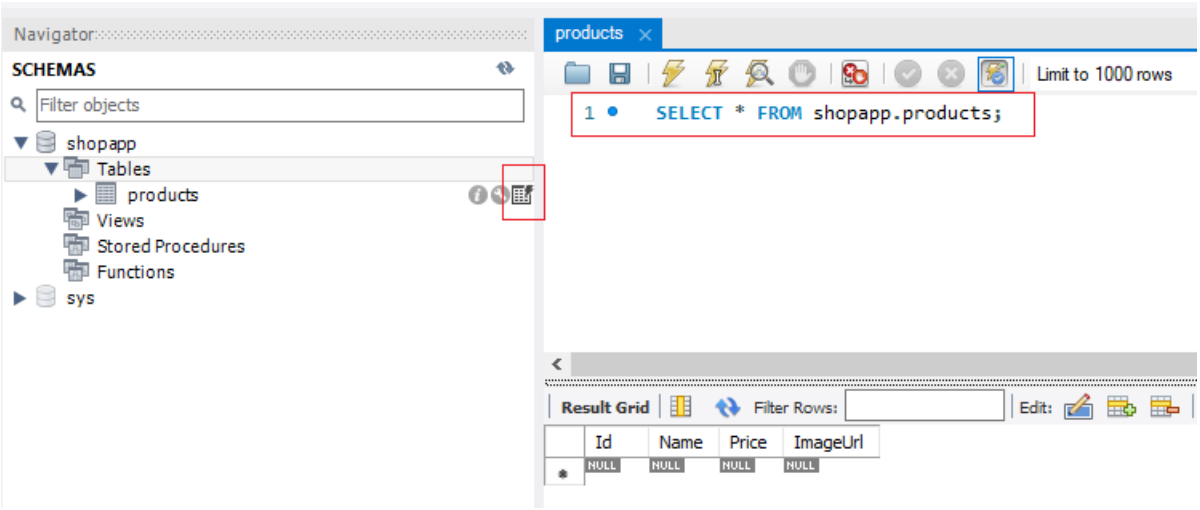
1. SELECT

SELECT Query From IDE

Önceki word dosyasının sonunda shopapp isimli database'imize boş bir tablo eklemiştik.

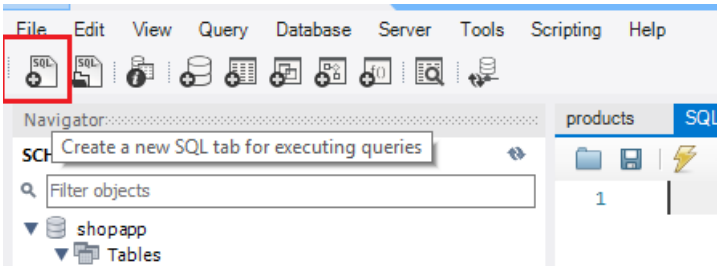
Şimdi bu tablo için bir select sorgusu nasıl yapılabilir bakalım:



- Yukarıdaki gibi products tablosunun yanındaki tablo ikonuna tıklarsak veya tabloya sağ tıklayıp select rows dersek yukarıdaki gibi bir select sorgusu bizim için otomatik olarak yazılacaktır.
- Burada söylenen shopapp database'inin products tablosundan tüm satırları getir.
- Getirilen satırlar aşağıda gösterilir.

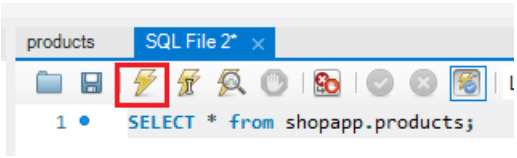
SELECT Query By Hand

Bu yöntemde yine IDE bize yardımcı oldu biz query'i elle yazmadık. Elle yazmak istersek ne yapabileceğimize bakalım:

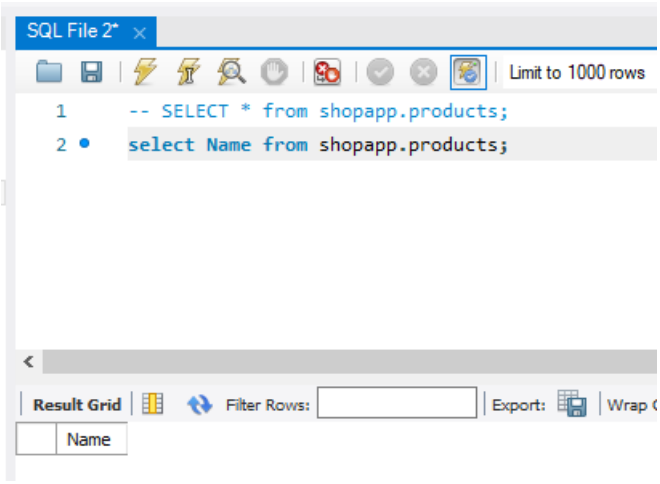


Yukarıdaki + butonu ile yeni bir boş SQL Query sayfası yaratılır ve artık bu sayfa içine istenilen query elle yazılabilir.

Aşağıdaki sorgu yazıldıktan sonra şimşek iconlu buton ile çalıştırılabilir, tablo boş olduğu için sonuçta boş bir tablo göreceğiz.



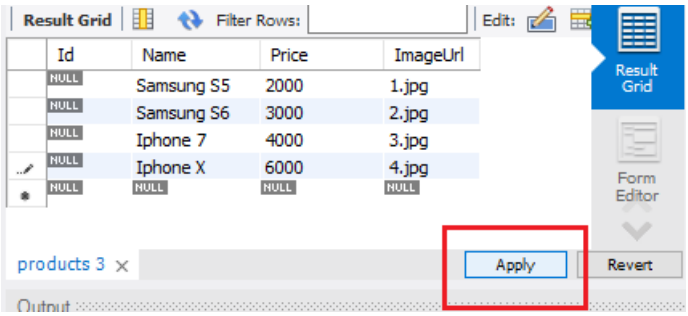
Tek bir column'u seçmek için SELECT columnName from … yapısını kullanırız:



Comment satırı için - - kullanıldığına dikkat et.

