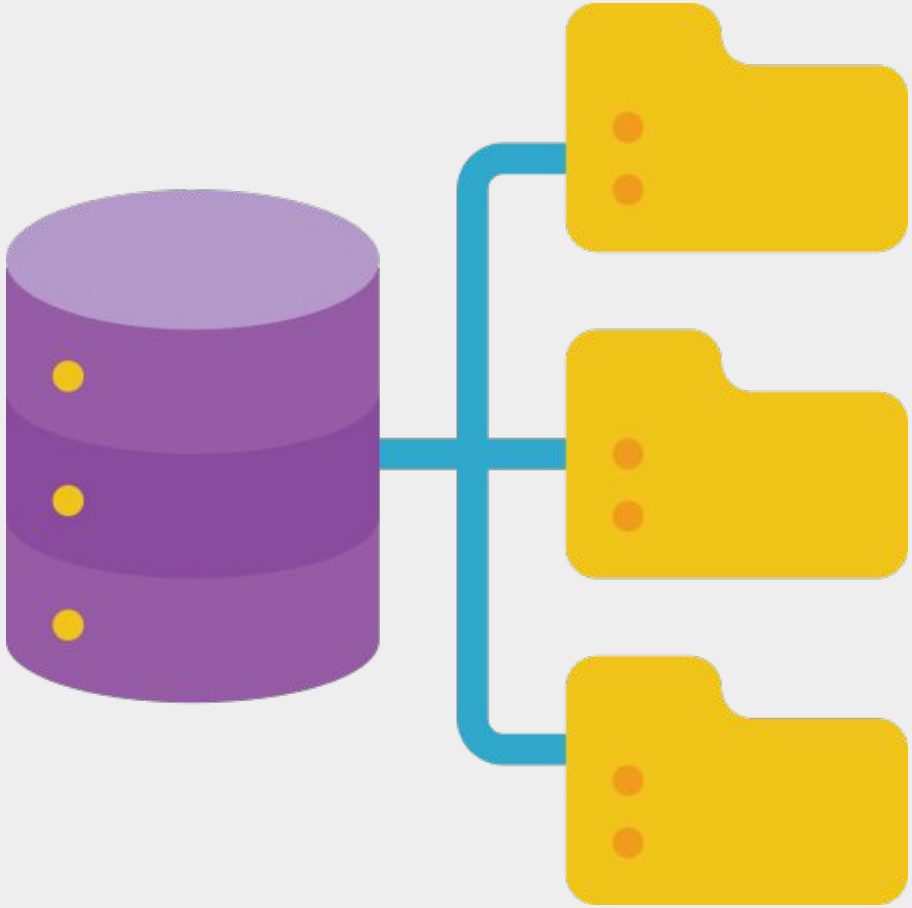
SQL Zero2End

Özge Usta

Senior Data Scientist



Eğitim İçeriği

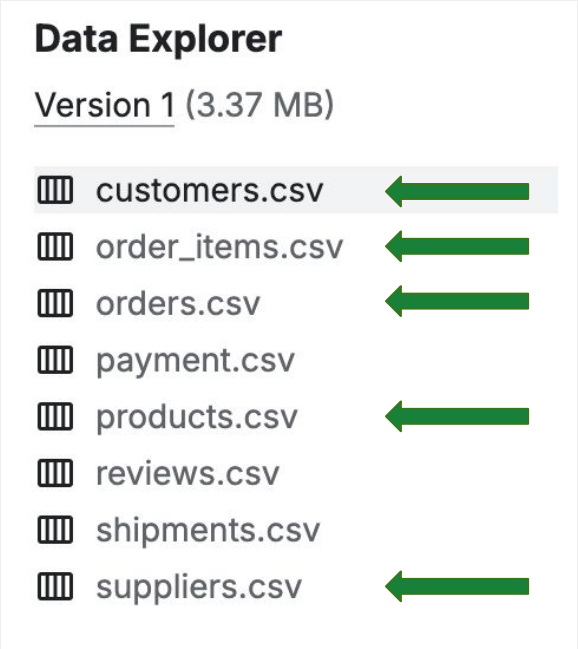
- SQL Nedir?
- Temel Veri Sorgulama
- Veri Filtreleme
- Veri Sıralama
- Koşullu Veri Dönüşümü
- Toplama (Aggregation) Fonksiyonları ve Gruplama
- Tablo Birleştirme (JOINs)
- Alt Sorgular (Subqueries)
- Pencere Fonksiyonları (Window Functions)
- İleri Konular ve Öneriler

