Marketing Analytics – Lab 2

The Very Basic Concepts

Database: A database is a collection of related data that is organized and stored in a structured manner. For example, a company might have a database of employee information, including names, addresses, and salaries.

Table: A table is a collection of data stored in rows and columns. Each column represents a field or attribute of the data, and each row represents a record. For example, a company might have a table of employee data with columns like "EmployeeID", "Name", "Department", and "Salary".

Query: A query is a request for data from a database. It is used to retrieve, modify, or manipulate data in a database. For example, a query might be used to retrieve all employees who work in the IT department.

The Very Basic Syntax

SELECT…FROM

WHERE…

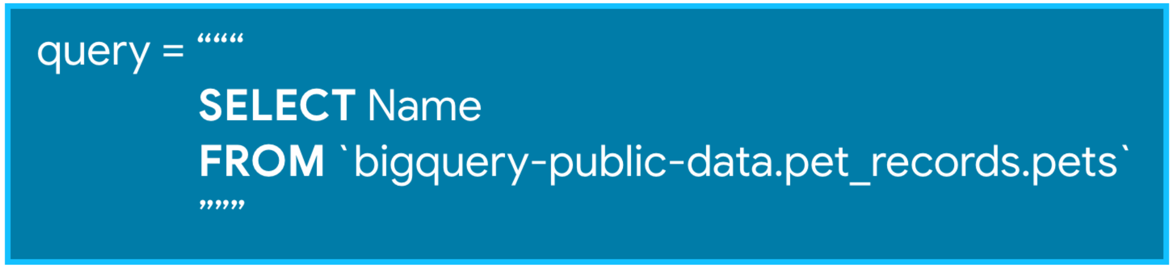
COUNT()

GROUP BY

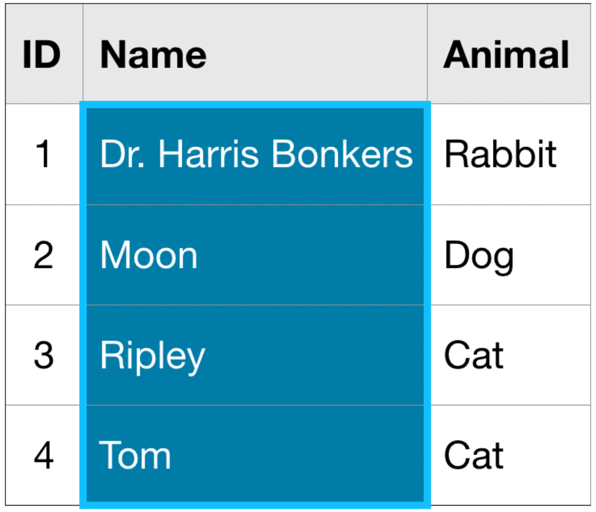
GROUP BY… HAVING

ORDER BY

SELECT... FROM

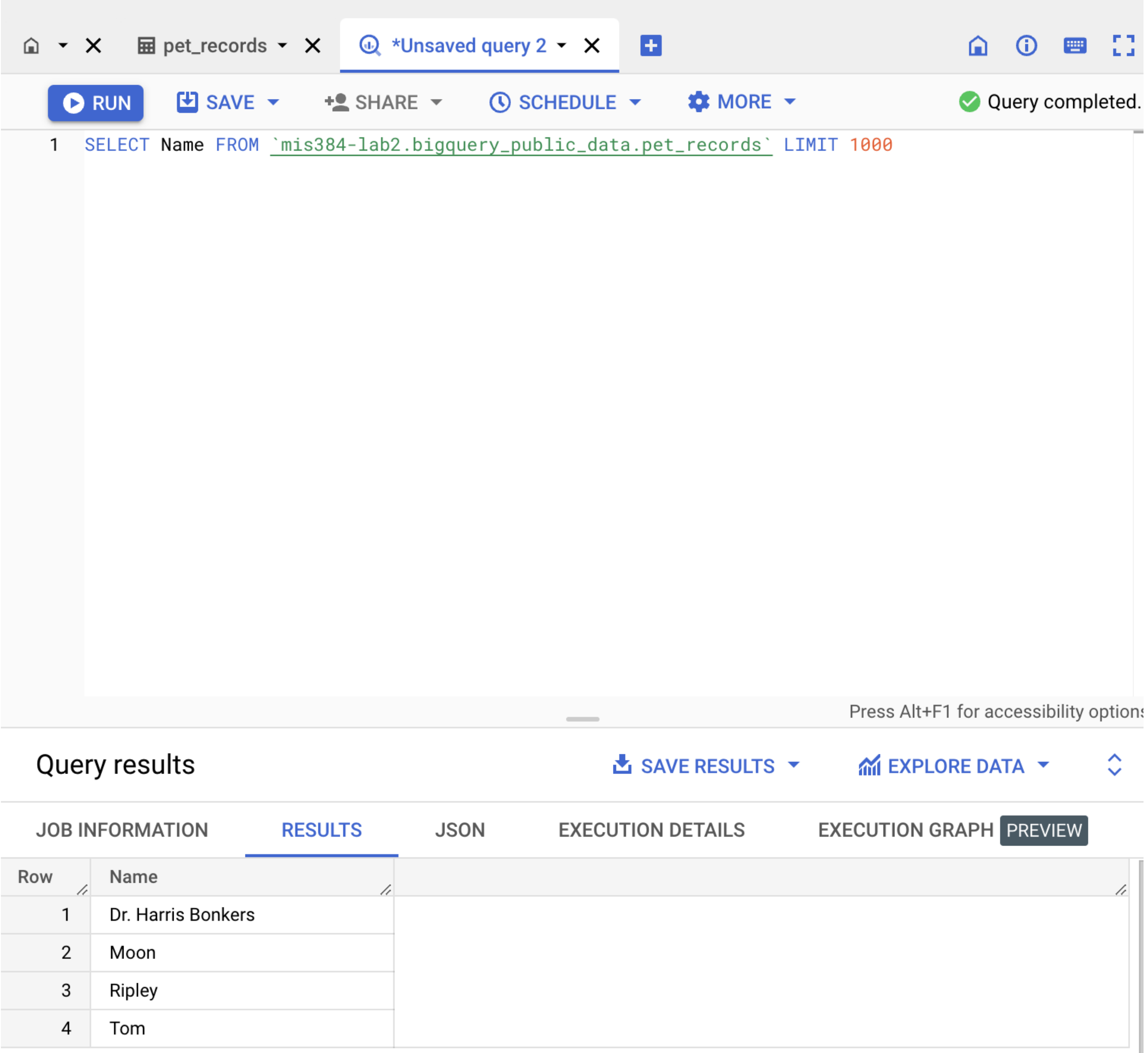


```
query = “““
          SELECT Name
          FROM `bigquery-public-data.pet_records.pets`
          ”””
```



ID	Name	Animal
1	Dr. Harris Bonkers	Rabbit
2	Moon	Dog
3	Ripley	Cat
4	Tom	Cat

Note that when writing an SQL query, the argument we pass to **FROM** is *not* in single or double quotation marks (' or "). It is in backticks (`).

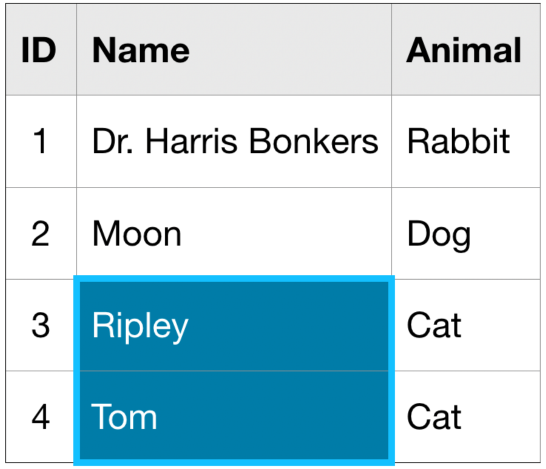


```
SELECT Name FROM `mis384-lab2.bigquery_public_data.pet_records` LIMIT 1000
```

