1. Birden Fazla Tablodan Kayıt Seçme – SQL Joins

Birden fazla tablodan kayıt almak istiyorsak, tabloları birleştirmemiz gerekecektir yani join işlemi yapmalıyız.

CustomerId	Name	Surname
3	Ada	Bilgi
4	Sadık	Turan
5	Çınar	Turan
6	Yiğit	Bilgi

OrderId	CustomerId	OrderDate
502	3	07-05-2018
503	4	03-05-2019
504	5	02-04-2020
505	3	03-05-2020

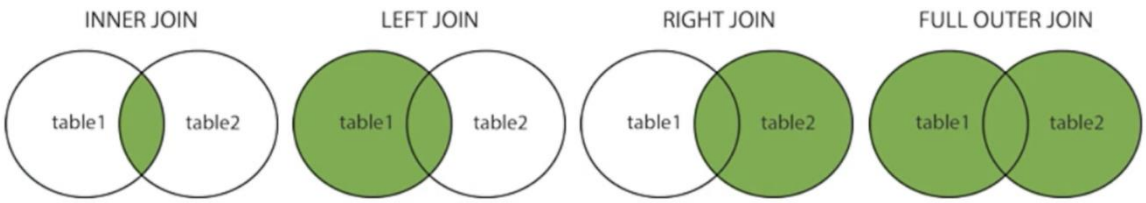
OrderId	CustomerName	OrderDate
502	Ada Bilgi	07-05-2018
503	Sadık Turan	03-05-2019
504	Çınar Turan	02-04-2020
505	Ada Bilgi	03-05-2020

Yukarıdaki gibi 2 tablomuz olsun: Customer ve Order tabloları. Bizim amacımız hangi order'ı hangi customer'ın hangi tarihte verdiğini gösteren tek bir tablo elde etmek.

Bu amaçla yukarıdaki iki tablo customerId column'ına göre join edilir. Böylece ili tabloda da ortak olan customer id satırları yanyana eklenmiş olur bunların arasından istediğimiz bilgileri seçebiliriz.

Bu join tipine INNER JOIN denir, iki tablonun da ortak kısımları alınır, yani örneğin 6 numaralı customer hiçbir order vermediği için join tablosunda yer almaz, benzer şekilde eğer bir order'ı customer tablosunda yer almayan bir customer vermiş olsaydı o da join tablosunda yer almazdı.

Join Tipleri



Inner join'i zaten yukarıda gördük, iki tablonunda ortak customerId olan satırları birleştiriliyor.

Buna karşın **left join** bu ortak kısımların yanısıra siparişi olmayan customer'ları da gösterir, bu siparişsiz customer'ların siparişle ilgili olan satırlarına (mesela OrderDate) null değer gelecek.

Bunun yanında **right join** de aynı mantık sadece hangi tablonun left/right olduğuyla ilgili, bizim örnekte bu customer tablosunda yer almayan müşterilen yaptığı siparişleri de göstermek anlamına geliyor. Örneğin bazı siparişler guest hesabı ile verilmiş karşısında bir customerId yok, işte bu siparişler de right join tablosunda görünecek, müşteri bilgileri kısmı null olarak gelecek.

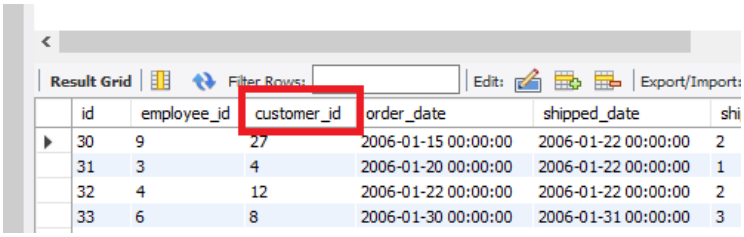
Son olarak **full outer join** hem iki tabloda ortak olan verileri gösterir hem sipariş tablosunda yer almayan customerları hem de customer tablosunda yer almayan siparişleri gösterir özetle önceki 3 join tablosunun birleşimidir diyebiliriz.

Şimdi bu join tiplerini daha detaylı anlayalım ve pratik yapalım.

2. Inner Join

Zaten mantığını biliyoruz, nortwind datasetinde de customers ve orders olmak üzere iki tablo var bu iki tablo üzerinde bir inner join oluşturmak istiyoruz.

Orders tablosunda müşteriler customer_id column'unda tutluyor:



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

id	employee_id	customer_id	order_date	shipped_date	shi
30	9	27	2006-01-15 00:00:00	2006-01-22 00:00:00	2
31	3	4	2006-01-20 00:00:00	2006-01-22 00:00:00	1
32	4	12	2006-01-22 00:00:00	2006-01-22 00:00:00	2
33	6	8	2006-01-30 00:00:00	2006-01-31 00:00:00	3

O halde orders tablosunun customer_id'si ile customers tablosunun id'sini kullanarak bir inner join tanımlayabilirim böylece tüm ortak id'li kayıtlar yanyana eklenmiş olur!

```
1 •  select * from orders inner join customers
2    on orders.customer_id = customers.id
```

Sonuçta elde edilen tablonun bir kısmı aşağıda soldaki kısım orders tablosundan geliyor sağdaki kısım customers tablosundan.

paid_date	notes	tax_rate	tax_status_id	status_id	id	company	last_name	first_name	email_address	job_title
NULL	NULL	0	NULL	0	1	Company A	Bedecs	Anna	NULL	Owner
NULL	NULL	0	NULL	0	1	Company A	Bedecs	Anna	NULL	Owner
2006-02-23 00:00:00	NULL	0	NULL	3	3	Company C	Axen	Thomas	NULL	Purchasing Represen
2006-04-25 00:00:00	NULL	0	NULL	3	3	Company C	Axen	Thomas	NULL	Purchasing Represen
NULL	NULL	0	NULL	0	3	Company C	Axen	Thomas	NULL	Purchasing Represen
2006-01-20 00:00:00	NULL	0	NULL	3	4	Company D	Lee	Christina	NULL	Purchasing Manager

Tabloların yerleri de değiştirilebilirdi aynı tablonun yerleri değişmiş versiyonunu elde ederdik.

```
1 •  select * from customers inner join orders
2    on orders.customer_id = customers.id
```

Sonuçta iki tablo belirtilen id'ler üzerinde inner join edildi ve tüm satırlar select edildi!