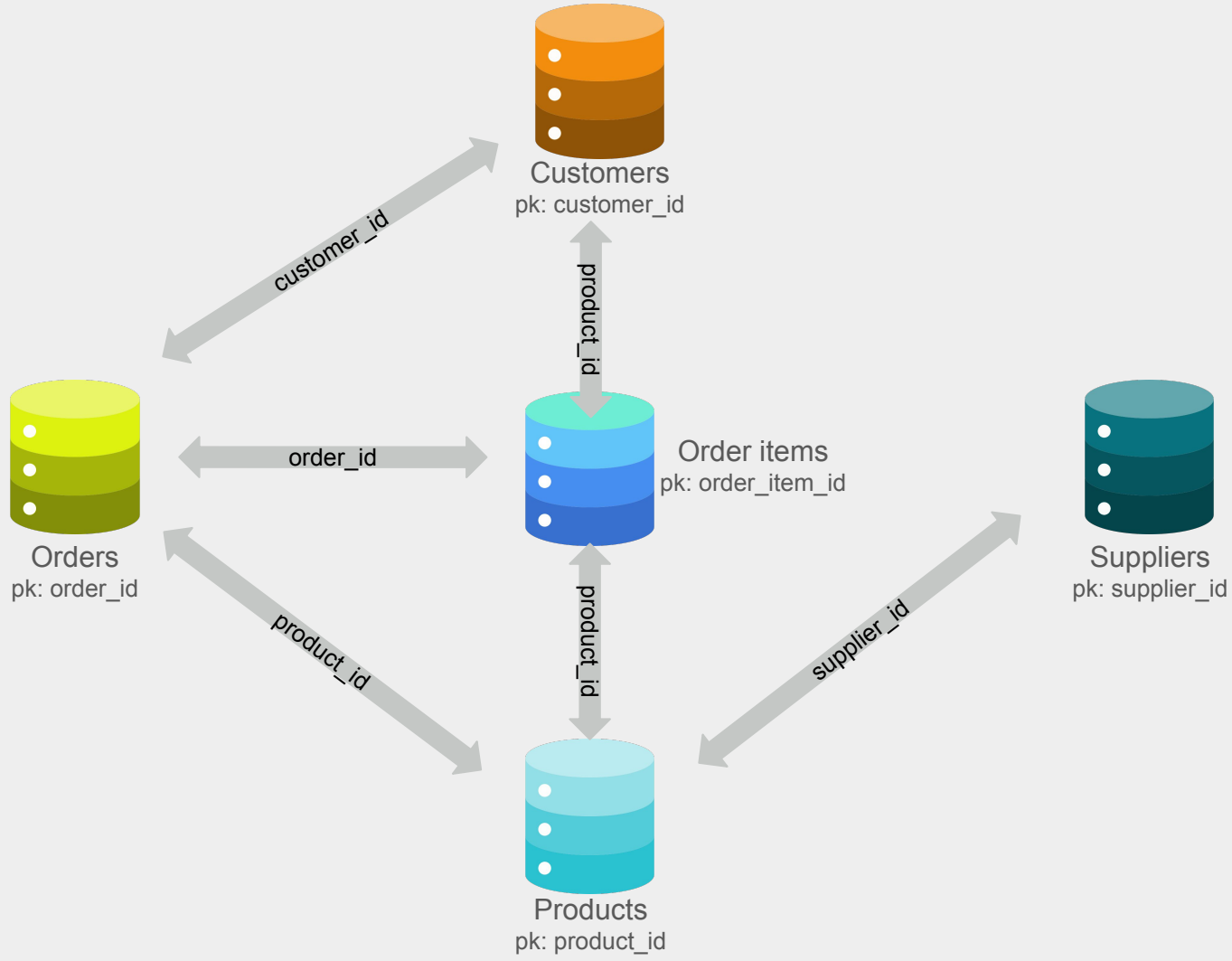
SQL Nedir?

