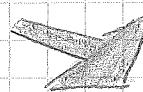


SQL Server Relational Database Management System (RDBMS)

- SQL'in içine kuruldu.
- Sunucusu Transact-SQL (T-SQL)'e bağlıdır.
- 20 yıldan fazladır Windows'ta kullanılır.
- 2016'da Microsoft, SQL'ı Linux'ta kullanılabileceğine getirdi.

SQL Server



SQLoS

- (*) It provides many operations system service.
 (Such as → Memory I/O management)

① Relational Engine

(Query Processor):

- (*) Memory management
- (*) Buffer management
- (*) Thread and task management
- (*) Distributed query processing

② Storage Engine:

- (*) Storage and retrieval of data from the storage systems.
 (Such as disks and SAN)

10 GB' a kadar disk depolama
Kapasitesi olan küçük veritabanları için → SQL Server
Expression

Daha büyük ve kritik uygulamalar için → Enterprise
Seri

! SQL Server 'da tabloları ekranда görmek
için :

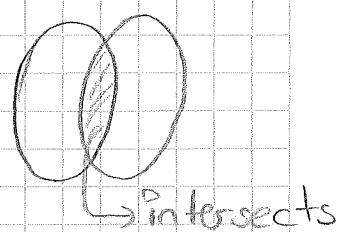
Database Diagrams → New Database Diagram
seçip ordan istediğiniz tabloları seçiyorsunuz

SQL \Rightarrow Yapılan dirilmis, songu dilidir.

Relational data base ile ilgili konumuzda
söylediğimiz circa.

Normalize edilmiş relational database: Her bilgi ayrı
tabloda tutulur ve bunlar birbirine primary key ile
ile ilişkilendirilir. (SUBJUNKT)

INNER JOIN, \Rightarrow RESİSTİM
NULL geçiş mevcut



① List products with category names;

Product tablosu 4 tabloyla doğrudan ilişkilidir.

Product tablosunda hangi bilgiler var bak.

Category bilgisini category tablosundan alacaktır.

Product ve Category tablolarını join yapmak gereklidir.

Bunu, INNER JOIN ile yapacagız.

Category - Id \Rightarrow Category tablosu için primary key

Ama Product tablosu için değil

SELECT *

FROM product.product AS A \rightarrow Bilgilerle bağlı.



INNER JOIN yerine sadece JOIN kullanma.
bilirsin

```

SELECT TOP 10 *
FROM product.product AS A
INNER JOIN product.category AS B ON
A.category_id = B.category_id

```

ilk 10 kaydını category_id'lerin döndürdü.

Yukardaki kodda

?

SELECT product_name, B.category_id derslik hatalı verdi

TASK

List employees of stores with their store information

Select employee name, surname, store names

{ SELECT s.first-name, s.last-name, st.store-name

From sale.staff s

JOIN sale.store st

ON s.store_id = st.store_id

AS A yazmak zorunda
AS s yazilmamalidir.

SELECT A.first-name, A.last-name, B.store-name

From sale.staff A

INNER JOIN sale.store B

ON A.store_id = B.store_id

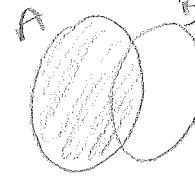
one to one

base

base

table A

table B



A
LEFT JOIN

LEFT JOIN

! LEFT JOIN = LEFT OUTER JOIN

SELECT columns

FROM table-A

LEFT JOIN table-B ON join-condition

→ Sıfırda bir veya fazla
diğer tablo A

LEFT JOIN'da ki: Birçok önce yazılan temel tablo
olmak alırız, sağda? tabloda
uygun olanları getiririz.
Sadece hepsini Sağda? olanları!

TASK 1

Write a query that returns products that have never
been ordered.

Select product ID, product name, order ID

SELECT A.product-id, A.product-name, B.order-id

FROM product.product A

LEFT JOIN sale.order-item B

ON A.product-id = B.product-id

WHERE B.order-id IS NULL

↓

NULL'ları yani hiç sipariş verilmeyenleri getir.

Bunu INNER JOIN ile yapamam.

1. Günlük o kesileri getir.

Birdinde olus ettiğinde diğerlerini getirez.

TASK

Report the stock status of the products that product id greater than 310 in the stores

Expected columns: Product_id, Product_name, Store_id, quantity

SELECT A.product_id, A.product_name, B.store_id, B.quantity

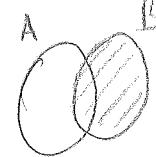
FROM product.product A

LEFT JOIN product.stock B

ON A.product_id = B.product_id

WHERE A.product_id > 310

ORDER BY store_id



Subject :

Date : / /

RIGHT JOIN

SELECT columns
FROM table

RIGHT JOIN table-B
ON join-conditions

Sadece tabloyu esas alır

Sadece tabloyla eşleşenler yazılır.

B TABLE
TAKİP EDİLEN TABLO
YAZILIR

! RIGHT JOIN = RIGHT OUTER JOIN

→ Sadece bak onu esas
al diyor.
Yani table-B

TASK

Report (AGAIN WITH RIGHT JOIN) the stock status
of the products that product id greater than 310
in the stores.

Expected columns: product-id, product-name, store-id,
quantity

SELECT A.product_id, A.product_name, A.store_id, A.quantity
FROM product.stock A
RIGHT JOIN product.product B
ON A.product_id = B.product_id
WHERE A.product_id > 310
ORDER BY store_id

NULL'lar da geldi NULL göstermek için her satırı:

[WHERE A.product_id > 310 AND IS NOT NULL]
olarak yazarım.

TASK

Report the orders information made by all staffs.

Expected columns: staff_id, first_name, last_name,
all the information about orders.

SELECT *

FROM SALE.staff → (Basered table no 2)

Once tabloya baktik.

SELECT COUNT(staff_id)

FROM SALE.staff

↙
Kac staff var
onu saydik.

Bunlari bilgisi ne

order by ile left

join yaparak
ulaşiyorum.

SELECT COUNT(DISTINCT A.staff_id)

↗ Farklı olanları
bul dedik.

FROM sale.staff A

INNER JOIN sale.orders B

ON A.staff_id = B.staff_id

ilkinde 10, ikinci de 6 bonus vardı. Deneb ki 4

kis hiz sifaris olmamis

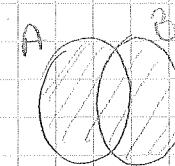
```

SELECT A.staff_id, A.first_name, A.last_name, B.order_id
FROM sale.staff A
LEFT JOIN sale.orders B
ON A.staff_id = B.staff_id

```

id'lerle çalışmak güvenli değil.

FULL OUTER JOIN



$A \cup B$ (NULL'lar gelir.)

Her iki tabloda ne var ne yok hepsini getir.
Sadece sola ne yazdığını önceliğe alır.