Inner Join bu kadar ancak şimdi biraz yukarıdaki query'e eklenebilecek şeyleri görelim:

Tüm column'ları değil sadece istediğimiz columnları görelim:

```
1 •  select orders.id, customers.first_name, orders.order_date from customers inner join orders
2    on orders.customer_id = customers.id
```

id	first_name	order_date
38	Amritansh	2006-03-10 00:00:00
45	Amritansh	2006-04-07 00:00:00
65	Amritansh	2006-05-11 00:00:00
72	Amritansh	2006-06-07 00:00:00
44	Anna	2006-03-24 00:00:00
71	Anna	2006-05-24 00:00:00

Tablolara Kısaltma Atayalım, böylece yukarıdaki sorguyu kısaltalım:

```
select o.id, c.first_name, o.order_date from orders o inner join customers c
on o.customer_id = c.id
```

Yukarıdaki şekilde query'nin heryerinde orders tablosuna o diyerek customers tablosuna c diyerek ulaşabiliyoruz, bu da query'i kısaltıyor.

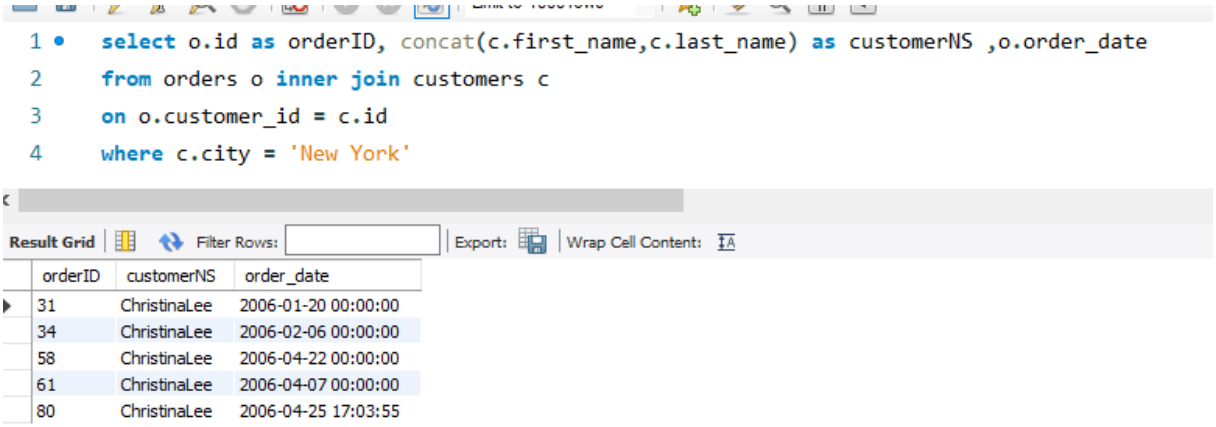
Column İsimlerini Değiştirelim ve Concat Fonksiyonunu Kullanalım:

```
1 •  select o.id as orderID, concat(c.first_name,c.last_name) as customerNS ,o.order_date
2    from orders o inner join customers c
3    on o.customer_id = c.id
```

orderID	customerNS	order_date
44	AnnaBedecs	2006-03-24 00:00:00
71	AnnaBedecs	2006-05-24 00:00:00
36	ThomasAxen	2006-02-23 00:00:00
63	ThomasAxen	2006-04-25 00:00:00
81	ThomasAxen	2006-04-25 17:26:53
31	ChristinaLee	2006-01-20 00:00:00
34	ChristinaLee	2006-02-06 00:00:00
58	ChristinaLee	2006-04-22 00:00:00

Concat fonksiyonu ile iki sütun string olarak concatanate edildi.

Join Edilmiş Tabloya Daha Önce Kullandığımız Sorguları Kullanabiliriz (Group by, Where, Etc.)



```
1 •  select o.id as orderID, concat(c.first_name,c.last_name) as customerNS ,o.order_date
2    from orders o inner join customers c
3    on o.customer_id = c.id
4    where c.city = 'New York'
```

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

orderID	customerNS	order_date
31	ChristinaLee	2006-01-20 00:00:00
34	ChristinaLee	2006-02-06 00:00:00
58	ChristinaLee	2006-04-22 00:00:00
61	ChristinaLee	2006-04-07 00:00:00
80	ChristinaLee	2006-04-25 17:03:55

Mesela where ile belirli satırlar elimine edilmiş.

Inner join için önemli bir diğer nokta bence şu, örneğin bir customer'ın birden fazla siparişi olabilir o halde join tablosunda aynu customer'a birden fazla satır ayrılacak her customer-order kombinasyonu ayrı ayrı bir kayıt olarak ele alınacak!

3. **İkiden Fazla Tabloyu Join Etmek**

Şimdi önceki kısımda yapılan şey orders tablosu ile customers tablosunu id'lere göre join etmekti:

```
1 •  select  o.id as orderID, concat(c.first_name,c.last_name) as customerNS ,o.order_date
2    from orders o
3    inner join customers c on o.customer_id = c.id
4    order by customerNS
```

Join edildiğinde orderID'ye karşılık customer isim ve soyadlarını aşağıdaki gibi elde etmiştik.

orderID	customerNS	order_date
38	AmritanshRaghav	2006-03-10 00:00:00
45	AmritanshRaghav	2006-04-07 00:00:00
65	AmritanshRaghav	2006-05-11 00:00:00
72	AmritanshRaghav	2006-06-07 00:00:00
44	AnnaBedecs	2006-03-24 00:00:00
71	AnnaBedecs	2006-05-24 00:00:00

Buna karşın bir order'ın içerisinde hangi product'lar olduğu bilgisi ise order_details tablosunda yer alıyor:

id	order_id	product_id	quantity	unit_price	discount	status_id	date_allocated	purchase_order_id	inventory_id
27	30	34	100.0000	14.0000	0	2	NULL	96	83
28	30	80	30.0000	3.5000	0	2	NULL	NULL	63
29	31	7	10.0000	30.0000	0	2	NULL	NULL	64
30	31	51	10.0000	53.0000	0	2	NULL	NULL	65

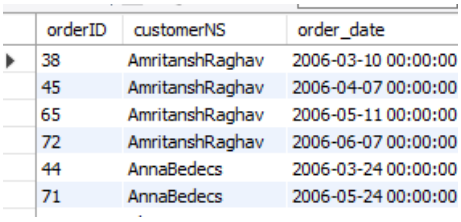
Görüldüğü gibi bir oder'ın içinde birden fazla product olabilir.