Tabloya IDE ile Eleman Ekleyelim:

Select ile tüm tabloyu seçtikten sonra aşağıda NULL bir tablo görünüyordu buna tıklayıp elle doldurduk:



Eğer Apply dersek, IDE bizim için server'a gerekli eleman ekleme query'lerini gönderecek, tabi önce bizden onay alacak:

IDE aşağıdaki gibi bir kodu server'a yolluyorum emin misin diyor, evet dersek artık database içindeki products tablomuz güncellenmiş olacak.

```
1   INSERT INTO `shopapp`.`products` (`Name`, `Price`, `ImageUrl`) VALUES ('Samsung S5', '2000', '1.jpg');
2   INSERT INTO `shopapp`.`products` (`Name`, `Price`, `ImageUrl`) VALUES ('Samsung S6', '3000', '2.jpg');
3   INSERT INTO `shopapp`.`products` (`Name`, `Price`, `ImageUrl`) VALUES ('Iphone 7', '4000', '3.jpg');
4   INSERT INTO `shopapp`.`products` (`Name`, `Price`, `ImageUrl`) VALUES ('Iphone X', '6000', '4.jpg');
5
```

Artık aşağıdaki gibi bir sorgu yaptığımızda sonucunu göreceğiz.

```
3 •  SELECT name,price from shopapp.products;
4
```

Result Grid | Filter Rows: | Export: | Wra

name	price
Samsung S5	2000
Samsung S6	3000
Iphone 7	4000
Iphone X	6000

Query language case sensitive değil, o yüzden column name'ler bile aynı olduğu sürece büyük küçük harf yazılması önemli değil.

Column'u Farklı İsimle Çağırmak

İstersek bir column'u ismiyle çağırıp, sonuçta görünecek tabloda ismini değiştirebiliriz:

Select columnName as customName from …

Limit to 1000 rows

```
1 •  SELECT name as İSİM from shopapp.products;
2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

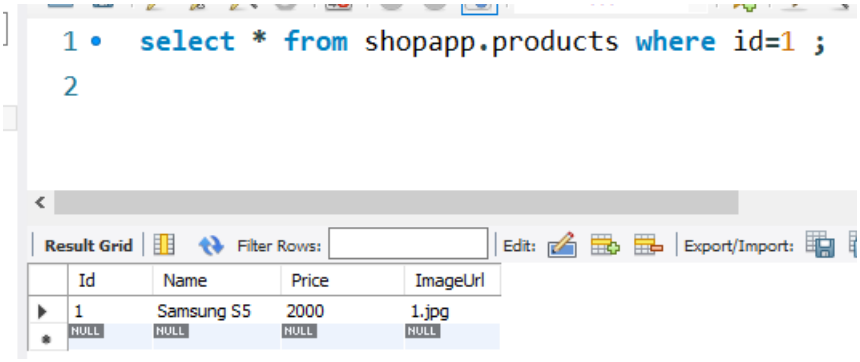
İSİM
Samsung S5
Samsung S6
Iphone 7
Iphone X

2. WHERE

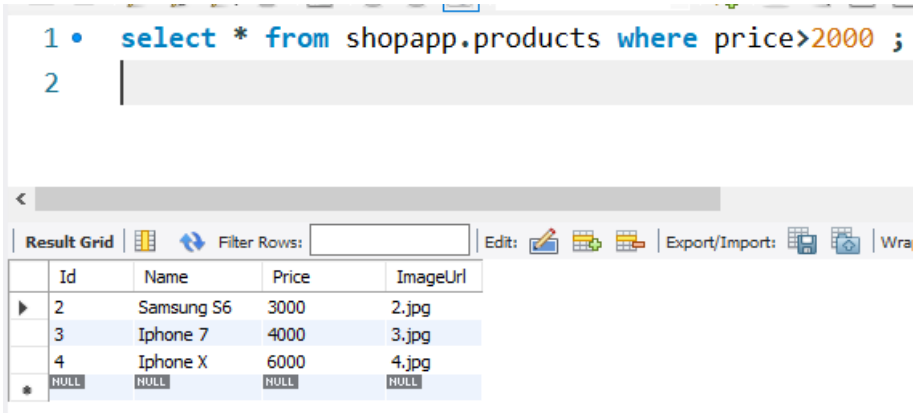
Select column1,column2 from … diyerek yalnızca istediğimiz columnları sorgulayabiliyorduk yani column bazında bir filtreleme yapabiliyorduk.

Where ile de satır bazında bir filtreleme yapabiliriz:

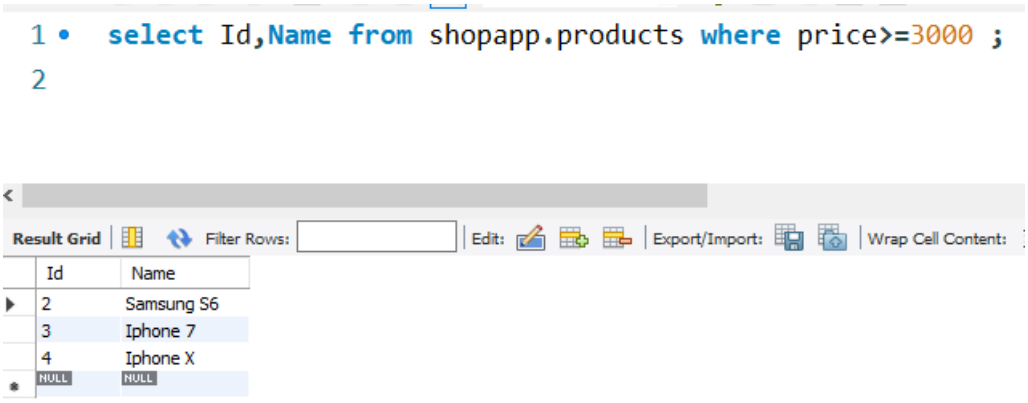
id=1 olan satırı filtreleyelim:



Price>2000 olan satırları filtreleyelim:

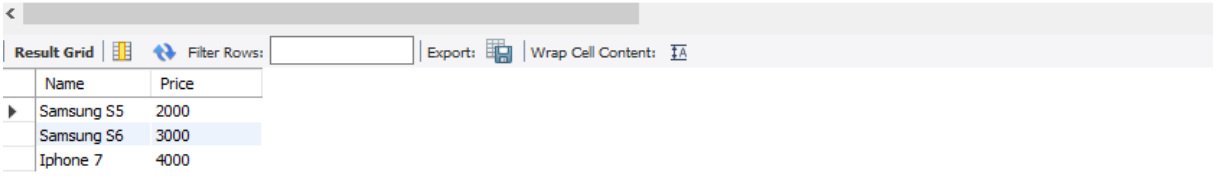


price>=3000 olan elemanların Id ve Name columnlarını filtreleyelim:



AND operatörü ile iki farklı where koşulunu bağlama:

```
1 •  select Name,Price from shopapp.products where price>=2000 and price<=4000 ;
2
```



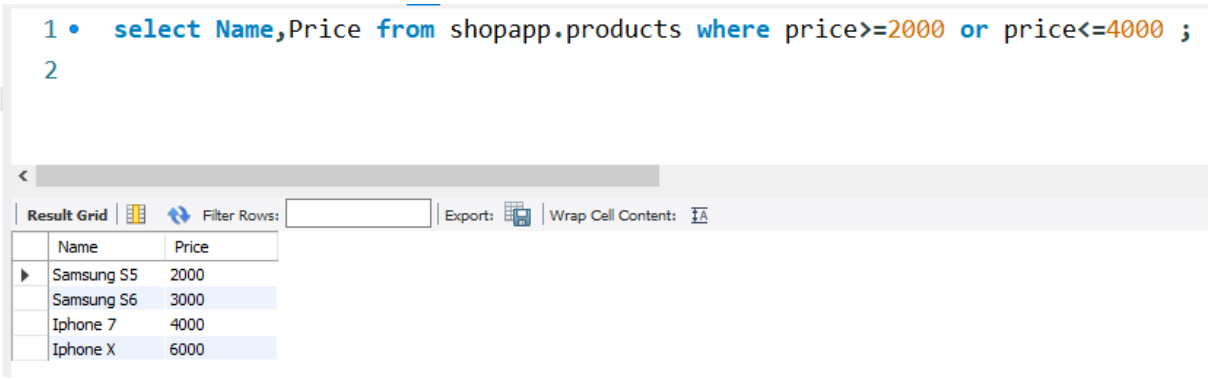
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	Price
Samsung S5	2000
Samsung S6	3000
Iphone 7	4000

- Price>=2000 **ve** <=4000 olan elemanların Name ve Price column'ları getirildi.

OR operatörü ile where koşullarını bağlama:

```
1 •  select Name,Price from shopapp.products where price>=2000 or price<=4000 ;
2
```



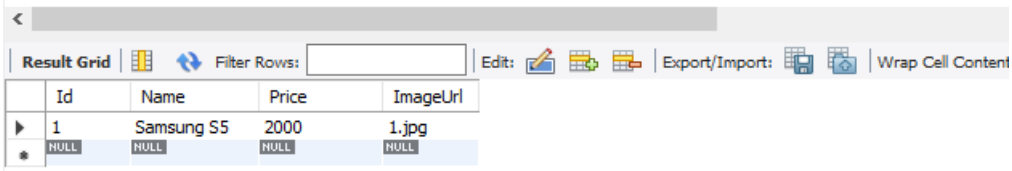
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	Price
Samsung S5	2000
Samsung S6	3000
Iphone 7	4000
Iphone X	6000

- İki koşuldan herhangi birini sağlayan satırlar döndürülecek.

Name="Samsung s5" olan satırlar getirilsin.

```
1 •  select * from shopapp.products where name="Samsung S5";
2
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

Id	Name	Price	ImageUrl
1	Samsung S5	2000	1.jpg
NULL	NULL	NULL	NULL

Name!=”Samsung s5” olan satırlar getirilsin.

```
1 •  select * from shopapp.products where name!="Samsung S5";
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl
2	Samsung S6	3000	2.jpg
3	Iphone 7	4000	3.jpg
4	Iphone X	6000	4.jpg
NULL	NULL	NULL	NULL

Aynı işlem için NOT operatörü de kullanılabilirdi:

```
1 •  select * from shopapp.products where NOT name="Samsung S5";
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl
2	Samsung S6	3000	2.jpg
3	Iphone 7	4000	3.jpg
4	Iphone X	6000	4.jpg
NULL	NULL	NULL	NULL

Burada name alanı Samsung S5’e eşit olan satırlar bulunsun, sonra bunların dışında kalan tüm alanlar getirilsin demiş olduk.

3 Koşulu Bağlama

```
1 •  select * from shopapp.products where id>2 and (price=4000 or price=6000);
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl
3	Iphone 7	4000	3.jpg
4	Iphone X	6000	4.jpg
NULL	NULL	NULL	NULL

İd>2 olan ve price=4000 veya 6000 olan tüm satırlar gelsin.

3. WHERE Yardımcı Operatörler (Between, In, Like)

Between Yardımcı Operatörü

Önceki kısımda fiyatı 2000 ile 4000 arasında olan satırları aşağıdaki şekilde elde etmiştik:

```
1 •  select * from shopapp.products where price>=2000 and price<=4000
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl
1	Samsung S5	2000	1.jpg
2	Samsung S6	3000	2.jpg
3	Iphone 7	4000	3.jpg
NULL	NULL	NULL	NULL

Buna alternatif olarak between operatörü kullanılabilir:

```
1 •  select * from shopapp.products where price between 2000 and 4000;
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl
1	Samsung S5	2000	1.jpg
2	Samsung S6	3000	2.jpg
3	Iphone 7	4000	3.jpg
NULL	NULL	NULL	NULL

Benzer şekilde **NOT BETWEEN** ile id'si 2-3 arasında olmayan satırları elde edebiliriz:

```
1 •  select * from shopapp.products where id not between 2 and 3;
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl
1	Samsung S5	2000	1.jpg
4	Iphone X	6000	4.jpg
NULL	NULL	NULL	NULL

Bir sonraki yardımcı where operatörüne geçmeden önce tabloya yeni bir category column'u ekledik bunun için tablonun yanındaki anahtar işaretini kullandığımızı unutma! Daha sonra da tabloya yeni girdiler ekledik:

▶	1	Samsung S5	2000	1.jpg	Telefon
	2	Samsung S6	3000	2.jpg	Telefon
	3	Iphone 7	4000	3.jpg	Telefon
	4	Iphone X	6000	4.jpg	Telefon
	5	Samsung S5	3000	1.jpg	Telefon
	6	Lenova	7000	5.jpg	Bilgisayar
*	NULL	NULL	NULL	NULL	NULL

Daha önceden bildiğimiz gibi mesela category='Telefon' ile telefonları sorgulayabiliriz.

```
1 • select * from shopapp.products where category='Telefon';
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	Id	Name	Price	ImageUrl	Category
▶	1	Samsung S5	2000	1.jpg	Telefon
	2	Samsung S6	3000	2.jpg	Telefon
	3	Iphone 7	4000	3.jpg	Telefon
	4	Iphone X	6000	4.jpg	Telefon
	5	Samsung S5	3000	1.jpg	Telefon
*	NULL	NULL	NULL	NULL	NULL

Benzer şekilde category'si telefon veya bilgisayar olanları da or ile isteyebiliriz:

```
1 • select * from shopapp.products where category='Telefon' or category='Bilgisayar';
2
```

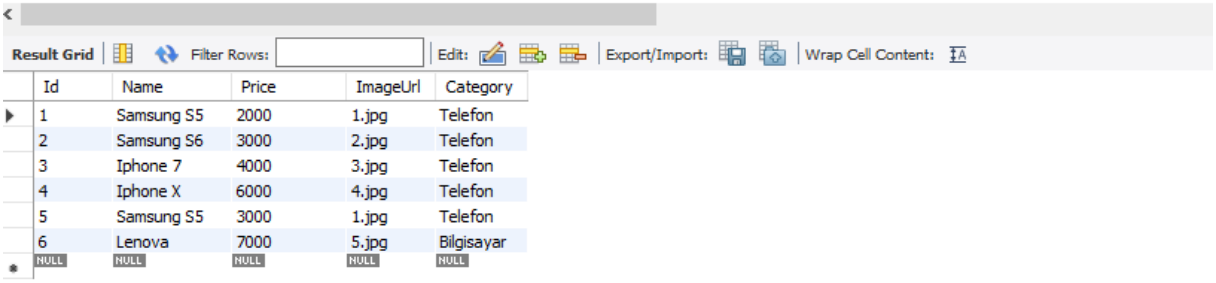
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	Id	Name	Price	ImageUrl	Category
▶	1	Samsung S5	2000	1.jpg	Telefon
	2	Samsung S6	3000	2.jpg	Telefon
	3	Iphone 7	4000	3.jpg	Telefon
	4	Iphone X	6000	4.jpg	Telefon
	5	Samsung S5	3000	1.jpg	Telefon
	6	Lenova	7000	5.jpg	Bilgisayar
*	NULL	NULL	NULL	NULL	NULL

In Yardımcı Operatörü

Az önce or ile yapılan sorgunun daha temizi In ile yapılabilir:

```
1 •  select * from shopapp.products where category in ('Telefon','Bilgisayar');
2
```



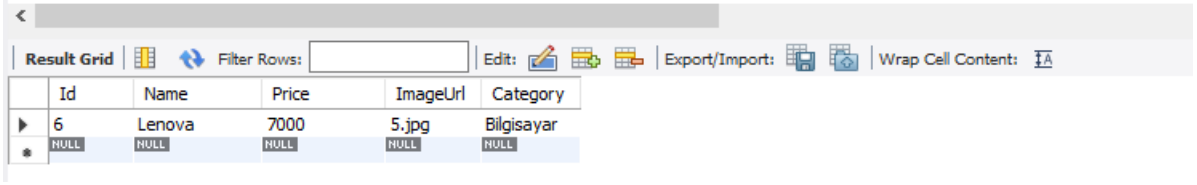
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category
1	Samsung S5	2000	1.jpg	Telefon
2	Samsung S6	3000	2.jpg	Telefon
3	Iphone 7	4000	3.jpg	Telefon
4	Iphone X	6000	4.jpg	Telefon
5	Samsung S5	3000	1.jpg	Telefon
6	Lenova	7000	5.jpg	Bilgisayar
NULL	NULL	NULL	NULL	NULL

Yani in operatörü ile bir categorical variable'a göre row filtering yapabiliyoruz.

Not In

```
1 •  select * from shopapp.products where category not in ("telefon");
2
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category
6	Lenova	7000	5.jpg	Bilgisayar
NULL	NULL	NULL	NULL	NULL

Category'si telefon olmayan satırlar döndürülür

In'in parametresi bir başka select statement olabilir:

```
WHERE value IN (SELECT value FROM table_name)
```

★ List of values can be a result set of a **SELECT** statement

Like Yardımcı Operatörü

Bu operatörü örneğin isim alanı “Sam” ile başlayan ya da içerisinde “ov” geçen ya da “ken” ile biten elemanları sorgulamak için kullanırız.

Like ile daha detaylı sorgular da yapılabilir.

Örneğin ürün özellikerini belirten bir description column’u olsun burada da ürünlerin özellikleri yazsın diyelim, like yardımcı operatörü sayesinde mesela hızlı şarj özelliğine sahip olan ürünleri sorgulayabiliriz.

İçinde name alanında samsung geçen tüm elemanları döndürür.

```
1 • select * from shopapp.products where name like '%samsung%'
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category
1	Samsung S5	2000	1.jpg	Telefon
2	Samsung S6	3000	2.jpg	Telefon
5	Samsung S5	3000	1.jpg	Telefon
NULL	NULL	NULL	NULL	NULL

%samsung% demek başında sonunda ne olduğu önemli değil, içinde samsung geçsin yeter demektir.

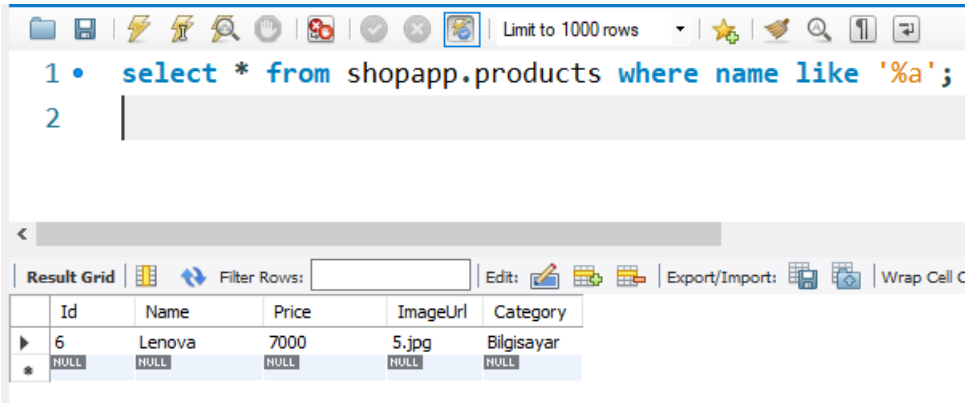
Name alanı l ile başlayan gerisi önemli olmayan elemanlar:

```
1 • select * from shopapp.products where name like 'l%';
2
```

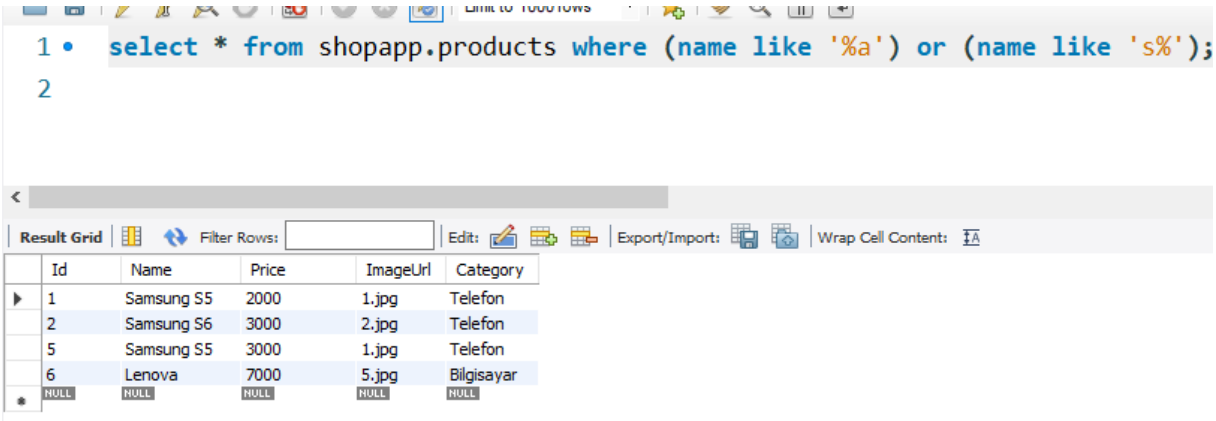
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell C

Id	Name	Price	ImageUrl	Category
6	Lenova	7000	5.jpg	Bilgisayar
NULL	NULL	NULL	NULL	NULL

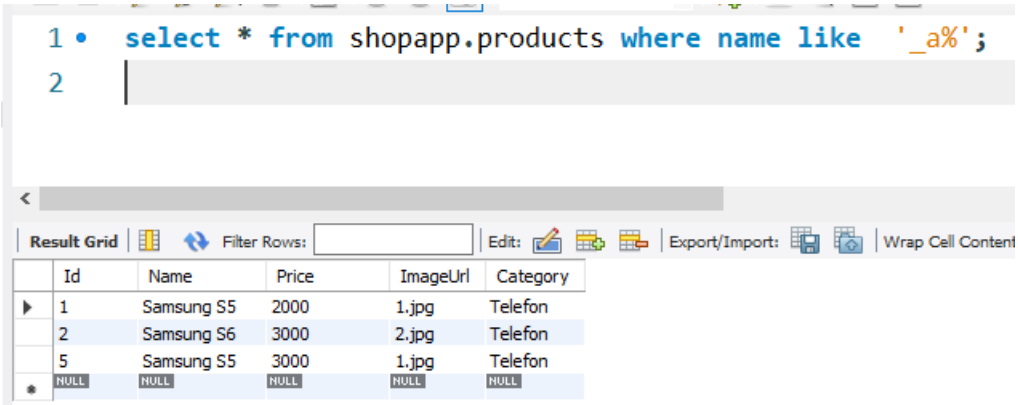
Name alanı a ile biten başı önemli olmayan elemanlar:



Name alanı a ile biten ya da s ile başlayan elemanlar:



Name alanının ikinci harfi a olan ve gerisi önemli olmayan elemanlar:



İlk ve üçünce harfi i ve h olan gerisi önemli olmayan kayıtlar:

```
1 • select * from shopapp.products where name like  'i_h%';
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category
3	Iphone 7	4000	3.jpg	Telefon
4	Iphone X	6000	4.jpg	Telefon
NULL	NULL	NULL	NULL	NULL

İçinde samsung geçmeyen kayıtları getir:

```
1 • select * from shopapp.products where name not like '%samsung%';
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category
3	Iphone 7	4000	3.jpg	Telefon
4	Iphone X	6000	4.jpg	Telefon
6	Lenova	7000	5.jpg	Bilgisayar
NULL	NULL	NULL	NULL	NULL

İsim alanında samsung geçen ve fiyatı 2000'den bütük olan kayıtları getir:

```
1 • select * from shopapp.products where (name like '%samsung%') and (price>2000);
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category
2	Samsung S6	3000	2.jpg	Telefon
5	Samsung S5	3000	1.jpg	Telefon
NULL	NULL	NULL	NULL	NULL

Tabloya aşağıdaki gibi bir description alanı eklenmiş olduğun varsayalım.

Id	Name	Price	ImageUrl	Category	Description
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
NULL	NULL	NULL	NULL	NULL	NULL

Şimdi siyah iphone kayıtlarını filtreleyelim:

```
1 • select * from shopapp.products where name like "%iphone%" and description like "%siyah%";
2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category	Description
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
NULL	NULL	NULL	NULL	NULL	NULL

4. ORDER - Kayıt Sıralama

Fiyata göre artan sıralayalım:

```
2 •  select * from shopapp.products order by price;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Id	Name	Price	ImageUrl	Category	Description
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
NULL	NULL	NULL	NULL	NULL	NULL

Fiyata göre azalan sıralayalım:

```
2 •  select * from shopapp.products order by price desc;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

Id	Name	Price	ImageUrl	Category	Description
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
NULL	NULL	NULL	NULL	NULL	NULL

Name'e göre alfabetik artan sıralayalım:

```
2 •  select * from shopapp.products order by Name;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Id	Name	Price	ImageUrl	Category	Description
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
NULL	NULL	NULL	NULL	NULL	NULL

Category'e göre asc sıralama yapalım, zaten default ascending olduğu için asc yazmıyoruz.

```
2 •  select * from shopapp.products order by category;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

Id	Name	Price	ImageUrl	Category	Description
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
NULL	NULL	NULL	NULL	NULL	NULL

Burada önemli bir nokta şu tamam kayıtlar kategoriye göre sıralandı önce bilgisayarlar sonra telefonlar geldi ancak telefonların kendi içerisindeki sıralama neye göre yapıldı?

Şuanda bu Id'ye göre yapılıyor, işte istersek biz bu ikinci sıralama kriterini de belirtebiliriz, hatta daha fazla sıralama kriterini de ekleyebiliriz:

Birincil olarak kategoriye göre artan ikincil olarak da price'a göre azalan bir sıralama yapılsın:

```
2 •  select * from shopapp.products order by category asc,price desc;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category	Description
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
NULL	NULL	NULL	NULL	NULL	NULL

5. SQL Fonksiyonları – Hesaplama – Min, Max, Count, Avg, Sum

Bu fonksiyonlara giriş yapmadan önce bi kaç önemli komuttan daha bahsedelim:

```
SELECT distinct column1 FROM table_name;
```

★Column1'in unique elemanları elde edilecek.

★ Let's say I want to see the first 10 rows of the customer table. Like df.head() function in pandas.

```
SELECT * FROM customer limit 10;
```

Challenges:

★ Get the customer IDs for the top 10 highest payment amounts from payment_table:

```
SELECT cutomer_id
FROM payment_table
ORDER BY amount DESC
LIMIT 10;
```

Challenges:

★ How many payment transactions were greater than $5.00?

```
SELECT COUNT(amount)
FROM payment_table
WHERE amount>5;
```

Where amount>5 ile döndürülen tablodaki amount sütununun sayısı sayılacak. İstersen count(*) de aynı sonuç çıkacak.

★ How many actors have a first name that starts with the letter P?

```
SELECT COUNT(*)
FROM actor_table
WHERE first_name LIKE 'P%';
```

★ How many unique districts are our customers from?

```
SELECT COUNT(DISTINCT district)
FROM address;
```

Aynen bir column'u saydırır gibi distinct columnName dediğimizde o column'un distinct halini saymış oluyoruz.

★ Retrieve the list of names for those distinct districts from the previous challenge.

```
SELECT DISTINCT district
FROM address;
```

★ How many films have a rating of R and a replacement cost between $5 and $15?

```
SELECT COUNT(*)
FROM film
WHERE rating='R' AND replacement_cost BETWEEN 5 AND 15;
```

Önce where gerçekleşiyor döndürülen listede satırlar sayılıyor.