TASK

Write a query that returns stock and order information together for all products. (Top 20)

Expected columns: product_id, stock_id, quantity, order_id, list_price

(order item ve stock tablolardan sonra gelenlerdir.)

```

SELECT A.product_id, B.store_id, B.quantity, A.list_price
FROM sale.order_item A

```

FULL OUTER JOIN product.stock B

ON A.product_id = B.product_id

ORDER By B.product_id, A.order_id

Sıralama çok önemlidir

A tablosundan olan B'de yoksa sadece NULL
yazar.

16.10.2021

Subject :

Date :

! Sonda 'each' kullanırsan GROUP BY kullanmak
sondaymaz.

SELECT count(DISTINCT A.staff_id), count(DISTINCT B.staff_id)
FROM sale.staff A

LEFT JOIN sale.orders B
ON A.staff_id = B.staff_id

By sekilde [10 6] ciktisini veriyor

RIGHT JOIN yarindan [6 6] ciktisini veriyor

CHINOOK DATABASE'den PRE-CLASS SORUSU

How many artists don't have album title info?

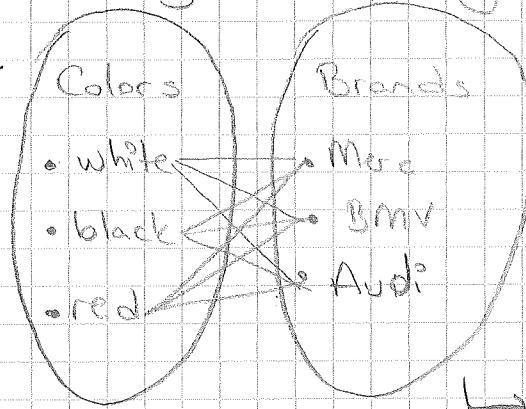
SELECT COUNT(*)
FROM artists ar
LEFT JOIN albums al
ON ar.ArtistId = al.ArtistId
WHERE al.Title IS NULL

(Cartesian Join)

Date:

CROSS JOIN

Kartesyan çarpım yapısı



SELECT colors.color, brands.brand
FROM colors
CROSS JOIN brands;

← İainde WHERE clause kullanırsan
INNER JOIN gibi salısmış.

SELF JOIN

Bir tablonun kendisyle join yapılması.

SELECT A.first_name, A.last_name, A.job_title,
B.first_name AS manager_name

FROM employees A

← Kendinden kendine

INNER JOIN employees B

ON B.emp_id = A.reports_to;

Employees' i A olasık tanımladık.

Sonra B olasık tanımladık. (Sanki 2 farklı

tabloyanus gibi kendisyle
join'ledik.)



SELF JOIN'de



INNER JOIN veya

LEFT JOIN kullanılır.

HAVING

Group By ile birlikte kullanılır (Group By'dan sonra
AGG kullanmak zorundayız)

SELECT column-1, agg-func(column-2)
FROM table-name

GROUP BY column-1

HAVING search-condition;

FROM

WHERE

} SAL'in çalışma sırası

GROUP BY

HAVING

SELECT

ORDER BY

TASK

Soru: Ort. Ün foyatı 1000'den yüksekk olan markaları getir.

```

SELECT brand-name AVG(list-price) AS avg-list-price
FROM product.product A, product.brand B
WHERE A.brand-id = B.brand-id
AND A.model-year > 2016
GROUP BY brand-name
HAVING AVG(list-price) > 1000
ORDER BY AVG(list-price) ASC;
    
```

Virgülle
inner join
yapın
fakta

TASK

Write a query that checks if any product id is repeated in more than one row in the product table.

Expected output: product-id, num-of-rows

①

```

SELECT * } Product tablosuna bak
FROM product
    
```

②

```

SELECT product-id, COUNT(*) CNT_ROW } Product id
    
```

Bize bunun
CNT_PRODUCTIN
sayısı.

```

    FROM product
    GROUP BY product-id
        
```

```

HAVING COUNT(*) > 1
    
```

ye göre
satırları
saydı.

TASK

Write a query that returns category ids with a max list price above 4000 or a min list price below 500
 (No need category name)

```
SELECT *
```

```
FROM product.product
```

{ Ünlüefin id bilgisi var.

max de 2000
min de 500

```
SELECT category_id, MAX(list_price) AS max_price,
```

MIN(list_price) MIN_PRICE

```
FROM product.product
```

```
GROUP BY category_id
```

```
SELECT category_id, list_price { 1 ve 2 nolu category_id'ları
```

```
FROM product.product
```

} getirdi?

```
ORDER BY 1,2
```

```
SELECT category_id, MAX(list_price) AS max_price,
```

MIN(list_price) MIN_PRICE

```
FROM product.product
```

```
GROUP BY category_id
```

HAVING MAX(list_price) > 400 AND MIN(list_price) < 500

Subject :

Date : / /

TASK

Markaların ort. Fiyat listesini getir. (Azalan sırayla)

Exp. output : brand-name , avg-list-price

```
SELECT b.brand-name, AVG(p.list-price) AS ort  
FROM product.product p  
JOIN product.brand b ON p.brand-id = b.brand-id  
GROUP BY b.brand-name  
ORDER BY ort DESC ;
```

Subject :

Date :

Farklı gruplamaları
tek vadede yazınız.

GROUPING SETS

Group By altında yapmak istedığınız farklı gruplamaları
yazınınız.

SELECT column1, column2, agg-func (column3)
FROM table-name

Group By

GROUPING SETS (

(column1, column2),

(column1),

(column2),

()

) ;

Li farklı
gruplama
yapın.

TASK

SELECT *

FROM sale.sales_summary

İçinde neler var

baklık

(

Hoca bir tablo attı bundan sonra "set table" olarak bunu kullanacağınız. AMA BEN YAZMIYOLUM



Toplam sales miktarını hesaplayınız:

SELECT sum(total_sales_prices)

FROM sale.sales_summary



Markaların Toplam sales miktarını hesaplayınız:

SELECT brand, sum(total_sales_prices)

FROM sale.sales_summary

GROUP BY brand



Kategori bazında yapan toplam sales miktarı:

SELECT Category, sum(total_sales_prices) total_sales

FROM sale.sales_summary

GROUP BY Category

Subject :

Date : / /

* Marka ve kategori birlikteki toplam sales miktarını hesaplayınız :

```
SELECT brand, Category, SUM((total_sales - price) / total_sales)
FROM sale.sales_summary
GROUP BY brand, Category
ORDER BY brand
```

Onceki 4 kdu tek sorguda yapıyanız :

```
SELECT brand, category, sum((total_sales - price))
FROM sale.sales_summary
GROUP BY
: GROUPING SETS (
    (brand, category),
    (brand),
    (category),
    (),
    () );
```

Grouping Sets' te optimize etmek daha kolay.