- Veritabanlarındaki veriyi **oluşturmak**, **okumak**, **güncellemek** ve **silmek** için kullanılan komutlar bütünüdür.
- Veritabanı yönetim sistemleri ile kullanılır.
- Sözdizimi (syntax) büyük ölçüde standarttır. Bu yüzden birini öğrenince diğerlerini kullanmak kolaydır.

Verimizi Tanıyalım

🛒 Online Shop 2024 | Kaggle




Customers
pk: customer_id
customer_id
product_id
Orders
pk: order_id
order_id
Order items
pk: order_item_id
Suppliers
pk: supplier_id
product_id
product_id
supplier_id
Products
pk: product_id

Temel Veri Sorgulama

SELECT * FROM …

——————————

SELECT

    column1, column2, …

FROM …

——————————

SELECT DISTINCT

    column1, column2, …

FROM …

```
SELECT DISTINCT
  supplier_id,
  category
FROM `case_study.products`
;
```

Veri Filtreleme

1. Mantıksal Operatörler
   - AND/OR
2. Karşılaştırmalar
   - Temel Karşılaştırmalar (=, >, <, <>)
   - BETWEEN
   - IN
   - IS (NOT) NULL
   - LIKE
   - …

```
SELECT
  product_id,
  price
FROM `case_study.products`
WHERE category IN ("Home & Kitchen", "Furniture")
  OR price BETWEEN 500 AND 1000
```

Veri Sıralama

SELECT

…

FROM …

WHERE …

ORDER BY … ASC/DESC

LIMIT …

```sql
SELECT
  order_id,
  customer_id,
  total_price
FROM `case_study.orders`
ORDER BY total_price DESC
LIMIT 5;
```

Koşullu Veri Dönüşümü

- IF

SELECT

IF(condition1, value_1, value_2)

FROM …

```
SELECT
  order_id,
  IF(total_price <= 500, "<=500", ">500") as price_segment
FROM `case_study.orders`
```

- CASE WHEN

SELECT

CASE WHEN condition1 THEN value_1

WHEN condition2 THEN value_2

… ELSE value_n END

FROM …

Toplama (Aggregation) Fonksiyonları ve Gruplama

- SUM
- COUNT
- MIN
- MAX
- AVG
- STDDEV
- …

```
SELECT
  customer_id,
  SUM(total_price) AS customer_total_price
FROM `case_study.orders`
GROUP BY 1
HAVING SUM(total_price) > 1000
```

```
SELECT
    column1,
    column2, …
    (toplama fonksiyonu)
FROM …
WHERE …
GROUP BY …
HAVING conditions
ORDER BY …
```

Sorular

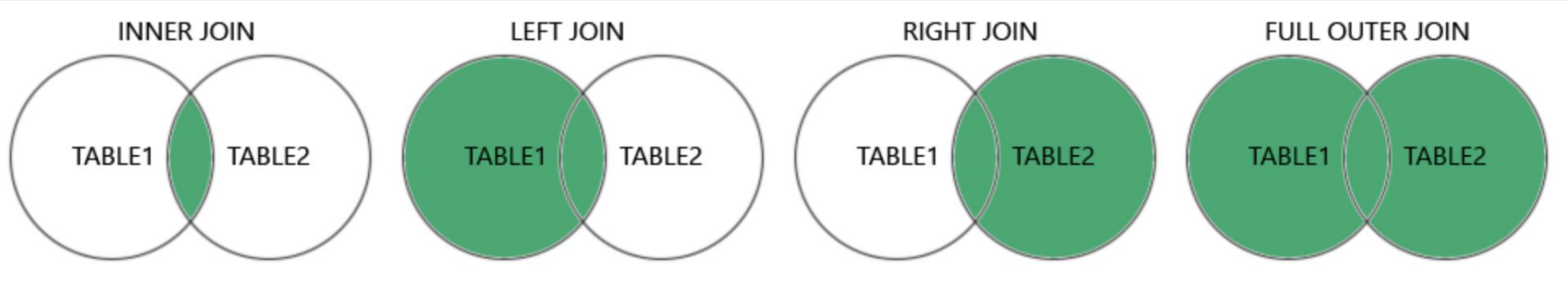
- Her category için:
  - kaç farklı ürün
  - kaç farklı supplier
  - category başına ortalama ürün fiyatı
- Siparişlerin toplam tutarlarına göre gruplayıp (1000-2000-3000-...) her grup için sipariş adedini bulalım.
- 10'dan fazla ürün içeren tüm siparişlerin ID'lerini (order_id) listeleyelim.

```sql
select
  category,
  count(distinct product_id) as product_count,
  count(distinct supplier_id) as supplier_count,
  avg(price) as avg_price
from `case_study.products`
group by 1
```

```sql
select
  case when total_price <= 1000 then "<=1000"
    when total_price <= 2000 then "1000< <=2000"
    when total_price <= 3000 then "2000< <=3000"
  else ">3000" end as price_segment,
  count(order_id) as total_order
from `case_study.orders`
group by 1
```

```sql
select
  order_id,
  sum(quantity) as total_product_count
from `case_study.order_items`
group by 1
having total_product_count > 10
```

