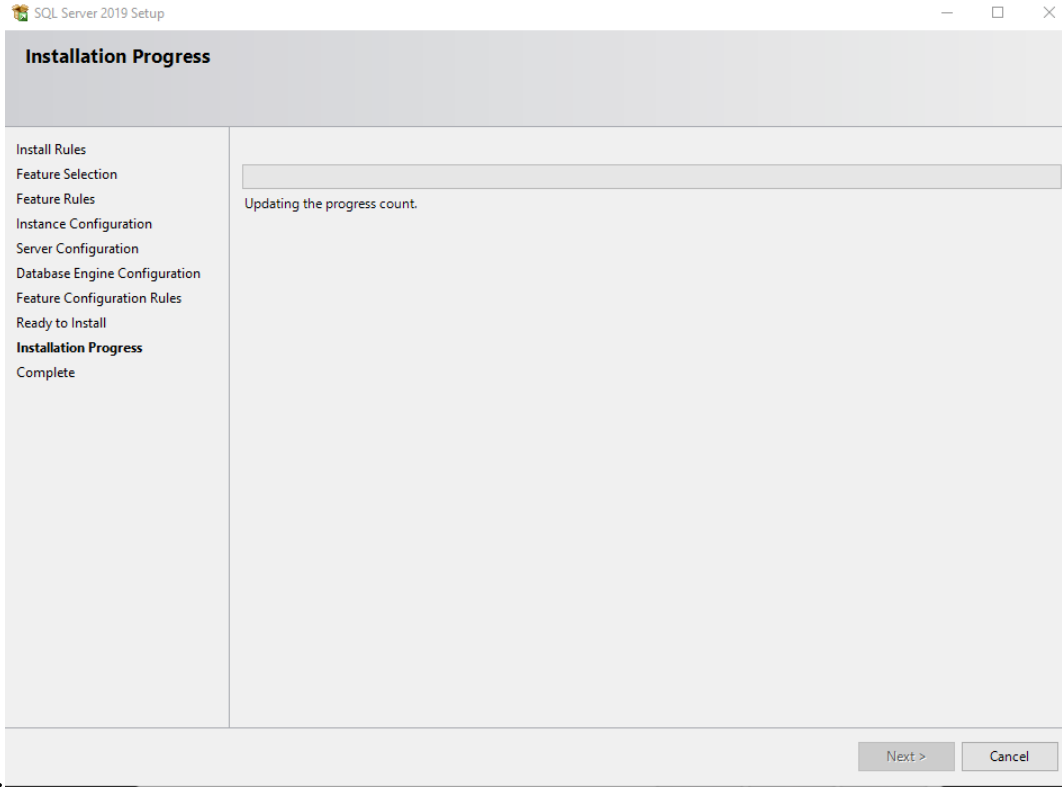
< Back Next > Cancel

20. "Windows authentication mode" seçeneğinin seçili olmasına dikkat edilir. "Add Current User" butonuna tıklanarak mevcut kullanıcı "Specify SQL Server administrators" listesine eklenir. "Next" butonuna tıklanır.



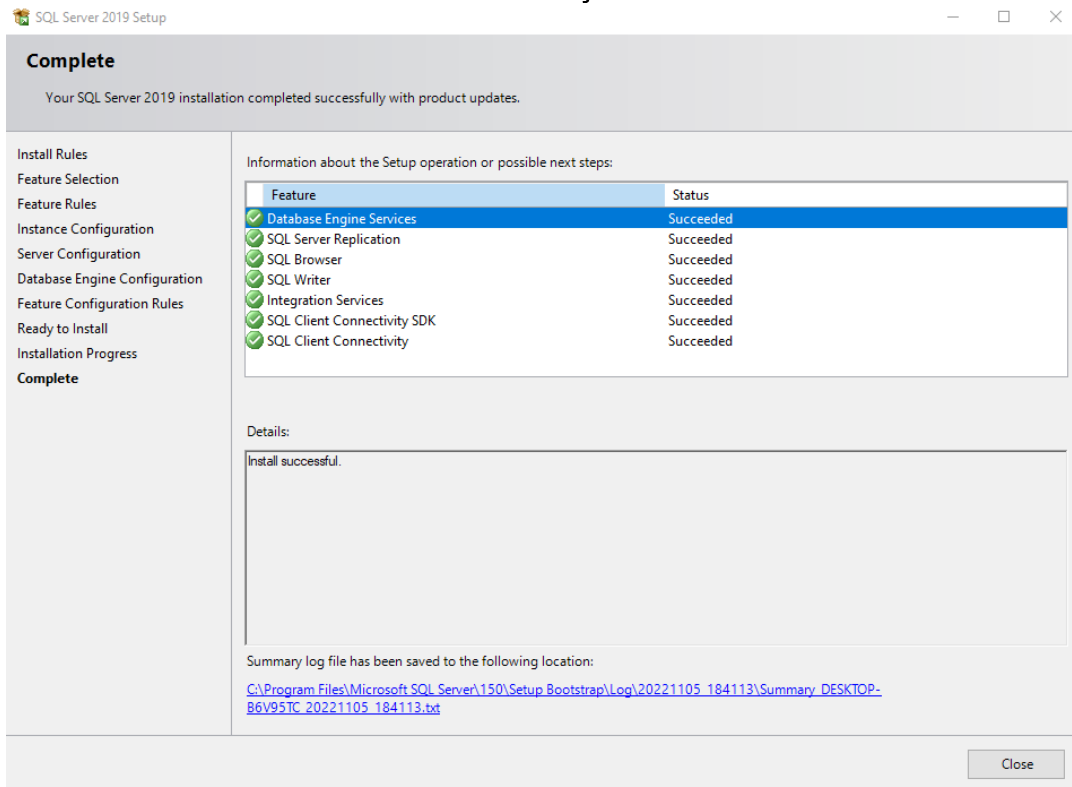
21. "Next" butonuna tıklanır.





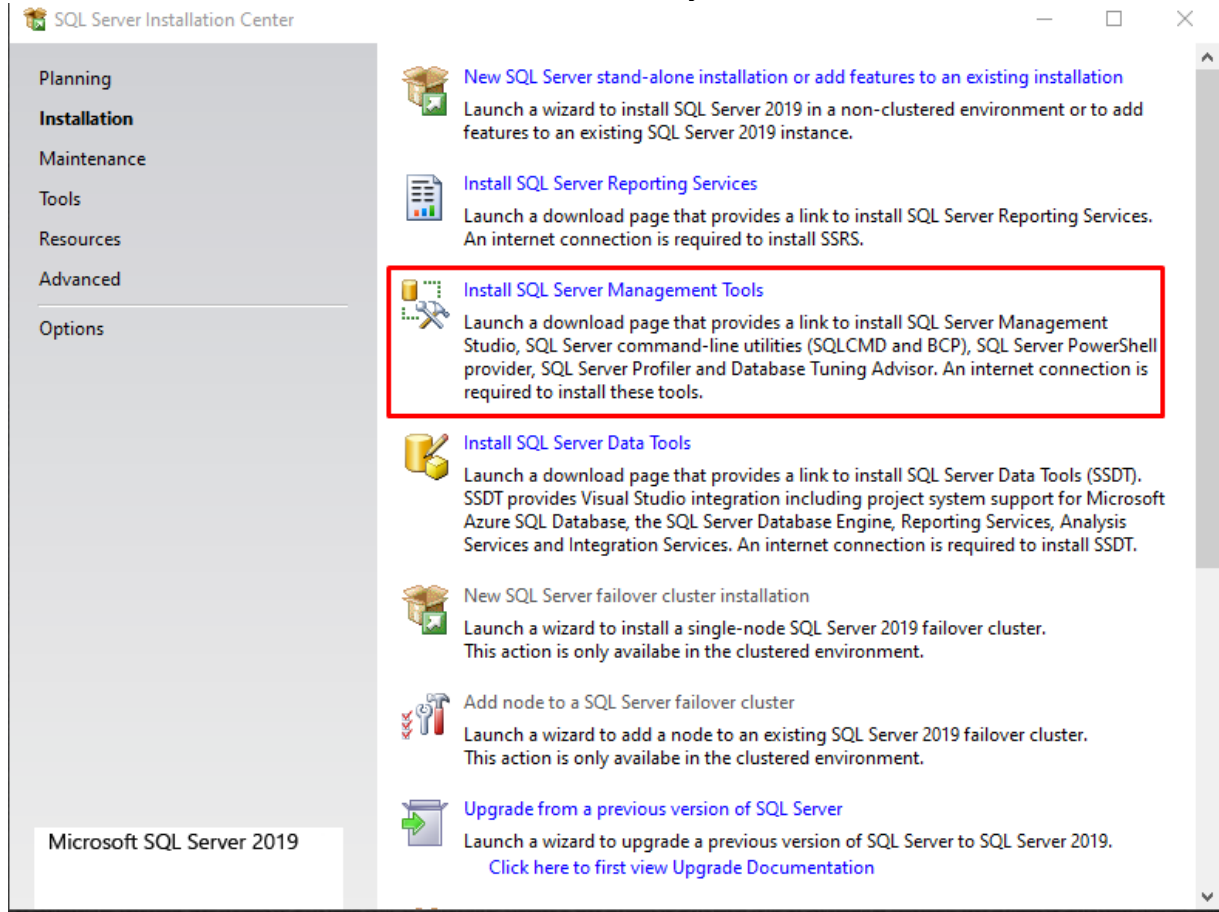
22.

23. SQL Server kurulumu tamamlanmış olur.

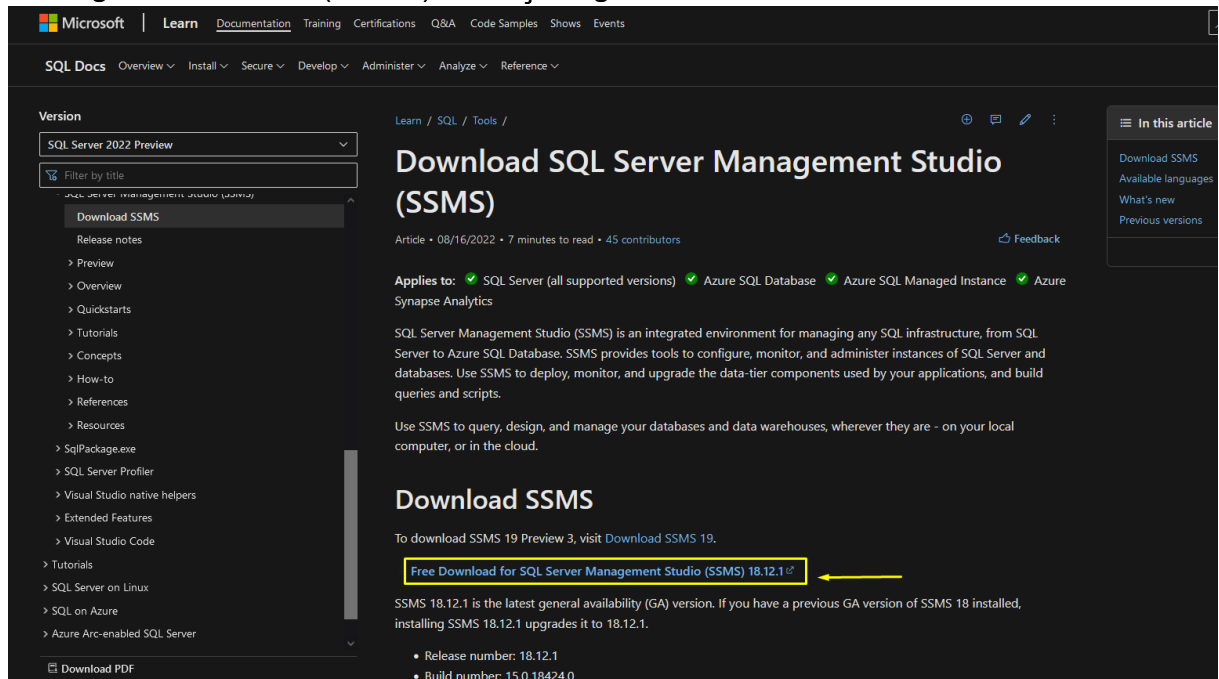


4. SQL SERVER MANAGEMENT STUDIO KURULUMU

1. "Install SQL Server Management Tools" seçeneğine tıklanır.