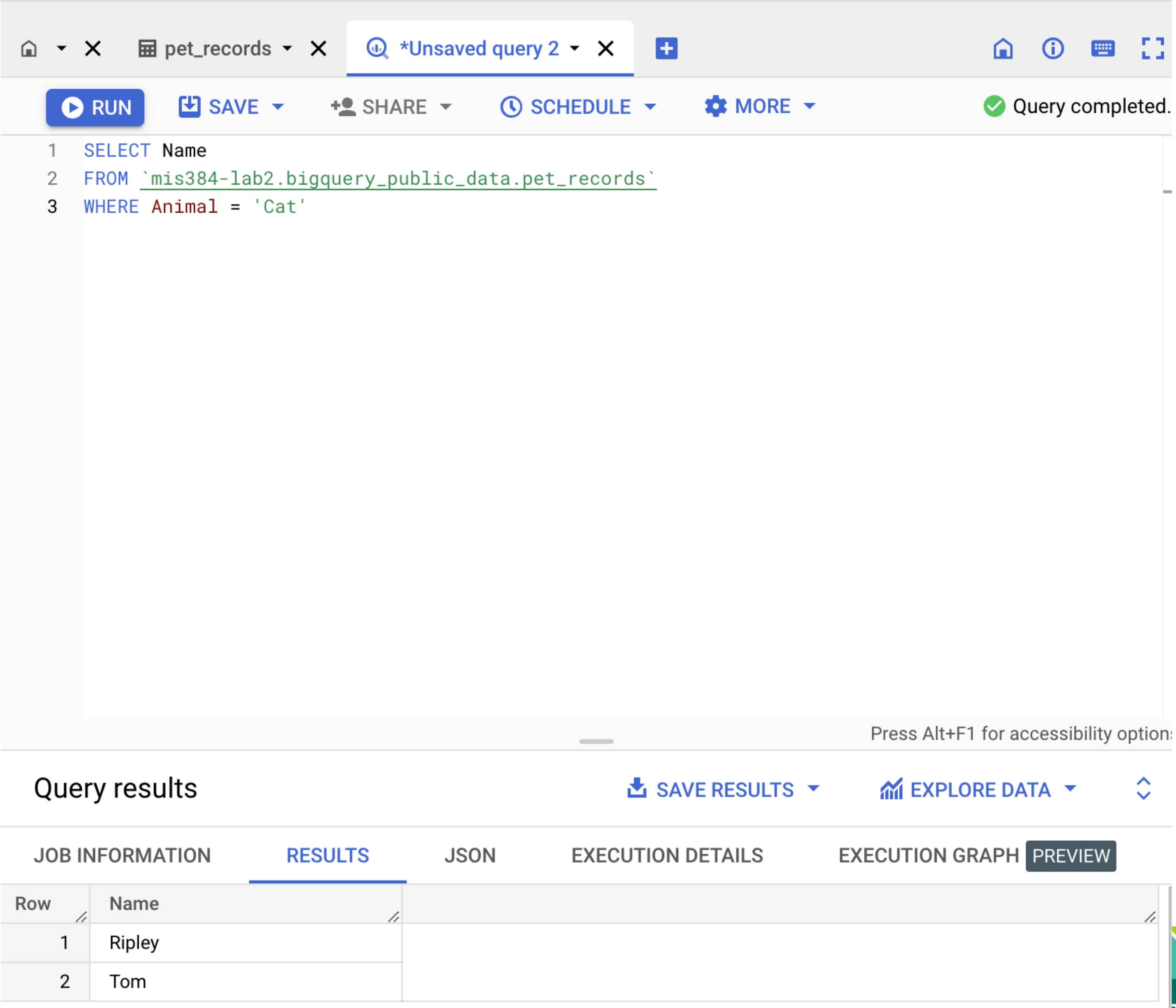
Row	Name
1	Dr. Harris Bonkers
2	Moon
3	Ripley
4	Tom

WHERE …

```
query = “““
        SELECT Name
        FROM `bigquery-public-data.pet_records.pets`
        WHERE Animal = 'Cat'
        ”””
```



ID	Name	Animal
1	Dr. Harris Bonkers	Rabbit
2	Moon	Dog
3	Ripley	Cat
4	Tom	Cat

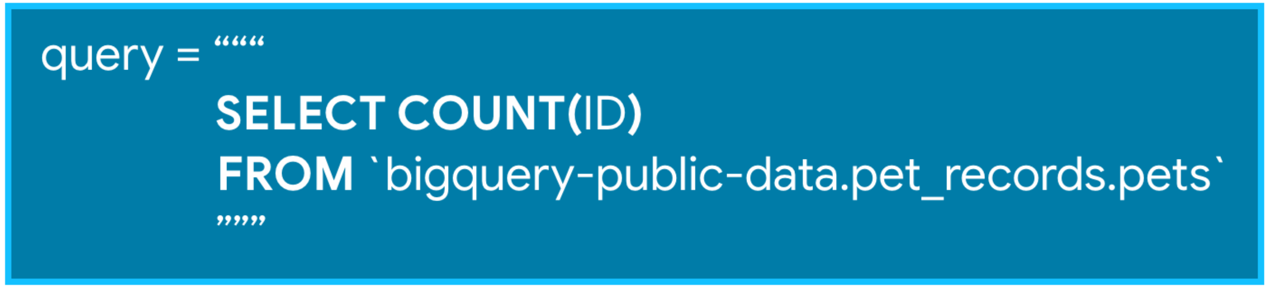


```sql
SELECT Name
FROM `mis384-lab2.bigquery_public_data.pet_records`
WHERE Animal = 'Cat'
```

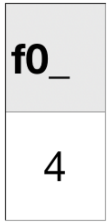
Query results

Row	Name
1	Ripley
2	Tom

COUNT()



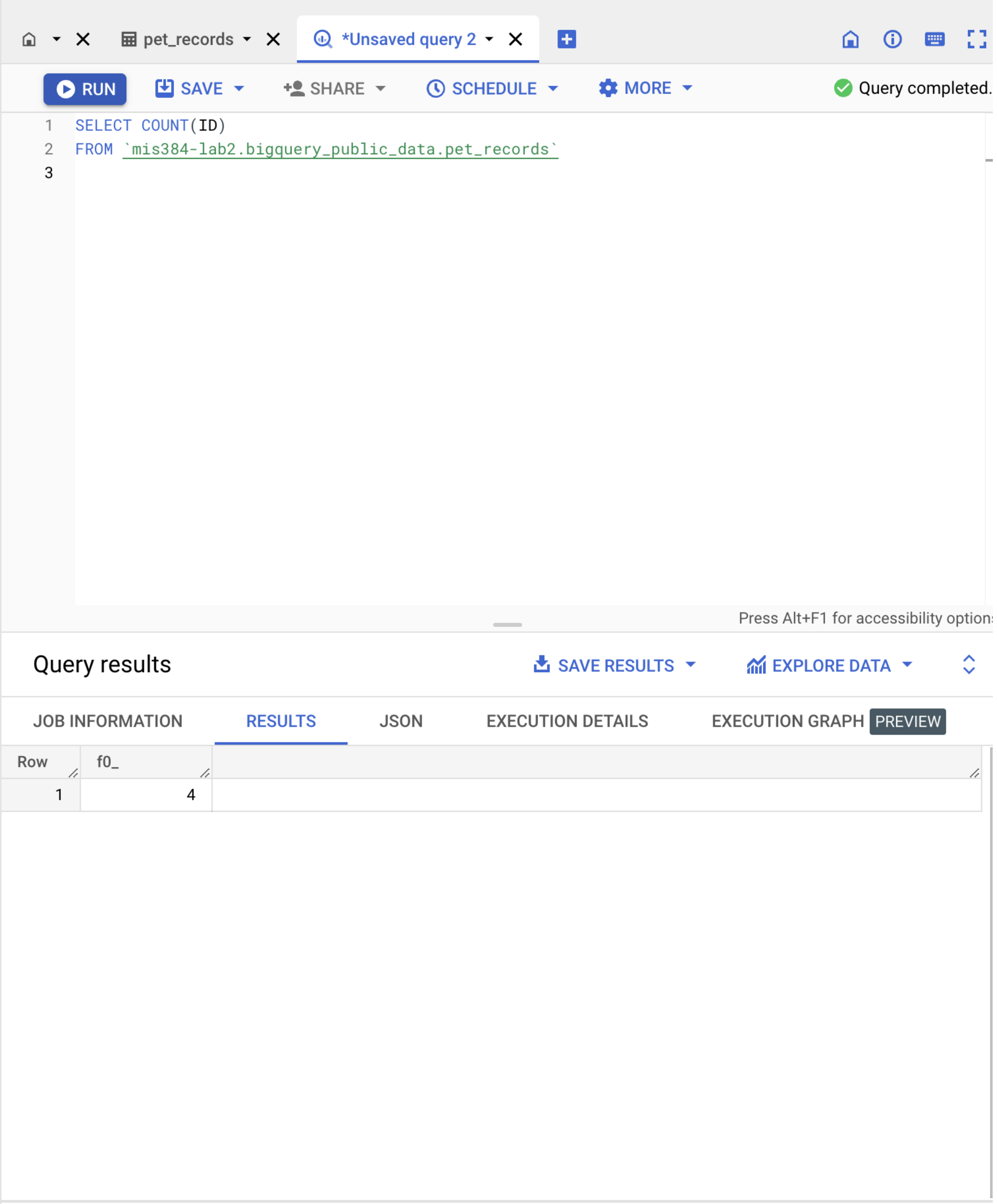
```
query = “““
        SELECT COUNT(ID)
        FROM `bigquery-public-data.pet_records.pets`
        ”””
```



f0_
4

COUNT(DISTINCT) allows you to count the number of distinct or unique values in a given column. The **DISTINCT** keyword can be used in conjunction with **COUNT** to count the number of unique values within a specific column or set of columns.