Tablo Birleştirme (JOINs)



Tablo Birleştirme (JOINs)

```
SELECT
    *
FROM table1 AS t1
INNER/LEFT/RIGHT/FULL JOIN table2 AS t2
    ON t1.column_1 = t2.column_2
    ...
```

```sql
SELECT
  order_id,
  t1.customer_id,
  first_name,
  last_name
FROM `case_study.orders` t1
LEFT JOIN `case_study.customers` t2
  ON t1.customer_id = t1.customer_id
```

Sorular

- Her category için:
  - toplam geliri
  - satılan ürün sayısını
  - kaç farklı kullanıcının aldığını

hesaplayalım ve satış hacmine göre sıralayalım.

- Hiç satışı olmamış ürünlerin isim ve kategori bilgilerini alalım.
- İçinde “City” geçen adreslerin her birine kaç farklı tedarikçi ürün satmış bulalım.

```sql
select
   category,
   sum(price_at_purchase * quantity) as total_revenue,
   sum(quantity) as total_product_count,
   count(distinct customer_id) as distinct_customer_count
from `case_study.orders` t1
join `case_study.order_items` t2
  --on t1.order_id = t2.order_id
  using(order_id)
join `case_study.products` t3
  --on t2.product_id = t3.product_id
  using(product_id)
group by 1
```

```sql
select distinct
  t1.product_id,
  category
from `case_study.products` t1
left join `case_study.order_items` t2
  using(product_id)
where t2.product_id is null
```

```sql
select
  address,
  count(distinct supplier_id) as distnct_supplier_count
from `case_study.customers`
join `case_study.orders`
  using(customer_id)
join `case_study.order_items`
  using(order_id)
join `case_study.products`
  using(product_id)
where lower(address) like "%city%"
-- address like "%City%"
group by 1
```

Alt Sorgular (Subqueries)

Türetilmiş Tablolar (Nested Subqueries):

```
SELECT
    …
FROM …
WHERE condition IN (SELECT * FROM …)
```

Ortak Tablo İfadeleri - CTE (Common Table Expressions):

```
WITH base AS (
    SELECT … FROM … WHERE …
)
SELECT
    …
FROM base
```

Sorular

- Hiç satışı olmamış ürünleri tedarik eden tedarikçilerin bilgilerini getirelim.
- Öncelikle sattığı toplam ürün miktarına göre en çok ürün satan 3 tedarikçiyi bulalım ve bu tedarikçilerin hangi üründen ne kadar sattığını bulalım.
- Her müşterinin birinci siparişler dışındaki tüm siparişlerinin total hacmini bulalım.

```sql
with not_purchased_products as (
  select distinct
    t1.product_id,
    supplier_id
  from `case_study.products` t1
  left join `case_study.order_items` t2
    using(product_id)
  where t2.product_id is null
)
select
  supplier_id,
  supplier_name
from `case_study.suppliers`
join not_purchased_products
  using(supplier_id)
```

```sql
with top_suppliers as (
  select
    supplier_id,
    sum(quantity) as product_count
  from `case_study.suppliers`
  join `case_study.products`
    using(supplier_id)
  join `case_study.order_items`
    using(product_id)
  group by 1
  order by product_count desc
  limit 3
)
select
  supplier_id,
  product_id,
  sum(quantity) as supplier_product_count
from `case_study.suppliers`
join top_suppliers
  using(supplier_id)
join `case_study.products`
  using(supplier_id)
join `case_study.order_items`
  using(product_id)
group by 1,2
```

```sql
with first_orders as(
  select
    customer_id,
    min(order_date) as first_order_date
  from `case_study.orders`
  group by 1
)
select
  sum(total_price) as total_revenue,
  sum(if(t2.customer_id is null, total_price, 0)) as total_revenue_wout_first_orders

from `case_study.orders` t1
left join first_orders t2
  on t1.customer_id = t2.customer_id
  and t1.order_date = t2.first_order_date
```

