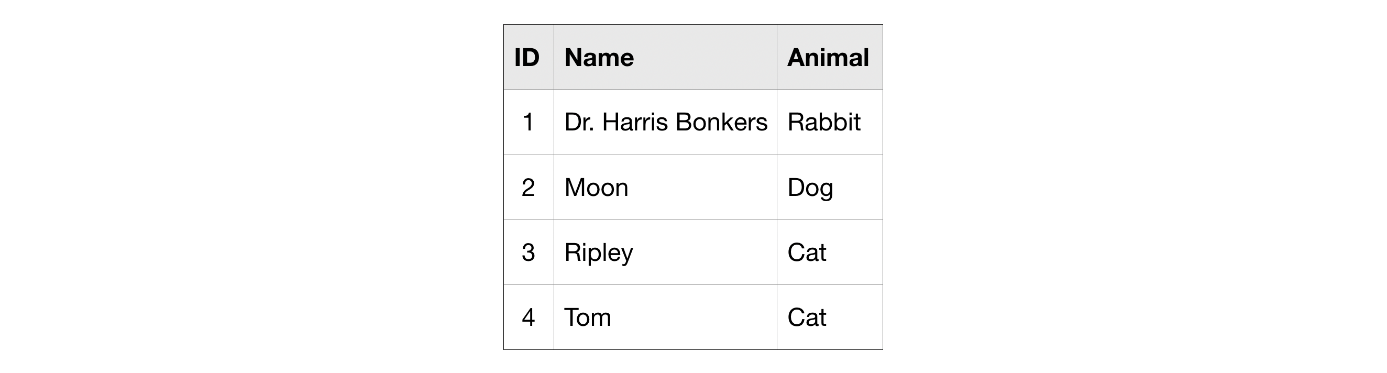
Introduction

Artık bir veri setine nasıl erişeceğinizi ve onu nasıl inceleyeceğinizi bildiğinize göre, ilk SQL sorgunuzu yazmaya hazırsınız! Yakında göreceğiniz gibi, SQL sorguları çok büyük bir veri setini ayıklamanıza ve yalnızca ihtiyacınız olan bilgileri almanıza yardımcı olacaktır.

Belirlediğiniz koşullara göre belirli sütunlardan veri almak için **SELECT**, **FROM** ve **WHERE** anahtar kelimelerini kullanarak başlayacağız.

Açıklık sağlamak için, yalnızca **pets** adlı bir tablo içeren küçük, hayali bir veri seti olan **pet\_records** ile çalışacağız.



# SELECT ... FROM

En temel SQL sorgusu, tek bir tablodan tek bir sütun seçer. Bunu yapmak için:

* **SELECT** kelimesinden sonra istediğiniz sütunu belirtin ve
* **FROM** kelimesinden sonra tabloyu belirtin.

Örneğin, **Name** sütununu (**bigquery-public-data** projesindeki **pet\_records** veritabanında yer alan **pets** tablosundan) seçmek için sorgumuz şu şekilde olacaktır:



Bir SQL sorgusu yazarken FROM'a geçirdiğimiz argümanın tek veya çift tırnak (' veya ") içinde olmadığını unutmayın. Ters tırnak (`) içindedir.

# WHERE ...

BigQuery veri setleri büyük olduğundan, genellikle yalnızca belirli koşulları karşılayan satırları döndürmek istersiniz. Bunu **WHERE** ifadesini kullanarak yapabilirsiniz.

Aşağıdaki sorgu, **Animal** sütununun **'Cat'** metnini içerdiği satırlardaki **Name** sütununa ait girdileri döndürür.



# Example: What are all the U.S. cities in the OpenAQ dataset?

Artık temelleri öğrendiğinize göre, gerçek bir veri setiyle bir örnek üzerinden gidelim. Hava kalitesiyle ilgili bir **OpenAQ** veri setini kullanacağız.

