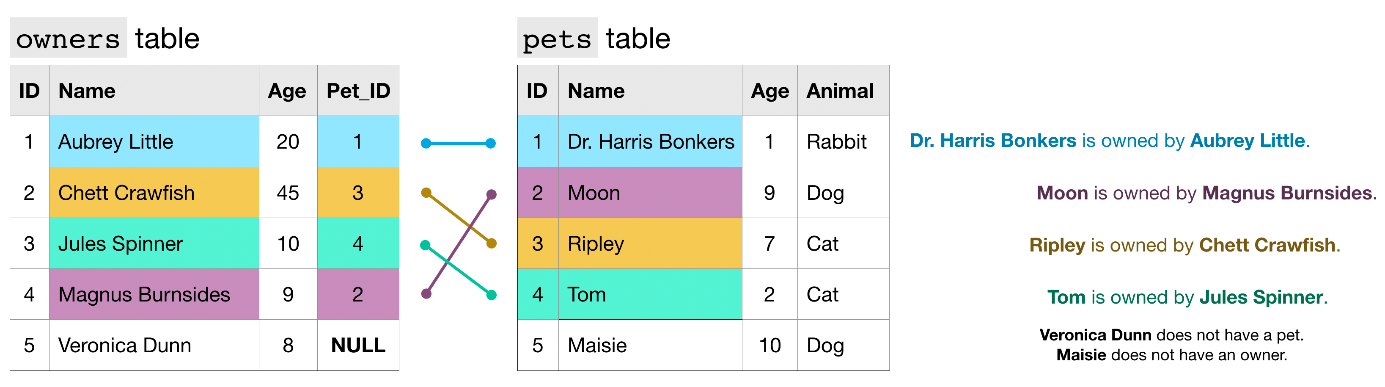
Introduction

SQL'e Giriş mikro-kursunda, iki farklı tablodaki bilgileri birleştirmek için **INNER JOIN** komutunu nasıl kullanacağınızı öğrendiniz. Şimdi, **UNION** komutunu kullanarak birden fazla tablodan bilgi çekmenin yanı sıra, birkaç farklı **JOIN** türü hakkında daha bilgi edineceksiniz.

## Bu süreçte, owners ve pets adlı iki hayali tabloyla çalışacağız.



**owners** tablosunun her satırı, farklı bir evcil hayvan sahibini temsil eder ve **ID** sütunu benzersiz bir tanımlayıcıdır. **Pet\_ID** sütunu (**owners** tablosunda), sahibine ait olan evcil hayvanın kimlik numarasını içerir (bu numara, **pets** tablosundaki evcil hayvanın kimlik numarasıyla eşleşir).

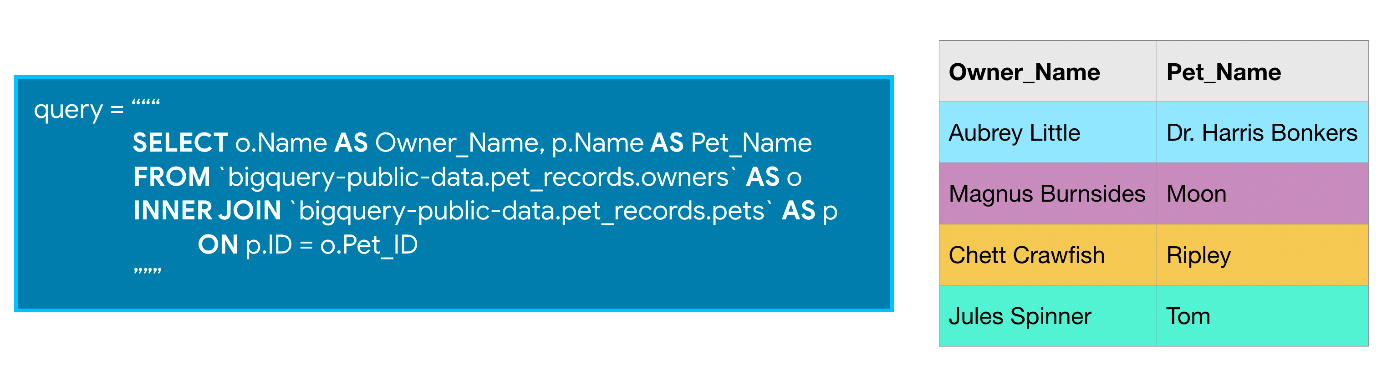
Örneğin;

