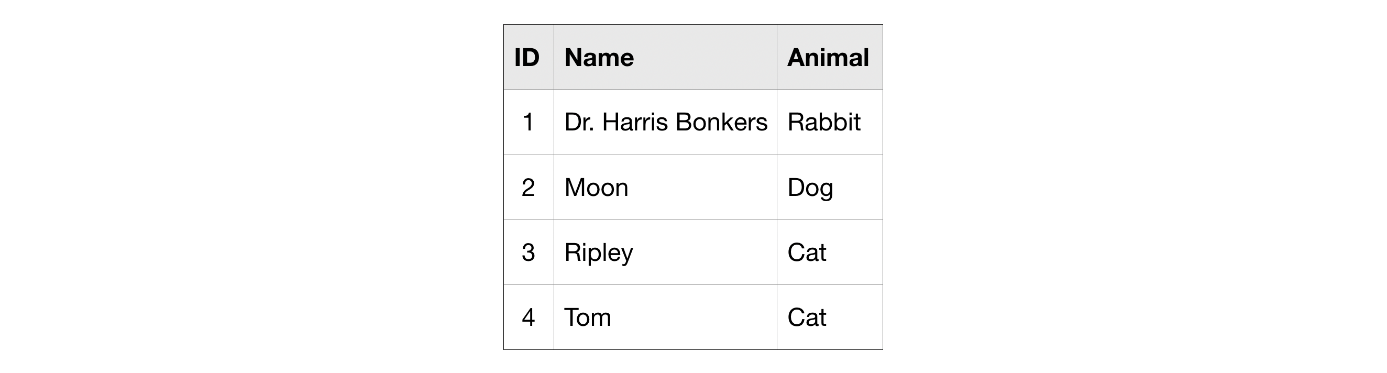
Introduction

Artık ham veriyi seçebildiğinize göre, verilerinizi gruplandırmayı ve bu gruplar içindeki öğeleri saymayı öğrenmeye hazırsınız. Bu, aşağıdaki gibi soruları yanıtlamanıza yardımcı olabilir:

* Mağazamız her bir meyve türünden kaç tane sattı?
* Veteriner kliniği kaç hayvan türünü tedavi etti?

Bunu yapmak için üç yeni teknik hakkında bilgi edineceksiniz: **GROUP BY**, **HAVING** ve **COUNT()**. Yine, evcil hayvanlar hakkındaki bu hayali bilgi tablosunu kullanacağız.



# COUNT()

Adından da tahmin edebileceğiniz gibi, **COUNT()** bir şeylerin sayısını döndürür. Eğer ona bir sütun adı verirseniz, o sütundaki girdi sayısını döndürür.

Örneğin, **pets** tablosundaki **ID** sütununun **COUNT()** değerini **SELECT** yaparsak, tabloda 4 adet ID olduğu için 4 değerini döndürecektir.



**COUNT()** , birçok değer alıp tek bir değer döndüren bir **toplama işlevi (aggregate function)** örneğidir. (Diğer toplama işlevi örnekleri arasında **SUM()**, **AVG()**, **MIN()** ve **MAX()** bulunur.) Yukarıdaki resimde de fark edeceğiniz gibi, toplama işlevleri garip sütun adları (**f0\_\_** gibi) oluşturur. Bu eğitimde ilerleyen kısımlarda, adı daha açıklayıcı bir şeyle nasıl değiştireceğinizi öğreneceksiniz.

# GROUP BY[¶](https://www.kaggle.com/code/dansbecker/group-by-having-count#GROUP-BY)

**GROUP BY**, bir veya daha fazla sütunun adını alır ve **COUNT()** gibi toplama işlevlerini uyguladığınızda, o sütunda aynı değere sahip tüm satırları tek bir grup olarak ele alır.

Örneğin, **pets** tablosunda her bir hayvan türünden kaç tane olduğunu öğrenmek istediğimizi varsayalım. **Animal** sütununda aynı değere sahip satırları bir araya getirmek için **GROUP BY**'ı, her grupta kaç **ID** olduğunu bulmak için ise **COUNT()**'u kullanabiliriz.

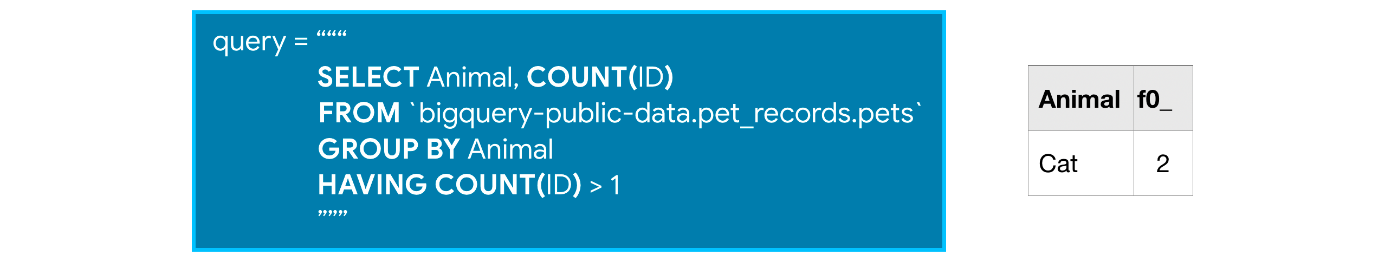


Üç satırdan (her biri farklı bir hayvan için) oluşan bir tablo döndürüyor. Evcil hayvanlar tablosunun 1 tavşan, 1 köpek ve 2 kedi içerdiğini görebiliyoruz.