★ How many films have the word Truman somewhere in the title?

```
SELECT COUNT(*)
FROM film
WHERE title LIKE '%Truman%';
```

Yukarıdaki örneklerde zaten count'ı görmüş olduk ancak şimdi baştan önemli fonksiyonlara giriş yapalım:

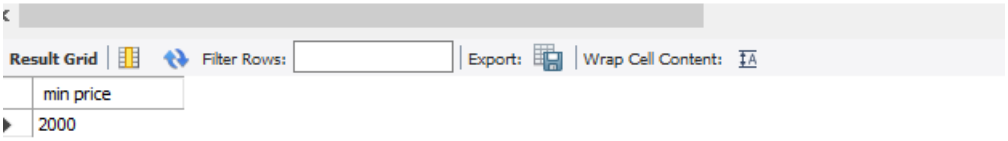
MIN( )

Aşağıda price column'unun minimum değeri hesaplandı ve sonucunda min(price) başlığı altında getirildi.

```
2 •  select min(price) from shopapp.products;
```

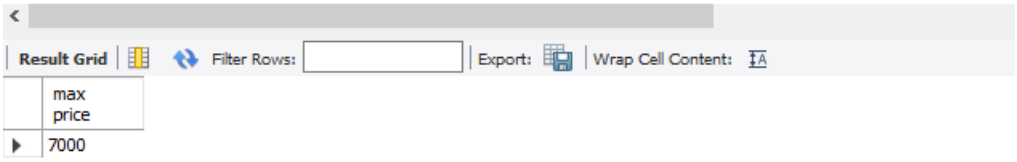
Column ismini değiştirmek istersek:

```
2 •  select min(price) as 'min price' from shopapp.products;
```



MAX( )

```
2 •  select max(price) as 'max price' from shopapp.products;
```



Max price'a sahip elamanın diğer özelliklerini de görüntülemek isteseydim:

```
2 •  select * from shopapp.products order by price desc limit 1
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category	Description
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
NULL	NULL	NULL	NULL	NULL	NULL

Ya da

```
2 •  select * from shopapp.products where price=(select max(price) from shopapp.products);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category	Description
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
NULL	NULL	NULL	NULL	NULL	NULL

Order by where'den sonra gelir:

```
2 •  select * from shopapp.products
3    where price>2000
4    order by price desc
```

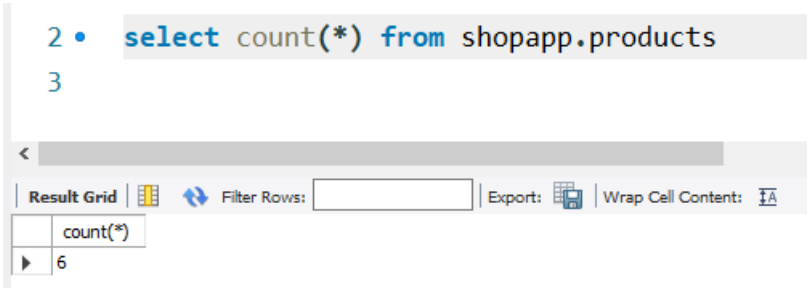
Result Grid | Filter Rows: | Edit: | Export/Import:

Id	Name	Price	ImageUrl	Category	Description
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
NULL	NULL	NULL	NULL	NULL	NULL

Önce price>2000 kayıtları getirilir sonra azalan şekilde sıralanır, sona limit deseydik en son da o eklenirdi.

COUNT( )

Döndürülen sonuç tablosunda tüm satırların sayısını hesapla ve count(*) olarak göster.



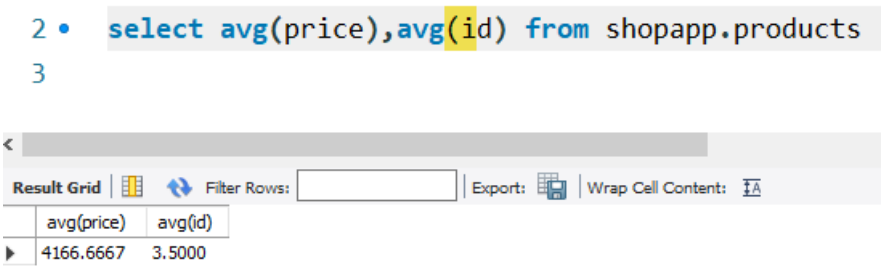
Burada count(*) yerine herhangi bir column versek de sonuç aynı çıkacaktı.

★ Finally we can use COUNT with DISTINCT to count only the distinct elements of a specified column.

```
SELECT COUNT(DISTINC column) FROM table;
```

AVG( )

Price column'unun ve id column'unun ortalamaları hesaplandı ve select edildi.



SUM( )

```
2 •  select sum(price) from shopapp.products
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(price)
25000

Price ile id columnları elementwise çarpılıp yeni column'un sum'ı döndürülebilir

```
1
2 •  select sum(price*id) from shopapp.products
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(price*id)
101000

Daha farklı bir şey de yapılabilir:

```
2 •  select *, price*id from shopapp.products
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category	Description	price*id
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp	2000
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp	6000
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp	12000
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp	24000
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp	15000
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz	42000

Price*id ismindeki yeni column direk iki column'un çarpımı şeklinde elde edildi, bunun yanında diğer tüm satırlar da getirildi.

6. SQL Fonksiyonları – String – Length, Left, Right, Concat, Lower, Upper, Trim

LOWER() / UPPER()

```
2 • select lower(category) from shopapp.products
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

lower(category)
telefon
telefon
telefon
telefon
telefon
bilgisayar

Burada yapılan şey aynı select category demek gibi. Bu yüzden bu döndürülen listenin order'ı id sırasına göre yanı orijinal liste gibi.

Yani şöyle düşün category column'unu seçmişiz sonra tüm elemanları lowercase yapmışız.

Bunu daha iyi anlamak için:

```
2 • select *,lower(category) from shopapp.products
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category	Description	lower(category)
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp	telefon
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp	telefon
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp	telefon
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp	telefon
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp	telefon
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz	bilgisayar

Hatta:

```
2 • select *,lower(category) from shopapp.products order by name
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Id	Name	Price	ImageUrl	Category	Description	lower(category)
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp	telefon
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp	telefon
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz	bilgisayar
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp	telefon
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp	telefon
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp	telefon

Burada tüm liste ve category column seçildi yerleştirildi sonra da name'e göre order edildi.