Subject :

Date :

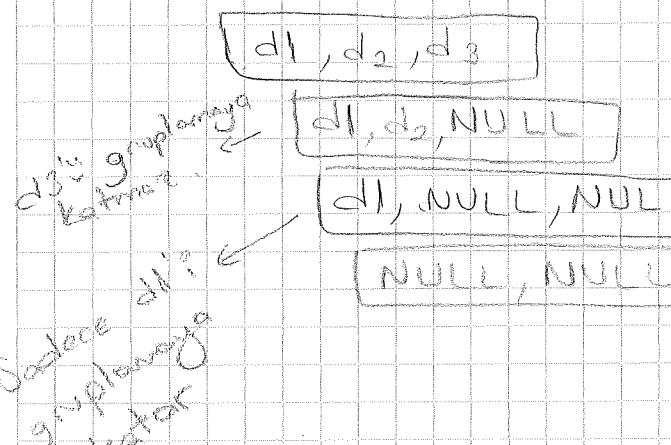
Rollup

SELECT d1, d2, d3, agg-func(c4)

FROM Table-name

GROUP BY

ROLLUP (d1, d2, d3)



ROLLUP aşıgı doğrultu
sondan sırayla
silerek grüplene
yapılır.

brand, category, model-year sütunları ile Rollup kullanarak

total sales hesaplanması yapın.

(3 sütun için 4 farklı grüplama)

SELECT brand, category, model-year, SUM(total_sales * price) total_price
FROM sale.sales_summary

GROUP BY

ROLLUP (brand, category, model-year)

→ Rollup için
değer fazla
combination olur.

Cube

```
SELECT d1,
       d2,
       d3,
       agg_func(cu)
FROM Table-name
GROUP BY
       CUBE (d1,d2,d3)
```

[d1,d2,d3]

[d1,d2,NULL]

[d1,d3,NULL]

[d2,d3,NULL]

[d1,NULL,NULL]

[d2,NULL,NULL]

[d3,NULL,NULL]

[NULL,NULL,NULL]

2nd Zone
initial value.

Cube Rollup'ın
değer ayrıntılı
gibi 2nd Zone initial
value.

2nd Zone

Subject :

Date : / /

TASK

Onceki senyu CUBE ile yapalim:

(3 wütn 8 in 8 kombinasyon)

```
SELECT brand, category, model_year, sum(total_sale_price) total_price
FROM sale.sales_summary
GROUP BY
    CUBE (brand, category, model_year)
ORDER BY brand, category;
```

→ Tekli yorum
satırı

/*
* ↗ Göklu yorum satırı
*/

Subject:

Date:

PIVOT

Unique satırları alt, sütun olarak yazar.

Yani da AGG sonucu elde edilen verileri yazar.

GROUP BY ile kullanılmış.

(Cunki satır kediği grupta yapıyor.)

SELECT [column-name], [pivot-value1], [pivot-value2] ...

FROM table-name

PIVOT (

agg-func(agg-column)

FOR pivot-column

IN ([pivot-value1], [pivot-value2] ... pivot-tables))

AS pivot-table-name;

Pivot altındaki katagorilerin
AGG istenimi yap.

Pivot-table adıktan ibar.

Pre-Class Log

SELECT * FROM

(SELECT CountryName, filmId
FROM tblCountry AS c
INNER JOIN tblFilm AS f
ON c.CountryId=f.FilmCountryId) AS BaseData

PIVOT (

COUNT(filmId) FOR CountryName
IN ([China], [France], [Germany]) AS PivotTable

TASK

Kategori ve model bilgisi göre toplam sales miktarını, summary tablosu üzerinden hesaplayınız.

```

SELECT category, model-year, sum(total-sales-price)
FROM sale.sales-summary
Group By category, model-year
ORDER BY 1, 2
  
```

Aşağıda bu
pivot ile yapılabilir

Bunu pivot table'a çevireceğiz

```

SELECT *
FROM (
    Kaynak { SELECT category, model-year, total-sales-price }
    Tablosu { FROM sale.sales-summary }
) as A
PIVOT (
    sum(total-sales-price)
    FOR category
    IN (
        [Category-name] [ ]
        [ ] [ ]
        [ ] [ ]
        ...
    )
) A as PI
  
```

SUBAQUERIES

- * Veriler daha hızlı çalışın, daha iyi okunsun.
 - * JOIN ile tabloların hepsine gitmeye adılır, milyonlık satırarda bu zordur.
 - * Subquery ana tablonun bir kismini ayırip buna yönelik alt sorular oluşturucu.

```
SELECT column_name  
FROM table_1,table_2
```

WHERE column-name OPERATOR

```
SELECT column_name  
FROM table_1, table_2;
```

Outer quirky

Outer Shells

(Enclosing)

INNER OVERLAY (Nested)

(Nested query = subquery)

(*) SELECT } Subquery bantla ile kullanılır.
FROM }
WHERE }

SELECT file:

SELECT order_id, list_price

三

Select AVG(lost-pfd)

from sale or use.

) AS avg-price

FROM sale, order, item;

WHERE file:

```
SELECT order_id, order_date  
FROM sale.orders
```

```
WHERE order_date IN (
```

```
    SELECT TOP 5 order_date  
    FROM sale.orders  
    ORDER BY order_date DESC
```

From file:

```
SELECT order_id, order_date  
FROM (SELECT TOP 5  
      FROM sale.orders  
      ORDER BY order_date DESC  
) A;
```

Subqueries

Single-row

Multiple-row

Correlated

① Single-row Subqueries

$=, >, <, >=, \leq, \geq, !=$ ile kullanılır.

(* Jointerle yapılanlar subquery ile yapılabilir. Ama subqueryler daha hızlı.)

TASK

Write a query that returns the total list price by each order id's.

(Order id'lerde göre toplam list-price ları hesapla.)

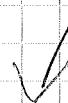
SELECT *
FROM sale.order-item
} Bilgilere bak

SELECT order_id,
(SELECT sum(list-price)
FROM sale.order-item B)

FROM sale.order-item

→ WHERE A.order_id = B.order_id

Buraya bunu da ekleyebilirim



Correlated subquery yapmışız

Subject :

Date : / /

TASK

Bring all the staff from the store that Maria Cussons works at. (Maria Cussons take her personnel)

SELECT store_id

FROM sale.staff

WHERE first_name = "Maria" AND last_name = "Cussons"

Single query

SELECT

*

From sale.staff

WHERE store_id = (Yukardaki

Kadu buraya
ya)

TASK

List the staff that Jane Destrey is the manager of.

SELECT *

FROM sale.staff

WHERE manager_id =

(SELECT staff_id

FROM sale.staff

WHERE first_name = "Jane" AND
last_name = "Destrey")



Esiitlik yendne IN külalitsak da agni sõnavega getida.

Teek row'lu sõnularda ükski de külalitsus.

Subject
2

Multiple-row Subqueries

Date: / /

İçinde IN, NOT IN, ANY, ALL olur.

► Birdeksiz sorma standartı.