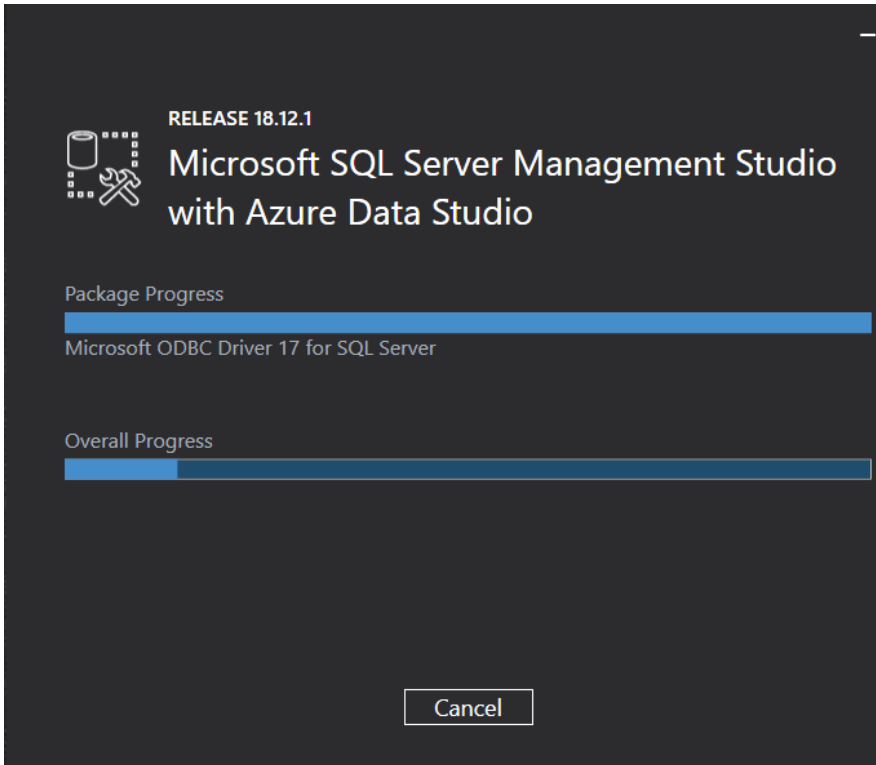
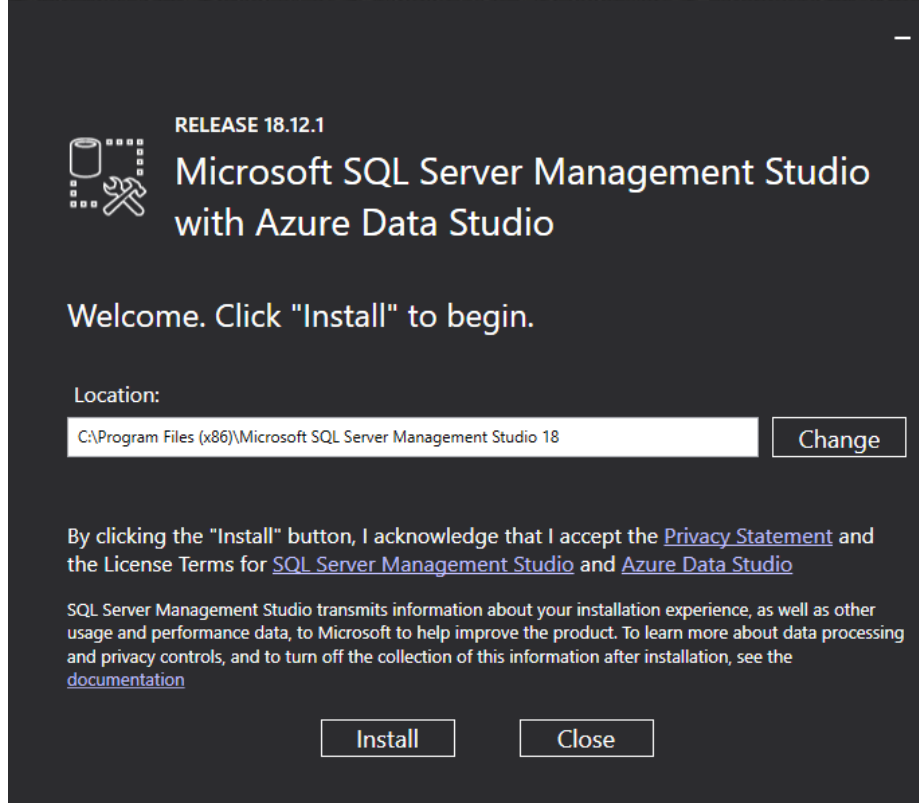
2. Tarayıcı da yönlendirilen bağlantıda "Free Download for SQL Server Management Studio (SMSS)..." seçeneğine tıklanır.



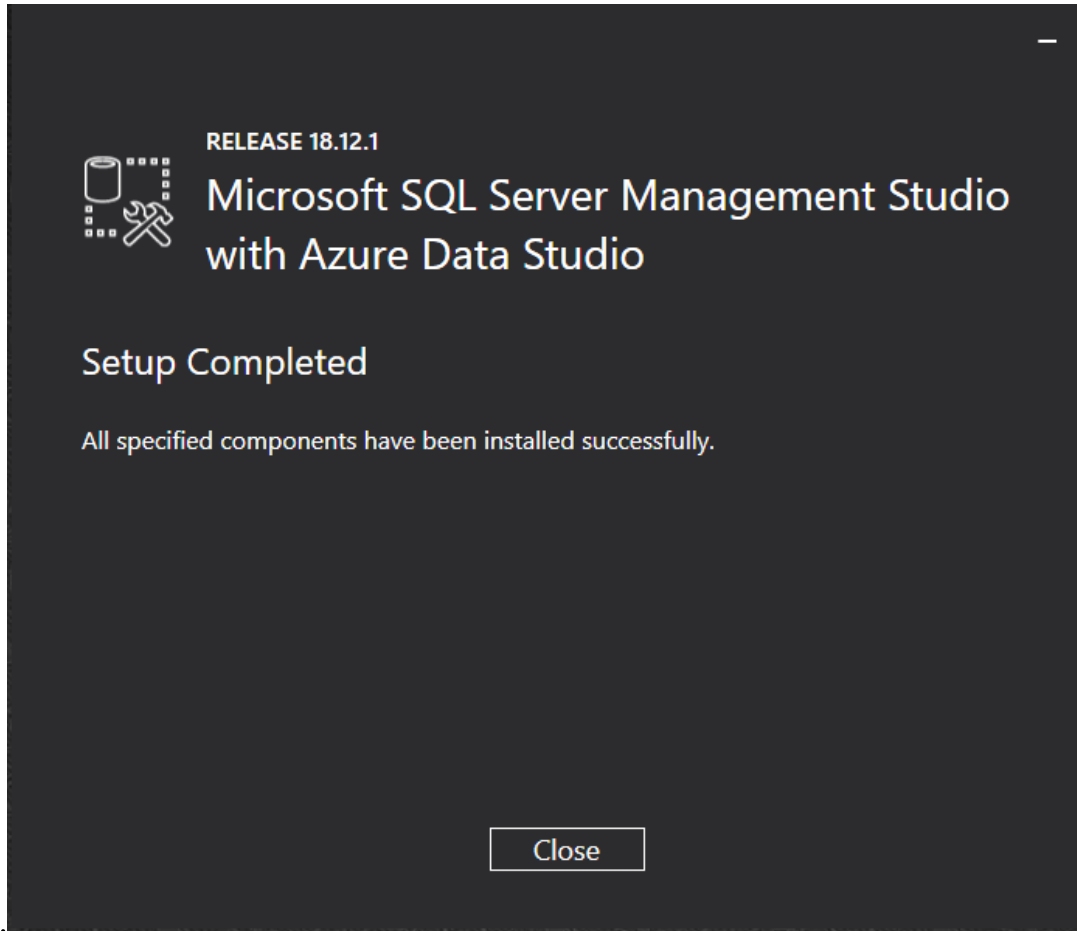


3.

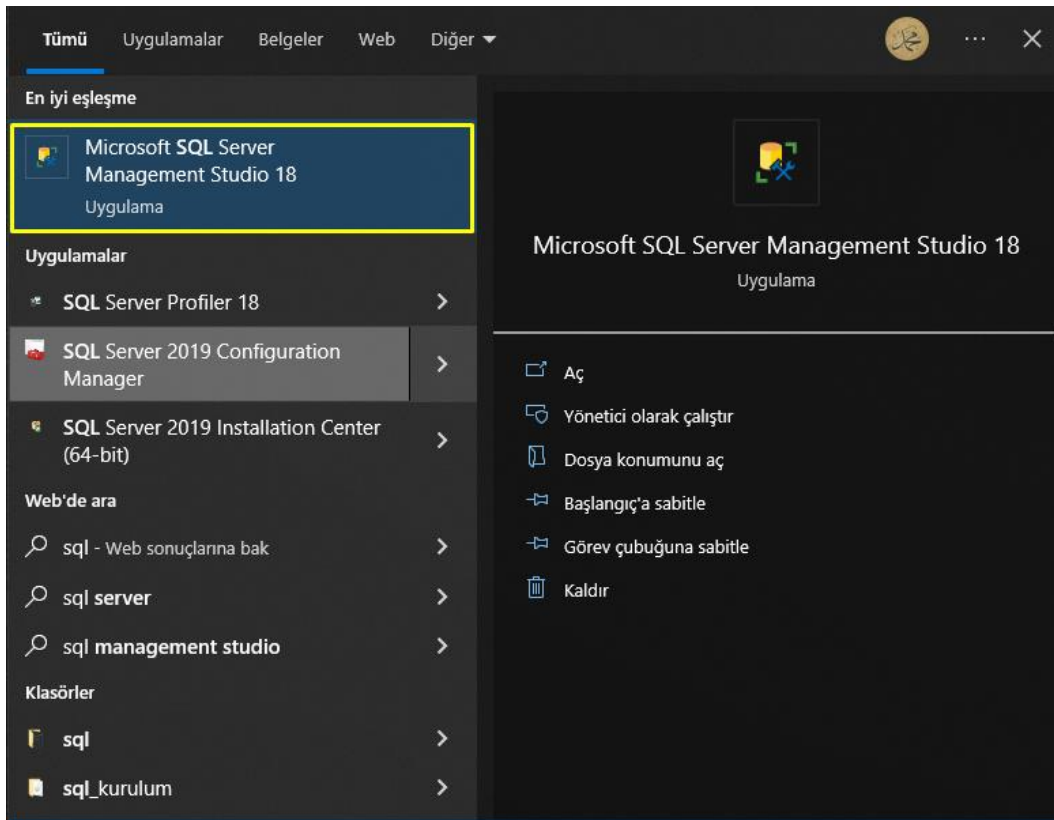
4. "Install" butonuna tıklanır.



5.

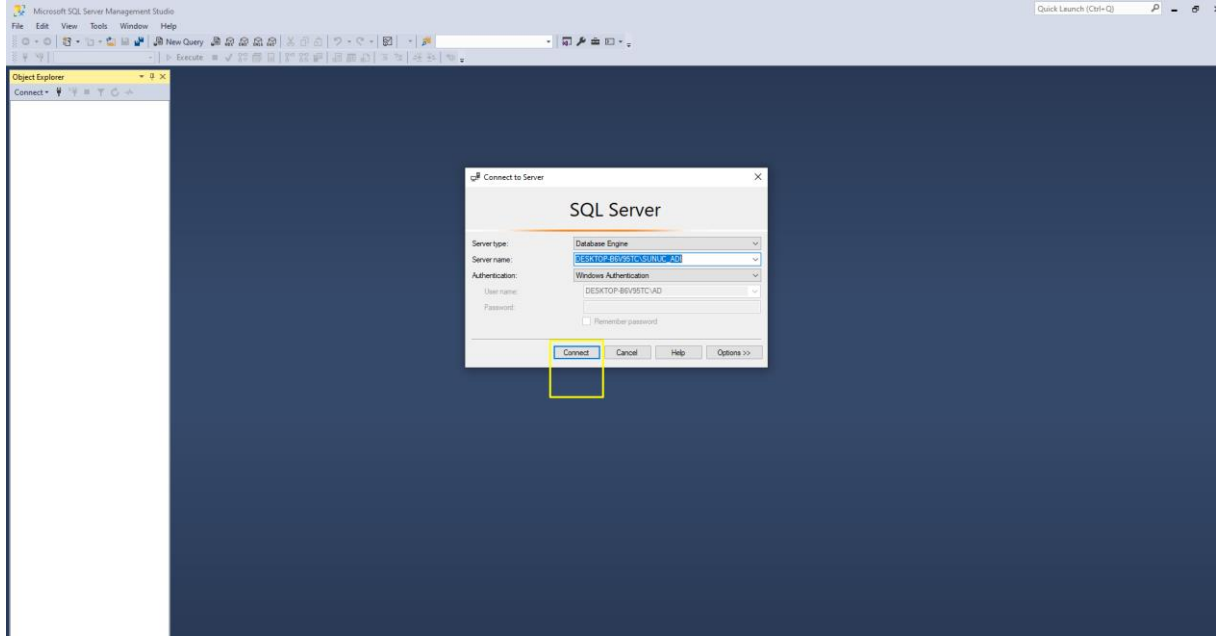


6.