COUNT()

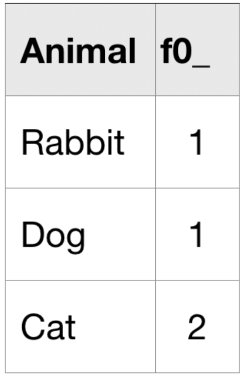


```sql
SELECT COUNT(ID)
FROM `mis384-lab2.bigquery_public_data.pet_records`
```

Row	f0_
1	4

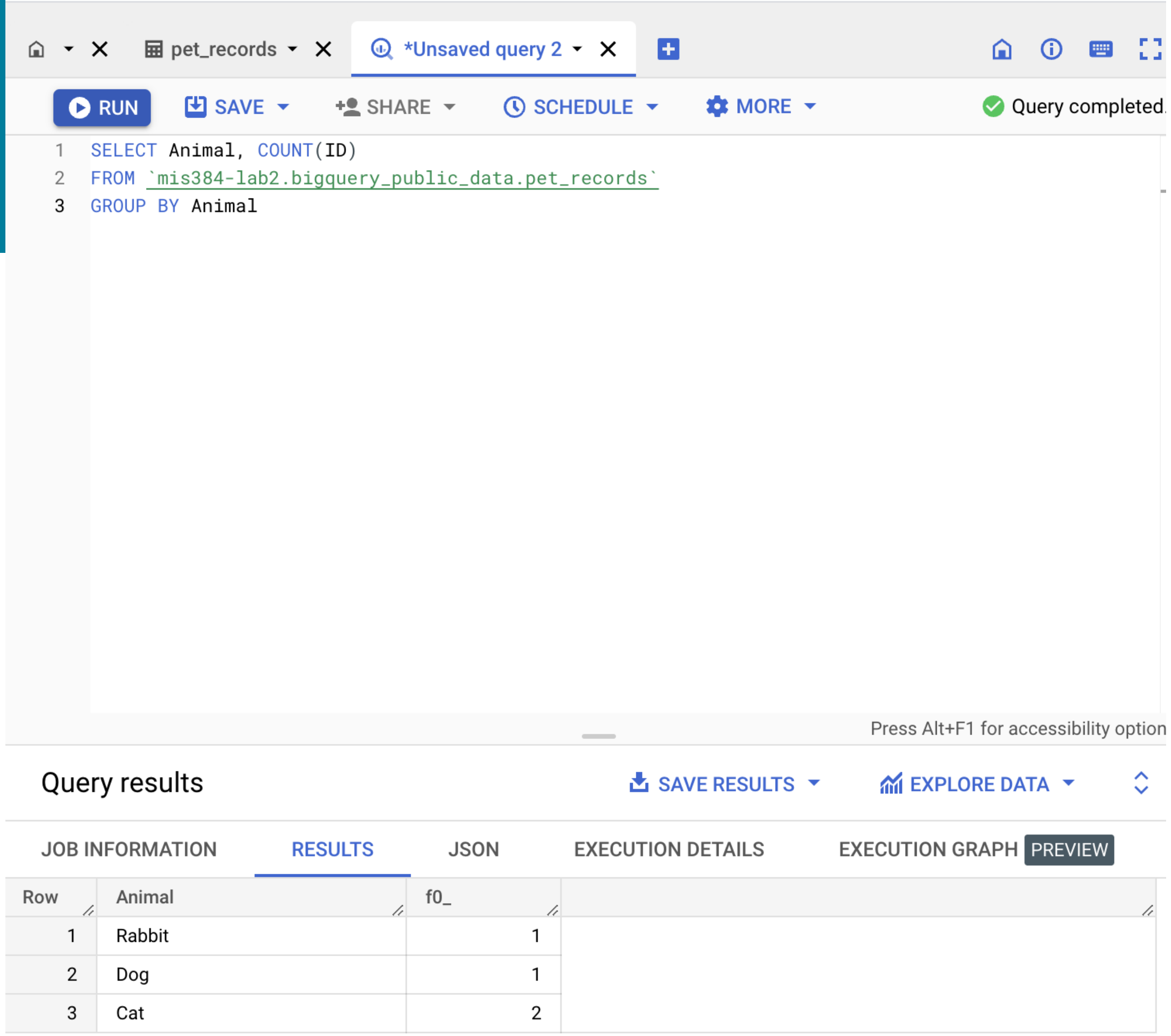
GROUP BY

```
query = “““
            SELECT Animal, COUNT(ID)
            FROM `bigquery-public-data.pet_records.pets`
            GROUP BY Animal
            ”””
```



Animal	f0_
Rabbit	1
Dog	1
Cat	2

GROUP BY



Query completed.

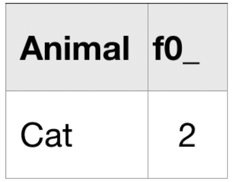
```sql
SELECT Animal, COUNT(ID)
FROM `mis384-lab2.bigquery_public_data.pet_records`
GROUP BY Animal
```

Query results

Row	Animal	f0_
1	Rabbit	1
2	Dog	1
3	Cat	2

GROUP BY… HAVING

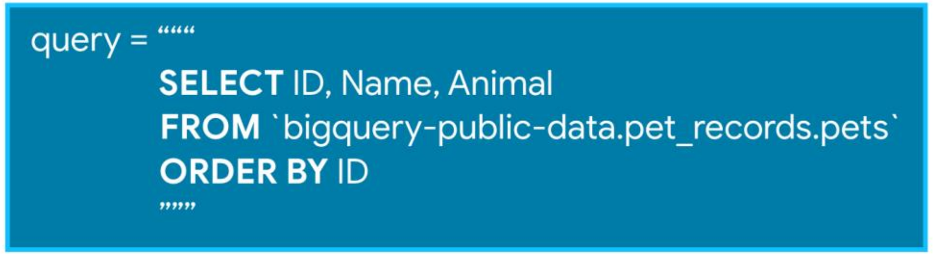
```
query = “““
          SELECT Animal, COUNT(ID)
          FROM `bigquery-public-data.pet_records.pets`
          GROUP BY Animal
          HAVING COUNT(ID) > 1
          ”””
```



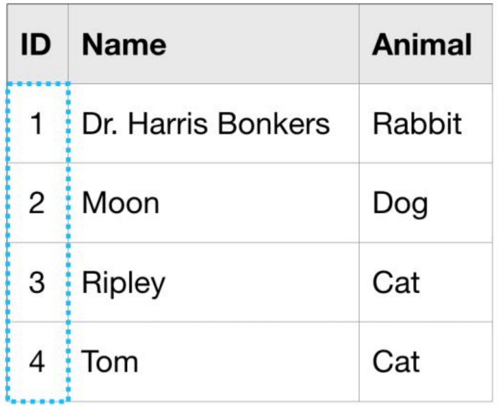
Animal	f0_
Cat	2

HAVING is used in combination with **GROUP BY** to ignore groups that don't meet certain criteria. So this query, for example, will only include groups that have more than one ID in them.