Peki ya biz, hem order bilgisini, hem customer bilgisini hem de o order'a karşılık hangi ürünlerin sipariş edildiğini elde etmek istersek ne yapacağız?

Başka tabloları da joinimize dahil edeceğiz!

Order Details tablosunu Join'e dahil edelim:

Şimdi orders ile customers'ın join hali aşağıdaki gibiydi:

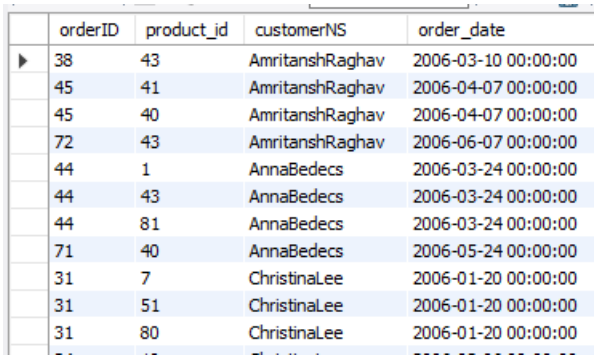


orderID	customerNS	order_date
38	AmritanshRaghav	2006-03-10 00:00:00
45	AmritanshRaghav	2006-04-07 00:00:00
65	AmritanshRaghav	2006-05-11 00:00:00
72	AmritanshRaghav	2006-06-07 00:00:00
44	AnnaBedecs	2006-03-24 00:00:00
71	AnnaBedecs	2006-05-24 00:00:00

Buna üçüncü bir tabloyu aşağıdaki gibi join edersek:

```
select  o.id as orderID, od.product_id,concat(c.first_name,c.last_name) as customerNS ,o.order_date
from orders o
inner join customers c on o.customer_id = c.id
inner join order_details od on od.order_id = o.id
order by customerNS
```

Artık her order_id'ye karşılık gelen order_detail satırı da yeni tabloya eklenecek, elbette bir order_id'ye birden fazla order_detail id gelebileceği için (çünkü bir order'da birden fazla product olabiliyor) join tablosunda aynı orderID'nin tekrarlamasını bekleriz:



orderID	product_id	customerNS	order_date
38	43	AmritanshRaghav	2006-03-10 00:00:00
45	41	AmritanshRaghav	2006-04-07 00:00:00
45	40	AmritanshRaghav	2006-04-07 00:00:00
72	43	AmritanshRaghav	2006-06-07 00:00:00
44	1	AnnaBedecs	2006-03-24 00:00:00
44	43	AnnaBedecs	2006-03-24 00:00:00
44	81	AnnaBedecs	2006-03-24 00:00:00
71	40	AnnaBedecs	2006-05-24 00:00:00
31	7	ChristinaLee	2006-01-20 00:00:00
31	51	ChristinaLee	2006-01-20 00:00:00
31	80	ChristinaLee	2006-01-20 00:00:00

Beklediğimiz gibi mesela 45 numaralı order'a karşılık 2 farklı order_detail satırı geliyormuş yani 40 ve 41 nolu product'lar alınmış.

Elbette artık detail tablosu da join edildiği için product_id column'una detail tablosu içinden ulaşabildik!

Products tablosunu da join'e ekleyelim:

Bunun yani her order'a karşılık alınan product'ları ve bunların isimlerini de görmek istiyorum diyelim, zaten product_id'leri görebiliyorduk.

Bu id'ye order_details tablosu ile ulaşabildik bu tablonun yapısı aşağıda:

id	order_id	product_id	quantity	unit_price	discount	status_id	date_allocated	purchase_order_id	inventory_id
27	30	34	100.0000	14.0000	0	2	NULL	96	83
28	30	80	30.0000	3.5000	0	2	NULL	NULL	63
29	31	7	10.0000	30.0000	0	2	NULL	NULL	64
30	31	51	10.0000	53.0000	0	2	NULL	NULL	65

Buna karşılık hangi product_id'nin hangi product'a karşılık geldiği ise aşağıdaki product tablosunda tutuluyor:

supplier_ids	id	product_code	product_name	description	standard_cost
4	1	NWTB-1	Northwind Traders Chai	NULL	13.5000
10	3	NWTCO-3	Northwind Traders Syrup	NULL	7.5000
10	4	NWTCO-4	Northwind Traders Cajun Seasoning	NULL	16.5000
10	5	NWTO-5	Northwind Traders Olive Oil	NULL	16.0125
2;6	6	NWTJP-6	Northwind Traders Boysenberry Spread	NULL	18.7500

Bu yüzden önceki kısımdaki üçlü join'imize bir tablo daha join ediyoruz, böylece her product_id'ye karşılık product_name'lerin de işin içine dahil edilmesini istiyoruz:

```
select  o.id as orderID, od.product_id, p.product_name, concat(c.first_name,c.last_name) as customerNS ,o.order_date
from orders o
inner join customers c on o.customer_id = c.id
inner join order_details od on od.order_id = o.id
inner join products p on od.product_id = p.id
order by customerNS
```