7.

8. "Connect" butonuna tıklanarak veritabanına bağlantı sağlanır ve kullanıma hazır hale gelir.

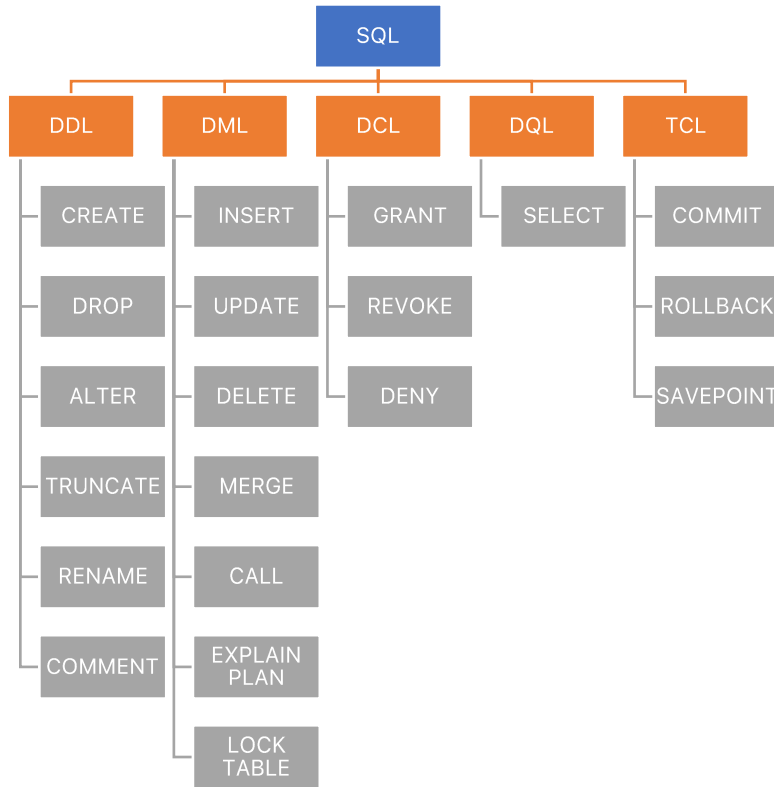


5.SQL (STRUCTURED QUERY LANGUAGE) - YAPISAL SORGULAMA DİLİ

Structured Query Language baş harflerinden oluşan SQL, ilişkisel veritabanları üzerinde verilere ulaşmak ve verileri işlemek üzerine yazılmış bir dildir. Türkçe karşılığı "Yapılandırılmış Sorgu Dili" dir. PHP, C, Java, Python gibi bir programlama dili değildir. Veritabanları için kullanılan alt dil olarak kabul edilir. Programlama dilleri için platform bağımsızdır. Yani hem Oracle, hem PHP, hem Python kullanılan üç farklı sistem aynı veritabanına bağlanabilir ve işlemler yapılabilir.

Veritabanı sistemlerinde SQL ile işlemler gerçekleştirilir. Bu sistemlere örnek verecek olursak;

- MySQL
- MsSQL
- Microsoft SQL Server
- PostgreSQL
- Oracle
- Access



Veritabanları satır ve sütun mantığı ile çalışır. Sütunlar/kolonlar feature (özellik) görevi görürken, satırlar verileri taşır. SQL ile veritabanı, tablolar, veriler üzerinde oluşturma, güncelleme, silme işlemleri yapılabilir.

SQL diline ait komutları kullanarak veritabanı üzerinde aşağıdaki işlemler yapılabilir:

- Veritabanı üzerinde tablolar oluşturmak, güncellemek ve silmek.
- Tablolar üzerinde feature (özellik)/Kolon-Sütun oluşturmak, güncellemek, silmek.
- Tablo içinde ki veriler üzerinde, oluşturma, güncelleme, silme işlemleri.
- Verileri çağırma işlemleri.

NOT: "CREATE", "UPDATE", "SELECT" ve "DELETE" komutları bazı kaynaklarda "CRUD (Create Read Update Delete)" kısaltması ile anılmaktadır.

5.0. Veri Türleri

Veri Tipi	Türü	Açıklama
char(n)	String	Unicode'u desteklemeyip char(n) şeklinde kullanılırlar. 8000 karaktere kadar değer alabilirler. Belirtilenden(n) az karakter girilse dahi giriş yapılan boyut kadar yer kaplar.
varchar(n)	String	Char'dan farklı olarak verinin boyutu kadar yere kaplar. 8000 karaktere kadar depolama yapar. Varchar (MAX) kullanımı ile 2GB'a kadar depolama yapılabilir.

text	String	Belirtilenden az değer girilse bile boyutu kadar yer kaplar.2GB'a kadar metinsel veri depolar. Unicode'u desteklemez.
Ntext	String	Text'den farklı olarak girilen karakter boyutu kadar yer kaplar ve unicode'u destekler.
nchar	String	Unicode (uluslararası karakter setini, tanımlı tüm alfabeleri içerir)destekler. Char'dan farklı olarak maksimum 4000 karaktere kadar değer alabilir.
nvarchar	String	Verinin boyutu kadar yer kaplar.Varchardan farklı olarak unicode'u destekler.4000 karaktere kadar değer alabilir. Unicode desteği bulunmaktadır.
bit	Boolean	0, 1 ve null değerini saklar.
tinyint	Number	0 ile 255 arasında değerleri saklar. 1 byte büyüklüğüne sahip, 0–255 arası tam sayı veriler için kullanılan tam sayı veri tipidir.
smallint	Number	32,768 ile 32,767 arasında değerleri saklar. 2 byte büyüklüğünde -32.768 ve 32.768 arası değer alabilen tam sayı veri tipidir.
int	Number	2,147,483,648 ile 2,147,483,647 arasında değerleri saklar. 4 byte büyüklüğünde, -2 milyar /+2 milyar arasında değer tutabilen tam sayı veri tipidir.
bigint	Number	9,223,372,036,854,775,808 ile 9,223,372,036,854,775,807 arasında değerleri saklar. 8 byte büyüklüğünde -2^{63} ve 2^{63} arasında değer tutabilen tam sayı veri tipidir.
datetime	Date	YYYY-AA-GG şeklinde tarih ve zaman verilerini tutan 8 byte uzunluğunda veri tipidir. 1 Ocak 1753 – 31 Aralık 9999. 3.33 milisaniye doğruluk hassasiyeti vardır.
smalldatetime	Date	1 Ocak 1900 – 6 Haziran 2079. 1 dakikalık doğruluk hassasiyeti vardır. Tarih ve zaman verilerini yıl-ay-gün ve saat-dakika-saniye-salise şekilde saklar. 4 byte uzunluğunda veri tipidir.
date	Date	1 Ocak 0001 – 31 Aralık 9999. Sadece tarih içerir, saati saklamaz. Tarihleri YYYY-AA-GG (yıl-ay-gün) formatında saklar. 3 byte uzunluğunda veri tipidir.
money	Number	8 byte boyutunda, yaklaşık -2^{64} ile 2^{64} arasında parasal değerleri tutmak için kullanılır. 4 basamağa kadar duyarlı ondalık tipli verileri saklar.
smallmoney	Number	4 byte uzunluğunda yaklaşık -214.000 ile 214.000 arası parasal değerleri tutmak için kullanılır.Money tipinde olduğu gibi 4 basamağa kadar duyarlı ondalık tipli verileri saklarken kullanılır.

5.1. DQL (Data Query) – Veri Sorgulama

Veri sorgulama işlemleri için kullanılır.

5.1.1 Select

Veritabanındaki kayıtları sorgulama ve listeleme için kullanılır.

```
-- Syntax for SQL Server and Azure SQL Database
```

```
SELECT [KOLONLAR] FROM [TABLO_ADI]
```

```
-- ogrenciler tablosu
```

```
/*
```

id	ad	soyad	yas
1	Ali	Yılmaz	19
2	Ayşe	Demir	20

```
*/
```

```
-- Örnek - 1
```

```
SELECT * FROM ogrenciler
```

```
/* Örnek - 1 Çıktı
```

id	ad	soyad	yas
1	Ali	Yılmaz	19
2	Ayşe	Demir	20

```
*/
```

```
-- Örnek - 2
```

```
SELECT ad,soyad,yas FROM ogrenciler
```

```
/* Örnek - 2 Çıktı
```

ad	soyad	yas
Ali	Yılmaz	19
Ayşe	Demir	20

```
*/
```

```
-- Örnek - 3
```

```
SELECT id FROM ogrenciler
```

```
/* Örnek - 3 Çıktı
```

id
1
2

```
*/
```

```
-- NOT: * ifadesi tablodaki tüm kolonları/sütunları kapsamaktadır. SELECT ifadesinde kullanıldığında tüm kolonları listeler.
```

“SELECT” ifadesi ile tablodan kayıt çekildiğinde bu kayıtların hangi sıralamada geleceğini belirlemek için “ORDER BY” parametresi kullanılır.

```
-- Syntax
```

```
SELECT [KOLONLAR] FROM [TABLO_ADI] ORDER BY [KOLON_ADI]
```

```
SELECT [KOLONLAR] FROM [TABLO_ADI] ORDER BY [KOLON_ADI] DESC, [KOLON2] ASC
```

```
SELECT [KOLONLAR] FROM [TABLO_ADI] ORDER BY [KOLON_ADI] ASC
```

```
-- ORDER BY ifadesinden sonra yer alan [KOLON_ADI] sütunu referans alınarak sıralama gerçekleştirilir.
-- ORDER BY [KOLON_ADI] şeklinde kullanılırsa varsayılan olarak ASC (Ascending - Artan) olarak sıralar.
-- ORDER BY [KOLON_ADI] DESC (Descending - Azalan) şeklinde kullanılırsa ilgili kolonun değerini en yüksekten en düşüğe doğru sıralar.
```

```
/* kullanıcı tablosu
```

id	ad	soyad	yas
1	Ali	Yılmaz	19
2	Ahmet	Mutlu	18
3	Sare	Aygün	21
4	Ayşe	Demir	25
5	Abdülmelik	Derinkök	40

```
*/
```

```
-- Örnek - 1
```

```
SELECT * FROM kullanıcı ORDER BY yas
```

```
SELECT * FROM kullanıcı ORDER BY yas ASC
```

```
/* Örnek - 1 Çıktı
```

id	ad	soyad	yas
2	Ahmet	Mutlu	18
1	Ali	Yılmaz	19
3	Sare	Aygün	21
4	Ayşe	Demir	25
5	Abdülmelik	Derinkök	40

```
*/
```

```
-- Örnek - 2
```

```
SELECT * FROM kullanıcı ORDER BY yas DESC
```

```
/* Örnek - 1 Çıktı
```

id	ad	soyad	yas
5	Abdülmelik	Derinkök	40
4	Ayşe	Demir	25
3	Sare	Aygün	21
1	Ali	Yılmaz	19
2	Ahmet	Mutlu	18

```
*/
```

"SELECT" sorgusu kullanılırken bazı durumlarda tüm kayıtları çekmek yerine sayılı kaydın getirilmesi gerekebilir. Böyle durumlar için "TOP" ifadesi "SELECT" sorgusu ile birlikte kullanılır.