Pencere Fonksiyonları (Window Functions)

1. Sıralama Fonksiyonu
    - ROW_RANK, DENSE_RANK, LAG, LEAD…
2. Toplama (Aggregation) Fonksiyonu)
    - SUM, COUNT, AVG…

```
SELECT
    …,
    (function) OVER (PARTITION BY ... ORDER BY ...)
FROM base
```

Sorular

- Öncelikle sattığı toplam ürün miktarına göre en çok ürün satan 3 tedarikçiyi bulalım ve bu tedarikçilerin portföyündeki **en çok satan 5 ürün**ün listesini getirelim.
- Her müşterinin son siparişine ait zaman ve sipariş ID’sini bulalım.
- Her müşteri için
  - total sipariş geliri
  - bulunduğu adresin toplam sipariş geliri
  - zaman içinde (sipariş tarihine göre - order_date bazında) kümülatif olarak artan toplam sipariş gelirini

tek bir tabloda toplayalım.

```sql
with top_suppliers as (
  select
    supplier_id,
    sum(quantity) as product_count
  from `case_study.suppliers`
  join `case_study.products`
    using(supplier_id)
  join `case_study.order_items`
    using(product_id)
  group by 1
  order by product_count desc
  limit 3
),
base as (
  select
    supplier_id,
    product_id,
    sum(quantity) as total_product_count,
    row_number() over(partition by supplier_id order by sum(quantity) desc) as rank_

  from `case_study.suppliers`
  join top_suppliers
    using(supplier_id)
  join `case_study.products`
    using(supplier_id)
  join `case_study.order_items`
    using(product_id)
  group by 1,2
)
select
  *
from base
where rank_ <= 3
```

```sql
select
  customer_id,
  order_id,
  order_date,
  row_number() over(partition by customer_id order by order_date desc) as rank_
from `case_study.orders`
qualify rank_ = 1
```

```sql
select
  customer_id,
  address,
  order_date,
  sum(total_price) over(partition by customer_id) as customer_revenue,
  sum(total_price) over(partition by address) as address_id,
  sum(total_price) over(partition by customer_id order by order_date) as
customer_cum_revenue
from `case_study.customers`
join `case_study.orders`
  using(customer_id)
```

İleri Konular

- Veri Manipülasyonu:
  - CREATE (Veri Yaratma)
  - INSERT INTO (Veri Ekleme)
  - UPDATE (Veri Güncelleme)
  - DELETE FROM (Veri Silme)
- Tarih ve Saat Fonksiyonları:
  - CURRENT_TIMESTAMP
  - EXTRACT
  - DATEPART
  - DATE_TRUNC
  - DATE_DIFF
  - DATE_ADD
- Metin (String) Fonksiyonları:
  - CONCAT
  - SUBSTRING/SUBSTR
  - LENGTH
  - UPPER/LOWER
  - TRIM
- Birlik İşlemleri:
  - UNION (ALL/DISTINCT)
  - INTERSECT
  - EXCEPT

SQL Öğrenimini Pekiştirme Yolları

1. İnteraktif eğitim sitelerini değerlendirin.
2. Mülakat odaklı, gerçek hayata yakın zorluktaki SQL problemleri çözün
3. Gerçek dünya verisiyle Çalışın
4. İleri SQL fonksiyonlarını araştırın

Kaynaklar

- W3School: https://www.w3schools.com/sql/
- Hackerrank: https://www.hackerrank.com/domains/sql
- LeetCode: https://leetcode.com/problemset/
- Datacamp: https://www.datacamp.com/
- Kaggle: https://www.kaggle.com/learn/advanced-sql

Dinlediğiniz için Teşekkür Ederim

linkedin.com/in/ozge-usta

medium.com/@ousta15