Sonuç:

orderID	product_id	product_name	customerNS	order_date
38	43	Northwind Traders Coffee	AmritanshRaghav	2006-03-10 00:00:00
45	41	Northwind Traders Clam Chowder	AmritanshRaghav	2006-04-07 00:00:00
45	40	Northwind Traders Crab Meat	AmritanshRaghav	2006-04-07 00:00:00
72	43	Northwind Traders Coffee	AmritanshRaghav	2006-06-07 00:00:00
44	1	Northwind Traders Chai	AnnaBedecs	2006-03-24 00:00:00
44	43	Northwind Traders Coffee	AnnaBedecs	2006-03-24 00:00:00
44	81	Northwind Traders Green Tea	AnnaBedecs	2006-03-24 00:00:00
71	40	Northwind Traders Crab Meat	AnnaBedecs	2006-05-24 00:00:00
31	7	Northwind Traders Dried Pears	ChristinaLee	2006-01-20 00:00:00
31	51	Northwind Traders Dried Apples	ChristinaLee	2006-01-20 00:00:00
31	80	Northwind Traders Dried Plums	ChristinaLee	2006-01-20 00:00:00

Total Order Harcamasını da Join Tablosunda Görelim

Bu işlem için yeni bir joine gerek yok çünkü order_details tablsounda hangi product'tan kaç adet satıldığını ve unit price'ı öğrenebiliyoruz:

id	order_id	product_id	quantity	unit_price	discount	status_id	date_allocated	purchase_order_id	inventory_id
27	30	34	100.0000	14.0000	0	2	NULL	96	83
28	30	80	30.0000	3.5000	0	2	NULL	NULL	63
29	31	7	10.0000	30.0000	0	2	NULL	NULL	64
30	31	51	10.0000	53.0000	0	2	NULL	NULL	65

Yani yuakrıdaki tabloda quantity ve unit_price'ı çarparak her bir order için kaç müşteriden kaç para temin edildiği hesaplanabilir:

```
select  o.id as orderID, (od.unit_price*od.quantity) as totalSpend,od.product_id, p.product_name, concat(
from orders o
inner join customers c on o.customer_id = c.id
inner join order_details od on od.order_id = o.id
inner join products p on od.product_id = p.id
order by customerNS
```

orderID	totalSpend	product_id	product_name	customerNS	order_date
38	13800.00000000	43	Northwind Traders Coffee	AmritanshRaghav	2006-03-10 00:00:00
45	482.50000000	41	Northwind Traders Clam Chowder	AmritanshRaghav	2006-04-07 00:00:00
45	920.00000000	40	Northwind Traders Crab Meat	AmritanshRaghav	2006-04-07 00:00:00
72	230.00000000	43	Northwind Traders Coffee	AmritanshRaghav	2006-06-07 00:00:00
44	450.00000000	1	Northwind Traders Chai	AnnaBedecs	2006-03-24 00:00:00
44	1150.00000000	43	Northwind Traders Coffee	AnnaBedecs	2006-03-24 00:00:00
44	74.75000000	81	Northwind Traders Green Tea	AnnaBedecs	2006-03-24 00:00:00
71	736.00000000	40	Northwind Traders Crab Meat	AnnaBedecs	2006-05-24 00:00:00
31	300.00000000	7	Northwind Traders Dried Pears	ChristinaLee	2006-01-20 00:00:00
31	530.00000000	51	Northwind Traders Dried Apples	ChristinaLee	2006-01-20 00:00:00
31	35.00000000	80	Northwind Traders Dried Plums	ChristinaLee	2006-01-20 00:00:00
34	184.00000000	19	Northwind Traders Chocolate Bis...	ChristinaLee	2006-02-06 00:00:00

Görüldüğü gibi artık 45 numarı order için 40 numaralı product'a 920 41 numaralı product'a ise 482.5 dolar harcandığını görebiliyorum.

Şimdi ise totalde her bir oder için kaç para harcandığını group by kullanarak hesaplayalım

Bunun için tabloyu orderID'ye göre veya o.id'ye göre gruplayacağız ve sonra totalspend column'u üzerinde sum methodunu kullanacağız.

```
select  o.id as orderID, sum(od.unit_price*od.quantity) as totalSpend, concat(c.first_name,c.last_name) as customerNS ,o.order_da
from orders o
inner join customers c on o.customer_id = c.id
inner join order_details od on od.order_id = o.id
inner join products p on od.product_id = p.id
group by orderID
order by customerNS
```

Ayrıca product_id ve product_name gibi bilgiler yeni tabloda pek anlamlı olmayacağı için (çünkü bir grubun içinde birden fazla product olmasına rağmen sadece ilki gösterilecek) bu kısımları göstermeyeceğiz:

orderID	totalSpend	customerNS	order_date
38	13800.00000000	AmritanshRaghav	2006-03-10 00:00:0(
45	1402.50000000	AmritanshRaghav	2006-04-07 00:00:0(
72	230.00000000	AmritanshRaghav	2006-06-07 00:00:0(
44	1674.75000000	AnnaBedecs	2006-03-24 00:00:0(
71	736.00000000	AnnaBedecs	2006-05-24 00:00:0(
31	865.00000000	ChristinaLee	2006-01-20 00:00:0(
34	184.00000000	ChristinaLee	2006-02-06 00:00:0(
58	3520.00000000	ChristinaLee	2006-04-22 00:00:0(
80	380.00000000	ChristinaLee	2006-04-25 17:03:5!
33	276.00000000	ElizabethAndersen	2006-01-30 00:00:0(
39	1275.00000000	ElizabethAndersen	2006-03-22 00:00:0(

Sonuçta yukarıdaki gibi orderID'ye karşılık totalSpend elde edilmiş oluyor.

4. Left Join

Bunun mantığını zaten biliyorduk bi tekrar edelim:

Öncelikle inner join ile customer ve order tablosunu inner join edip customer bilgileri ile order bilgilerini aynı tabloda görebiliyoruz

```
1 •  select c.first_name, o.id
2    from customers c
3    inner join orders o
4    on o.customer_id = c.id
5    order by c.first_name
```

➔

first_name	id
Amritansh	38
Amritansh	45
Amritansh	65
Amritansh	72
Anna	44
Anna	71
Christina	31
Christina	34
Christina	58

Fakat unutma ki burada gelen join tablosu hem customer hem de order tablosunda denk düşen kayıtları bize getirir.

Yani eğer customer tablosunda henüz sipariş vermemiş bir customer bilgisi yer alıyorsa bu customer yukarıdaki tabloda görünmeyecektir.

İşte hem sipariş veren hem de vermeyen customer'ları aynı tablo da görmek istersek o halde LEFT veya RIGHT join kullanabiliriz.