TASK

List order dates for customers residing in the Holbrook city: (Holbrook şehrindeki müşterilerin sipariş tarihleri)

SELECT customer_id, order_dates

FROM sale.orders

WHERE customer_id IN

(SELECT customer_id

FROM sale.customers

WHERE city = 'Holbrook')

= yazınca birdeksiz
yeni IN. forma
birdeksiz
Customer
bileşik
tarihlerini
listeler

TASK

List all customers who orders on the same dates as Abby Parks.

(Abby Parks ile aynı tarihlerde sipariş veren müşteriler.)

Subject :

Date : / /

```
SELECT *  
FROM sale.orders A  
INNER JOIN (  
    SELECT A.first_name, A.last_name, B.customer_id,  
           B.order_id, B.order_date  
    FROM sale.customer A  
    INNER JOIN sale.orders ON A.customer_id = B.customer_id  
    WHERE A.last_name = "Parks" AND A.first_name = "Abby"  
) B  
ON A.order_date = B.order_date
```

TASK

List bikes that model year equal to 2020 and
its prices more than all electric bikes.

(B3 den elektrikli bisikletlerden pahali olan bisikletlerin
listesini)

```
SELECT product_name, list_price
```

```
FROM product.product
```

```
WHERE list_price > ALL (
```

```
    SELECT list_price
```

B3 den
max
asln
danesi

```
    FROM product.product A
```

```
    INNER JOIN product.category B
```

```
    ON A.category_id = B.category_id
```

```
    WHERE B.category_name = 'Electric Bikes'
```

```
)
```

```
AND model_year = 2020
```

! ALL yine ANY yeari mi ne bakarsiniz

Exists

Kullanımı IN gibiymis

Subquery'in herhangi bir değer döndürse döndürme
değeri sağlanır. (Dönen değerlerle eşleştirilecektir.)

WHERE ve HAVING ile kullanılır.

SELECT *
FROM sale.customer
WHERE EXISTS (SELECT 1)

Burada bir değer döndürse
customer tablosundaki tüm
değerlerin getirilir.

SELECT *
FROM sale.customers A
WHERE EXISTS (

SELECT 1
FROM sale.orders B
WHERE B.order_date > 2020-01-01
AND A.customer_id = B.customer_id

Durdurulması gereken
başka bir sorgunun
değerleri esittirlerse
bu nüta geçilecektir.

Subject:

Date: 21.09.2021

```
SELECT *  
FROM sale.customers A  
WHERE EXISTS (
```

```
    SELECT 1  
    FROM sale.orders B  
    WHERE B.orders_date > '2020-01-01'  
)
```

Not Exists

- İcerdeki sorgu döner miyorsa,
diğerdeki sorguyu boş döndürür, yani çalışmaz.
- İcerdeki cevap evet ise → Çalışırma.
- İcerdeki cevap hayır ise → Çalışır.

Write a query that returns State where "Trek Remedy 9.8-2019" product is not ordered

```

SELECT A.product_id, A.product_name, B.order_id,
FROM product.product A, sale.order_item B,
      sale.orders C, sale.customer D
WHERE A.product_id = B.product_id
AND B.order_id = C.order_id
AND C.customer_id = D.customer_id
AND A.product_name = 'Trek Remedy 9.8-2019'
  
```

SELECT

FROM sale.customer

WHERE NOT EXISTS (

Yukardaki kodu butayla kopyala

) ;

Bu ünkle ilgili bilgi
icermeyecektir.

```
CREATIVE VIEW view-name AS  
SELECT columns  
FROM tables  
[WHERE conditions];
```

TASK

2019 yılindan sonra üretilen ürünlerin bulunduğu bir 'New Products' view'i oluşturun.

```
CREATE VIEW new-view AS  
SELECT product_id, model_year  
FROM product.product  
WHERE model_year > 2018
```

① DROP VIEW new-view → dedi neden bilmiyorum

VIEW gibi

Common Table Expression

Subject:

Date:/..../..



- (*) CTE subquery'lerden hizli calisir.
- (*) Subquery'lerde gosterilen degerlerin dolasma araligi daha kolay.
- (*) Sorgu ile genclik tablolari olusturup olusan tablo Istenilen sorgu yapmak icin kullanilir.
- (*) VIEW'e benzer. (Statement scoped views.)

Ordinary

WITH query_name [(column-name1, ...)]

(SELECT ...) -- CTE Definition

SQL Statement;

Recursive *→ Hizsizlik nedeni sorunludur yararli*

WITH table_name (column-list)

AS

(

— Anchor member

Initial query

UNION ALL

— Recursive member that references table_name

recursive_query

)

— references table_name

SELECT *

FROM table_name

TASK

Sharyn Hopkins'inin en son siparişindeki tarih
verisi ve San Diego şehrinde ikamet eden müşterileri
bul.

bir sonraki
tarih verdict

WITH T1 AS

(

SELECT max(order-date)
FROM sale.customer A, sale.orders B
WHERE A.customer_id = B.customer_id
AND A.first-name = 'Sharyn'
AND A.last-name = 'Hopkins'

)

DISTINCT

SELECT A.order-date, A.order-id, B.customer_id, B.first-name,
B.last-name, B.city
FROM sale.orders A, sale.customers B, T1
WHERE A.customer_id = B.customer_id
AND A.order-date < T1.last-purchase
AND B.city = 'San Diego'

TASK

List all customers who orders on the same dates as Abby Parks.

WITH T1 AS

(

SELECT order_date

FROM sale.customer A, sale.orders B

WHERE A.customer_id = B.customer_id
AND A.first_name = 'Abby'

AND A.last_name = 'Parks'

)

SELECT A.order_date, A.order_id, B.first_name, B.last_name

FROM sale.orders A, sale.customer B, T1

WHERE

AND A.order_date = T1.order_date

Subject:

Veri İretmek istedığımızda kullanılır.

AnaBölde çok fazla verimizde sıkırmaz.

→ Matematiksel problemler

Date:

Recursive CTE Example

WITH CTE

AS (SELECT 1 AS n (anchor member))

UNION ALL

SELECT n+1 (recursive member)

FROM CTE

"Üstteki soruyayla WHERE n < 10 (terminator)

alttaki soruyu

alt alta birleşir.

