# SQL

The SQL commands we’ll use to answer these questions are:

* SELECT
* WHERE
* LIMIT
* ORDER BY
* GROUP BY
* AND
* OR
* MIN
* MAX
* AVG
* SUM
* COUNT

To do this, we’ll define a function that takes our query (stored as a string) as an input and shows the result as a formatted dataframe:

### 引入sqlite必须要打的语句。

**import** **sqlite3**

**import** **pandas** **as** **pd**

db **=** sqlite3**.**connect('hubway.db')

**def** **run\_query**(query):

**return** pd**.**read\_sql\_query(query,db)

## SELECT

1. Most basic sentence。
2. You can either **specify columns by name (separated by commas)** or use the wildcard **\* to return every column** in the table.
3. Use **From** followed : which table to get them from.
4. Eg: For example, if we wanted to see the start\_date and bike\_number for every trip in the trips table, we could use the following query:

**SELECT** start\_date, bike\_number **FROM** trips;

1. Notice: 每一句语句的结尾需要(;).

## LIMIT

1. **LIMIT**simply tells the database **how many rows you want it to return**.

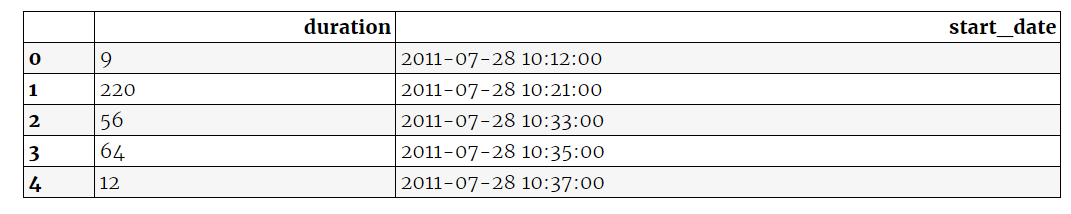
Eg: **SELECT** start\_date, bike\_number **FROM** trips **LIMIT** 5;

1. In Python,

query **=** 'SELECT duration, start\_date FROM trips LIMIT 5;'

run\_query(query)

>>>



## Order By

1. This command allows you to sort the database on a given column.
2. By default**, ORDER BY sorts in** ascending order(ASC). descending(DESC).

**ORDER** **BY** duration **ASC**

1. Q: **What was the duration of the longest trip?**

思路:

* Use SELECT to retrieve the duration column FROM the trips table
* Use ORDER BY to sort the duration column and use the DESC keyword to specify that you want to sort in descending order
* Use LIMIT to restrict the output to 1 row

Code(in py):

query **=** '''

SELECT duration

FROM trips

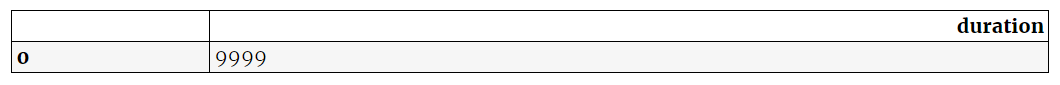
ORDER BY duration DESC

LIMIT 1;

'''

run\_query(query)

>>>



## WHERE

1. Where: to input a specific subset of the data. Eg:
   1. **WHERE** bike\_number **=** "B00400"

query **=** '''

SELECT \*

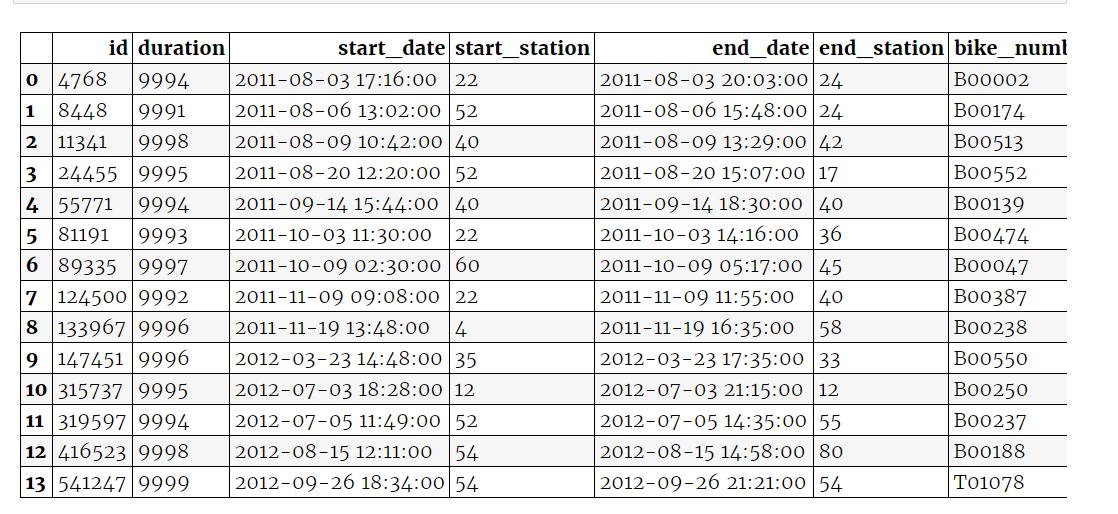
FROM trips

WHERE duration > 9990;