DİKKAT: "TOP" ifadesi, SQL dilinin MSSQL/T-SQL implementasyonunda kullanılmaktadır. Yaygın olarak kullanılan MySQL implementasyonunda aynı işlem için "LIMIT" ifadesi kullanılmaktadır. Başka SQL implementasyonlarında da farklılık gösterebilir.

```
-- Syntax
```

```
SELECT TOP [N] [KOLONLAR] FROM [TABLO_ADI]
```

```
-- Tablodan yalnızca [N] adet kaydı getirir.
```

```
/* kullanıcı tablosu
```

id	ad	soyad	yas
1	Ali	Yılmaz	19
2	Ahmet	Mutlu	18
3	Sare	Aygün	21

```

| 4 | Ayşe | Demir | 25 |
| 5 | Abdülmelik | Derinkök | 40 |
*/
-- Örnek - 1

SELECT TOP 1 * FROM kullanıcı
/* Örnek - 1 Çıktı
| id | ad | soyad | yas |
| 1 | Ali | Yılmaz | 19 |
*/

-- Örnek - 2

SELECT TOP 3 * FROM kullanıcı
/* Örnek - 2 Çıktı
| id | ad | soyad | yas |
| 1 | Ali | Yılmaz | 19 |
| 2 | Ahmet | Mutlu | 18 |
| 3 | Sare | Aygün | 21 |
*/

```

"SELECT" ifadesi kullanılırken kolonda tekrar eden aynı verilerin gelmemesi istenebilir. Örneğin sınıf kolonu birçok kayıt aynı olabilir ve hangi sınıfların bulunduğu bilgisini elde etmek için tekrar eden verilerin listelenmesi istenmeyen bir durumdur. Böyle durumlar için "DISTINCT" ifadesi kullanılmaktadır.

```

-- Syntax
SELECT DISTINCT [KOLON] FROM [TABLO_ADI]

/* ogrenciler tablosu
| id | ad | soyad | sinif | vize | final |
| 1 | Ali | Yılmaz | 1 | 65 | 82 |
| 2 | Ahmet | Mutlu | 1 | 30 | 45 |
| 3 | Asel Sare | Aygün | 2 | 85 | 79 |
| 4 | Ayşe | Demir | 2 | 55 | 63 |
| 5 | Abdülmelik | Derinkök | 2 | 100 | 100 |
| 5 | Muhammed Yağız | Mutlu | 2 | 95 | 68 |
| 5 | Elif Eylül | Mutlu | 2 | 90 | 90 |
*/

-- Örnek - 1

SELECT DISTINCT sinif FROM ogrenciler
/* Örnek - 1 Çıktı
| sinif |
| 1 |
| 2 |
*/

```

"SELECT" komutu veya kullanılabilir diğer komutlarda koşul belirtme ihtiyacı olabilir. Örneğin, yaş özelliği x'den büyük kayıtlar, ad özelliği "a" ile başlayan kayıtlar, ad özelliği "Abdölmelik" ve soyad özelliği "Derinkök" olan

kayıtlar. Bu gibi durumlarda "SELECT" komutu için "WHERE" ifadesi eklenerek istenen koşula uygun kayıtlar sorgulanabilir.

```
-- Syntax
SELECT [KOLONLAR] FROM [TABLO_ADI] WHERE [KOŞUL]
SELECT [KOLONLAR] FROM [TABLO_ADI] WHERE [KOŞUL] AND [KOŞUL2]
SELECT [KOLONLAR] FROM [TABLO_ADI] WHERE [KOŞUL] OR [KOŞUL2]

-- Birden fazla koşul belirtmek için "AND" ya da "OR" kullanılır.
-- WHERE ifadesi ile istenildiği kadar koşul belirtilebilir.
-- AND kullanımında belirtilen koşulların tümünü sağlayan kayıtlar getirilir.
-- OR kullanımında koşullardan herhangi birini sağlayan tüm kayıtlar getirilir.

-- kullanıcı tablosu
/*
| id | ad      | soyad   | yas |
| 1  | Ali     | Yılmaz  | 19  |
| 2  | Ahmet   | Mutlu   | 18  |
| 3  | Sare    | Aygün   | 21  |
| 4  | Ayşe    | Demir   | 25  |
| 5  | Abdülmelik | Derinkök | 40  |
*/
-- Örnek - 1

SELECT * FROM kullanıcı WHERE ad='Sare'

/* Örnek - 1 Çıktı
| id | ad      | soyad   | yas |
| 3  | Sare    | Aygün   | 21  |
*/

-- Örnek - 2

SELECT * FROM kullanıcı WHERE yas>20

/* Örnek - 2 Çıktı
| id | ad      | soyad   | yas |
| 3  | Sare    | Aygün   | 21  |
| 4  | Ayşe    | Demir   | 25  |
| 5  | Abdülmelik | Derinkök | 40  |
*/

-- Örnek - 3

SELECT * FROM kullanıcı WHERE id=1

/* Örnek - 3 Çıktı
| id | ad      | soyad   | yas |
| 1  | Ali     | Yılmaz  | 19  |
*/

--Örnek - 4
SELECT * FROM kullanıcı WHERE id=1 AND yas=20

/* Örnek - 4 Çıktı
| id | ad      | soyad   | yas |
--ŞARTLARA UYGUN KAYIT OLMADIĞINDAN BOŞ DÖNDÜRÜR--
*/
```

```
--Örnek -5
SELECT * FROM kullanıcı WHERE id=1 OR yas=40
```

```
/* Örnek - 5 Çıktı
| id | ad      | soyad   | yas |
| 1  | Ali     | Yılmaz  | 19  |
| 5  | Abdülmelik | Derinkök | 40  |
*/
```

“WHERE” komutu ile “SELECT” sorgusu üzerinde filtreleme işlemi yapılabilir de bazı durumlarda yalnızca “WHERE” kullanımı ihtiyacı karşılayamamaktadır. Örneğin, ad özelliği “A” ile başlayan kayıtlar, soyad özelliği “Y” ile başlayan ve 5 karakter olan kayıtlar gibi sorgulamaları yalnızca “WHERE” ifadesi kullanarak gerçekleştiremeyiz. Bu gibi işlemler için “WHERE” ifadesi ile birlikte kullanılan “LIKE” parametresini ekleriz.

“LIKE” ifadesinin parametresi String veri türündedir. Filtreleme yapılacak desen ‘DESEN’ şeklinde kullanılmalıdır. Desen içerisinde kullanılacak “Wildcard” (Joker karakter) semboller bulunmaktadır, bunlar:

Sembol	Açıklama
%	Sıfır veya daha fazla karakteri temsil eder. (A%)
_	Tek bir karakteri temsil eder. (A_____k)
[]	Parantez içindeki herhangi bir karakteri temsil eder. [xyz]
^	Parantez içinde olmayan herhangi bir karakteri temsil eder. (^a)
-	Bir dizi karakteri temsil eder. (x-y)

“Wildcard” sembolleri birbirleri ile kombine edilerek yazılabilirler.

```
-- Syntax
```

```
SELECT [KOLONLAR] FROM [TABLO_ADI] WHERE [ARANACAK_KOLON] LIKE '[ARAMA_DESENI]'
-- WHERE ifadesinden sonraki "[ARANACAK_KOLON] LIKE '[ARAMA_DESENI]'" alanından
sonra çoklu filtreleme yapılacaksa AND/OR bağlaçları kullanılabilir.
```

```
-- il tablosu
```

```
/*
| id | ad      |
| 1  | Adana   |
| 2  | Adıyaman |
| 3  | Afyonkarahisar |
...
| 11 | Bilecik |
...
| 44 | Malatya |
...
| 81 | Düzce   |
*/
```



```

--Örnek - 1
SELECT * FROM il WHERE ad LIKE 'Bi%'
-- Örnek 1: ad özelliği "Bi" ile başlayan kayıtları çeker.
/* Örnek - 1 Çıktı
| id | ad      |
| 11 | Bilecik |
| 12 | Bingöl  |
| 13 | Bitlis  |

*/

--Örnek - 2
SELECT * FROM il WHERE ad LIKE 'MA%A'
-- Örnek 2: ad özelliği "Ma" ile başlayan ve "a" ile biten kayıtları çeker.
/* Örnek - 2 Çıktı
| id | ad      |
| 44 | Malatya |
| 45 | Manisa  |

*/

--Örnek - 3
SELECT * FROM il WHERE ad LIKE 'MA____A'
-- Örnek 3: ad özelliği "Ma" ile başlayan ve "a" ile biten ve uzunluğu 7 karakter
olan kayıtları çeker.
/* Örnek - 3 Çıktı
| id | ad      |
| 44 | Malatya |

*/

--Örnek - 4
SELECT * FROM il WHERE ad LIKE '%E_A%'
-- Örnek 4: ad özelliğinin herhangi bir yerinde "e" ile başlayıp "a" ile biten 3
karakter uzunluğunda ifade barındıran kayıtları çeker.
/* Örnek - 4 Çıktı
| id | ad      |
| 23 | Elazığ  |

*/

--Örnek -5
SELECT * FROM il WHERE ad LIKE '%ğ%'
-- Örnek 5: ad özelliğinde "ğ" bulunan kayıtları çeker.
/* Örnek -5 Çıktı
| id | ad      |
| 4  | Ağrı    |
| 23 | Elazığ  |
| 48 | Muğla   |
| 51 | Niğde   |
| 59 | Tekirdağ |
| 76 | Iğdır   |

*/

--Örnek -6
SELECT * FROM il WHERE ad LIKE '%[ke]%'
-- Örnek 6: ad özelliği "k" veya "e" ile biten kayıtları çeker.
/* Örnek - 6 Çıktı
| id | ad      |
| 11 | Bilecik |

```

```

|17| Çanakkale |
|22| Edirne   |
|29| Gümüşhane|
|51| Niğde     |
|53| Rize      |
|64| Uşak      |
|67| Zonguldak |
|71| Kırıkkale |
|73| Şırnak    |
|78| Karabük   |
|80| Osmaniye  |
|81| Düzce     |
*/
-- Örnek - 7
SELECT * FROM il WHERE ad LIKE '^a-y%'
-- Örnek 7: ad özelliği "a-y" aralığındaki karakterlerle BAŞLAMAYAN kayıtları çeker.
/* Örnek - 7 Çıktı
| id | ad       |
| 77 | Zonguldak|
*/

```

"WHERE" ifadesinin kullanımında sayısal iki veri arasındaki kayıtların listelenmesi için "WHERE [kolon1] < x AND [kolon1] > y" şeklinde bir filtreleme yapılabilir. Bu türde bir işlem için "WHERE" ifadesinde kullanılmak üzere "BETWEEN" ifadesi de kullanılabilir.

```

-- Syntax
SELECT [KOLONLAR] FROM [TABLO_ADI] WHERE [KOLON_ADI] BETWEEN [ALT_LIMIT] AND
[UST_LIMIT]

-- BETWEEN ifadesinde belirtilen alt ve üst limit değerleri de dahil edilerek
filtreleme gerçekleştirilir.
-- kullanıcı tablosu
/*
| id | ad       | soyad  | yas |
| 1  | Ali      | Yılmaz | 19  |
| 2  | Ahmet    | Mutlu  | 18  |
| 3  | Sare     | Aygün  | 21  |
| 4  | Ayşe     | Demir  | 25  |
| 5  | Abdülmelik | Derinkök | 40  |
*/
-- Örnek - 1

SELECT * FROM kullanıcı WHERE yas BETWEEN 20 AND 40

/* Örnek - 1 Çıktı
| id | ad       | soyad  | yas |
| 3  | Sare     | Aygün  | 21  |
| 4  | Ayşe     | Demir  | 25  |
| 5  | Abdülmelik | Derinkök | 40  |
*/

```

Metinsel ifadelerin filtrelenebilmesi için "WHERE [kolon]='değer'" kalıbı kullanılır. Ancak çoklu metin ifadelerini filtrelemek için "OR" kullanmak yazılan

kodun pratiklik ve okunabilirliğini düşürecektir. Bu türde işlemler için de "IN" parametresi kullanılmalıdır.

```
-- Syntax
SELECT [KOLONLAR] FROM [TABLO_ADI] WHERE [KOLON_ADI]
IN([veri1],[veri2],[veri3],[veriN])

-- IN() ifadesi yalnızca metinsel işlemlerde kullanılmayabilir.
-- IN() ifadesi içerisine yazılan ifadeler aynı veri türünden olmalıdır.

-- kullanıcı tablosu
/*
| id | ad      | soyad   | yas |
| 1  | Ali     | Yılmaz  | 19  |
| 2  | Ahmet   | Mutlu   | 18  |
| 3  | Sare    | Aygün   | 21  |
| 4  | Ayşe    | Demir   | 25  |
| 5  | Abdülmelik | Derinkök | 40  |
*/
-- Örnek - 1

SELECT * FROM kullanıcı WHERE ad IN('Sare','Abdülmelik')
/* Örnek - 1 Çıktı
| id | ad      | soyad   | yas |
| 3  | Sare    | Aygün   | 21  |
| 5  | Abdülmelik | Derinkök | 40  |
*/
-- Örnek - 2

SELECT * FROM kullanıcı WHERE yas IN(18,19,40)
/* Örnek - 2 Çıktı
| id | ad      | soyad   | yas |
| 1  | Ali     | Yılmaz  | 19  |
| 2  | Ahmet   | Mutlu   | 18  |
| 5  | Abdülmelik | Derinkök | 40  |
*/
```

Bazı durumlarda kayıtların gruplanması gerekebilir. Örneğin, öğrenci kayıtlarının tutulduğu bir tablo ve bu tabloda id, ad, soyad, sınıf sütunları bulunsun. Hangi sınıfta kaç öğrenci mevcut sorusunun cevabı ve bunun gibi örnek durumlar için "GROUP BY" ifadesi kullanılmaktadır. Aynı zamanda "GROUP BY" ifadesine konu olan veri grubunun filtreleme işlemi için de "HAVING" ifadesi kullanılmaktadır. "GROUP BY" ifadesi "AGGREGATE" fonksiyonlar ile birlikte kullanılabilir. "AGGREGATE" fonksiyonlar tablodaki ilgili sütunun tüm kayıtları ile işlem yapılan fonksiyonlardır. "Aggregate" fonksiyonları aşağıda belirtilmiştir:

SUM: Kolondaki değerlerin kümülatif(toplam) ifadesini verir.