İlk olarak, sorguları çalıştırmak için ihtiyacımız olan her şeyi ayarlayacak ve veri tabanımızda hangi tabloların olduğuna hızlıca göz atacağız. (Bunu önceki derste nasıl yapacağınızı öğrendiğiniz için, kodu gizledik. Ancak göz atmak isterseniz, aşağıdaki "Code" düğmesine tıklamanız yeterlidir.)

from google.cloud import bigquery

*# Create a "Client" object*

client = bigquery.Client()

*# Construct a reference to the "openaq" dataset*

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

*# API request - fetch the dataset*

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

*# List all the tables in the "openaq" dataset*

tables = list(client.list\_tables(dataset))

*# Print names of all tables in the dataset (there's only one!)*

for table **in** tables:

print(table.table\_id)

Using Kaggle's public dataset BigQuery integration.

global\_air\_quality

Veri kümesi yalnızca global\_air\_quality adlı bir tablo içeriyor. Tabloyu alıp ilk birkaç satırına göz atarak ne tür veriler içerdiğini göreceğiz. (Yine, kodu gizledik. Göz atmak için aşağıdaki "Kod" butonuna tıklayın.)

*# Construct a reference to the "global\_air\_quality" table*

table\_ref = dataset\_ref.table("global\_air\_quality")

*# API request - fetch the table*

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

*# Preview the first five lines of the "global\_air\_quality" table*

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

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:8: UserWarning: Cannot use bqstorage\_client if max\_results is set, reverting to fetching data with the tabledata.list endpoint

| location | city | country | pollutant | value | timestamp | unit | source\_name | latitude | longitude | averaged\_over\_in\_hours | location\_geom |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Borówiec, ul. Drapałka | Borówiec | PL | bc | 0.85217 | 2022-04-28 07:00:00+00:00 | µg/m³ | GIOS | 1.0 | 52.276794 | 17.074114 | POINT(52.276794 1) |
| 1 | Kraków, ul. Bulwarowa | Kraków | PL | bc | 0.91284 | 2022-04-27 23:00:00+00:00 | µg/m³ | GIOS | 1.0 | 50.069308 | 20.053492 | POINT(50.069308 1) |
| 2 | Płock, ul. Reja | Płock | PL | bc | 1.41000 | 2022-03-30 04:00:00+00:00 | µg/m³ | GIOS | 1.0 | 52.550938 | 19.709791 | POINT(52.550938 1) |
| 3 | Elbląg, ul. Bażyńskiego | Elbląg | PL | bc | 0.33607 | 2022-05-03 13:00:00+00:00 | µg/m³ | GIOS | 1.0 | 54.167847 | 19.410942 | POINT(54.167847 1) |
| 4 | Piastów, ul. Pułaskiego | Piastów | PL | bc | 0.51000 | 2022-05-11 05:00:00+00:00 | µg/m³ | GIOS | 1.0 | 52.191728 | 20.837489 | POINT(52.191728 1) |

Her şey yolunda görünüyor! Öyleyse bir sorgu oluşturalım. Diyelim ki şehir sütununda, ülke sütununda "ABD" ("Amerika Birleşik Devletleri") olan satırlardaki tüm değerleri seçmek istiyoruz.

*# Query to select all the items from the "city" column where the "country" column is 'US'*

query = """

SELECT city

FROM `bigquery-public-data.openaq.global\_air\_quality`

WHERE country = 'US'

"""

Şimdi bu sorgunun yukarıda öğrendiklerinizle uyumlu olduğundan emin olmak için zaman ayırın.

# Submitting the query to the dataset[¶](https://www.kaggle.com/code/dansbecker/select-from-where" \l "Submitting-the-query-to-the-dataset" \t "_self)

Bu sorguyu kullanarak OpenAQ veri kümesinden bilgi almaya hazırız. Önceki eğitimde olduğu gibi, ilk adım bir İstemci nesnesi oluşturmaktır.

*# Create a "Client" object*

client = bigquery.Client()

Using Kaggle's public dataset BigQuery integration.