Şuan customer tablosu sol tarafta olduğu için LEFT join kullanacağız:

```
1 •  select c.first_name, o.id
2    from customers c
3    left join orders o
4    on o.customer_id = c.id
5    order by c.first_name
```

Göründüğü gibi bu kez bazı customer kayıtlarınının order_id kısmı null olarak görünür çünkü burada sipariş vermeyen müşteri bilgileri de yer alır.

first_name	id
Alexander	NULL
Amritansh	38
Amritansh	45
Amritansh	65
Amritansh	72
Andre	NULL
Anna	44
Anna	71
Antonio	NULL

5. Righ Join

Right join'in mantığı left join ile birebir aynı sadece sağ ve sol farkı var yani geçenki örnekte customer tablosu soldaki tablo olduğu için left join yaptığımızda hem sipariş veren customer'lar hem de vermeyenler getirilmişti.

Eğer ki geçenki örnekte right join yapsaydık, sipariş veren customer'ların yanında bir customer'ı olmayan yani sahipsiz siparişleri görüntüleyecektik, elbette böyle bir sipariş kaydı varsa.

6. Ekstra

Zaten bildiğimiz şeyler, ekstra belirtmek gereken bir iki yer var, o da where ile join kullanılırsa hangi sıraya işler yürür:

```
SELECT title, COUNT(store_id) AS copies_at_store1
FROM film
INNER JOIN inventory  ON inventory.film_id = film.film_id
WHERE store_id=1
GROUP BY title
ORDER BY title;
```

★First, the joining happens, then where statement make an elimination, later the resultant set is grouped by title and count function is executed, finally the set is ordered.

Ayrıca HAVING statement'ı ile sadece groupby ile gruplanan kolunu test etmek zorunda değiliz yani mesela group by ile bir şirketin ortalama ne kadar satış yaptığını buluyoruz, her şirket_id'si içerisinde birsürü satış bilgisi var avg ile bunların avg'ını hesaplıyoruz, having ile bu avg bilgisini test edebiliriz ancak bunun yanında her bir grubun içinde kalan değişkenleri de test edebiliriz.

```
select s.company, avg(o.shipping_fee) as ortalama from orders o i
on s.id = o.shipper_id
where o.shipping_fee > 0
group by s.id
having count(o.id) > 5
```

Mesela yukarıdacount(o.id) > 5 olan kayıtlar gösterilecektir, yani her grubun içerisinde

	company	ortalama
▶	Shipping Company A	36.66666667
	Shipping Company B	87.77777778
	Shipping Company C	28.16666667

Her bir company içerisinde birçok sipariş bilgisi bulunuyor işte having ile her grup içerisindeki order sayısına bakıyorum ve test ediyorum.

7. Bir Tablodan Seçilen Kayıtların Başka bir Tabloya Eklenmesi

Şimdi aşağıdaki query ile employees tablosunu ve orders tablosunu kullanarak, her çalışanın sattığı order bilgisini elde edebilirim. Left join yapıyorum ve böylece çalışana karşılık order'ı elde etmekle kalmıyoruz, henüz herhangi bir order verdirememiş çalışanları da tablomuzda elde ediyoruz.

```
1 •  select e.id,  CONCAT(e.first_name,' ', e.last_name) as fullname, e.email_address, o.id
2    from employees e
3    left join orders o on e.id = o.employee_id
4
```

Elbette aşağıdaki görüldüğü gibi bir çalışan birden fazla order verdirmiş olabilir her order'ı için ayrı bir satır söz konusu

	id	fullname	email_address	id
▶	1	Nancy Freehafer	nancy@northwindtraders.com	41
	1	Nancy Freehafer	nancy@northwindtraders.com	42
	1	Nancy Freehafer	nancy@northwindtraders.com	43
	1	Nancy Freehafer	nancy@northwindtraders.com	44
	1	Nancy Freehafer	nancy@northwindtraders.com	45

Şimdi istiyorum ki her bir çalışanın kaç sipariş verdiğini saydıralım:

```
1 •  select e.id,  CONCAT(e.first_name,' ', e.last_name) as fullname, e.email_address, count(o.id)
2    from employees e
3    left join orders o on e.id = o.employee_id
4    group by e.id
5
```

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

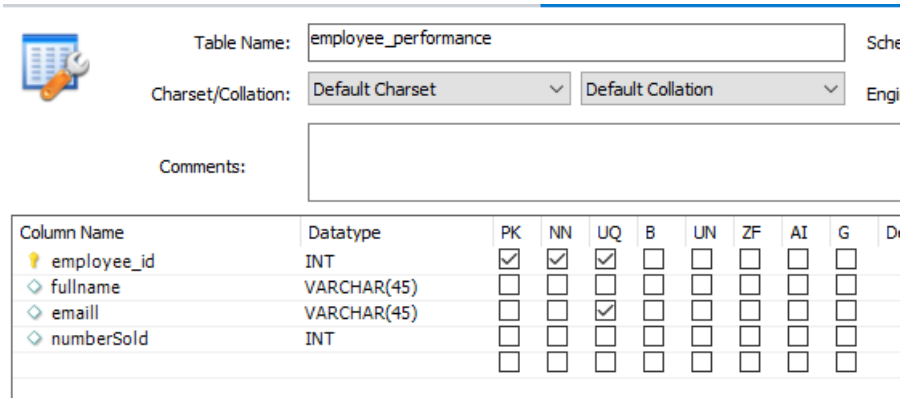
	id	fullname	email_address	count(o.id)
▶	1	Nancy Freehafer	nancy@northwindtraders.com	12
	2	Andrew Cencini	andrew@northwindtraders.com	4
	3	Jan Kotas	jan@northwindtraders.com	6
	4	Mariya Sergienko	mariya@northwindtraders.com	8
	5	Steven Thorpe	steven@northwindtraders.com	0
	6	Michael Neipper	michael@northwindtraders.com	4
	7	Robert Zare	robert@northwindtraders.com	2
	8	Laura Giussani	laura@northwindtraders.com	2
	9	Anne Hellung-Larsen	anne@northwindtraders.com	10

Bunun için her employee id'ye göre bir gruplama yaptık ve her bir grup içindeki order_id sayısını count ettirdik.