AVG: Kolondaki ifadelerin ortalamasını verir.

MAX: Kolondaki ifadelerin en yüksekini verir.

MIN: Kolondaki ifadelerin en düşüğünü verir.

COUNT: Kolondaki ifadelerin adet bilgisini verir.

-- Aggregate fonksiyonlar

/* ogrenciler tablosu

id	ad	soyad	sinif	vize	final
1	Ali	Yılmaz	1	65	82
2	Ahmet	Mutlu	1	30	45
3	Asel Sare	Aygün	2	85	79
4	Ayşe	Demir	2	55	63
5	Abdülmelik	Derinkök	2	100	100
6	Muhammed Yağız	Mutlu	2	95	68
7	Elif Eylül	Mutlu	2	90	90

*/

-- Örnek - 1 COUNT

```
SELECT COUNT(*) FROM ogrenciler
SELECT COUNT(ad) FROM ogrenciler
```

--Örnek -1 Açıklama: Tablodaki kayıt sayısını döndürür. Örnek tabloda toplam 7 kayıt bulunmaktadır. Örnek 1'de yer alan sorgularda herhangi bir koşul belirtilmediğinden COUNT fonksiyonunun içine hangi değer girilirse girilsin aynı sonucu verecektir.

/* Örnek - 1 Çıktı

(No Column Name)

7

*/

-- Örnek - 2 SUM

```
SELECT SUM(vize) FROM ogrenciler
```

--Örnek - 2 Açıklama: Tablodaki vize sütununda yer alan tüm kayıtları toplar ve sonucu ekrana yazdırır. İlgili tablodaki vize sütununda yer alan kayıtların toplamı 520'dir.

/* Örnek - 2 Çıktı

(No Column Name)

520

*/

-- Örnek - 3 MIN

```
SELECT MIN(vize) FROM ogrenciler
```

--Örnek - 3 Açıklama: Tablodaki vize sütununda yer alan tüm kayıtları karşılaştırır ve en düşük olan değeri yazdırır. İlgili tablodaki en düşük vize değeri 30'dur.

/* Örnek - 3 Çıktı

(No Column Name)

30

*/

-- Örnek - 4 MAX

```
SELECT MAX(final) FROM ogrenciler
```

--Örnek - 4 Açıklama: Tablodaki final sütununda yer alan tüm kayıtları karşılaştırır ve en yüksek olan değeri yazdırır. İlgili tablodaki en yüksek final değeri 100'dür.

```

/* Örnek - 4 Çıktı
(No Column Name)
100
*/

-- Örnek - 5 AVG

SELECT AVG(final) FROM ogrenciler
--Örnek - 5 Açıklama: Tablodaki final sütununda yer alan tüm kayıtları toplar ve
sattır sayısına böler yani aritmetik ortalamasını alır. İlgili tablodaki final
kolonunda yer alan değerlerin ortalaması değeri 75'tir.
-- 75,xx şeklinde bir sonuç olmasına rağmen elde edilen ifade tam sayı olarak ekrana
yazdırılmıştır.

/* Örnek - 5 Çıktı
(No Column Name)
75
*/

-- Syntax GROUP BY
SELECT [KOLON_X],AGGREGATE_FONKSIYON([KOLON]) FROM [TABLO_ADI] GROUP BY [KOLON_X]
SELECT [KOLON_X],AGGREGATE_FONKSIYON([KOLON]) FROM [TABLO_ADI] GROUP BY [KOLON_X]
HAVING AGGREGATE_FONKSIYON([KOLON]) [OPERATOR] [KOSUL]

-- AGGREGATE_FONKSIYON alanına SUM,MIN,MAX,AVG,COUNT ifadeleri getirilebilir.
-- [KOLON_X] ifadesi gruplandırılacak kolon manasına gelmektedir.
-- HAVING ifadesinde yer alan [OPERATOR] <,>,<=>,<=>,<=> ifadelerinde biri olabilir.
-- Örnek HAVING COUNT(*) < 5
-- Eğer sorguda WHERE ifadesi kullanılacaksa GROUP BY ifadesinden önce yazılmalıdır.

--Örnek 6
SELECT sınıf,COUNT(*) FROM ogrenciler GROUP BY sınıf

--Örnek -6 Açıklama: sınıf kolonuna göre gruplandırma işlemi bulunmaktadır. sınıf
kolonu 1 değerinde olan 2 kayıt ve sınıf kolonu 2 değerinde olan 5 kayıt
bulunmaktadır.

/*Örnek - 6 Çıktı

| sınıf |(No Column Name)|
| 1 | 2 |
| 2 | 5 |

*/

--Örnek 6
SELECT sınıf,AVG(vize) FROM ogrenciler GROUP BY sınıf HAVING AVG(vize) > 50

--Örnek -6 Açıklama: sınıf kolonuna göre gruplandırma işlemi bulunmaktadır. Sınıf
kolonu aynı olan kayıtların vize notlarının aritmetik ortalaması alınır ve having
ifadesinde de ortalama değeri 50'den büyük olan sınıflar gösterilir.
-- sınıf değeri 1 olan kayıtların vize ortalaması 47 olduğu için şarta uygunluk
taşımaz bu nedenle de listelenmez.
-- sınıf değeri 2 olan kayıtların vize ortalaması ise 85'tir ve şarta uygunluk
sağladığı için listelenir.

/*Örnek - 6 Çıktı

| sınıf |(No Column Name)|
| 2 | 85 |

```

SELECT sorgusu kullanarak tek tablo ile veri sorgulama işlemleri şu anda kadar farklı şekillerde gösterildi ancak birden fazla tablonun gösterilmesi gereken durumlarda olabilir. Örneğin, personel bilgilerinin tutulduğu "personel" tablosu ve "personel" tablosundaki kayıtlarla ilişkili verilerin bulunduğu "personel_detay" tablosunun birleştirilerek sorgulamaların yapılması. Bu gibi ilişkisel ya da ilişkisi bulunmayan tabloları birleştirerek tek tablo ile çıktı almak için "JOIN" yapısı kullanılmaktadır.

JOIN yapısının üç farklı kullanımı vardır:

- Inner Join
- Outer Join (left, right, full)
- Cross Join

INNER JOIN: İki tabloda eşleşen verileri tek tabloda birleştirerek getirir.

OUTER JOIN: İki tabloda eşleşen verileri tek tabloda getirir ancak eşleşmeyen verileri de getirir. LEFT, RIGHT ve FULL olmak üzere üç ayrı kullanımı vardır. LEFT JOIN yapısı kullanıldığında JOIN ifadesinin solundaki tabloda bulunan tüm veriler getirilir ve sağda bulunan tabloda eşleşmeyen ifadeler NULL olarak görüntülenir. RIGHT JOIN yapısı kullanıldığında JOIN ifadesinin sağındaki tüm veriler getirilir ve JOIN ifadesinin solunda bulunan tablodaki eşleşmeyen ifadeler NULL olarak görüntülenir. FULL JOIN yapısı kullanıldığında ise JOIN ifadesinin sağında ve solunda bulunan tabloda eşleşen veya eşleşmeyen tüm veriler getirilmektedir.

CROSS JOIN: Belirtilen tabloların verilerinin kartezyen çarpımı şeklinde çıktısını verir.

```
--syntax

--INNER JOIN
SELECT * FROM [TABLO_1] INNER JOIN [TABLO_2] ON [TABLO_1].[İLİSKİLİ_KOLON] =
[TABLO_2].[İLİSKİLİ_KOLON]
SELECT * FROM [TABLO_1] LEFT OUTER JOIN [TABLO_2] ON [TABLO_1].[İLİSKİLİ_KOLON] =
[TABLO_2].[İLİSKİLİ_KOLON]
SELECT * FROM [TABLO_1] RIGHT OUTER JOIN [TABLO_2] ON [TABLO_1].[İLİSKİLİ_KOLON] =
[TABLO_2].[İLİSKİLİ_KOLON]
SELECT * FROM [TABLO_1] FULL OUTER JOIN [TABLO_2] ON [TABLO_1].[İLİSKİLİ_KOLON] =
[TABLO_2].[İLİSKİLİ_KOLON]
SELECT * FROM [TABLO_1] CROSS JOIN [TABLO_2]
-- Tablo isimlerine alias (takma ad) ataması yapılabilir. Ör: SELECT * FROM
[TABLO_1] t1 INNER...
-- * ifadesi iki tablodaki tüm kolonları çeker. Belirli kolonların gelmesi
isteniyorsa [TABLO_ADI].[KOLON_ADI] yapısı ile belirtilmelidir.
-- OUTER ifadesinin kullanımı opsiyoneldir.
/*
tablo1: ogrenci
|id |ad |soyad |bolum|dogum_yeri |
| 1 |Ali |Yılmaz |1 |Sivas |
| 2 |Ahmet |Mutlu |3 |Bilecik |
| 3 |Asel Sare |Aygün |2 |Bursa |
| 4 |Ayşe |Demir |1 |Bursa |
| 5 |Abdülmelik |Derinkök |1 |Balıkesir |
| 6 |Muhammed Yağız |Mutlu |4 |Eskişehir |
```

7	Elif Eylül	Mutlu	3	Elazığ
8	Orhan	Veli	NULL	Malatya

```

tablo2: program
| id | ad |
| 1 | Bilgisayar |
| 2 | Elektronik |
| 3 | İşletme |
| 4 | Muhasebe |

-- tablo1 ve tablo2 arasında ilişkili kolonlar tablo2.id = tablo1.bolum şeklindedir.

*/

--Örnek - 1 (INNER JOIN)
SELECT o.ad,o.soyad,p.ad FROM ogrenciler o INNER JOIN program p ON o.bolum=p.id
-- id=8 olan kaydın bolum alanı NULL olduğu için bu sorguda listelenmez.
/*Örnek - 1 Çıktı
|ad      |soyad      |ad      |
| Ali     | Yılmaz    | Bilgisayar |
| Ahmet   | Mutlu     | İşletme   |
| Asel Sare | Aygün     | Elektronik |
| Ayşe    | Demir     | Bilgisayar |
| Abdülmelik | Derinkök  | Bilgisayar |
| Muhammed Yağız | Mutlu    | Muhasebe   |
| Elif Eylül | Mutlu     | İşletme    |
*/

--Örnek - 2 (LEFT JOIN)
--1
SELECT o.ad,o.soyad,p.ad FROM ogrenci o LEFT JOIN program p ON o.bolum=p.id
--2
SELECT o.ad,o.soyad,p.ad FROM ogrenci o LEFT OUTER JOIN program p ON o.bolum=p.id
-- iki kodda aynı çıktıyı verir.
/*Örnek - 2 Çıktı
|ad      |soyad      |ad      |
| Ali     | Yılmaz    | Bilgisayar |
| Ahmet   | Mutlu     | İşletme   |
| Asel Sare | Aygün     | Elektronik |
| Ayşe    | Demir     | Bilgisayar |
| Abdülmelik | Derinkök  | Bilgisayar |
| Muhammed Yağız | Mutlu    | Muhasebe   |
| Elif Eylül | Mutlu     | İşletme    |
| Orhan    | Veli      | NULL       |
*/

--Örnek - 3 (RIGHT JOIN)
SELECT o.ad,o.soyad,p.ad FROM ogrenci o RIGHT JOIN program p ON o.bolum=p.id
SELECT o.ad,o.soyad,p.ad FROM ogrenci o RIGHT OUTER JOIN program p ON o.bolum=p.id
/*Örnek - 3 Çıktı
|ad      |soyad      |ad      |
| Ali     | Yılmaz    | Bilgisayar |
| Ayşe    | Demir     | Bilgisayar |
| Abdülmelik | Derinkök  | Bilgisayar |
| Asel Sare | Aygün     | Elektronik |
| Ahmet   | Mutlu     | İşletme   |
| Elif Eylül | Mutlu     | İşletme    |
| Muhammed Yağız | Mutlu    | Muhasebe   |
*/