LENGTH()

```
2 • select length('Erdogan Yıldız') as CharLen
3
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CharLen
16

```
2 • select name,length(name) from shopapp.products;
3
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	length(name)
Samsung S5	10
Samsung S6	10
Iphone 7	8
Iphone X	8
Samsung S5	10
Lenova	6

LEFT(column, #) / RIGHT((column, #))

```
2 • select name,left(name,3) as first3let from shopapp.products;
3
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	first3let
Samsung S5	Sam
Samsung S6	Sam
Iphone 7	Iph
Iphone X	Iph
Samsung S5	Sam
Lenova	Len

CONCAT()

Bir column ile bir '…' string'ini bağladı bunu aynı python'da olduğu gibi broadcasting yaptı otomatik olarak column'daki her elemana … ekledi.

```
2 •  select name, concat(left(name,3),'...') from shopapp.products;
3
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	concat(left(name,3),'...')
Samsung S5	Sam...
Samsung S6	Sam...
Iphone 7	Iph...
Iphone X	Iph...
Samsung S5	Sam...
Lenova	Len...

REPLACE(column,string,string)

Name alanı içindeki space karakterlerini – ile değiştirelim, bu yeni column'u da select edelim.

```
2 •  select name, replace(name,' ','-') from shopapp.products;
3
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	replace(name,' ','-')
Samsung S5	Samsung-S5
Samsung S6	Samsung-S6
Iphone 7	Iphone-7
Iphone X	Iphone-X
Samsung S5	Samsung-S5
Lenova	Lenova

TRIM()/LTRIM()/RTRIM()

Elemanın soldan sağdan veya heriki taraftan olan boşluklarını siler. Yani bir name elemanı eğer ' Samsung S5 ' şeklinde ise trim edildiğinde 'Samsung S5' haline gelir.

Elbette daha birçok numeric ve string fonksiyonları kullanılabilir bunlara bakabilirsin.

7. Group By – Distinct, Having

Groupby ile bir tabloyu istediğimiz satıra göre gruplarız. Yani örneğin aşağıdaki tabloda DeptId'ye göre bir gruplama yapılmış, bu gruplandırma ile her grubun içerisindeki diğer bilgiler aslında tutuluyor. Yani deptId'ye göre guruplandırdıktan sonra avg(salary) dediğimizde her grubun average salary'si hesaplanmış olacak.

Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	300
1006	Steve	3	4500
1007	Alice	3	3500

```
SELECT DeptID, AVG(Salary)
FROM Employee
GROUP BY DeptID;
```

GROUP BY Employee Table using DeptID

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

Önce distinct keyword'ünü hatırlayalım:

```
3 •  select category from shopapp.products;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category
Telefon
Telefon
Telefon
Telefon
Telefon
Bilgisayar

```
3 •  select distinct category from shopapp.products;
```

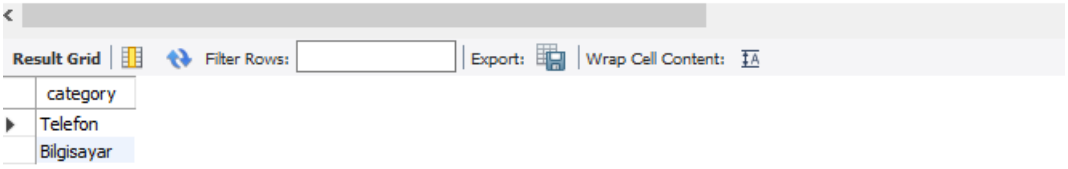
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category
Telefon
Bilgisayar

Aslında aggregate function olmadan groupby statement'ı kullanıldığında aynı distinct kullanmışız gibi bir görüntü elde ediyoruz, groupby'ın asıl olayı aggregate function'larla orataya çıkıyor.

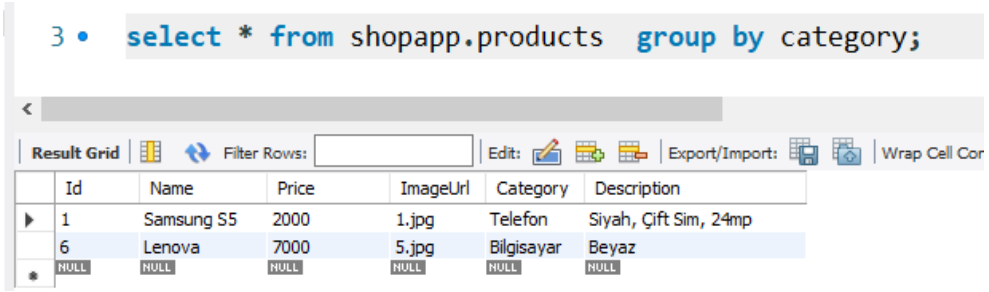
Groupby ile gruplandırılan her grubun içerisindeki diğer eleman bilgileri tutuluyor, bu elemanlar üzerinde functions çalıştırabiliyoruz.

```
3 •  select category from shopapp.products group by category;
```



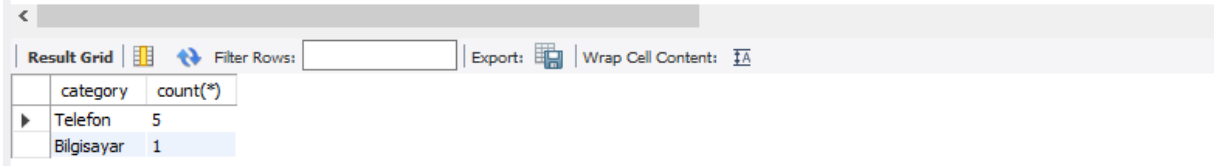
Önce tablo gruplandırılıyor daha sonra ilgili tablodan category sütunu seçiliyor, eğer category yerine * deseydik yine sadece 2 satır görecektik çünkü 2 grup oluştu, 2 grubun ilk elemanları görünecekti:

```
3 •  select * from shopapp.products  group by category;
```



Group By with Aggregate Functions

```
3 •  select category, count(*) from shopapp.products  group by category;
```



Kategoriye göre gruplandırma yapılıyor, daha sonra aggregate functions bu grupları baz alarak çalışıyor.

Count fonksiyonu grupların elemanlarını saydı ve yeni bir column olarak select edildi.