Sorguyu query() metoduyla ayarlayarak başlıyoruz. Metodu varsayılan parametrelerle çalıştırıyoruz, ancak bu metot aynı zamanda dokümanlarda okuyabileceğiniz daha karmaşık ayarlar belirlememize de olanak tanıyor. Bu konuya daha sonra tekrar değineceğiz.

*# Set up the query*

query\_job = client.query(query)

Daha sonra sorguyu çalıştırıp sonuçları pandas DataFrame'e dönüştürüyoruz.

*# API request - run the query, and return a pandas DataFrame*

us\_cities = query\_job.to\_dataframe()

/opt/conda/lib/python3.7/site-packages/google/cloud/bigquery/client.py:440: UserWarning: Cannot create BigQuery Storage client, the dependency google-cloud-bigquery-storage is not installed.

"Cannot create BigQuery Storage client, the dependency "

Şimdi us\_cities adında bir pandas DataFrame'imiz var, bunu da diğer DataFrame'ler gibi kullanabiliriz.

*# What five cities have the most measurements?*

us\_cities.city.value\_counts().head()

Phoenix-Mesa-Scottsdale 39414

Los Angeles-Long Beach-Santa Ana 27479

Riverside-San Bernardino-Ontario 26887

New York-Northern New Jersey-Long Island 25417

San Francisco-Oakland-Fremont 22710

Name: city, dtype: int64

# More queries

Birden fazla sütun istiyorsanız, adları arasına virgül koyarak seçebilirsiniz:

query = """

SELECT city, country

FROM `bigquery-public-data.openaq.global\_air\_quality`

WHERE country = 'US'

"""

\* işareti olan tüm sütunları şu şekilde seçebilirsiniz:

query = """

SELECT \*

FROM `bigquery-public-data.openaq.global\_air\_quality`

WHERE country = 'US'

"""

# Q&A: Notes on formatting[¶](https://www.kaggle.com/code/dansbecker/select-from-where" \l "Q&A:-Notes-on-formatting" \t "_self)

SQL sorgusunun biçimlendirmesi size yabancı gelebilir. Herhangi bir sorunuz varsa, bu sayfanın altındaki yorumlar bölümünden sorabilirsiniz. İşte iki yaygın sorunun yanıtları:

### **Question: What's up with the triple quotation marks (""")?**[**¶**](https://www.kaggle.com/code/dansbecker/select-from-where#Question:-What's-up-with-the-triple-quotation-marks-(%22%22%22)?)

Cevap: Bunlar, Python'a satır sonları olsa bile, içlerindeki her şeyin tek bir dize olduğunu söyler. Satır sonları gerekli değildir, ancak sorgunuzu okumayı kolaylaştırırlar.

### **Question: Do you need to capitalize SELECT and FROM?**

Cevap: Hayır, SQL büyük harf kullanımına önem vermez. Ancak, SQL komutlarınızı büyük harfle yazmak gelenekseldir ve bu, sorgularınızın okunmasını biraz daha kolaylaştırır.

# Working with big datasets

BigQuery veri setleri çok büyük olabilir. Size ücretsiz olarak çok sayıda hesaplama yapma imkanı sunuyoruz, ancak herkesin bir limiti vardır.

Her Kaggle kullanıcısı 30 günde bir 5 TB veriyi ücretsiz olarak tarayabilir. Bu sınıra ulaştığınızda, sıfırlanmasını beklemeniz gerekir.

**Şu anda Kaggle'daki en büyük veri seti** 3 TB'dir, bu yüzden dikkatli olmazsanız 30 günlük limitinizi birkaç sorguda bitirebilirsiniz.

Ancak endişelenmeyin: Sınırınızı aşmamak için bir kerede çok fazla veri taramaktan nasıl kaçınacağınızı size öğreteceğiz.

Başlamak için, herhangi bir sorguyu çalıştırmadan önce boyutunu tahmin edebilirsiniz. İşte (çok büyük olan!) Hacker News veri setini kullanan bir örnek. Bir sorgunun ne kadar veri tarayacağını görmek için, bir **QueryJobConfig** nesnesi oluştururuz ve **dry\_run** parametresini **True** olarak ayarlarız.