```

```
--Örnek - 4 (FULL JOIN)
SELECT [o r].ad,[o r].soyad,p.ad FROM ogrenci [o r] FULL JOIN program p ON [o
r].bolum=p.id
SELECT o.ad,o.soyad,p.ad FROM ogrenci o FULL OUTER JOIN program p ON o.bolum=p.id
/*Örnek - 4 Çıktı
```

ad	soyad	ad
Ali	Yılmaz	Bilgisayar
Ahmet	Mutlu	İşletme
Asel Sare	Aygün	Elektronik
Ayşe	Demir	Bilgisayar
Abdumelik	Derinkök	Bilgisayar
Muhammed Yağız	Mutlu	Muhasebe
Elif Eylül	Mutlu	İşletme
Orhan	Veli	NULL

```
*/
```

```
--Örnek - 5 (CROSS JOIN)
SELECT o.ad,o.soyad,p.ad FROM ogrenci o CROSS JOIN program p
```

```
/* Örnek - 5 Çıktı
```

ad	soyad	ad
Ali	Yılmaz	Bilgisayar
Ahmet	Mutlu	Bilgisayar
Asel Sare	Aygün	Bilgisayar
Ayşe	Demir	Bilgisayar
Abdumelik	Derinkök	Bilgisayar
Muhammed Yağız	Mutlu	Bilgisayar
Elif Eylül	Mutlu	Bilgisayar
Orhan	Veli	Bilgisayar
Ali	Yılmaz	Elektronik
Ahmet	Mutlu	Elektronik
Asel Sare	Aygün	Elektronik
Ayşe	Demir	Elektronik
Abdumelik	Derinkök	Elektronik
Muhammed Yağız	Mutlu	Elektronik
Elif Eylül	Mutlu	Elektronik
Orhan	Veli	Elektronik
Ali	Yılmaz	İşletme
Ahmet	Mutlu	İşletme
Asel Sare	Aygün	İşletme
Ayşe	Demir	İşletme
Abdumelik	Derinkök	İşletme
Muhammed Yağız	Mutlu	İşletme
Elif Eylül	Mutlu	İşletme
Orhan	Veli	İşletme
Ali	Yılmaz	Muhasebe
Ahmet	Mutlu	Muhasebe
Asel Sare	Aygün	Muhasebe
Ayşe	Demir	Muhasebe
Abdumelik	Derinkök	Muhasebe
Muhammed Yağız	Mutlu	Muhasebe
Elif Eylül	Mutlu	Muhasebe
Orhan	Veli	Muhasebe

```
*/
```


5.2. DDL (Data Defination) – Veri Tanımlama

5.2.1. Create

“CREATE” komutu veritabanı oluşturmak ya da veritabanında, tabloda nesne (tablo, kolon vs.) oluşturmak için kullanılır.

```
--syntax

CREATE [TUR] [NESNE_ADI] [PARAMETRELER]

--VERİTABANI Oluşturma - 1

CREATE DATABASE Yeni

--VERİTABANI Oluşturma - 1 - Çıktı
/*
  Commands completed successfully.

Veritabanı mevcut değilse yukarıdaki gibi çıktı verir ve veritabanını varsayılan yapılandırma ile oluşturur. Ancak veritabanı daha önceden eklenmişse aşağıdaki gibi çıktı verecektir.

  Database 'Yeni' already exists. Choose a different database name.
*/

--VERİTABANI Oluşturma - 2

CREATE DATABASE Yeni2
ON
(
  Name= 'dbAdi',
  Filename= 'D:\dbb.mdf',
  Size=9,
  Filegrowth=3
)

--VERİTABANI Oluşturma - 2 - Çıktı
/*
  Commands completed successfully.

Veritabanı mevcut değilse yukarıdaki gibi çıktı verir ve veritabanını varsayılan yapılandırma ile oluşturur. Ancak veritabanı daha önceden eklenmişse aşağıdaki gibi çıktı verecektir.

  Database 'Yeni2' already exists. Choose a different database name.

NAME: SQL Server veritabanımıza başvurduğunda bu ismi kullanır. İsim
unique(benzersiz) olmak zorundadır.
FILENAME: Veritabanımızın kaydedileceği yolu ifade eder.
SIZE: Veritabanınızın ilk boyutu
MAXSIZE: Veritabanınızın en fazla ulaşabileceği boyut, MB cinsinden değeri ifade eder.
FILEGROWTH: Yeterli boşluk kalmadığında veritabanınızın genişleme miktarı, MB cinsinden değeri ifade eder.
*/
```

```
--TABLO oluşturma - 1
CREATE TABLE [Tablo Adı]
(
  Id INT PRIMARY KEY Identity(2,4),
  metin VARCHAR(50)
)

-- Id kolonunun veri türü (INT) belirtildikten sonra Constraint (Kısıtlayıcı)
ifadeler kullanılmıştır. Constraint kullanımı kolon değerinin kısıtlanması için
kullanılır.
```

CONSTRAINTS (KISITLAYICILAR)

NOT NULL - Bir sütunun değere sahip olmamasını sağlar.

UNIQUE - bir sütundaki tüm değerlerin farklı olmasını sağlar.

PRIMARY KEY -bir tablodaki her satırı benzersiz şekilde tanımlar.

FOREIGN KEY -Başka bir tablodaki bir satırı / kaydı benzersiz şekilde tanımlar.

```
CREATE TABLE [Tablo Adı]
(
  Id INT PRIMARY KEY Identity(2,4),
  metin VARCHAR(50)
  x_id INT FOREIGN KEY REFERENCES [TABLO_2]([KOLON])
    ON DELETE [ISLEM]
    ON UPDATE [ISLEM]
)
```

NOT: [ISLEM] şeklinde belirtilen parametrenin alabileceği değerler ile ilgili:

CASCADE: REFERENCES ile belirtilen sütunda bir eylem (UPDATE, DELETE) olduğunda Foreign key ile belirtilen ilişkili sütunda benzer eylemi yapar.

NO ACTION: Foreign key sütunu ve REFERENCES ile belirtilen sütunda bir ilişki varsa, REFERENCES ile belirtilen sütunda bir eyleme (UPDATE, DELETE) izin vermez.

SET NULL: REFERENCES ile belirtilen sütunda bir eylem (UPDATE, DELETE) olduğunda Foreign key ile belirtilen ilişkili sütunu NULL yapar (!!! İlgili sütunda NOT NULL kısıtlaması varsa hata verir.).

SET DEFAULT: REFERENCES ile belirtilen sütunda bir eylem (UPDATE, DELETE) olduğunda Foreign key ile belirtilen ilişkili sütuna DEFAULT değerini verir(!!! İlgili sütunda DEFAULT kısıtlaması yoksa hata verir.).

RESTRICT: MySQL VTYS içerisinde bulunan bu eylem NO ACTION ile aynı işleve sahiptir.

CHECK - Bir sütundaki tüm verilerin belirli bir koşulu karşılamasını sağlar.

DEFAULT -Değer belirtilmediğinde bir sütun için varsayılan değer atar.

INDEX -veritabanından çok hızlı bir şekilde veri almak için kullanılır.

```
--TABLO oluşturma - 1 - Çıktı
/*
  Commands completed successfully.
```

Tablo ilgili veritabanında mevcut değilse yukarıdaki gibi çıktı verir ve tabloyu oluşturur. Ancak tablo daha önceden eklenmişse aşağıdaki gibi çıktı verecektir.

```

There is already an object named 'Tablo Adı' in the database.

*/

--TABLO oluşturma - 2

CREATE TABLE personel(

    id    INT    IDENTITY(1,1) NOT NULL,
    ad    NVARCHAR (20) NOT NULL,
    soyad NVARCHAR (20),
    unvan  NVARCHAR (20) ,
    PRIMARY KEY (id)
)
--TABLO oluşturma - 2 - Çıktı
/*
    Commands completed successfully.

Tablo ilgili veritabanında mevcut değilse yukarıdaki gibi çıktı verir ve tabloyu
oluşturur. Ancak tablo daha önceden eklenmişse aşağıdaki gibi çıktı verecektir.

There is already an object named 'personel' in the database.

- IDENTITY(1,1) yazımında ilk sayı başlangıç değerini, ikinci sayı ise artış
miktarını ifade etmektedir.
- NOT NULL ifadesi kolon değerinin boş bırakılamayacağı anlamını taşımaktadır. Eğer
NOT NULL ifadesi kullanılmazsa kolon NULL değer alabilir yani ilgili alan boş
bırakılabilir.
*/

```

5.2.2. Drop

Veritabanını kaldırmak için veya veritabanında bulunan nesneleri kaldırmak için kullanılır.

```

--syntax

DROP [TUR] [NESNE_ADI]

-- DROP TABLE - 1
DROP TABLE personel

/* DROP TABLE - 1 - ÇIKTI
    Commands completed successfully.

personel isimli tabloyu siler.
*/

-- DROP DATABASE - 1

DROP DATABASE Yeni

/* DROP DATABASE - 1 - ÇIKTI
    Commands completed successfully.

Yeni isimli veritabanını siler.
*/

```

5.2.3. Alter

Var olan bir veritabanı üzerinde düzenleme ve değişiklik yapmak için kullanılır.

```
--syntax

ALTER [TUR] [NESNE_ADI] [PARAMETRELER]

--ALTER DATABASE - 1

ALTER DATABASE Yeni2
MODIFY FILE
(
Name = 'dbAdi',
Size = 12
)

/* --ALTER DATABASE - 1 - ÇIKTI
Commands completed successfully.

- Mevcut veritabanı yapılandırmasında değişiklik yapılacağı zaman Name
parametresinin belirtilmesi gerekmektedir. Örnekte CREATE bölümünde oluşturulan
veritabanının boyutu değiştirilmiştir.

*/

-- ALTER TABLE - 1

ALTER TABLE personel
ADD
memleket VARCHAR(50),
yas INT

/* --ALTER DATABASE - 1 - ÇIKTI
Commands completed successfully.

- CREATE bölümünde oluşturulan personel tablosuna memleket ve yas kolonları
oluşturulmuştur.

*/

-- ALTER TABLE - 2
ALTER TABLE personel
ALTER COLUMN
memleket NVARCHAR(50)

/* --ALTER DATABASE - 2 - ÇIKTI
Commands completed successfully.

- CREATE bölümünde oluşturulan personel tablosunda bulunan memleket kolonunun veri
türü nvarchar olarak değiştirilmiştir.

*/

-- ALTER TABLE - 3
ALTER TABLE personel
DROP COLUMN memleket,yas
```

```

/* --ALTER DATABASE - 3 - ÇIKTI
Commands completed successfully.

- personel tablosunda bulunan memleket ve yas kolonları silinmiştir.

*/

```

5.2.4. Truncate

Tabloda bulunan tüm satırları/kayıtları "NULL" ifadeler dahil olmak üzere kaldırır.

```

--syntax

TRUNCATE TABLE [TABLO_ADI]

-- TRUNCATE TABLE - 1

TRUNCATE TABLE ogrenci

/* - TRUNCATE TABLE - 1
Commands completed successfully.

Tablodaki tüm verileri siler ve tablolunun yapılandırmasını yeniler. Örneğin,
identity alanını tablonun ilk yapılandırma değerinden başlatır.

*/

```

5.2.5. Rename

Nesneleri yeniden adlandırmak için kullanılır.

```

-- syntax

-- Veritabanı adı değiştirme
SP_RENAME '[MEVCUT_AD]' , '[YENİ_AD]' , 'database'

-- Tablo adı değiştirme
SP_RENAME '[TABLO_ADI]' , '[YENİ_TABLO_ADI]'
-- Tablo adı değişikliğinde tablo adının tamamı dbo. ifadesi de [TABLO_ADI] alanında
bulunmalıdır.

--Kolon adı değiştirme

SP_RENAME '[TABLO_ADI].[MEVCUT_KOLON_ADI]' , '[YENİ_KOLON_ADI]' , 'column'

```

5.2.6. Comment

Veri sözlüğüne yorum eklemek için kullanılır.

```

--Syntax

-- ile tek satırlık yorum ekleme işlemi yapılır.