Buraya kadarki şeyler zaten bildiğimiz şeylerim bir pratiğiydi.

Şimdi amacımız, bu elde edilen yeni tablonun içeriklerini yeni bir tabloya gömmek!

Bunun için öncelikle soldaki tables kısmına sağ tıklayıp create new table diyerek yeni bir employee_performance tablosu yaratıyorum:



Bilgileri bu tablonun içerisine aktaracağım, bunun için biraz önce hazırladığımız select komutunun hemen üstüne insert into sorgusunu ekliyorum.

```
insert into employee_performance(employee_id, fullname, email, numberSold )
select e.id,  CONCAT(e.first_name,' ', e.last_name) as fullname, e.email_address, count(o.id)
from employees e
left join orders o on e.id = o.employee_id
group by e.id
```

Eklenen sorgudaki parametrelerin aktarım yapılacak tablonun column'ları ile aynı olduğuna dikkat et.

Sonuçta select ile elde edilen tablonun kayıtları yeni yaratılan tabloya aktarılacaktır:

employee_id	fullname	email	numberSold
1	Nancy Freehafer	nancy@northwindtraders.com	12
2	Andrew Cencini	andrew@northwindtraders.com	4
3	Jan Kotas	jan@northwindtraders.com	6
4	Mariya Sergienko	mariya@northwindtraders.com	8
5	Steven Thorpe	steven@northwindtraders.com	0
6	Michael Neipper	michael@northwindtraders.com	4
7	Robert Zare	robert@northwindtraders.com	2
8	Laura Giussani	laura@northwindtraders.com	2
9	Anne Hellung-Larsen	anne@northwindtraders.com	10
NULL	NULL	NULL	NULL

Aslında burada yazılan insert sorgusu, 2. Word dosyasındakine çok benzer, orada eklenecek value'yu elle belirtmiştik, burada ise eklenecek value bir select sorgusu sonucunda elde edilmiş o kadar!

8. Join ile Update Kullanımı

Geçen örnekte, employees tablosu ile orders tablosunu join ettim ve ve sonuçta elde edilen verileri yeni bir tablo olan employee_performance tablosuna ekledim.

Şimdi ise yapmak istediğim şey bu yeni tablodaki kayıtlarda yapılan değişikliklerin aynısını gerisingeri orijinal join tablosunda da görmek. Yani join tablosunu update etmek istiyorum, ki join tablosunu update etmek demek aslında, join olan tabloları update etmek demek!

Değişiklikleri direkt join tablosunda yapmadım, önce ayrı bir tabloda test ettim emin olduktan sonra join tablosunu update edeceğim.

Diyelim ki employee_performance tablsunda aşağıdkai gibi bazı email adresleri güncellendi:

employee_id	fullname	email	numberSold
1	Nancy Freehafer	nancy@northwindtraders.com UPDATED	12
2	Andrew Cencini	andrew@northwindtraders.com	4
3	Jan Kotas	jan@northwindtraders.com	6
4	Mariya Sergienko	mariya@northwindtraders.com UPDATED	8
5	Steven Thorpe	steven@northwindtraders.com	0
6	Michael Neipper	michael@northwindtraders.com	4
7	Robert Zare	robert@northwindtraders.com UPDATED	2
8	Laura Giussani	laura@northwindtraders.com	2
9	Anne Hellung-Larsen	anne@northwindtraders.com UPDATED	10
NULL	NULL		NULL

Join ile Update

Aşağıdaki query ile, employees tablosunu update etmek istediğimizi söylüyoruz, bunun için employees tablosu ile employee_performance tablosu join ediliyor.

```
update employees e inner join employee_performance ep
on e.id = ep.employee_id
set e.email_address = ep.email
```

Daha sonra join edilmiş tablolardaan ep'nin email değerleri e'nin email değeri olarak atansın diyoruz.

Böyle dersek e'nin tüm email kayıtları ep'ninkiler ile update edilecek.

Şimdi burada ep zaten e'nin kopyalanmış üzerinde bazı değişiklikler yapılmış versiyonunu temsil ediyor, bu yüzden biz e'nin tüm kayıtları güncellensin istemeyiz, verimsiz olur.

O yüzden ep'deki updated kayıtları buluruz ve e tablosunda da sadece bu email adresleri güncellensin deriz.

Bu sebeple aşağıdaki gibi where statement kullanırız:

```
2 •  update
3    employees e inner join employee_performance ep on e.id = ep.employee_id
4    set e.email_address = ep.email
5    where ep.email like '%updated%'
```

9 . Insert – Update with Joins Özet

Sonuçta geçen section'da yapılan bildiğimiz bir tabloya insert işleminde elle değer atamak yerine bir join tablodan select ile kayıtları filtreleyip sonra insert etmekti. Zaten 2. Word dosyasında insert'i görmüştük!

Eskiden insert şöyle yapılıyordu:

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

Şimdi ise values yerine bir select statement'ı ile join edilmiş tablodan elde edilen değerler otomatik verildi:

```
insert into employee_performance(employee_id, fullname, email, numberSold )
select e.id,  CONCAT(e.first_name,' ', e.last_name) as fullname, e.email_address, count(o.id)
from employees e
left join orders o on e.id = o.employee_id
group by e.id
```

Benzer şekilde eskiden update şöyle yapılıyordu:

```
update shopapp.products
set name='Samsung S5 New'
where id=1
```

Böylelikle ilgili tablodaki tüm bir column değerlerini veya seçilen satırlar için column değerlerini verebiliyorduk.

Şimdi ise aşağıdaki şekilde

```
2 •  update
3    employees e inner join employee_performance ep on e.id = ep.employee_id
4    set e.email_address = ep.email
5    where ep.email like '%updated%'
```

Yine update dilecek tabloyu belirtiyoruz ancak burada update edilecek tablo bir başka tablo ile join halinde, böylece join edilen tabloların içindeki column değerlerini parametrik olarak kullanabiliyoruz, eskiye benzer şekilde where sorgusu ile join tablosundaki hangi satırlar için update işleminin yapılacağını belirtebiliyoruz.