'''

run\_query(query)

>>>



1. Where + AND/OR  recommend using parentheses () here.

query **=** '''

SELECT \*

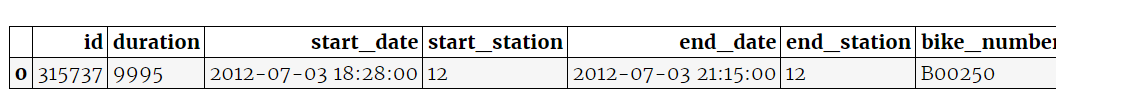
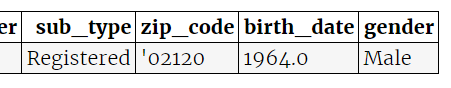
FROM trips

WHERE (duration >= 9990) AND (sub\_type = "Registered")

ORDER BY duration DESC;

'''

run\_query(query)

## COUNT

1. COUNT(column\_name): This allows us to shift the calculation to the database and save us the trouble of writing **additional scripts to count up results**. EG:

**SELECT** **COUNT**(id)

**FROM** trips

1. Q: **“How many trips were taken by ‘registered’ users?”**

query **=** '''

SELECT COUNT(\*)

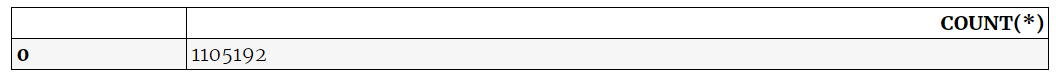
FROM trips

WHERE sub\_type = "Registered";

'''

run\_query(query)

>>>



**对title 进行命名 (AS)**

query **=** '''

SELECT COUNT(\*) AS "Total Trips by Registered Users"

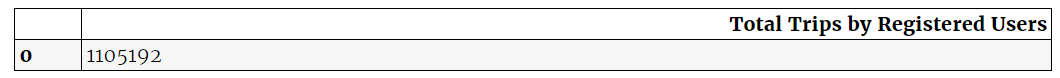
FROM trips

WHERE sub\_type = "Registered";

'''

run\_query(query)

>>>



## Aggregate Functions

1. **aggregate functions :** SUM, AVG, MIN and MAX
2. **Q :**  **“What was the average trip duration?”**

query **=** '''

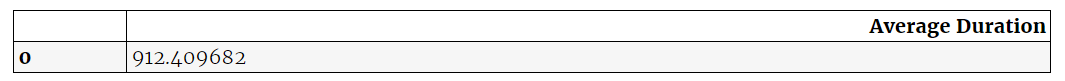
SELECT AVG(duration) AS "Average Duration"

FROM trips;

'''

run\_query(query)

>>>



## GROUP BY

1. GROUP BY separates the rows into groups based on the contents of a particular column and allows us to perform aggregate functions on each group.
2. **Q : do registered or casual users take longer trips?**

query **=** '''

SELECT sub\_type, AVG(duration) AS "Average Duration"

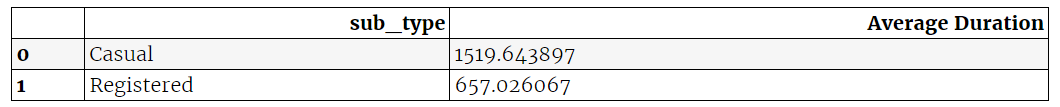
FROM trips

GROUP BY sub\_type;

'''

run\_query(query)

**>>>**



1. **Q : which bike was used for the most trips**?

query **=** '''

SELECT bike\_number as "Bike Number", COUNT(\*) AS "Number of Trips"

FROM trips

GROUP BY bike\_number

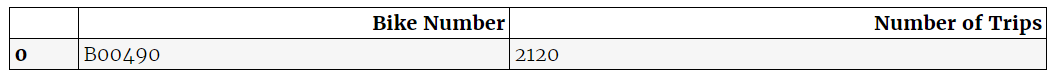
ORDER BY COUNT(\*) DESC

LIMIT 1;

'''

run\_query(query)

**>>>**



## Arithmetic Operators

1. **可以用: + ,- ,\*,/**
2. **Q : average duration of trips by registered members over the age of 30?**

query **=** '''

SELECT AVG(duration)