```

```
-- /* */ ifadeleri ile birden fazla satıra yorum ekleme işlemi yapılır.

-- Tek satırlık yorum ifadesi.

/*
Birden fazla satıra sahip,

Yorum ifadesi.
*/
```

5.3. DML (Data Manipulation) – Veri İşleme

5.3.1 Insert

Veritabanında bulunan ilgili tabloya veri (satır) ekler.

```
--Syntax

--1. kullanım
INSERT [TABLO_ADI] VALUES ([Veri1],[Veri2],[Veri3],...,[VeriN])
--2. kullanım
INSERT INTO [TABLO_ADI] VALUES ([Veri1],[Veri2],[Veri3],...,[VeriN])
--3. kullanım
INSERT [TABLO_ADI] ([Kolon1],[Kolon2],[Kolon3],...,[KolonN]) VALUES
([Veri1],[Veri2],[Veri3],...,[VeriN])
--4. kullanım
INSERT [TABLO_ADI] ([Kolon1],[Kolon2]) VALUES ([Veri1],[Veri2])
--5. kullanım
INSERT [TABLO_ADI] VALUES
([Veri1],[Veri2],[Veri3],...,[VeriN]),
([2Veri1],[2Veri2],[2Veri3],...,[2VeriN]),
...,
([nVeri1],[nVeri2],[nVeri3],...,[nVeriN])

/*
Birinci kullanımda VALUES alanından sonra parantez içinde tablodaki kolon sıralamasına göre eklenmesi istenen veriler virgül ile ayrılarak belirtilir. Birinci ve beşinci kullanımlarda olduğu gibi "VALUES" ifadesinden önce kolon ad veya adları belirtilmemişse tüm kolonlar için değer belirtilmelidir. İkinci kullanımda birinciden farklı olarak "INTO" ifadesi bulunmaktadır. Ancak T-SQL syntax'ı için opsiyonel bir ifadedir yani kullanılması veya kullanılmaması durumu işlemi etkilememektedir. Üçüncü kullanımda [TABLO_ADI] ifadesinden sonra parantez içinde kolon adı veya adları belirtilerek yalnızca belirtilen kolonlara veri girişi yapılabilir. Dördüncü kullanımda üçüncü ile aynı ifadeyi temsil etmektedir yalnızca veri girişi yapılacak kolon sayısı farklıdır. Beşinci kullanım şeklinde birden fazla satır ekleme işlemi yapılmaktadır. "VALUES" ifadesinden sonra parantez içinde veriler belirtilip parantez kapatıldıktan sonra virgül ile 2, 3, 4 ... n adet satır aynı kod ile eklenebilmektedir.

NOT: Eğer kolon değeri NOT NULL şeklinde ayarlanmış ise mutlaka değer girilmelidir.
NOT-2: Otomatik artan Identity/Primary Key olarak belirlenmiş kolonlara değer girişi yapılmaz.
NOT-3: T-SQL'de tek INSERT ifadesi ile en fazla 1000 satır veri eklenebilmektedir.

- ogrenciler tablosu
```

id	ad	soyad	sinif	vize	final
1	Ali	Yılmaz	1	65	82
2	Ahmet	Mutlu	1	30	45
3	Asel Sare	Aygün	2	85	79

4	Ayşe	Demir	2	55	63
5	Abdülmelik	Derinkök	2	100	100
6	Muhammed Yağız	Mutlu	2	95	68
7	Elif Eylül	Mutlu	2	90	90

*/

--Örnek - 1

INSERT ogrenciler VALUES('Uğur','Erdem',1,68,92)

-- Örnek - 1 Çıktı

/*

(1 row affected)

NOT: Tablo olarak çıktı vermez ancak bilgi mesajı olarak etkilenen satır sayısını yazdırır.

İşlemden sonra ogrenciler tablosu:

id	ad	soyad	sinif	vize	final
1	Ali	Yılmaz	1	65	82
2	Ahmet	Mutlu	1	30	45
3	Asel Sare	Aygün	2	85	79
4	Ayşe	Demir	2	55	63
5	Abdülmelik	Derinkök	2	100	100
6	Muhammed Yağız	Mutlu	2	95	68
7	Elif Eylül	Mutlu	2	68	92
8	Uğur	Erdem	1	90	90

*/

5.3.2 Update

Tablodaki verileri istenen şarta göre günceller. Eğer şart ifadesi (WHERE) belirtilmemişse, ilgili sütun/lardaki tüm kayıtları değiştirir.

--syntax

UPDATE [TABLO_ADI] SET [KOLON1] = [DEĞER1], [KOLON2] = [DEĞER2],..., [KOLONn] = [DEĞERn] WHERE [ŞART]

/*

- DİKKAT: UPDATE komutu yazıldıktan sonra WHERE ifadesi ile şart belirtilmezse veri güncellemesi yapılacak kolonlardaki tüm veriler değişir.

- ogrenciler tablosu

id	ad	soyad	sinif	vize	final
1	Ali	Yılmaz	1	65	82
2	Ahmet	Mutlu	1	30	45
3	Asel Sare	Aygün	2	85	79
4	Ayşe	Demir	2	55	63
5	Abdülmelik	Derinkök	2	100	100
6	Muhammed Yağız	Mutlu	2	95	68
7	Elif Eylül	Mutlu	2	90	90

*/

```
--Örnek - 1
UPDATE ogrenciler SET ad='Mehmet' WHERE id=2

/* Örnek - 1 Çıktı

(1 row affected)
-- Tablo olarak çıktı vermez ancak bilgi mesajı olarak etkilenen satır sayısını yazdırır.

İşlemden sonra ogrenciler tablosu:
| id | ad       | soyad   | sinif | vize | final |
| 1  | Ali      | Yılmaz  | 1     | 65  | 82     |
| 2  | Mehmet   | Mutlu   | 1     | 30  | 45     |
| 3  | Asel Sare | Aygün   | 2     | 85  | 79     |
| 4  | Ayşe     | Demir   | 2     | 55  | 63     |
| 5  | Abdülmelik | Derinkök | 2     | 100 | 100    |
| 6  | Muhammed Yağız | Mutlu   | 2     | 95  | 68     |
| 7  | Elif Eylül | Mutlu   | 2     | 90  | 90     |

*/
--Örnek - 2
UPDATE ogrenciler SET ad='Veli'

/* Örnek - 2 Çıktı

(7 rows affected)
-- Tablo olarak çıktı vermez ancak bilgi mesajı olarak etkilenen satır sayısını yazdırır.

İşlemden sonra ogrenciler tablosu:
| id | ad  | soyad   | sinif | vize | final |
| 1  | Veli | Yılmaz  | 1     | 65  | 82     |
| 2  | Veli | Mutlu   | 1     | 30  | 45     |
| 3  | Veli | Aygün   | 2     | 85  | 79     |
| 4  | Veli | Demir   | 2     | 55  | 63     |
| 5  | Veli | Derinkök | 2     | 100 | 100    |
| 6  | Veli | Mutlu   | 2     | 95  | 68     |
| 7  | Veli | Mutlu   | 2     | 90  | 90     |

*/
```

5.3.3 Delete

Tablodaki kayıtların (satır) koşula uygun olanlarını siler. Eğer koşul ifadesi (WHERE) belirtilmemişse, ilgili sütun/lardaki tüm kayıtları siler.

```
--syntax

DELETE FROM [TABLO_ADI] WHERE [ŞART]

--DİKKAT: DELETE komutu kullanılırken WHERE ifadesi kullanılmazsa yani şart belirtilmezse tablodaki tüm veriler silinir.

/*

- ogrenciler tablosu

| id | ad       | soyad   | sinif | vize | final |
| 1  | Ali      | Yılmaz  | 1     | 65  | 82     |
| 2  | Ahmet    | Mutlu   | 1     | 30  | 45     |
```



```

| 3 | Asel Sare | Aygün | 2 | 85 | 79 |
| 4 | Ayşe | Demir | 2 | 55 | 63 |
| 5 | Abdülmelik | Derinkök | 2 | 100 | 100 |
| 6 | Muhammed Yağız | Mutlu | 2 | 95 | 68 |
| 7 | Elif Eylül | Mutlu | 2 | 90 | 90 |
*/

--Örnek - 1
DELETE FROM ogrenciler WHERE id=2

/* Örnek - 1 Çıktı
(1 row affected)
-- Tablo olarak çıktı vermez ancak bilgi mesajı olarak etkilenen satır sayısını
yazdırır.

İşlemden sonra ogrenciler tablosu:
| id | ad | soyad | sinif | vize | final |
| 1 | Ali | Yılmaz | 1 | 65 | 82 |
| 3 | Asel Sare | Aygün | 2 | 85 | 79 |
| 4 | Ayşe | Demir | 2 | 55 | 63 |
| 5 | Abdülmelik | Derinkök | 2 | 100 | 100 |
| 6 | Muhammed Yağız | Mutlu | 2 | 95 | 68 |
| 7 | Elif Eylül | Mutlu | 2 | 90 | 90 |
*/

--Örnek - 2
DELETE FROM ogrenciler

/* Örnek - 2 Çıktı
(6 rows affected)
-- Tablo olarak çıktı vermez ancak bilgi mesajı olarak etkilenen satır sayısını
yazdırır.

İşlemden sonra ogrenciler tablosu:
| id | ad | soyad | sinif | vize | final |

[TABLODA KAYIT BULUNMAZ]

*/

```

5.3.4 Merge

İlgili satırın şart koşulan duruma uygunluğuna göre "Insert" veya "Update" komutlarının işlevlerini gerçekleştirir. "UPSERT" anahtar kelimesi ile de kaynaklarda yer almaktadır.

5.3.5 Call

Oracle veritabanı sistemlerinde kullanılan PL/SQL ya da Java alt programlarını çalıştırmak için kullanılır (T-SQL'de karşılığı EXEC komutudur).

5.3.6 Explain Plan

SQL sorgularının çalıştırılması sonucunda gerçekleşecek veri hareketlerini ön izlemek ve sorguların kaynak maliyetini görüntülemek için kullanılır.

5.3.7 Lock Table

Tablonun farklı oturumlardan gelen istekleri kuyruğa alarak sunucu yükünü azaltma amacıyla kullanılır.

5.4. DCL (Data Control) – Veri Kontrol

Kullanıcı sayısı birden fazla olan veritabanlarında yetkilendirme işlemleri için "DCL" komutları kullanılmaktadır.

5.4.1. Grant

Kullanıcıların veritabanı veya tablolara dair yetki verilmesi amacıyla kullanılır.

5.4.2. Revoke

Kullanıcılara "GRANT" komutu ile verilen yetkilerin kaldırılması amacıyla kullanılır.

5.4.3. Deny

Kullanıcıların veritabanı veya tablolara dair yetki sınırlaması amacıyla kullanılır.

5.5. TCL (Transaction Control)

"DML" işlemlerini yönetmek için kullanılan komutlardır.

5.5.1. Commit

Gerçekleştirilen Transaction işleminin başarılı olması durumunda yapılan değişikliklerin veritabanına kalıcı olarak aktarılması için kullanılır.

5.5.2 Rollback

Gerçekleştirilen Transaction işleminin başarısız olması durumunda o ana kadar yapılan değişikliklerin geri alınması ve verilerin Transaction işleminden önceki haline getirilmesi için kullanılır.

5.5.3. Savepoint

Transaction işleminin başarısız olması durumunda gerçekleşecek "rollback" işleminin, "transaction" eyleminin en başa döndürülmesinin istenmediği durumlarda geri dönüş noktası belirlemek için kullanılır.

5.6. Fonksiyonlar

5.6.1 String (Metinsel) Fonksiyonlar

T-SQL'de metinsel ifadeler için kullanılan bazı fonksiyonlar ve bunların açıklaması aşağıda yer almaktadır:

LOWER: Metni küçük harf düzenine dönüştürür.

UPPER: Metni büyük harf düzenine dönüştürür.

LEFT : Metnin solundan belirtilen karakter kadar olan ifadeyi alır.

RIGHT: Metnin sağından belirtilen karakter kadar olan ifadeyi alır.

LTRIM: Metnin solunda bulunan boşlukları temizler.

RTRIM: Metnin sağında bulunan boşlukları temizler.

SUBSTRING: Belirtilen index'ten itibaren belirtilen karakter sayısı kadar olan ifadeyi alır.

CHARINDEX: Karakterin metin içerisindeki index'ini döndürür.

REVERSE: Metni tersine çevirerek döndürür.

REPLACE: Metnin içerisinde yer alan ifadeyi istenen ifade ile değiştirmek için kullanılır.

```
-- String Fonksiyonlar

--syntax

LOWER('[METİN]')
UPPER('[METİN]')
LEFT('[METİN]',[SAYI])
RIGHT('[METİN]',[SAYI])
LTRIM('[METİN]')
RTRIM('[METİN]')
SUBSTRING('[METİN]',[INDEX],[UZUNLUK])
CHARINDEX('[KARAKTER]','[METİN]')
REVERSE('[METİN]')
REPLACE('[METİN]','[ARANAN_DEĞER]','[YENİ_DEĞER]')
```

--Örnek - LOWER

```
SELECT LOWER('AbdulMelik')
```

-- Örnek - LOWER Çıktı

```
/*
|(No column name)|
| abdulmelik      |
*/
```

--Örnek - UPPER

```
SELECT UPPER('AbdulMelik')
```

-- Örnek - UPPER Çıktı

```
/*
|(No column name)|
| ABDULMELİK      |
*/
```

--Örnek - LEFT

```
SELECT LEFT('Abdulmelik',5)
```

-- Örnek - LEFT Çıktı

```
/*
|(No column name)|
| Abdul           |
*/
```

--Örnek - RIGHT

```
SELECT RIGHT('Abdulmelik',5)
```

-- Örnek - RIGHT Çıktı

```
/*
|(No column name)|
```

```

| melik          |
*/
--Örnek - LTRIM

SELECT LTRIM(' Abdulmelik ')

-- Örnek - LTRIM Çıktı
/*
|(No column name)|
| Abdulmelik      |
*/

--Örnek - RTRIM

SELECT RTRIM(' Abdulmelik ')

-- Örnek - RTRIM Çıktı
/*
|(No column name)|
| Abdulmelik      |
*/

--Örnek - SUBSTRING

SELECT SUBSTRING('Abdulmelik',2,4)

-- Örnek - SUBSTRING Çıktı
/*
|(No column name)|
| bdul            |
*/

--Örnek - CHARINDEX

SELECT CHARINDEX('u','Abdulmelik')

-- Örnek - CHARINDEX Çıktı
/*
|(No column name)|
| 4               |
*/

--Örnek - REVERSE

SELECT REVERSE('Abdulmelik')

-- Örnek - REVERSE Çıktı
/*
|(No column name)|
| kilemludbA     |
*/

--Örnek - REPLACE

SELECT REPLACE('Fatih Sultan Yavuz','Yavuz','Mehmet')

-- Örnek - REPLACE Çıktı
/*
|(No column name)|
| Fatih Sultan Mehmet |

```

```
*/
```

5.6.2 Sayısal Fonksiyonlar

T-SQL'de sayısal ifadeler için kullanılan bazı fonksiyonlar ve bunların açıklaması aşağıda yer almaktadır:

POWER: Üs alma işlemi yapar.