* **pets** tablosu, Dr. Harris Bonkers'ın kimlik numarası 1 olan evcil hayvan olduğunu gösterir.
* **owners** tablosu, Aubrey Little'ın kimlik numarası 1 olan evcil hayvanın sahibi olduğunu gösterir.

Bu iki bilgiyi bir araya getirdiğimizde, Dr. Harris Bonkers'ın sahibi Aubrey Little'dır. Benzer şekilde, Veronica Dunn'ın karşılık gelen bir **Pet\_ID**'si olmadığı için onun bir evcil hayvanı yoktur. Ve 5, **Pet\_ID** sütununda görünmediği için Maisie'nin de bir sahibi yoktur.

# JOINs

Sahipler tablosundaki Pet\_ID sütunundaki değerin evcil hayvanlar tablosundaki ID sütununda eşleşmesi olan satırları her iki tablodan çekmek için INNER JOIN kullanabileceğimizi hatırlayın.



Bu durumda, Veronica Dunn ve Maisie sonuçlara dahil edilmez. Peki, bunun yerine sahipleri olsun olmasın tüm evcil hayvanları içeren bir tablo oluşturmak istersek ne olurdu? Ya da her iki tablodaki tüm satırları birleştirmek istersek? Bu durumlarda, sadece farklı bir **JOIN** türü kullanmamız gerekir.

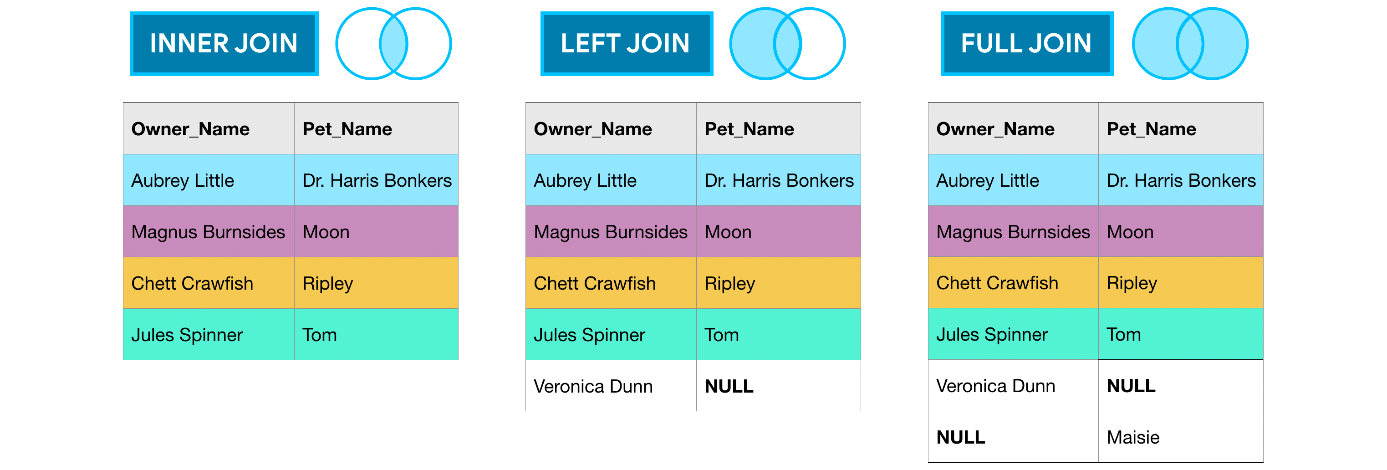
Örneğin, **owners** tablosundaki tüm satırları içeren bir tablo oluşturmak için bir **LEFT JOIN** kullanırız. Bu durumda, "sol" terimi sorgudaki **JOIN** komutundan önce gelen tabloyu ifade eder. ("Sağ" terimi ise **JOIN**'den sonraki tabloyu ifade eder.)