*# Query to get the score column from every row where the type column has value "job"*

query = """

SELECT score, title

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

WHERE type = "job"

"""

*# Create a QueryJobConfig object to estimate size of query without running it*

dry\_run\_config = bigquery.QueryJobConfig(dry\_run=True)

*# API request - dry run query to estimate costs*

dry\_run\_query\_job = client.query(query, job\_config=dry\_run\_config)

print("This query will process **{}** bytes.".format(dry\_run\_query\_job.total\_bytes\_processed))

This query will process 553320240 bytes.

Sorguyu çalıştırırken, taramak istediğiniz veri miktarını sınırlamak için bir parametre de belirleyebilirsiniz. İşte düşük limitli bir örnek.

*# Only run the query if it's less than 1 MB*

ONE\_MB = 1000\*1000

safe\_config = bigquery.QueryJobConfig(maximum\_bytes\_billed=ONE\_MB)

*# Set up the query (will only run if it's less than 1 MB)*

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

*# API request - try to run the query, and return a pandas DataFrame*

safe\_query\_job.to\_dataframe()

---------------------------------------------------------------------------

InternalServerError Traceback (most recent call last)

/tmp/ipykernel\_19/2063017411.py in <module>

**7**

**8** # API request - try to run the query, and return a pandas DataFrame

----> 9 safe\_query\_job.to\_dataframe()

/opt/conda/lib/python3.7/site-packages/google/cloud/bigquery/job.py in to\_dataframe(self, bqstorage\_client, dtypes, progress\_bar\_type, create\_bqstorage\_client, date\_as\_object)

**3403** ValueError: If the `pandas` library cannot be imported.

**3404** """

-> 3405 return self.result().to\_dataframe(

**3406** bqstorage\_client=bqstorage\_client,

**3407** dtypes=dtypes,

/opt/conda/lib/python3.7/site-packages/google/cloud/bigquery/job.py in result(self, page\_size, max\_results, retry, timeout, start\_index)

**3232** """

**3233** try:

-> 3234 super(QueryJob, self).result(retry=retry, timeout=timeout)

**3235**

**3236** # Return an iterator instead of returning the job.

/opt/conda/lib/python3.7/site-packages/google/cloud/bigquery/job.py in result(self, retry, timeout)

**819** self.\_begin(retry=retry, timeout=timeout)

**820** # TODO: modify PollingFuture so it can pass a retry argument to done().

--> 821 return super(\_AsyncJob, self).result(timeout=timeout)

**822**

**823** def cancelled(self):

/opt/conda/lib/python3.7/site-packages/google/api\_core/future/polling.py in result(self, timeout, retry)

**135** # pylint: disable=raising-bad-type

**136** # Pylint doesn't recognize that this is valid in this case.

--> 137 raise self.\_exception

**138**

**139** return self.\_result

InternalServerError: 500 Query exceeded limit for bytes billed: 1000000. 553648128 or higher required.

(job ID: f9f41eac-073d-4fa7-ab3a-504909a72e67)

-----Query Job SQL Follows-----

| . | . | . | . | . |

1:

2: SELECT score, title

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

4: WHERE type = "job"

5:

| . | . | . | . | . |

Bu durumda, 1 MB sınırı aşıldığı için sorgu iptal edildi. Ancak, sorguyu başarıyla çalıştırmak için sınırı artırabiliriz!

*# Only run the query if it's less than 1 GB*

ONE\_GB = 1000\*1000\*1000

safe\_config = bigquery.QueryJobConfig(maximum\_bytes\_billed=ONE\_GB)

*# Set up the query (will only run if it's less than 1 GB)*

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

*# API request - try to run the query, and return a pandas DataFrame*

job\_post\_scores = safe\_query\_job.to\_dataframe()

*# Print average score for job posts*

job\_post\_scores.score.mean()

1.7267060367454068