SELECT n

FROM CTE;

~~~~~

WITH T1 AS

(

0 yassam 0'dan 9'a

kadar devam eder.

SELECT 1 AS NUM

UNION ALL

SELECT NUM+1

FROM T1

WHERE NUM < 9

)

9'a kadar "yaklaşık" 9'adır.  
iteration

SELECT \*

FROM T1

## SET OPERATORS

Subject:

1

UNION

$\rightarrow A \cup B$

Date: ..../.....

UNION ALL

'UNION' den hizli cokusur. Cunku

UNION ALL siralamasi yapmez. DISTINCT yapmez.

SELECT column1, column2 ...

FROM table-A

UNION

SELECT column1, column2

FROM table-B



iki tablodaki  
tüm sütunlar  
ayni olmalı

UNION ALL



UNION  $\rightarrow$  içeriğte aynı olmayanlar  
seçer

UNION ALL  $\rightarrow$  içeriğe baktırın  
direk birleşir.

## TASK

Sacramento şehrindeki müşteriler ile Monroe şehrindeki müşterilerin soyadlarını listeleyiniz.

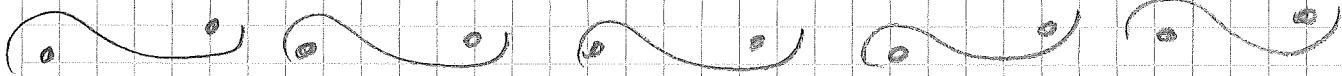
```
SELECT last_name
FROM sale.customers
WHERE city = 'Sacramento'
```

UNION ALL

```
SELECT last_name
FROM sale.customer
WHERE city = 'Monroe'
```

ORDER BY last\_name

Order by en son  
yazılır.



## SET OPERATORS (Kümeler)

① UNION → İki SELECT ifadesinin birleşimi. ( $A \cup B$ )  
Yineleme değerleri almaz

② INTERSECT → Her iki SELECT'in ortaklarını alır. ( $A \cap B$ )

③ EXCEPT → İlk SELECT'ten digerinin olduğu kısmını çıkarır. ( $A - B$ )

TASK

Write a query that returns customers who first name is 'Carter' or last name is 'Carter'

SELECT first\_name, last\_name

FROM sale.customer

WHERE first\_name = 'Carter'

UNION ALL

SELECT first\_name, last\_name

FROM sale.customer

WHERE last\_name = 'Carter'

OR ile çözüm:

SELECT first\_name, last\_name

FROM sale.customer

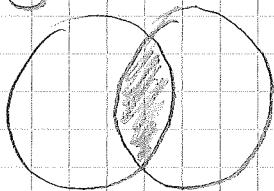
WHERE first\_name = 'Carter' OR last\_name = 'Carter'

$A \cap B$ 

②

INTERSECT

Her iki sorgu tablasının kesişimi alır.



SELECT column1, column2, ...

FROM table\_A

INTERSECT

SELECT column1, column2, ...

FROM table\_B



! Bir sütundaki değer her iki tabloda da aynı olmalı

DISTINCT degerler gelmiş oluyor.

TASK

Write a query that returns brands that have products for both 2018 and 2019.

SELECT B.brand\_name

FROM product.product A ; product.brand B

WHERE A.brand\_id = B.brand\_id

AND A.model\_year = 2018

INTERSECT

Aynı kodu yaz sadece model-year = 2019  
olacak

TASK

Write a query that returns customers who have orders for both 2018, 2019 and 2020.

Burası  
kod  
geliyor  
sağ  
ayrıca  
var

SELECT customer\_id

FROM sale.orders

WHERE order\_date BETWEEN '2018-01-01' AND '2018-12-31'

INTERSECT

(Yukardaki aynı kod sadece BETWEEN 2019 olacak)

INTERSECT

(Yukardaki aynı kod BETWEEN 2020 olacak)

)

```
SELECT first-name, last-name  
FROM sale.customer  
WHERE customer-id IN (  
    )
```

Onceki sayfadaeki  
kod getecek

VIEW → Görmek istedigin bir tabloyu istem vererek  
oluşturuyorsun. Sonra istersen drop  
ile silebilirsin

View ile CREATE kullanabilirim  
DROP

CREATE VIEW view-name AS

SELECT columns  
FROM tables  
[WHERE conditions];

DROP VIEW view-name; } Drop

! each id  
each grünce grpla. "id'ye görne grupby yap."

Subject :

Date : .....

## CASE - PRE-CLASS

SELECT first-name, last-name, hire-date,

CASE

WHEN hire-date < 2019 THEN 'old employee'

WHEN hire-date >= 2019 THEN 'new employee'

END AS empcat

→ year after act.

FROM employees

ORDER BY hire-date;

Veya :

WHEN hire\_date > '2019-01-01' THEN 'new employee'

ELSE 'old employee'

Subject :

25.10.2021

Date : .....

3

EXCEPT

A  
B  
C

B  
C  
D

T1



A  
B  
C  
D

T1 EXCEPT T2

SELECT column1, column2, ...

FROM table\_A

EXCEPT

SELECT column1, column2, ...

FROM table\_B

## TASK

2018 model birekleti olan markalarin hangileri?  
2019 model? yoktur?

SELECT brand\_id, brand\_name  
FROM product.brand  
WHERE brand\_id IN (

SELECT brand\_id  
FROM product.product  
WHERE model\_year = 2018

EXCEPT

SELECT brand\_id  
FROM product.product  
WHERE model\_year = 2019 )

TASK

Sadece 2019 yılınca sipariş verilen ürünler?

SELECT A.product\_id, A.product\_name

FROM product.product A

INNER JOIN (

SELECT A.product\_id

FROM sale.order\_item A, sale.orders B

WHERE A.order\_id = B.order\_id

AND B.order\_date BETWEEN '2019-01-01' AND '2019-12-31'

EXCEPT

SELECT A.product\_id

FROM sale.order\_item A, sale.orders B

WHERE A.order\_id = B.order\_id

AND B.order\_date NOT BETWEEN

)

ON A.product\_id = B.product\_id.

LIKE ?le : 2.yıl

SELECT COUNT(DISTINCT A.product\_id)

FROM sale.order\_item A, sale.orders B

WHERE A.order\_id = B.order\_id

AND B.order\_date LIKE "2019%"

# Simple Case

Subject :

Date : .....

**CASE** → Sartı, bir sistem

CASE case-expression

WHEN when-expression\_1 THEN result-expression\_1

WHEN when-expression\_2 THEN result-expression\_2

-----

[ELSE else-result-expression]

END

~~~~~

SELECT dept-name,

CASE dept-name

WHEN 'Computer-Science' THEN 'IT'

ELSE 'others'

END AS category

FROM department;

category adında

bir sonuç oluştur

Computer-Science → "IT"
jaz ger kalan
"others" jaz.

Subject :

Date : / /

TASK

Order status listesi alanlari degosterin ne onlara geldigini ticeren yeni bir olan dusunun.

```
SELECT order_id, order_status,
```

```
CASE order_status
```

```
WHEN 1 THEN 'Pending'
```

```
WHEN 2 THEN 'Processing'
```

```
WHEN 3 THEN 'Rejected'
```

```
'ELSE 'Completed'
```

```
END AS mean_of_status
```

```
FROM sale.orders
```

```
ORDER BY order_status
```

TASK

Staff tablosunu calisanlarin maftasi listesini ekleyin

```
SELECT first_name, last_name, store_id
```

```
CASE store_id
```

```
WHEN 1 THEN 'Sacramento Bikes'
```

```
WHEN 2 THEN 'Buffalo Bikes'
```

```
'ELSE 'San Angelo Bikes'
```

```
END AS store_name
```

```
FROM sale.staff
```

SEARCHED CASE } → Simple Case expression'

Kapsar.