10 . Join ile Delete Kullanımı

Önce eskiden delete'i nasıl kullanıyorduk onu hatırlayalım:

```
DELETE FROM shopapp.products
```

Dersek tüm kayıtlar siliniyordu, bunun yanında istediğimiz kayıtları silmek istersek:

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

Şeklinde bir where sorgusundan yararlanıyorduk.

Şimdi delete'i join ile naısl kullanacağımıza bakalım.

Amacımız sipariş vermemiş müşterileri bulup bunları customer tablosundan silmek olsun, öncelikle bu sipariş vermeyen müşterileri bulmak için bi left join işlemi yapıyorum:

```
select c.id, c.first_name from customers c
left join orders o on o.customer_id = c.id
where o.customer_id is null
```

Yukarıdaki query ile customers ile orders'ı left join yaptım böyece orders tablosunda yer almayan customer'ları da elde edeceğim daha sonra da where statement'ı ile bu sipariş vermeyen müşterileri ayıklamış oldum.

Şimdi ifadayei aşağıdaki hale getirirsem, yani select yerine delete c kısmını getirirsem elde edilen kayıtlar customers tablosundan silinir.

```
delete c
from customers c left join orders o on o.customer_id = c.id
where o.customer_id is null
```

Böylece join edilmiş tablolardan biriden istenilen satırlar silindi, hangilerinin silineceğini de join tablosu sayesinde karar verdik.

11 . Union Operatörü

Join operatörü ile seçilen tabloları id'lerin eşitliğine göre yanyana getiriyorduk.

Union operatörü ise tabloları yanyana değil, altalta eklememizi sağlar.

Diyelim ki amacımız aşağıdaki employee_performance tablosundan

employee_id	fullname	email	numberSold
1	Nancy Freehafer	nancy@northwindtraders.com UPDATED	12
2	Andrew Cencini	andrew@northwindtraders.com	4
3	Jan Kotas	jan@northwindtraders.com	6
4	Mariya Sergienko	mariya@northwindtraders.com UPDATED	8
5	Steven Thorpe	steven@northwindtraders.com	0
6	Michael Neipper	michael@northwindtraders.com	4
7	Robert Zare	robert@northwindtraders.com UPDATED	2
8	Laura Giussani	laura@northwindtraders.com	2
9	Anne Hellung-Larsen	anne@northwindtraders.com UPDATED	10
NULL	NULL	NULL	NULL

Ve aşağıdaki employees tablosundan

id	company	last_name	first_name	email_address	job_title	business_phone	home_phone	mobile_phone
1	Northwind Traders	Freehafer	Nancy	nancy@northwindtraders.com	Sales Representative	(123)555-0100	(123)555-0102	NULL
2	Northwind Traders	Cencini	Andrew	andrew@northwindtraders.com	Vice President, Sales	(123)555-0100	(123)555-0102	NULL
3	Northwind Traders	Kotas	Jan	jan@northwindtraders.com	Sales Representative	(123)555-0100	(123)555-0102	NULL
4	Northwind Traders	Sergienko	Mariya	mariya@northwindtraders.com	Sales Representative	(123)555-0100	(123)555-0102	NULL
5	Northwind Traders	Thorpe	Steven	steven@northwindtraders.com	Sales Manager	(123)555-0100	(123)555-0102	NULL
6	Northwind Traders	Neipper	Michael	michael@northwindtraders.com	Sales Representative	(123)555-0100	(123)555-0102	NULL
7	Northwind Traders	Zare	Robert	robert@northwindtraders.com	Sales Representative	(123)555-0100	(123)555-0102	NULL
8	Northwind Traders	Giussani	Laura	laura@northwindtraders.com	Sales Coordinator	(123)555-0100	(123)555-0102	NULL
9	Northwind Traders	Hellung-Larsen	Anne	anne@northwindtraders.com	Sales Representative	(123)555-0100	(123)555-0102	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Mailleri çekip altalta eklemek yani tüm mailleri elde etmek.

```
select email from employee_performance
union
select email_address from employees
```

Sonuçta aşağıdaki mail listesi elde edilir:

email
andrew@northwindtraders.com
anne@northwindtraders.com UPDATED
jan@northwindtraders.com
laura@northwindtraders.com
mariya@northwindtraders.com UPDATED
michael@northwindtraders.com
nancy@northwindtraders.com UPDATED
robert@northwindtraders.com UPDATED
steven@northwindtraders.com
nancy@northwindtraders.com
mariya@northwindtraders.com
robert@northwindtraders.com
anne@northwindtraders.com

İyi ama ep tablosunda 9 mail adresi var, e tablosunda da 9 mail adresi var union sonucunda neden 18 değil de 9 mail adresi var?

Çünkü iki tablodaki mail adresleri birebir aynı! Union operatörü aynı olan kayıtların yalnızca bir tanesini alır!

Eğer UNION ALL kullanırsak tekrarlayan alanları da gösterir:

```
select email from employee_performance
union all
select email_address from employees
```

email
andrew@northwindtraders.com
anne@northwindtraders.com UPDATED
jan@northwindtraders.com
laura@northwindtraders.com
mariya@northwindtraders.com UPDATED
michael@northwindtraders.com
nancy@northwindtraders.com UPDATED
robert@northwindtraders.com UPDATED
steven@northwindtraders.com
nancy@northwindtraders.com
andrew@northwindtraders.com
jan@northwindtraders.com
mariya@northwindtraders.com
steven@northwindtraders.com
michael@northwindtraders.com
robert@northwindtraders.com
laura@northwindtraders.com
anne@northwindtraders.com

11 . SubQueries

Şimdi örneklerle bu subquery mevzusunu anlayalım.