PI: Pi sayısını verir.

SIN: Girilen sayısının sinüs değerini döndürür.

COS: Girilen sayısının kosinüs değerini döndürür.

COT: Girilen sayısının kotanjant değerini döndürür.

TAN: Girilen sayısının tanjant değerini döndürür.

ABS: Girilen sayının mutlak değer ifadesini verir.

RAND: 0-1 arası rastgele bir değer üretir.

FLOOR: Girilen sayıyı kendisine en yakın olan ve kendisinden küçük olan tam sayıya yuvarlar.

CEILING: Girilen sayıyı kendisine en yakın olan ve kendisinden büyük olan tam sayıya yuvarlar.

ROUND: Girilen sayıyı büyük veya küçük sayıya yuvarlar. Yuvarlama işleminde ilgili basamak değeri 5'ten küçükse, küçük olan sayıya 5 veya 5'ten büyükse, büyük olan sayıya yuvarlar.

```
-- Sayısal Fonksiyonlar
--syntax

POWER([SAYI],[US])
PI()
SIN([SAYI])
COS([SAYI])
COT([SAYI])
TAN([SAYI])
ABS([SAYI])
RAND()
FLOOR([SAYI])
CEILING([SAYI])
ROUND([SAYI],[BASAMAK])

--Örnek - POWER

SELECT POWER(5,2)

-- Örnek - POWER Çıktı
/*
|(No column name)|
```

```

| 25 |
*/

--Örnek - PI

SELECT PI()

-- Örnek - PI Çıktı
/*
|(No column name)|
|3,14159265358979|
*/
--Örnek - SIN

SELECT SIN(60)

-- Örnek - SIN Çıktı
/*
|(No column name) |
|-0,304810621102217|
*/
--Örnek - COS

SELECT COS(60)

-- Örnek - COS Çıktı
/*
|(No column name) |
|-0,952412980415156|
*/
--Örnek - COT

SELECT COT(60)

-- Örnek - COT Çıktı
/*
|(No column name) |
|3,12460562224231 |
*/
--Örnek - TAN

SELECT TAN(60)

-- Örnek - TAN Çıktı
/*
|(No column name) |
|0,320040389379563 |
*/
--Örnek - ABS

SELECT ABS(-20)

-- Örnek - ABS Çıktı
/*
|(No column name) |
| 20 |
*/
--Örnek - RAND

SELECT RAND()

-- Örnek - RAND Çıktı

```

```

/*
|(No column name) |
|0,xxxxxxxxxxxxxxxx |

!RASTGELE BİR DEĞER ÜRETİR.
*/
--Örnek - FLOOR

SELECT FLOOR(24.99999)

-- Örnek - FLOOR Çıktı
/*
|(No column name) |
| 24 |
*/
--Örnek - CEILING

SELECT CEILING(24.0001)

-- Örnek - CEILING Çıktı
/*
|(No column name) |
| 25 |
*/
--Örnek - ROUND

SELECT ROUND(24.992,2) --1
SELECT ROUND(24.999,0) --2

-- Örnek - ROUND Çıktı
/*
|(No column name) | [1]
| 24.990 |

|(No column name) | [2]
| 25.000 |
*/

```

5.6.3. Tarih Fonksiyonları

T-SQL'de tarih verileri için kullanılan bazı fonksiyonlar ve bunların açıklaması aşağıda yer almaktadır:

GETDATE: Komutun çalıştırıldığı andaki tarih bilgisini YYYY-AA-GG "SS:DD:ss.xxx" formatında yazdırır.

CURRENT_TIMESTAMP: GETDATE fonksiyonu ile aynı işi yapmaktadır. Komutun çalıştırıldığı andaki tarih bilgisini YYYY-AA-GG "SS:DD:ss.xxx" formatında yazdırır. GETDATE fonksiyonuna göre kesirli saniye değeri (milisaniye) daha kesin olduğu Microsoft tarafından belirtilmiştir.

SYSDATETIME: GETDATE fonksiyonu ile aynı işi yapmaktadır. Komutun çalıştırıldığı andaki tarih bilgisini YYYY-AA-GG "SS:DD:ss.xxxxxxx" formatında

yazdırır. GETDATE fonksiyonuna göre kesirli saniye değeri (milisaniye) daha kesin olduğu Microsoft tarafından belirtilmiştir.

DATEADD: Bir tarih ifadesine belirtilen türde zaman ekleme veya çıkarma işlemi yapar. Örneğin, 01.01.2022 tarihine ay türünden 5 ekleme yapmak (01.06.2022 tarihi elde edilir). Tür parametresi milisaniye, saniye, dakika, saat, gün, hafta, ay veya yıl cinsinden değer olabilir (MILLISECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH ya da YEAR).

DATEDIFF: İki tarih arasındaki farkın istenen türde görüntülenmesini sağlar. Tür parametresi milisaniye, saniye, dakika, saat, gün, hafta, ay veya yıl cinsinden değer olabilir (MILLISECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH ya da YEAR).

DATEPART: Belirtilen tarih ifadesindeki istenen türdeki değeri almak için kullanılır. Örneğin, "01.05.2022 04:27:15.123" ifadesinden yıl türünden değer alınırsa "2022", ay türünden değer alınırsa "5", saat türünden değer alınırsa "4" sonucunu verir. Tür parametresi milisaniye, saniye, dakika, saat, gün, hafta, ay veya yıl cinsinden değer olabilir (MILLISECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH ya da YEAR).

Hatırlatma: SQL'de tarih ifadesinin yazım düzeni "YYYY-AA-GG SS:DD:ss.xxx" şeklindedir (YYYY: Yıl, AA: Ay (01-12), GG: Gün (01-31), SS: Saat (00-23), DD: Dakika (00-59), ss: Saniye (00-59), xxx: Milisaniye (0000000-9999999)).

```
-- Tarih Fonksiyonları
--syntax

GETDATE()
CURRENT_TIMESTAMP
SYSDATETIME()
DATEADD([TÜR],[SAYI],[TARİH])
-- [TÜR] parametresi MILLISECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH ya da YEAR olabilir.
DATEDIFF([TÜR],[TARİH1],[TARİH2])
-- [TÜR] parametresi MILLISECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH ya da YEAR olabilir.
DATEPART([TÜR],[TARİH])
-- [TÜR] parametresi MILLISECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH ya da YEAR olabilir.

--Örnek - GETDATE

SELECT GETDATE()

-- Örnek - GETDATE Çıktı
/*
| (No column name)          |
| YYYY-AA-GG SS:DD:ss.xxx |
*/

Komutun çalıştırıldığı andaki tarih bilgisini YYYY-AA-GG SS:DD:ss.xxx formatında yazdırır.
*/

--Örnek - CURRENT_TIMESTAMP
```



```
SELECT CURRENT_TIMESTAMP
```

```
-- Örnek - CURRENT_TIMESTAMP Çıktı
```

```
/*
```

```
|(No column name)      |  
| YYYY-AA-GG SS:DD:ss.xxx |
```

Komutun çalıştırıldığı andaki tarih bilgisini YYYY-AA-GG SS:DD:ss.xxx formatında yazdırır.

```
*/
```

```
--Örnek - SYSDATETIME
```

```
SELECT SYSDATETIME()
```

```
-- Örnek - SYSDATETIME Çıktı
```

```
/*
```

```
|(No column name)      |  
| YYYY-AA-GG SS:DD:ss.xxxxxxx |
```

Komutun çalıştırıldığı andaki tarih bilgisini YYYY-AA-GG SS:DD:ss.xxx formatında yazdırır.

```
*/
```

```
--Örnek - DATEADD
```

```
SELECT DATEADD(YEAR,5,'2022-12-03') --1
```

```
SELECT DATEADD(MONTH,5,'2022-12-03') --2
```

```
SELECT DATEADD(DAY,5,'2022-12-03') --3
```

```
SELECT DATEADD(MONTH,-5,'2022-01-01') --4
```

```
-- Örnek - DATEADD Çıktı
```

```
/*
```

```
|(No column name)      | [1]  
| 2027-12-03 00:00:00.000 |
```

```
|(No column name)      | [2]  
| 2023-05-03 00:00:00.000 |
```

```
|(No column name)      | [3]  
| 2022-12-08 00:00:00.000 |
```

```
|(No column name)      | [4]  
| 2021-08-01 00:00:00.000 |
```

```
*/
```

```
--Örnek - DATEDIFF
```

```
SELECT DATEDIFF(YEAR,'2019-05-08','2022-12-03') --1
```

```
SELECT DATEDIFF(MONTH,'2019-05-08','2022-12-03') --2
```

```
SELECT DATEDIFF(DAY,'2019-05-08','2022-12-03') --3
```

```
-- Örnek - DATEDIFF Çıktı
```

```
/*
```

```
|(No column name) | [1]  
| 3 |
```

```
|(No column name) | [2]  
| 43 |
```

```

| (No column name) | [3]
| 1305              |

*/
--Örnek - DATEPART
SELECT DATEPART(YEAR, '2022-12-03') --1
SELECT DATEPART(MONTH, '2022-12-03') --2
SELECT DATEPART(DAY, '2022-12-03') --3

-- Örnek - DATEPART Çıktı
/*
| (No column name) | [1]
| 2022              |

| (No column name) | [2]
| 12                |

| (No column name) | [3]
| 3                 |

*/

```

KAYNAKÇA

<https://www.datascienceearth.com/sql-nedir-dml-ddl-dcl-dql-komutlari/>

<https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/>

https://en.wikipedia.org/wiki/Data_definition_language

[https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/07/veri-tan%C4%B1mlama-dili-\(ddl-data-definition-language\)-i-fadeleri](https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/07/veri-tan%C4%B1mlama-dili-(ddl-data-definition-language)-i-fadeleri)

<https://www.burakavci.com.tr/2017/03/plsql-ddl-dml-dcl-tcl.html>

<https://tr.wikipedia.org/wiki/CRUD>

https://en.wikipedia.org/wiki/Data_query_language

<https://learn.microsoft.com/en-us/sql/t-sql/queries/explain-transact-sql?view=azure-sqldw-latest>

<https://www.mshowto.org/sql-server-sunucu-ve-veritabani-duzeyinde-roller.html>

<https://medium.com/deeplab-tech/sql-serverda-transaction-kullan%C4%B1m%C4%B1-ve-y%C3%B6netimi-be7525ebb8d2>

https://www.bilgigunlugum.net/dbase/sql/sql_fonksiyon

<https://freenetworkk.blogspot.com/2015/01/t-sql-nedir-transact-sql-cesitleri.html>

<https://www.yusufsezer.com.tr/mysql-tablo-kilitleme/>

<https://www.atakdomain.com/blog/sql-wildcards-nedir-kullanimi-nasildir-orneklerle-anlatim>

<https://medium.com/machine-learning-t%C3%BCrkiye/sql-server-tarih-fonksiyonlar%C4%B1-d4cce45186b7>

<https://learn.microsoft.com/en-us/sql/t-sql/functions/current-timestamp-transact-sql?view=sql-server-ver16>

<https://learn.microsoft.com/en-us/sql/t-sql/functions/sysdatetime-transact-sql?view=sql-server-ver16>

<https://learn.microsoft.com/en-us/sql/t-sql/functions/datepart-transact-sql?view=sql-server-ver16>

<https://learn.microsoft.com/en-us/sql/t-sql/statements/create-database-transact-sql?view=sql-server-ver16&tabs=sqlpool>

<https://kod5.org/sql-veritabani-olusturma-kodlu-kodsuz/>

<https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?redirectedfrom=MSDN&view=sql-server-ver16>