ORDER BY

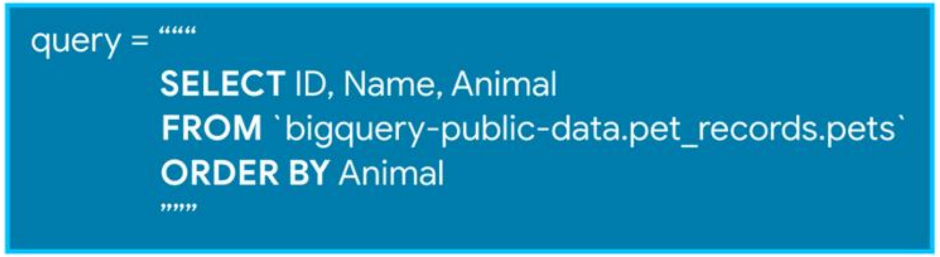


```
query = “““
          SELECT ID, Name, Animal
          FROM `bigquery-public-data.pet_records.pets`
          ORDER BY ID
          ”””
```

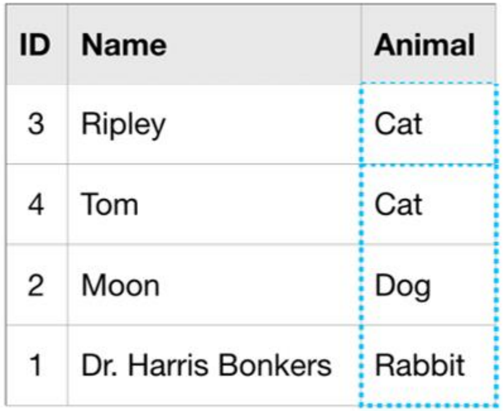


ID	Name	Animal
1	Dr. Harris Bonkers	Rabbit
2	Moon	Dog
3	Ripley	Cat
4	Tom	Cat

The **ORDER BY** clause also works for columns containing text, where the results show up in alphabetical order.



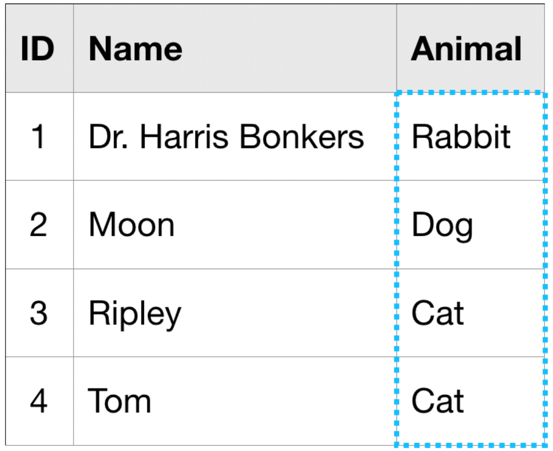
```
query = “““
          SELECT ID, Name, Animal
          FROM `bigquery-public-data.pet_records.pets`
          ORDER BY Animal
          ”””
```



ID	Name	Animal
3	Ripley	Cat
4	Tom	Cat
2	Moon	Dog
1	Dr. Harris Bonkers	Rabbit

ORDER BY

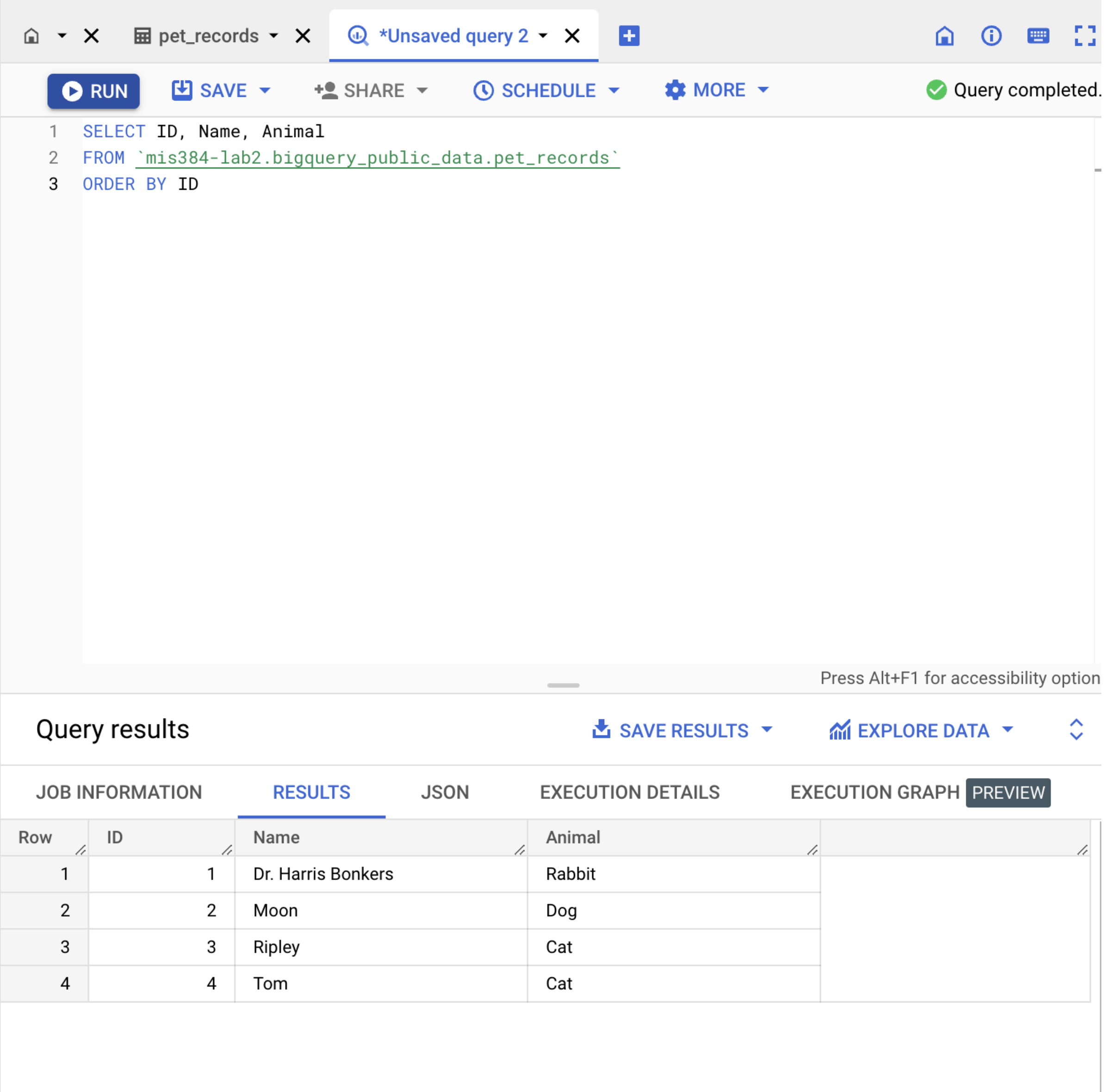
```
query = “““
          SELECT ID, Name, Animal
          FROM `bigquery-public-data.pet_records.pets`
          ORDER BY Animal DESC
          ”””
```



ID	Name	Animal
1	Dr. Harris Bonkers	Rabbit
2	Moon	Dog
3	Ripley	Cat
4	Tom	Cat

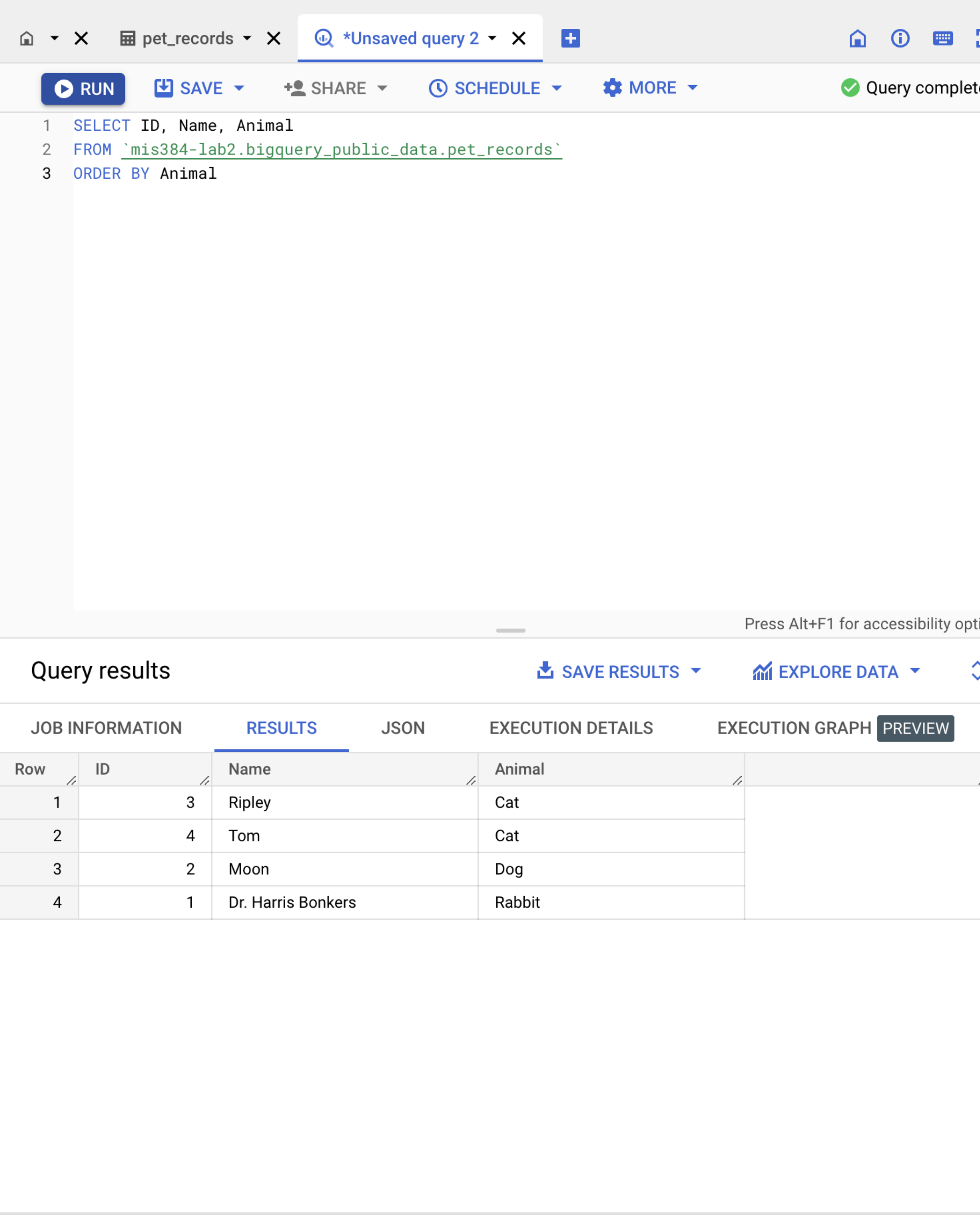
Reverse the order using the **DESC** argument (short for 'descending'). The next query sorts the table by the Animal column, where the values that are last in alphabetic order are returned first.