CASE

WHEN condition-1 THEN result-1

WHEN condition-2 THEN result-2

WHEN condition-N THEN result-3

[ELSE result]

END

"ONCEKİ SORU SEARCHED CASE ile:

SELECT first-name, last-name, store-id,

CASE

WHEN store-id = 1 THEN 'Sacramento Bikes'

WHEN store-id = 2 THEN 'Buffalo Bikes'

WHEN store-id = 3 THEN 'San Angelo Bikes'

END AS Store-name

FROM sale.staff

TASK

Müşterilerin e-mail adreslerindeki servis sağlayıcılarının
yeni bir sütun oluşturarak belirtiriz.

· SELECT first_name, last_name, email,

CASE

· WHEN email LIKE "%@yahoo%" THEN "Yahoo"
WHEN email LIKE "%gmail%" THEN "Gmail"
WHEN email LIKE "%hotmail%" THEN "Hotmail"
ELSE "Other"

END AS email-service

FROM sales.customers

ELSE email NOT NULL THEN others' da díyebilirim.

(IS NULL) da olmaması durumda göre.

Subject :

Date : / /

HOMEWORK

Aynı Siparişte hem Electric Bikes hem Comfort Bicycles
hem de Children Bicycles türlerini Sipariş veren müşterilerin jumlahı

DATES

SQL supports \Rightarrow DATE

TIME

DATETIME

JULIANDAY

STRFTIME

Subject:

Date : / /

SELECT CONVERT(VARCHAR, GETDATE(), 6)

J

Get date's VARCHAR string datatype

Time format is Olson

SELECT CONVERT(DATE, '25 Oct 21', 6)

J

VARCHAR's date's datatype

TASK

Functions for return date or time parts

SELECT A_date,
DATENAME(WEEKDAY, A_date) [weekday],
DAY(A_date) [day2],
MONTH(A_date),
YEAR(A_date),
A_time,
DATEPART(NANOSECOND, A_time),
DATEPART(MONTH, A_date)

Subject:

Date : / /

DATENAME (dw, A_date) [DAY],
DAY(A_date) [DAY_2],
MONTH(A_date),
YEAR(A_date),
A_time,

DATEPART (NANOSECOND, A_time),
DATEPART (MONTH, A_date)

Ppt - CLASS

SELECT

' , [' + CountryName + ']'
FROM
tblCountry

→ GIKTI ⇒ [China]
[France]

Australia
Seychelles

SELECT

QUOTENAME (CountryName)

From
tblCountry

→ GIKTI ⇒ [China]
[France]

↓
QUOTENAME (CountryName, " ")
→ GIKTI ⇒ ("China")
("France")

Subject :

Bunu → ekleyebilim.
Aylara göre yıl bir ti.
sister: / /

```
SELECT *  
FROM  
( SELECT CountryName,  
        DATENAME(MM, FilmReleaseDate) AS [FilmMonth]  
        ↑  
        YEAR(FilmReleaseDate) AS [FilmYear], FilmId  
    FROM tblCountry ASC  
    INNER JOIN tblFilm AS f  
    ON c.CountryId = f.FilmCountryId AS BaseDate
```

PIVOT (

COUNT(FilmId)

FOR CountryName

IN([China])

[France]

[]
[]
)

→ Bulular Column
ismi? oldular

) As PivotTable.

ÇIKTI \Rightarrow FilmYear FilmMonth China France Germany

1 - - -

2 - - -

3 - - -

DATE

Subject :

28.10.2021

Date : / /

```
SELECT A_time, A_date, GETDATE(),
       DATEDIFF(MINUTE, A_time, GETDATE()) AS MinuteDiff,
       DATEDIFF(WEEK, A_date, '2021-11-30') AS WeekDiff
```

From L_date-time

Sipariş tarihinden testim tarihini çıkardık.

```
SELECT DATEDIFF(DAY, shipped_date, order_date) DATE_DIFF,
      order_date, shipped_date
```

FROM sale.orders

Function

Syntax

Return Date Type



!iste dikkat cinsten (day, month, year, week) zaman farkını verir.

Subject :

Date :

DATEADD,

DATEADD \Rightarrow DATEADD(
datepart
number
date) \Rightarrow The data type
of date
argument

Gün, saat, dakika ekrabiller

EOMONTH

EOMONTH \Rightarrow EOMONTH(start_date [month_to_add])

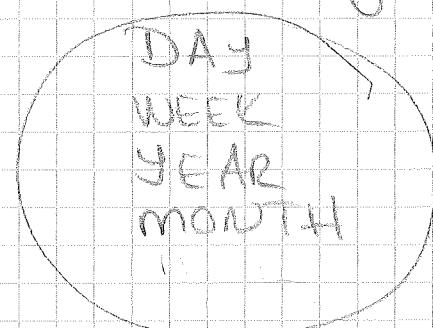
İkiinci bulunduğu ayın
son günün neşesi olduğunu getirir.

SELECT ORDER_DATE,

DATEADD(YEAR, 5, order_date)

FROM sale.orders

\Rightarrow Bu tür order_date'le
re 5'er yıl ekledi.



(-) deşer de yata
biraz

SELECT GETDATE(), DATEADD(HOUR, 5, GETDATE())

Saat 5 saat ekledi.

SELECT EOMONTH(GETDATE()),
EOMONTH(GETDATE(), 2)

2 ekte
yani 12. ay
aikt,

ISDATE \Rightarrow ISDATE(expression) \Rightarrow Int(Boolean)

Varchar tipindeki bir veri DATE mi değil mi onu söyleş

SELECT ISDATE('2021-10-01')

$\rightarrow 1$ (Evetse 1 sonucunu verir)
Hayırsa 0

TASK

Orders tablouna sipoislerin testimat hizyla ilgili bir alan ekleyin

Testimat olmadysa \Rightarrow Not Shipped

Sipois juri testim \Rightarrow Fast

Sipoislerin sonraki 2 gün \Rightarrow Normal

2 günden geese \Rightarrow Slow

SELECT order_id, DATEDIFF(day, order_date, shipped_date)

DATE-DIFF,

CASE

WHEN shipped_date IS NULL THEN 'Not Shipped'

WHEN DATEDIFF(day, order_date, shipped_date) = 0
THEN 'Fast'

WHEN DATEDIFF(" " " ") <= 2 THEN 'Normal'

WHEN DATEDIFF(" " " ") > 2 THEN 'Slow'

END AS Labels

FROM sale.orders

11

Hoca WITH file olsadu.

Subject:

Date:/..../.....

TASK

2 günlerde ger gönderilen sipariş bilgilerini getirin.

WITH TL

```
(SELECT *, DATEDIFF(day, order_date, shipped_date)
AS diffday
FROM sale.orders
WHERE DATEDIFF(day, order_date, shipped_date) > 2)
```

TASK

→ Projede bu yılın varmış

2 günlerde ger gönderilen siparişlerin nedenlerini
göre logilama

(Pivot ile de yapılabilir.)