# GROUP BY ... HAVING[¶](https://www.kaggle.com/code/dansbecker/group-by-having-count#GROUP-BY-...-HAVING)

**HAVING**, belirli kriterleri karşılamayan grupları dikkate almamak için **GROUP BY** ile birlikte kullanılır.

Yani, örneğin bu sorgu, yalnızca içinde birden fazla ID bulunan grupları dahil edecektir.



Belirtilen kriteri yalnızca bir grup karşıladığı için sorgu yalnızca bir satır içeren bir tablo döndürecektir.

# Example: Which Hacker News comments generated the most discussion?[¶](https://www.kaggle.com/code/dansbecker/group-by-having-count#Example:-Which-Hacker-News-comments-generated-the-most-discussion?)

Gerçek bir veri seti üzerinde bir örnek görmeye hazır mısınız? Hacker News veri seti, Hacker News sosyal ağ sitesindeki hikayeler ve yorumlar hakkında bilgiler içerir.

**full** tablosuyla çalışacağız ve ilk birkaç satırı yazdırarak başlayacağız. (İlgili kodu gizledik. Göz atmak için aşağıdaki "Code" düğmesine tıklayın.)

from google.cloud import bigquery

*# Create a "Client" object*

client = bigquery.Client()

*# Construct a reference to the "hacker\_news" dataset*

dataset\_ref = client.dataset("hacker\_news", project="bigquery-public-data")

*# API request - fetch the dataset*

dataset = client.get\_dataset(dataset\_ref)

*# Construct a reference to the "full" table*

table\_ref = dataset\_ref.table("full")

*# API request - fetch the table*

table = client.get\_table(table\_ref)

*# Preview the first five lines of the table*

client.list\_rows(table, max\_results=5).to\_dataframe()

Using Kaggle's public dataset BigQuery integration.

| title | url | text | dead | by | score | time | timestamp | type | id | parent | descendants | ranking | deleted |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | None | None | If the crocodile looked him up on Google, we b... | None | raxxorrax | NaN | 1633421535 | 2021-10-05 08:12:15+00:00 | comment | 28756662 | 28750122 | NaN | NaN | None |
| 1 | None | None | What exactly are you looking for? I think Pyto... | None | abiro | NaN | 1569141387 | 2019-09-22 08:36:27+00:00 | comment | 21040311 | 21040141 | NaN | NaN | None |
| 2 | None | None | Ironically, this very project might help out w... | None | mjevans | NaN | 1505769703 | 2017-09-18 21:21:43+00:00 | comment | 15279716 | 15276626 | NaN | NaN | None |
| 3 | None | None | As you start to gain some experience it can be... | None | every\_other | NaN | 1538575027 | 2018-10-03 13:57:07+00:00 | comment | 18130207 | 18128477 | NaN | NaN | None |
| 4 | None | None | That’s what I was referring to, yes. I heard o... | None | manmal | NaN | 1615664155 | 2021-03-13 19:35:55+00:00 | comment | 26449260 | 26449237 | NaN | NaN | None |

Tabloyu kullanarak hangi yorumların en çok yanıt aldığını görelim. Şöyle ki:

* **parent** sütunu hangi yoruma yanıt verildiğini gösterdiği için ve
* **id** sütunu her yorumu tanımlamak için kullanılan benzersiz kimliğe sahip olduğu için,

belirli bir yoruma yanıt olarak yapılan yorum sayısını bulmak için **parent** sütununa göre **GROUP BY** yapabilir ve **id** sütununu **COUNT()** ile sayabiliriz. (Bu hemen mantıklı gelmeyebilir; her şeyin net olduğundan emin olmak için bu kısma zaman ayırın!)

Ayrıca, yalnızca popüler yorumlarla ilgilendiğimiz için, ondan fazla yanıtı olan yorumlara bakacağız. Bu nedenle, yalnızca ondan fazla **ID**'ye sahip grupları **HAVING** ile döndüreceğiz.