ORDER BY



```sql
SELECT ID, Name, Animal
FROM `mis384-lab2.bigquery_public_data.pet_records`
ORDER BY ID
```

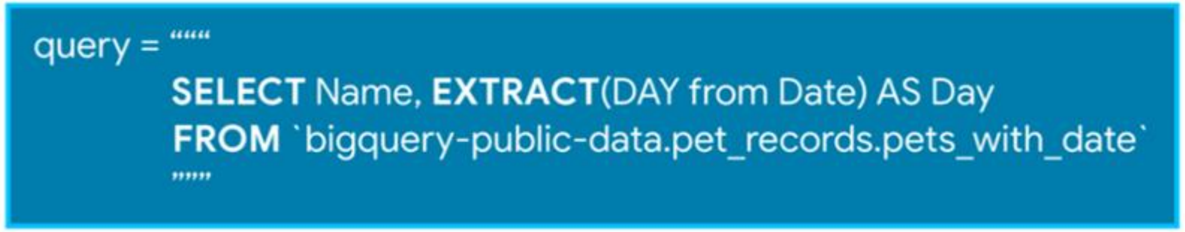
Row	ID	Name	Animal
1	1	Dr. Harris Bonkers	Rabbit
2	2	Moon	Dog
3	3	Ripley	Cat
4	4	Tom	Cat



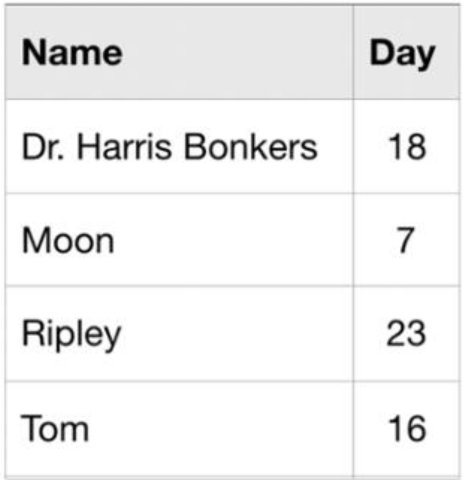
```sql
SELECT ID, Name, Animal
FROM `mis384-lab2.bigquery_public_data.pet_records`
ORDER BY Animal
```

Row	ID	Name	Animal
1	3	Ripley	Cat
2	4	Tom	Cat
3	2	Moon	Dog
4	1	Dr. Harris Bonkers	Rabbit

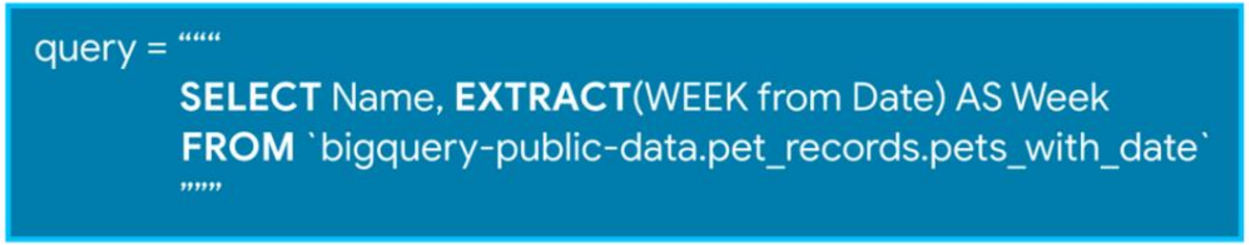
DATE



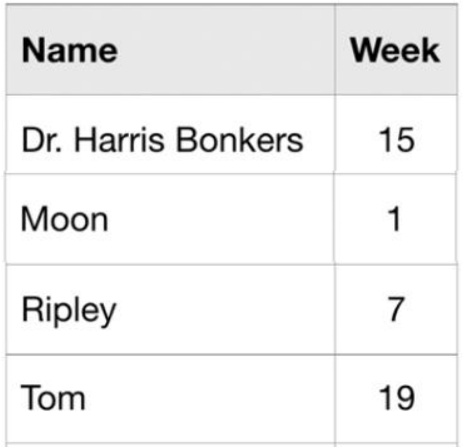
```
query = “““
        SELECT Name, EXTRACT(DAY from Date) AS Day
        FROM `bigquery-public-data.pet_records.pets_with_date`
        ”””
```



Name	Day
Dr. Harris Bonkers	18
Moon	7
Ripley	23
Tom	16



```
query = “““
        SELECT Name, EXTRACT(WEEK from Date) AS Week
        FROM `bigquery-public-data.pet_records.pets_with_date`
        ”””
```



Name	Week
Dr. Harris Bonkers	15
Moon	1
Ripley	7
Tom	19

Advanced Syntax: AS

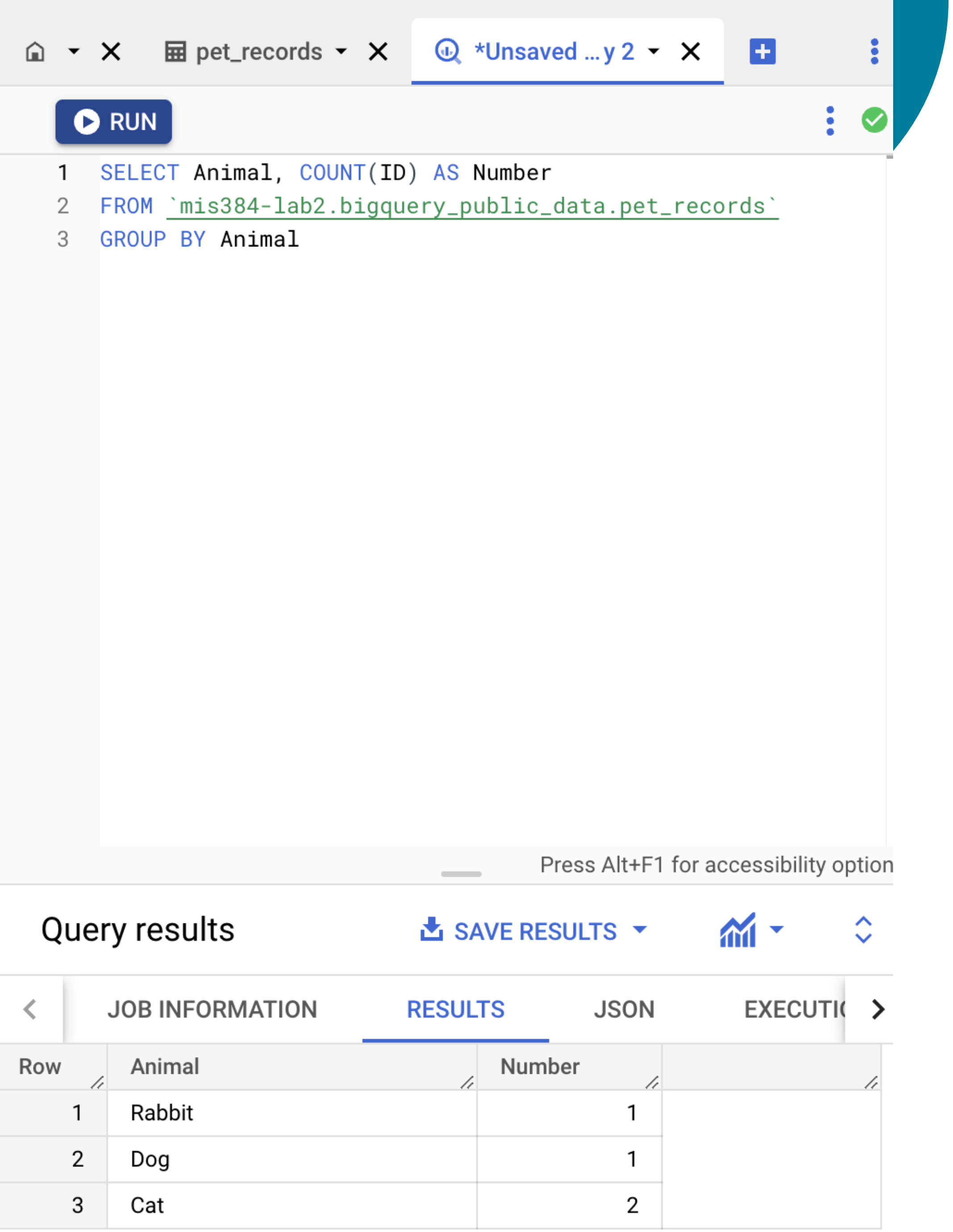
```
query = “““
        SELECT Animal, COUNT(ID)
        FROM `bigquery-public-data.pet_records.pets`
        GROUP BY Animal
        ”””
```