Yukarıdaki sorguda **INNER JOIN** yerine **LEFT JOIN** kullanmak, her iki tabloda eşleşen girdilerin olduğu tüm satırları ve ayrıca sol tablodaki tüm satırları (eşleşme olsun ya da olmasın) döndürür.

Bunun yerine **RIGHT JOIN** kullanırsak, eşleşen satırları ve sağ tablodaki tüm satırları (eşleşme olsun ya da olmasın) alırız.

Son olarak, **FULL JOIN** her iki tablodaki tüm satırları döndürür. Şunu unutmayın ki, her iki tabloda da eşleşmesi olmayan herhangi bir satır, eksik değerler için NULL girdilere sahip olacaktır. Bunu aşağıdaki görselde görebilirsiniz.



# UNIONs[¶](https://www.kaggle.com/code/alexisbcook/joins-and-unions" \l "UNIONs" \t "_self)

Gördüğünüz gibi, **JOIN**'ler farklı tablolardaki sonuçları yatay olarak birleştirir. Eğer bunun yerine sütunları dikey olarak birleştirmek isterseniz, bunu bir **UNION** ile yapabilirsiniz. Aşağıdaki örnek sorgu, her iki tablodaki **Age** sütunlarını birleştirir.



**UNION** komutuyla, her iki sütunun da veri tiplerinin aynı olması gerektiğini, ancak sütun adlarının farklı olabileceğini unutmayın. (Yani, örneğin **owners** tablosundaki **Age** sütunu ile **pets** tablosundaki **Pet\_Name** sütununun **UNION**'ını alamayız.)

Tekrarlanan değerleri dahil etmek için **UNION ALL** kullanırız. **9** sayısının hem **owners** tablosunda hem de **pets** tablosunda yer aldığını ve birleştirilmiş sonuçlarda iki kez göründüğünü fark edeceksiniz. Eğer tekrarlanan değerleri çıkarmak isterseniz, sorgudaki **UNION ALL** komutunu **UNION DISTINCT** olarak değiştirmeniz yeterlidir.

# Example[¶](https://www.kaggle.com/code/alexisbcook/joins-and-unions" \l "Example" \t "_self)

Hacker News veri setiyle çalışacağız. Yorumlar tablosunun ilk birkaç satırını inceleyerek başlayacağız. (İlgili kod gizlidir, ancak aşağıdaki "Kod" butonuna tıklayarak onu tekrar açabilirsiniz.)

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 "comments" table*

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

*# 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.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:19: UserWarning: Cannot use bqstorage\_client if max\_results is set, reverting to fetching data with the tabledata.list endpoint.

| d | by | author | time | time\_ts | text | parent | deleted | dead | ranking |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 9734136 | None | None | 1434565400 | 2015-06-17 18:23:20+00:00 | None | 9733698 | True | None | 0 |
| 1 | 4921158 | None | None | 1355496966 | 2012-12-14 14:56:06+00:00 | None | 4921100 | True | None | 0 |
| 2 | 7500568 | None | None | 1396261158 | 2014-03-31 10:19:18+00:00 | None | 7499385 | True | None | 0 |
| 3 | 8909635 | None | None | 1421627275 | 2015-01-19 00:27:55+00:00 | None | 8901135 | True | None | 0 |
| 4 | 9256463 | None | None | 1427204705 | 2015-03-24 13:45:05+00:00 | None | 9256346 | True | None | 0 |

Hikayeler tablosuyla da çalışacaksınız.

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

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

*# 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()

/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.