Sadece farklılıkla ilgili var. Bu günlerde order
tablosuna göreceğini.

DATEPART int dördün ogütde onu kullanmadım

select:

Date : / /

SELECT order_date,

sum (CAST WHEN DATE_NAME(WEEKDAY, order_date) =
'Monday' THEN 1 END)

in Monday



Sırası P1'ten verildiğine

1 yaz ve o

bırakır lopla.

Tüm günler için SUM P1'ni yapacak

From sale.orders

WHERE DATEDIFF(DAY, order_date, shipped_date) > 2

STRINGS

LEN(input_string) \Rightarrow İçine yazardığın str'inin kaç karakteri var?

CHARINDEX(substring, str) \Rightarrow str'de substring'in konusu location'da string[, start_location] olduğunu söyley

PATINDEX('pattern', input_string) \Rightarrow Bir str'de bir pattern arakten kullanılır.

LEFT(input_string, number_of_characters) \Rightarrow Sıdan sv kadar karakter getir

RIGHT(input_string, number_of_characters) \Rightarrow Sağdan sv kadar karakter getir

SUBSTRING(input_string, start, length) \Rightarrow Konuları karakterden başlayıp konuları karaktere kadar is tedbirini söylesit.

LOWER (input_string) \Rightarrow Str ifadeğin küçük harfe çevirir.

UPPER (input_string) \Rightarrow " " büyük harfe çevirir.

STRING_SPLIT (input_string)

seperator)

\Rightarrow FROM 'a yazılır.

Sprint teki virgüllerle ayrılmış değerleri alır ve bir satır oluyor olarak o satırın altına virgülle ayrılmış değerler satır olarak yazdırır.

TRIM ([removed; characters
from] input_string) \Rightarrow Sağ ve soldaki boşlukları atar.

LTRIM (input_string) \Rightarrow Soldaki boşlukları atar.

RTRIM (input_string) \Rightarrow Sağdaki boşlukları atar.

REPLACE (input_string, substring, new_string) \rightarrow Bir seyin yerine bir seyi kaynak.

STR (float_expression), \rightarrow İndeks değer float length[, decimal] olarak kabul eder ve

SCAST (expression AS target_type [(length)])

(CONVERT (target_type [(length)], expression [, style]))
Format değiştirmek için kullanılır.

COALESCE (Expression1, [E2, ..., En])

NULLIF (Expression1, Expression2) \rightarrow Sonuç eşitçe olursa NULL olur.
CASE 'lerde yapılacak işlem hali yapmamızı sağlar.

ROUND (number, decimals, [operation])

\rightarrow Yuvarlama yapar. Default 0 sifir.

SELECT ISNUMERIC(replace('1234557', 123, 222))

\downarrow Numeric mi değil mi dif baktık

SELECT CAST (0.333 AS NUMERIC(3,2)) } ^{Verify de}
 SELECT CAST (0.333 AS DECIMAL(3,2)) } ^{sym}

SELECT CONVERT(DATETIME, '2021-10-10')

SELECT CONVERT(INT, 30.48)

↳ Verify Pint'e convert.

SELECT CAST ('123135' AS VARCHAR) + 1

SELECT COALESCE(NULL,NULL,'Ahmed',NULL)

↳ 'Ahmed' oldu

SELECT NULLIF(10,9)

TASK

How many yahoo Mails in customer's email column?

(Use Case Expression and PATINDEX() function?)

SELECT

sum(CASE WHEN PATINDEX('%@yahoo%', email) < 20
 AND PATINDEX('%@yahoo%', email) IS NOT
 NULL
 THEN 1
 ELSE 0 END) AS yahoo

FROM sale.customer

⑥ ~~SELECT COUNT(*)~~

FROM sale.customer,

WHERE email LIKE '%@yahoo%'

→ 2.90 L

Subject:

Date:/..../..

TASK

İper tel no varsa yet, NULL ise email gelir.

```
SELECT * , COALESCE(phone, email) AS contact  
FROM sale.customers
```

WINDOW FUNCTION (ANALYTIC İŞLEMELER)

Group By' da griplanmış bir geçici tablo oluşturur.

WINDOW function arkasında geçici bir tablo oluşturur.

Griplama yapmaz. Her satırın karşılık AGG degerini
getirir. GROUP BY gibi DISTINCT islemi yapmaz.
Distinct yapmak istiyorsak yazmak zorundayız.

WINDOW funk.'da AGG yapmak zorunda değiliz

Group By' dan data null.

TASK

Stock amounts of products. (only stock table)

```
SELECT product_id, sum(quantity) cnt_stock  
FROM product.stock  
GROUP BY product_id  
GROUP BY 1
```

2. SQL

`SELECT *, sum(quantity) OVER (PARTITION BY product_id)`
`total_stock`
 From product_stock

`SELECT columns,`

`FUNCTION (...) OVER (PARTITION BY ... ORDER BY ...)`
 sum gibi
 window frame

From table1

Window frame
 (optional)

✓ UNBOUNDED PRECEDING AND CURRENT ROW

Window in default.

Bütün sıralı satırın kader bak

ROWS BETWEEN 1 PRECEDING AND CURRENT ROW

Bir öncek satırın en sonu satır kader
 hesaba kat

Window funk

TASK

What is the cheapest bike price?

```
SELECT MIN(list_price) OVER ()  
FROM product.product  
ORDER BY category_id, product_id
```

} En çıktı tüm
bisikletlerin en ucuz
fiyatı.

TASK

Her bir kategorideki en ucuz bisikletin fiyatı?

```
SELECT DISTINCT category_id, MIN(list_price) OVER (PARTITION  
BY category_id)
```

! PARTITION BY \Rightarrow Defines window partitions to form
groups of rows

ROW or RANGE \Rightarrow Defines the scope of the function

Pre-Class

Subject:

Date:/.....

SQLite Window Functions

✓ AGG

Avg()

Count()

Max()

Min()

Sum()

RANKING

CUME_DIST()

DENSE_RANK()

NTILE()

RANK()

ROW_NUMBER()

PERCENT_RANK()

OFFSET-FUNCTIONS

VALUE

FIRST_VALUE()

LAST_VALUE()

LAG()

LEAD()

NTH_VALUE()

Window function is similar to a normal agg func.

But main difference between them is the agg window func. doesn't change the number of rows returned.

→ AVG, COUNT, MAX, MIN, SUM stability

Window function (column-name)

OVER ([PARTITION BY expr-list] [ORDER BY order-list] frame-clause)

Isleni stradan
AGG funkcıdan
ayırır

Optional
Defines the window for
the window function

Optional
Satırları sınırlar
ORDER BY
varsayı da
çukur 2'nden

```
SELECT InvoiceDate, Total, SUM(Total)
    OVER (ORDER BY InvoiceDate) AS running_total
FROM Invoices
LIMIT 16;
```

FRAMING SYNTAX

Rows BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW:

Start at row 1 - include rows up to the current row.