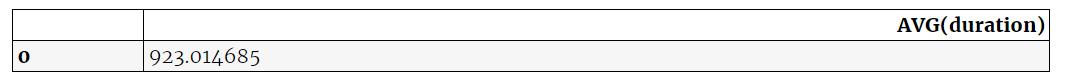
FROM trips

WHERE (2017 - birth\_date) > 30;

'''

run\_query(query)

>>>



## JOIN

1. In database, there are two tables: **Orders and Customers.**
2. If they were stored in same table, we can use:

**SELECT** order\_number, name ### 带上table.name: SELECT orders.order\_number, customers.name

**FROM** orders;

1. For **Connected two tables,** we use **JOIN** and **ON**. **JOIN** specifies which tables should be connected and ON specifies which columns in each table are related.
2. To tell customer\_id is the same:

**ON** orders.customer\_ID **=** customers.customer\_id

FROM orders JOIN customers or FROM customers JOIN orders here. ON 中写 FromLibrary.\*\*\* = ToLibrary.\*\*\*

**SELECT** orders.order\_number, customers.name

**FROM** orders

**JOIN** customers

**ON** orders.customer\_id **=** customers.customer\_id

1. Check all stations

query **=** '''

SELECT \*

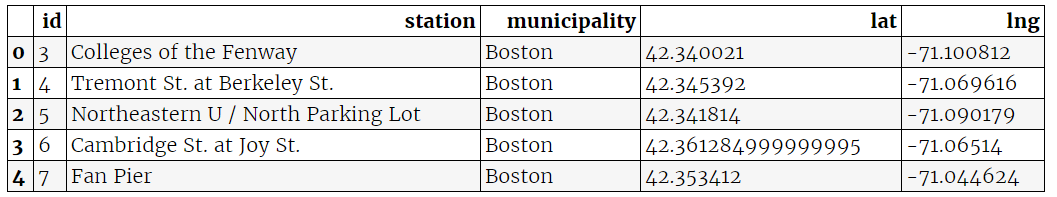
FROM stations

LIMIT 5;

'''

run\_query(query)

>>>



1. Q : **which station is the most frequent starting point?**

query **=** '''

SELECT stations.station AS "Station", COUNT(\*) AS "Count"

FROM trips

JOIN stations

ON trips.start\_station = stations.id

GROUP BY stations.station

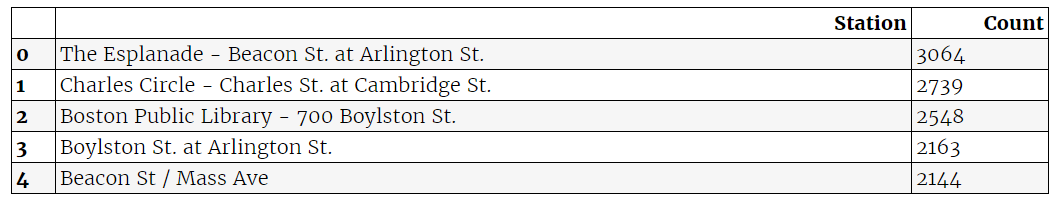
ORDER BY COUNT(\*) DESC

LIMIT 5;

'''

run\_query(query)

>>>



1. **Q : which stations are most frequently used for round trips?**

query **=** '''

SELECT stations.station AS "Station", COUNT(\*) AS "Count"

FROM trips

JOIN stations

ON trips.start\_station = stations.id

WHERE trips.start\_station = trips.end\_station

GROUP BY stations.station

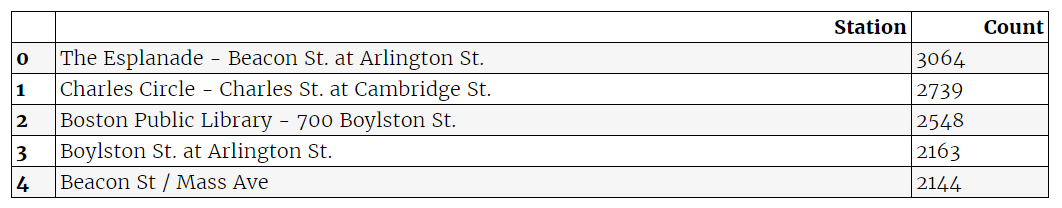
ORDER BY COUNT(\*) DESC

LIMIT 5;

'''

run\_query(query)

>>>



1. Q: **how many trips start and end in different municipalities(市区)?**

query **=** '''

SELECT COUNT(trips.id) AS "Count"

FROM trips

JOIN stations AS start

ON trips.start\_station = start.id

JOIN stations AS end

ON trips.end\_station = end.id

WHERE start.municipality <> end.municipality;

'''

run\_query(query)

>>>