Animal	f0_
Rabbit	1
Dog	1
Cat	2

```
query = “““
        SELECT Animal, COUNT(ID) AS Number
        FROM `bigquery-public-data.pet_records.pets`
        GROUP BY Animal
        ”””
```

Animal	Number
Rabbit	1
Dog	1
Cat	2

These queries return the same information, but in the second query the column returned by the **COUNT()** function will be called Number, rather than the default name of f0_.

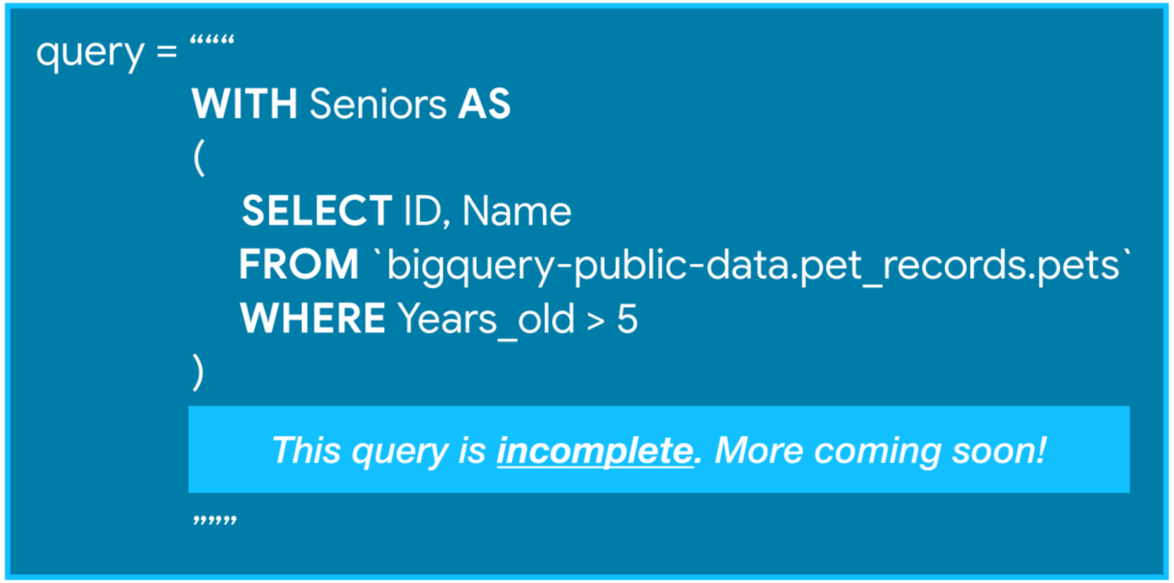


```
SELECT Animal, COUNT(ID) AS Number
FROM `mis384-lab2.bigquery_public_data.pet_records`
GROUP BY Animal
```

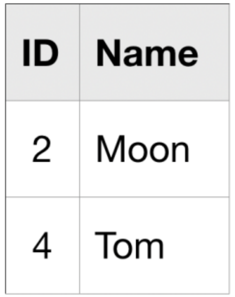
Row	Animal	Number
1	Rabbit	1
2	Dog	1
3	Cat	2

Advanced Syntax: WITH…AS

A **common table expression** (or **CTE**) is a temporary table that you return within your query. CTEs are helpful for splitting your queries into readable chunks, and you can write queries against them.



```
query = """
          WITH Seniors AS
          (
             SELECT ID, Name
             FROM `bigquery-public-data.pet_records.pets`
             WHERE Years_old > 5
          )
          This query is incomplete. More coming soon!
          """
```



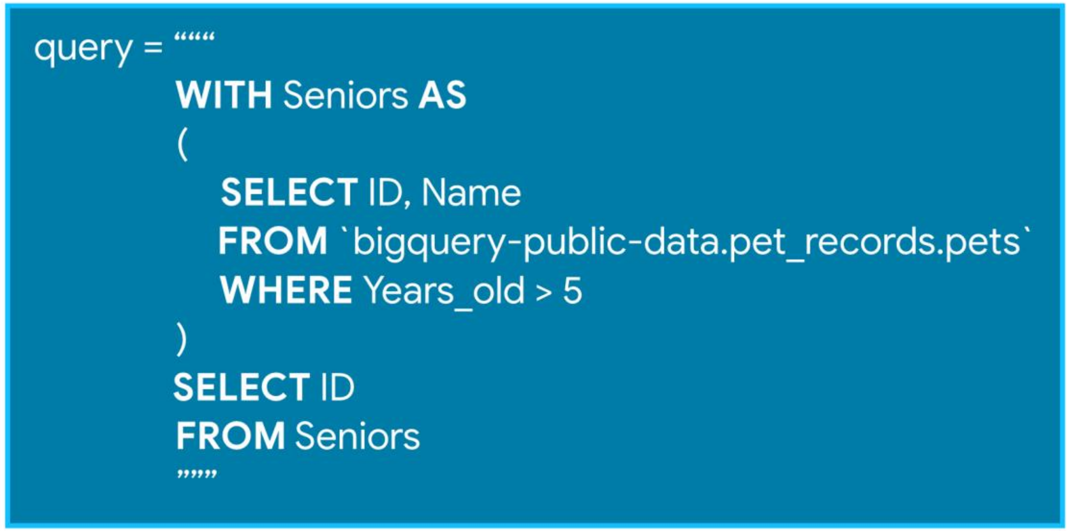
ID	Name
2	Moon
4	Tom

This is a **CTE** named `Seniors`.
(*It is not returned by the query*.)

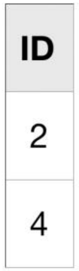
While this incomplete query above won't return anything, it creates a CTE that we can then refer to (as Seniors) while writing the rest of the query.