*# Query to select comments that received more than 10 replies*

query\_popular = """

SELECT parent, COUNT(id)

FROM `bigquery-public-data.hacker\_news.full`

GROUP BY parent

HAVING COUNT(id) > 10

"""

Sorgumuz hazır olduğuna göre, çalıştıralım ve sonuçları bir pandas DataFrame'ine kaydedelim:

*# Set up the query (cancel the query if it would use too much of*

*# your quota, with the limit set to 10 GB)*

safe\_config = bigquery.QueryJobConfig(maximum\_bytes\_billed=10\*\*10)

query\_job = client.query(query\_popular, job\_config=safe\_config)

*# API request - run the query, and convert the results to a pandas DataFrame*

popular\_comments = query\_job.to\_dataframe()

*# Print the first five rows of the DataFrame*

popular\_comments.head()

| parent | f0\_ |
| --- | --- |
| 0 | 29934192.0 | 175 |
| 1 | 3373702.0 | 75 |
| 2 | 24066748.0 | 45 |
| 3 | 31123102.0 | 42 |
| 4 | 9996333.0 | 754 |

popular\_comments veri çerçevesindeki her satır, ondan fazla yanıt alan bir yoruma karşılık gelir. Örneğin, 801208 kimlikli yorum 56 yanıt almıştır.

# Aliasing and other improvements[¶](https://www.kaggle.com/code/dansbecker/group-by-having-count" \l "Aliasing-and-other-improvements" \t "_self)

Sorgularınızı daha da iyi hale getirmek için birkaç ipucu:

* **COUNT(id)**'den dönen sütunun adı **f0\_\_** idi. Bu, pek açıklayıcı bir isim değil. Toplama işlemini belirttikten sonra **AS NumPosts** ekleyerek bu ismi değiştirebilirsiniz. Buna **takma ad verme (aliasing)** denir ve ileriki derslerde daha ayrıntılı olarak ele alınacaktır.
* **COUNT()** fonksiyonunun içine ne koyacağınızdan emin olamadığınız zamanlarda, her gruptaki satırları saymak için **COUNT(1)** kullanabilirsiniz. Çoğu insan bunu özellikle daha anlaşılır bulur, çünkü başka sütunlara odaklanmadığını biliriz. Ayrıca, sütun adları sağlanmasıyla kıyaslandığında daha az veri tarar (bu da daha hızlı olmasını ve veri erişim kotanızdan daha az harcamanızı sağlar).

Bu ipuçlarını kullanarak sorgumuzu yeniden yazabiliriz:

*# Improved version of earlier query, now with aliasing & improved readability*

query\_improved = """

SELECT parent, COUNT(1) AS NumPosts

FROM `bigquery-public-data.hacker\_news.full`

GROUP BY parent

HAVING COUNT(1) > 10

"""

safe\_config = bigquery.QueryJobConfig(maximum\_bytes\_billed=10\*\*10)

query\_job = client.query(query\_improved, job\_config=safe\_config)

*# API request - run the query, and convert the results to a pandas DataFrame*

improved\_df = query\_job.to\_dataframe()

*# Print the first five rows of the DataFrame*

improved\_df.head()

| parent | NumPosts |
| --- | --- |
| 0 | 24397272.0 | 87 |
| 1 | 28875764.0 | 43 |
| 2 | 30487933.0 | 40 |
| 3 | 27890790.0 | 96 |
| 4 | 19678914.0 | 48 |

Artık istediğin verilere sahipsin ve bunların tanımlayıcı isimleri var. Bu iyi bir stil.

# Note on using ****GROUP BY****[¶](https://www.kaggle.com/code/dansbecker/group-by-having-count#Note-on-using-GROUP-BY)

Toplama fonksiyonlarını (**COUNT()** gibi) SQL'e nasıl uygulayacağını söylediği için, **GROUP BY**'ı bir toplama fonksiyonu olmadan kullanmak mantıklı değildir. Benzer şekilde, herhangi bir **GROUP BY** ifadesi varsa, tüm değişkenler ya bir **GROUP BY** komutuna ya da bir toplama fonksiyonuna aktarılmalıdır.

Aşağıdaki sorguyu inceleyelim:

query\_good = """

SELECT parent, COUNT(id)

FROM `bigquery-public-data.hacker\_news.full`

GROUP BY parent

"""

Dikkat edin, iki değişken var: **parent** ve **id**.

* **parent**, bir **GROUP BY** komutuna (**GROUP BY parent** içinde) aktarılmış, ve
* **id**, bir toplama fonksiyonuna (**COUNT(id)** içinde) aktarılmış.

Ve bu sorgu çalışmaz, çünkü **author** sütunu bir toplama fonksiyonuna veya bir **GROUP BY** ifadesine aktarılmamıştır:

query\_bad = """

SELECT `by` AS author, parent, COUNT(id)

FROM `bigquery-public-data.hacker\_news.full`

GROUP BY parent

"""

Bu hatayı yaptığınızda, **"SELECT list expression references column (column's name) which is neither grouped nor aggregated at."** şeklinde bir hata mesajı alırsınız.

Bu sorgudaki by sütununun ters tırnak içine alındığını fark edebilirsiniz. Bunun nedeni, **BY**'ın **GROUP BY** gibi ifadelerde kullanılan ayrılmış bir anahtar kelime olmasıdır. BigQuery'de, tanımlayıcı olarak kullanılan ayrılmış anahtar kelimelerin hata vermesini önlemek için ters tırnak içine alınması gerekir. Ayrıca, bu sütuna daha sonra yapılacak referansları daha okunabilir hale getirmek için, sütunu **author** olarak yeniden adlandırmak üzere bir **takma ad (alias)** ekledik.