Orders tablosunda aşağıdaki gibi bir status_id kısmı var:

ayment_type	paid_date	notes	tax_rate	tax_status_id	status_id
eck	2006-01-15 00:00:00	NULL	0	NULL	3
edit Card	2006-01-20 00:00:00	NULL	0	NULL	3
edit Card	2006-01-22 00:00:00	NULL	0	NULL	3
edit Card	2006-01-30 00:00:00	NULL	0	NULL	3

Bu status_id'nin neye karşılık geldiği orders_status tablosunda yer alıyor:

id	status_name
0	New
1	Invoiced
2	Shipped
3	Closed
NULL	NULL

Peki biz istersek basit bir sorgu ile orders tablosunda status_id'si 3 olan order'ları çekebiliriz:

```
1 •  select id, order_date from orders
2    where status_id =  3
```

Aslında yaptığımız sorgu closed order'ları çekmiş oldu ancak bunu yapmak için id'yi elle girdik.

Örnek 1

Onun yerine subquery kullanarak orders_status tablosundan başka bir sorgu ile önce closed'a karşılık gelen id'yi çekelim ve bu id'yi yukarıdaki select sorgusuna gömelim:

```
1 •  select id, order_date from orders
2    where status_id =  (select id from orders_status where status_name = 'closed')
```

Yukarıdaki şekilde önce select sorgusu ile orders_status tablosunda 3 numaralı id elde edilir daha sonra bu değer diğer select sorgusu için kullanılır!

Örnek 2

Benzer şekilde bir başka örnek yapalım, diyelim ki products tablosundan belirli id'leri çekmek istiyorum, bunun için aşağıdaki gibi bir where ve in komutundna yararlanabiliriz.

```
select id, product_name from products
where id in (1,2,3)
```

Ancak diyelim ki bu 1,2,3 id'lerini elle vermek istemiyorum da bu liste başka bir select komutundan gelsin.

Örneğin 100'den fazla satılan ürünleri product tablosundan almak isityoruz diyelim, bu product_id'lerini elle giremem çünkü bilmiyorum o halde önce başka bir select komutu ile bu id'leri orders_detail tablosundan elde ederim:

```
select id, product_name from products
where id in (select product_id from order_details where quantity>100)
```

Burada subquery'nin döndüreceği aşağıdaki gibi bazı id bilgileridir ve bu bilgiler bir liste gibi kullanılabilir.

product_id
41
43
81
43
34

Örnek 3

Üçüncü örnek diğer ikisinden biraz daha farklı olacak, sindirmesi biraz daha zor.

Diyelim ki her bir customer'ın kaç sipariş verdiğini öğrenmek istiyorum. Bunun için daha önce yaptığımız gibi customers tablosu ile orders tablosunu join edebiliriz ve daha sonra group by ile her bir customer'ın kaç sipariş verdiğini hesaplayabiliriz.

Ancak burada ise subquery ile aynı amacı gerçekleştireceğiz.

Sorgu aşağıdaki gibi olacak:

```
select c.id,c.first_name, c.last_name, (select count(o.id) from orders o where o.customer_id = c.id)
from customers c
```

Anladığım kadarıyla burada olan şey aynı 2 loop'un içiçe olması gibi:

- Önce customers tablosundan bir satır seçiliyor, daha sonra o satır seçiliyken inner select komutu çalışıyor.
- Inner select komutu ise orders tablosunu satır satır geziyor ve order tablosundaki customer_id'si outer select'in o anki seçtiği customer'ın id'sine eşit olanları sayıyor ve kaydediyor.
- Daha sonra outer select komutu sıradaki customer'ı alıyor ve sıradaki customer'ın id'si de tüm order'larda aranıyor ve kaç tanesinde bulunduğu kaydoluyor.
- Bu şekilde ilerliyor ve sonucunda hangi customer'ın kaç order verdiğini bulmuş oluyoruz!

11 . Exists Operatörü

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns true if the subquery returns one or more records.

Products

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Suppliers

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA
4	Tokyo Traders	Yoshi Nagase	9-8 Sekimai Musashino-shi	Tokyo	100	Japan

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS
(SELECT ProductName FROM Products WHERE Products.SupplierID =
Suppliers.supplierID AND Price < 20);
```

Yukarıdaki sorgu ile yapılan şu oldu:

Eğer bir supplierID products tablosunda varsa ve bulunduğu satırdaki product'ın price'ı 20'den küçükse ilgili supplierID'ye karşılık gelen SupplierName'i select edelim.

Yani fiyatı 20'den düşük olan product'ları sağlayan supplier'ları listelemiş olduk.

Daha detaylı baktığımızda exists operatörünün işlevi şu oluyor:

- Önce supliers tablosundan ilk supplier seçiliyor.
- Sonra product tablsouna gidiliyor product tablosunda supplierid'si 1 olan ve fiyatı <20 olan productName'ler getiriliyor.
- Eğer herhang bir productName getirilebildiyse exists operatörü true döndürüyor.
- Böylece ilk supllier seçilmiş oluyor çünkü bu supplierID'nin products tablosunda 20'den aşağıda en az bir ürün sattığı bulundu.
- Şimdi sırada ikinci supplier'ı kontrol etmek var ...

12. Any & All Operatörleri

The ANY and ALL operators are used with a WHERE or HAVING clause.

The ANY operator returns true if any of the subquery values meet the condition.

The ALL operator returns true if all of the subquery values meet the condition.

Any Operatörü

Diyelim ki products tablosundan belirli id'leri seçmek istiyorum:

```
select id, product_name, list_price from products
where id in (2,3,5,10)
```

Any'nin mantığı da buna benzer:

```
select id, product_name, list_price from products
where id = any (select product_id from order_details where quantity>10)
```

Burada products'tan select edilen current id eğer inner query'nin döndürdüğü id'lerden birine eşitse current id select edilmiş olur.

Burada id = yerine >, <, >=, <=, not equal vesaire gibi başka ifadeler de kullanabilirdik.

Yani yukarıdaki örnekte diyelim ki inner query bize (2,3,5,10) listesini döndürüyor olsun, any operatörünün sağladığı şey şu:

id = 2 or id = 3 or id = 5 or id =10 ifadesi sağlanıyorsa true döndür yani current product'ı seç, yok sağlanmıyorsa product'ı seçme

All Operatörü

Yukarıdaki örneği baz alırsa All operatörünün yaptığu şeyse şu

İd =2 and id = 3 and id = 5 and id = 10 ifadesi sağlanıyorsa true döndür yani current product'ı seç!