Advanced Syntax: WITH…AS

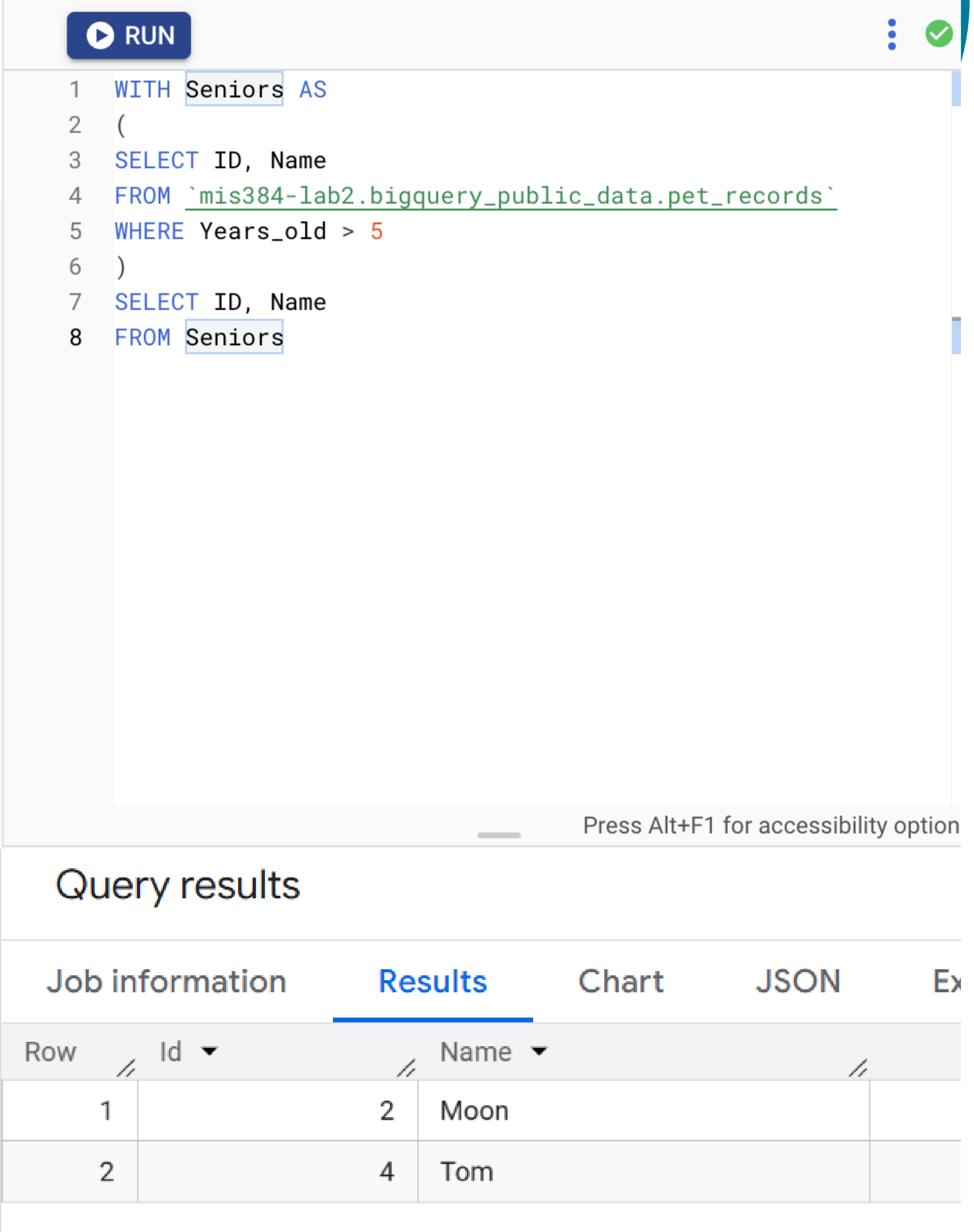
We can finish the query by pulling the information that we want from the CTE. The complete query below first creates the CTE, and then returns all of the IDs from it.



```
query = """
        WITH Seniors AS
        (
          SELECT ID, Name
          FROM `bigquery-public-data.pet_records.pets`
          WHERE Years_old > 5
        )
        SELECT ID
        FROM Seniors
        """
```



ID
2
4



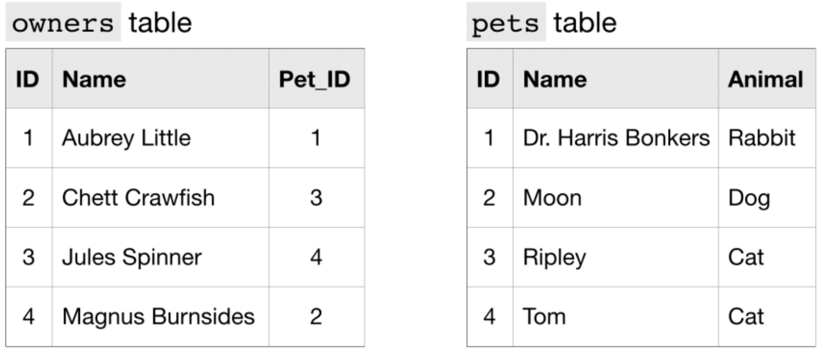
```
WITH Seniors AS
(
SELECT ID, Name
FROM `mis384-lab2.bigquery_public_data.pet_records`
WHERE Years_old > 5
)
SELECT ID, Name
FROM Seniors
```

Press Alt+F1 for accessibility option

Query results

Row	Id	Name
1	2	Moon
2	4	Tom

Advanced Syntax: JOIN

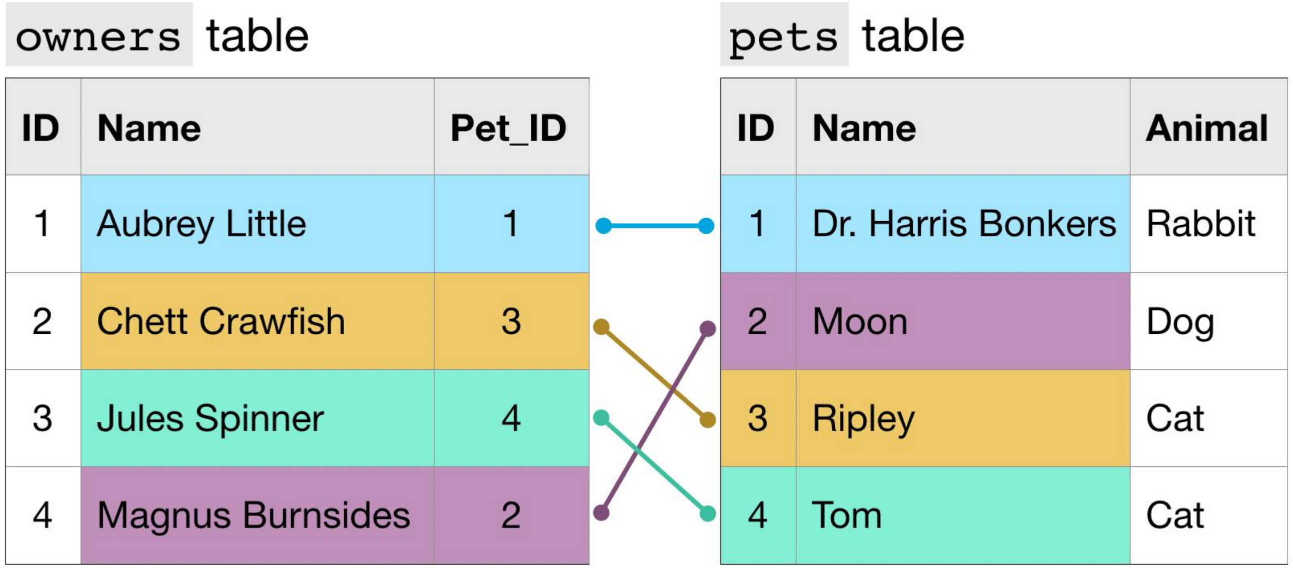


`owners` table

ID	Name	Pet_ID
1	Aubrey Little	1
2	Chett Crawfish	3
3	Jules Spinner	4
4	Magnus Burnsides	2

`pets` table

ID	Name	Animal
1	Dr. Harris Bonkers	Rabbit
2	Moon	Dog
3	Ripley	Cat
4	Tom	Cat



`owners` table

ID	Name	Pet_ID
1	Aubrey Little	1
2	Chett Crawfish	3
3	Jules Spinner	4
4	Magnus Burnsides	2