SQL'e Giriş mikro-kursundan **JOIN**'lere zaten aşina olduğunuz için, bu bölümde **ortak tablo ifadesi (CTE)** kullanan nispeten karmaşık bir JOIN örneğiyle çalışacağız.

Aşağıdaki sorgu, **stories** ve **comments** tablolarındaki bilgileri birleştirerek 1 Ocak 2012'de paylaşılan tüm hikayeleri ve bunlara gelen yorum sayısını gösteren bir tablo oluşturur. Sonuçların hiç yorum almamış hikayeleri de içermesi için **LEFT JOIN** kullanıyoruz.

*# Query to select all stories posted on January 1, 2012, with number of comments*

join\_query = """

WITH c AS

(

SELECT parent, COUNT(\*) as num\_comments

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

GROUP BY parent

)

SELECT s.id as story\_id, s.by, s.title, c.num\_comments

FROM `bigquery-public-data.hacker\_news.stories` AS s

LEFT JOIN c

ON s.id = c.parent

WHERE EXTRACT(DATE FROM s.time\_ts) = '2012-01-01'

ORDER BY c.num\_comments DESC

"""

*# Run the query, and return a pandas DataFrame*

join\_result = client.query(join\_query).result().to\_dataframe()

join\_result.head()

/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 "

| story\_id | by | title | num\_comments |
| --- | --- | --- | --- |
| 0 | 3412900 | whoishiring | Ask HN: Who is Hiring? (January 2012) | 154.0 |
| 1 | 3412901 | whoishiring | Ask HN: Freelancer? Seeking freelancer? (Janua... | 97.0 |
| 2 | 3412643 | jemeshsu | Avoid Apress | 30.0 |
| 3 | 3414012 | ramanujam | Impress.js - a Prezi like implementation using... | 27.0 |
| 4 | 3412891 | Brajeshwar | There's no shame in code that is simply "good ... | 27.0 |

Sonuçlar num\_comments sütununa göre sıralandığından, yorumu olmayan hikayeler DataFrame'in sonunda görünür. (NaN'ın "sayı değil" anlamına geldiğini unutmayın.)

*# None of these stories received any comments*

join\_result.tail()

| story\_id | by | title | num\_comments |
| --- | --- | --- | --- |
| 439 | 3413041 | ORioN63 | Solar days, sidereal days, solar years and sid... | NaN |
| 440 | 3412667 | Tez\_Dhar | How shall i Learn Hacking | NaN |
| 441 | 3412783 | mmaltiar | Working With Spring Data JPA | NaN |
| 442 | 3412821 | progga | Networking on the Network: A Guide to Professi... | NaN |
| 443 | 3412930 | shipcode | Project Zero Operating System – New Kernel | NaN |

Daha sonra, 1 Ocak 2014'te hikaye veya yorum yazan kullanıcılara karşılık gelen tüm kullanıcı adlarını seçmek için bir sorgu yazıyoruz. Her kullanıcının tabloda en fazla bir kez görünmesini sağlamak için UNION ALL yerine UNION DISTINCT kullanıyoruz.

*# Query to select all users who posted stories or comments on January 1, 2014*

union\_query = """

SELECT c.by

FROM `bigquery-public-data.hacker\_news.comments` AS c

WHERE EXTRACT(DATE FROM c.time\_ts) = '2014-01-01'

UNION DISTINCT

SELECT s.by

FROM `bigquery-public-data.hacker\_news.stories` AS s

WHERE EXTRACT(DATE FROM s.time\_ts) = '2014-01-01'

"""

*# Run the query, and return a pandas DataFrame*

union\_result = client.query(union\_query).result().to\_dataframe()

union\_result.head()

/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 "

| by |
| --- |
| 0 | learnlivegrow |
| 1 | egybreak |
| 2 | dclara |
| 3 | vram22 |
| 4 | espeed |

1 Ocak 2014'te mesaj gönderen kullanıcı sayısını bulmak için sadece DataFrame'in uzunluğunu almamız yeterli.

*# Number of users who posted stories or comments on January 1, 2014*

len(union\_result)

2282