Rows UNBOUNDED PRECEDING:

Start at row 1 - include rows up to the current row.

Rows BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING:

Start at current row - include rows up to the end

Rows BETWEEN CURRENT ROW AND N FOLLOWING:

Start at the current row - include N row

Rows BETWEEN N PRECEDING AND CURRENT ROW:

Start at N row - include the current row.

Rows BETWEEN N PRECEDING AND N FOLLOWING:

Start at N row - include last N row.
(The current row is included)

Subject :

Date : / /

```
SELECT InvoiceDate, Total, AVG(Total)
OVER(ORDER BY InvoiceDate
ROWS BETWEEN 4 PRECEDING AND CURRENT
ROW) AS
rolling-average
FROM Invoices;
```

Subject :

04.11.2021

Date : / /

Tüm bisikletler arasında en ucuz bisikletin adı:

First_Value ile :

```
SELECT DISTINCT FIRST_VALUE(product_name)
OVER (PARTITION BY category_id ORDER BY list_price)
AS F_V
FROM product.product
```

Subject:

önceki
satır

sonraki
satır

Date:

LAG() - LEAD()

Her bir satır için istekliliğin kaderi öncesi-sonrakı satır gösteriyor.

Arguman alımları ve kaç tane öncesi ve kaç deger sonrasını istekliliğini söyleyez.

SELECT order_date,

LEAD(order_date, 2) OVER (ORDER BY order_date)

next-second-w. LEAD

FROM sale.orders;

order_date'in 2 sonraki satır larini göster.

TASK

Her bir personelin bir önceki satısının sırası lâkini yapınız. (LAG fonk. kullan.)

Bunu yapanıza çok övgü!

PARTITION BY A.staff_id

SELECT

- LAG(order_date) OVER (PARTITION BY A.staff_id ORDER BY order_id) pre_order

FROM sale.orders A, sale.staff B

WHERE A.staff_id = B.staff_id

ORDER BY A.staff_id

Order_date gelersen ayın ilk 10 tane olur
da var, o yıldan
yazılacak

TASK

Herkâft müsteri idin bir sonrakı satışının tarihini
getirin. (use the LEAD func.)

```
SELECT A.staff_id, B.first_name, B.last_name, A.order_id, A.order_date,  
LEAD(A.order_date) OVER(PARTITION BY A.staff_id ORDER BY A.order_id  
next_order_of  
FROM sale.staff B, sale.orders A  
WHERE A.staff_id = B.staff_id  
ORDER BY A.staff_id
```

Bunlarda ORDER BY kullanmak

zoruldugut

Subject:

Date:/..../..

7

Row_Number()

Numaralandırma yapar.

Mesela order_id'yi sırala, yeni bir sutun oluştur.

Ortalık sutunu olmayan tablolardaki sıralamalar için kullanılır.

SELECT order_id, item_id

ROW_NUMBER() OVER (ORDER BY order_id) Row_Numb

FROM sale.order_items

Rank()

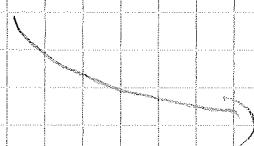
Numaralandırma yapar. (Veresaptılık sırasına göre)

Aynı değerlerde aynı no'yu verir.

SELECT order_id

RANK() OVER (ORDER BY order_id) [Rank]

FROM sale.order_items



1
1
1
1
5
5
7
7
7
10
10

(Piping)

Dense_Rank()

Numealandırma yapar.

SELECT order_id,

DENSE RANK() OVER (ORDER BY order_id) [dense_rank]

FROM sale.orders ?Hem

1
1
1
2
2
2
3

} Bu şekilde sıralar.

Subject :

Date : 15.10.2023

CUME_DIST()

Cumulative Distribution denek.

5 nolu değerin toplam değerler içindeki birincenin nesilimiz

$$\text{CUME_DIST} = \text{ROW_NUMBER} / \text{TOTAL ROWS}$$

PERCENT_RANK()

Göreceli pozisyon deneksi. Bir sutun içindeki bir değerin kendiinden önceki değerlere göre göreceli po-
zisyonunu söyle.

$$\text{PERCENT_RANK} = (\text{ROW_NUMBER} - 1) / (\text{TOTAL ROWS} - 1)$$

NTILE(N)

Sıralanmış bir sutunu verdiğimizde sayısına göre böler.

TASK

Her bir kategoriye göre bisiklet fiyatlarını artan sıraya
göre sırala ve bir sıra no'sunu.

```
SELECT category_id, list_price,  
ROW_NUMBER() OVER (PARTITION BY category_id ORDER BY list_price)  
FROM product.product
```

TASK

Aynı soruya aynı fiyatlı bisikletler aynı sıra no'sunu
alacak şekilde yanıtla. (RANK ile)

```
SELECT category_id, list_price  
ROW_NUMBER() OVER (PARTITION BY category_id  
ORDER BY list_price) row_num,  
RANK() OVER (PARTITION BY category_id ORDER BY list_price)  
ranknum  
FROM product.product
```

TASK

Müşterinin sipariş verdiği sipariş sayılarının liste içindeki
kümülatif dağılımını.

TASK

Müşterilerin sipariş verdiği günün, saylarının liste içindeki kümülatif olduğunu.

WITH T1 AS (

```
SELECT A.customer_id, sum(quantity) product_quantity
FROM sale.orders A, sale.order_item B
WHERE A.order_id = B.order_id
GROUP BY A.customer_id
```

)

DISTINCT

```
SELECT product_quantity,
ROUND(CUME_DIST() OVER(ORDER BY product_quantity), 2) cum_dist
FROM T1
ORDER BY 1
```

TASK

Ün saylarının görevdeki pozisyonunu göster.

Üstteki cevapta CUME_DIST() yerine PERCENT_RANK() kullanırsınız.

Kümülatif Toplam Order By ile çözülmür.

Subject :

Date : / /

TASK

Sipariş verdiğken JWN sayısına göre müşteriler 5 gruba bolun.

Yine önceki sonudaki TL'yi al sonda,

WITH TL AS

(

)

```
SELECT DISTINCT customer_id, NTILE(5)
OVER (ORDER BY product_quantity) percent_rnk
FROM TL
ORDER BY 1
```

INSTR Function

SELECT INSTR ('Reinvent yourself', 'yourself')

Output → 10

start-position;

Yourselves'in başla diğri pozisyonu çıktı
olarak verdi.

SELECT first_name, last_name, job_title,
INSTR (job_title, 'er') AS er

FROM employees
WHERE er > 0;

→ 'er'in 0'dan büyük pozisyonlarında olduğu
yerleri getir.

CONCAT ||

SQLite doesn't support CONCAT.

Onun yerine → || kullanılır.

SELECT 'Reinvent' || ' yourself'

Output → Reinvent yourself

Subject :

Date : / /

```
SELECT (first-name || " " || last-name) full-name  
FROM employees;
```