`pets` table

ID	Name	Animal
1	Dr. Harris Bonkers	Rabbit
2	Moon	Dog
3	Ripley	Cat
4	Tom	Cat

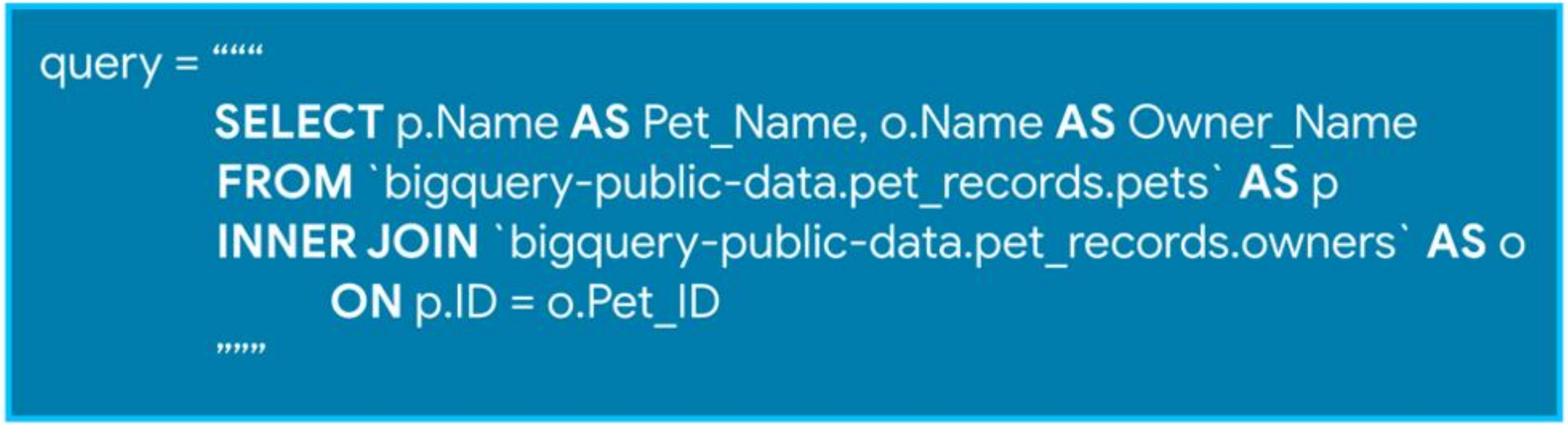
Dr. Harris Bonkers is owned by **Aubrey Little**.

Moon is owned by **Magnus Burnsides**.

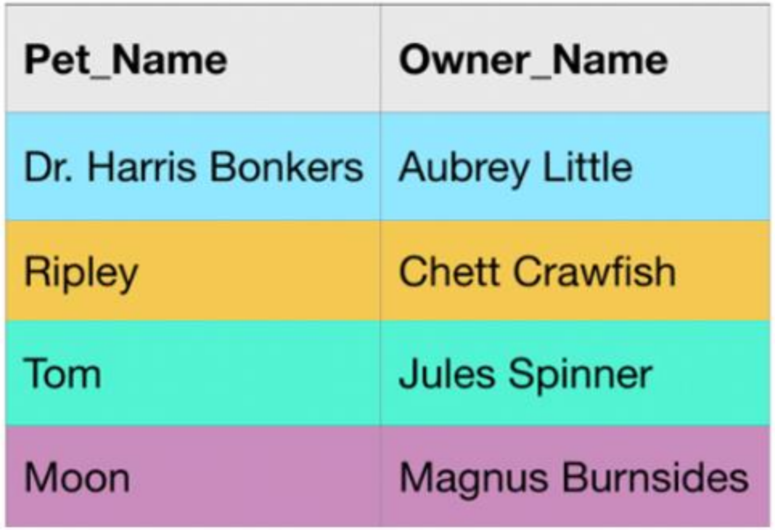
Ripley is owned by **Chett Crawfish**.

Tom is owned by **Jules Spinner**.

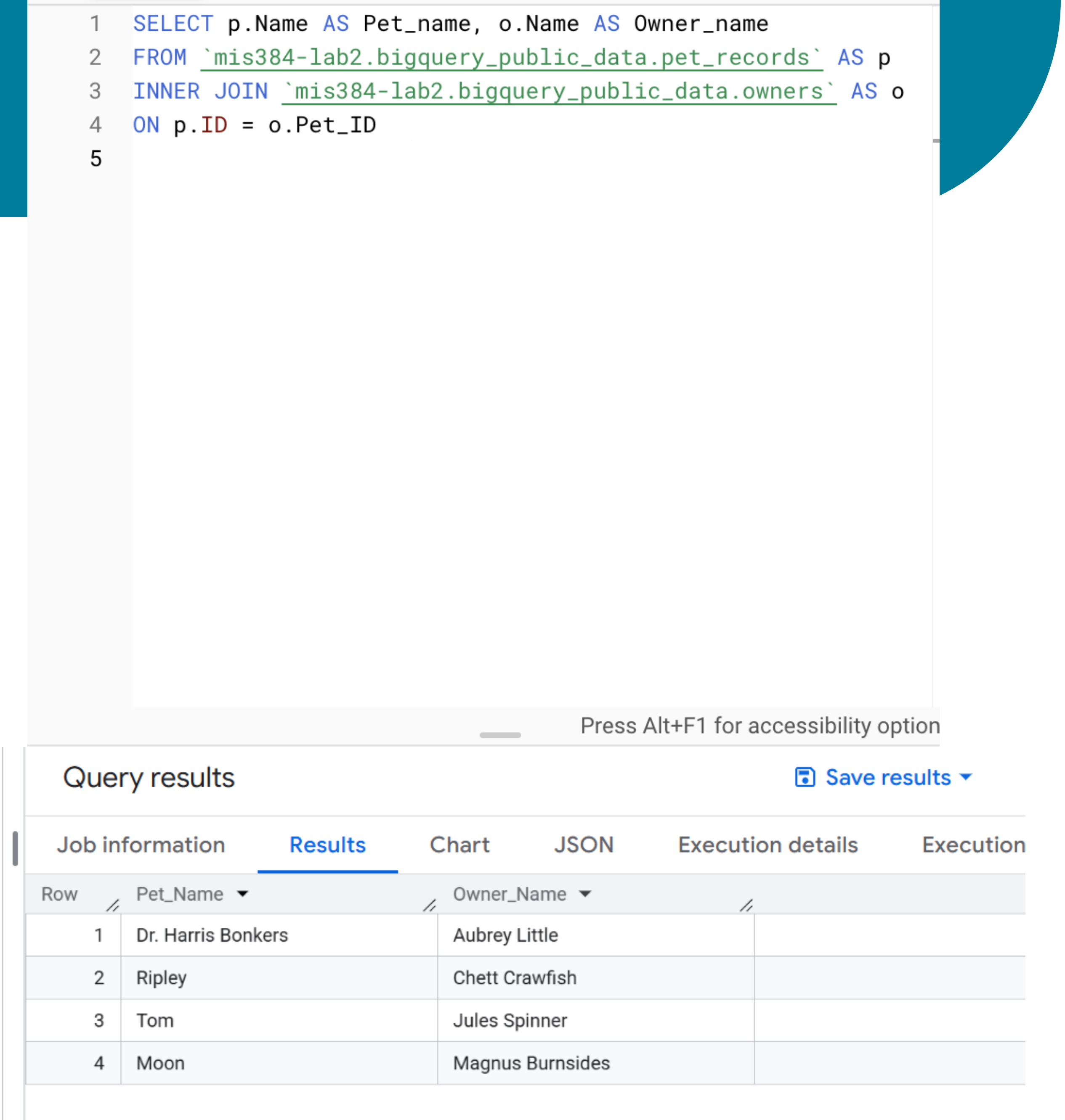
Advanced Syntax: JOIN



```
query = “““““
        SELECT p.Name AS Pet_Name, o.Name AS Owner_Name
        FROM `bigquery-public-data.pet_records.pets` AS p
        INNER JOIN `bigquery-public-data.pet_records.owners` AS o
            ON p.ID = o.Pet_ID
        ”””””
```



Pet_Name	Owner_Name
Dr. Harris Bonkers	Aubrey Little
Ripley	Chett Crawfish
Tom	Jules Spinner
Moon	Magnus Burnsides



```sql
SELECT p.Name AS Pet_name, o.Name AS Owner_name
FROM `mis384-lab2.bigquery_public_data.pet_records` AS p
INNER JOIN `mis384-lab2.bigquery_public_data.owners` AS o
ON p.ID = o.Pet_ID
```

Row	Pet_Name	Owner_Name
1	Dr. Harris Bonkers	Aubrey Little
2	Ripley	Chett Crawfish
3	Tom	Jules Spinner
4	Moon	Magnus Burnsides

Your Turn to Run Real Data

The Iconic is a popular Australian online fashion and sports retailer. It was launched in 2011 by Adam Jacobs and Cameron Votan, with the aim of bringing a diverse range of international and local brands to the Australian market. Over the years, the company has become a leading player in the online fashion retail industry in Australia.

The Iconic has adopted various marketing strategies to gain a competitive edge in the Australian ecommerce market. Now you have the access to its ecommerce database to evaluate the product related questions.

We'll work with the table ‘categories’ in the data-to-insights.ecommerce. The table provides the category of each product (productSKU) and its corresponding category.

Your Turn to Run Real Data

1. What is the number of categories available? (Hints: **COUNT (DISTINCT category))**
2. How many products are included in the category "Bags"?
3. The number of products in each category and in which category contained the greatest number of products?
4. Can you provide some suggestions to the purchasing team based on the above analysis?