Group By with Sum function

```
3 • select category,sum(price) from shopapp.products  group by category;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	sum(price)
Telefon	18000
Bilgisayar	7000

Görüldüğü sum(price) dediğimizde her kategori grubunun içinde kalan elemanların price değerlerinin toplamını elde etmiş oluyoruz.

Group By with Avg function

```
3 • select category,avg(price) from shopapp.products  group by category;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	avg(price)
Telefon	3600.0000
Bilgisayar	7000.0000

Fiyatı 2000'den büyük olan elemanlar groupby'a dahil edilsin:

```
3 • select category,avg(price) from shopapp.products
4   where price>2000
5   group by category
```

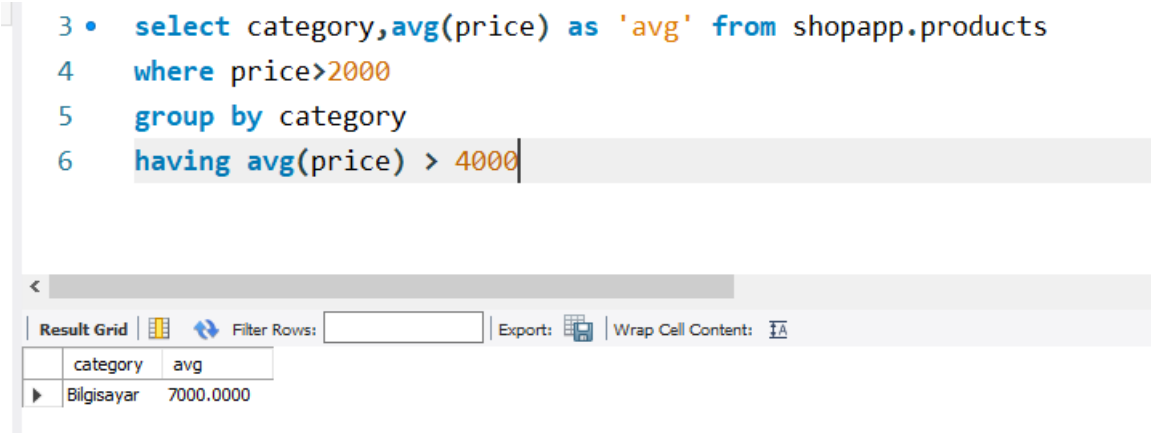
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	avg(price)
Telefon	4000.0000
Bilgisayar	7000.0000

Where kullanıldığında önce where statement'ı ile filtreleme yapılıyor daha sonra gruplandırma yapılıyor.

Eğer amacımız gruplandırma yaptıktan sonra tekrar bir satır bazlı filtreleme yapmaksa o halde **HAVING** kullanırız.

HAVING statement



Having aynı where gibi ancak where groupby'dan önce çalışırken having ise groupby'dan sonra çalışarak satır bazlı filtreleme yapar.

8. Insert – Kayıt Ekleme

Database'deki shopapp tablosunun güncel haline bir bakalım:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expressic
Id	INT	☑	☑	☑	☐	☐	☐	☑	☐	
Name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
Price	DECIMAL(10,0)	☐	☑	☐	☐	☐	☐	☐	☐	
ImageUrl	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Category	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Description	VARCHAR(500)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Burada görüyoruz ki bazı alanlar Not Null olarak belirtilmiş bunlar: Id,Name ve Price alanı. Id dışındaki diğer iki alana kesinlikle değer gönderilmesi lazım.

Id ise Auto Increment olarak belirtildiği için yeni bir kayıt eklerken bu değeri belirtmeyeceğiz.

Yeni satır eklemeyi ID ile yapabileceğimiz gibi SQL ile de aşağıdaki gibi yapabiliriz:

```
insert into shopapp.products (Name,Price,ImageUrl,Category,Description)
values ('Samsung S10', 7000, '1.jpg', 'Telefon' , 10);
```

Tablomuza yeni bir satır eklendi:

Id	Name	Price	ImageUrl	Category	Description
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
7	Samsung S10	7000	1.jpg	Telefon	10
NULL	NULL	NULL	NULL	NULL	NULL

Önemli bir hatırlatma COUNT methodu ilgili column'daki null entry'leri saymaz, sadece null olmayanların sayısını verir.

9. **Update Methodu – Kayıt Güncelleme**

Tablonun son hali aşağıda:

Id	Name	Price	ImageUrl	Category	Description
1	Samsung S5	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
7	Samsung S10	7000	1.jpg	Telefon	Mavi, Tek Sim, 64mp

Diyelim ki ilk kaydın isim alanını güncellemek istiyoruz:

```
3 •  update shopapp.products
4    set name='Samsung S5 New'
5    where id=1
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Id	Name	Price	ImageUrl	Category	Description
1	Samsung S5 New	2000	1.jpg	Telefon	Siyah, Çift Sim, 24mp
2	Samsung S6	3000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
5	Samsung S5	3000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
7	Samsung S10	7000	1.jpg	Telefon	Mavi, Tek Sim, 64mp
NULL	NULL	NULL	NULL	NULL	NULL

Eğer where id=1 demeseydik tüm name alanları güncellenirdi.

Hem name'i hem price'ı güncelleyelim:

```
3 •  update shopapp.products
4    set name='Samsung S5 New' , price= 2500
5    where id=1
```

Diyelim ki isminde samsung geçen ürünlerin fiyatına 1000 lira eklenecek şekilde bir update yapmak istiyorum:

```
3 •  update shopapp.products
4    set price = price + 1000
5    where name like '%samsung%'
```

Bu method bize hata verecek ancak başına bir satır eklersek aynı kodu çalıştırabiliriz:

```
3 •  set sql_safe_updates = 0;
4 •  update shopapp.products
5    set price = price + 1000
6    where name like '%samsung%'
```

Sonuçta yeni tabloda samsung isimli ürünlerin fiyatı 1000 arttı:

Id	Name	Price	ImageUrl	Category	Description
1	Samsung S5 New	3500	1.jpg	Telefon	Siyah, Çift Sim, 24mp
2	Samsung S6	4000	2.jpg	Telefon	Siyah, Çift Sim, 48mp
3	Iphone 7	4000	3.jpg	Telefon	Siyah, Tek Sim, 16mp
4	Iphone X	6000	4.jpg	Telefon	Gri, Tek Sim, 64mp
5	Samsung S5	4000	1.jpg	Telefon	Beyaz, Tek Sim, 16mp
6	Lenova	7000	5.jpg	Bilgisayar	Beyaz
7	Samsung S10	8000	1.jpg	Telefon	Mavi, Tek Sim, 64mp
NULL	NULL	NULL	NULL	NULL	NULL

Diyelim ki resim alanı NULL olan entry'lerin resim alanına noproduct.jpg'i yerleştirelim:

```
UPDATE shopapp.products
SET ImageUrl = 'noproduct.jpg'
WHERE ImageUrl IS NULL
```

10. **Delete Komutu ile Kayıt Silme**

Aşağıdaki komutu kullanırsak tüm kayıtlar silinecek.

```
DELETE FROM shopapp.products
```

İstediğimiz kriteri karşılayan kayıtları silmek istiyorsak:

```
DELETE FROM shopapp.products WHERE Id = 1
```

Price'ı 2000'den fazla ve kategorisi bilgisayar olan kayıtlar silinsin.

```
DELETE FROM shopapp.products WHERE price>2000 and category = 'bilgisayar'
```

Description alanı NULL olan kayıtları silelim:

```
DELETE FROM shopapp.products WHERE description is null
```