**Creator: Sridevi Muralidharan**

**Project 2**

**Trends and descriptive analytics**

**Analyzing bikeshare data from Bluebikes (Boston city) and Divvybikes (Chicago city)**

**Trends over time**

**How many trips were there in each month of each year?**

|  |
| --- |
| **Key findings** |
| **Jun - Oct increased activity - coincides with the summer weather in Boston - so that was good to see.** |
|  |
| **Bluebikes have doubled riders than the previous two years** |

**SQL Query**

WITH all\_bluebikes AS (SELECT \* FROM bluebikes\_2016

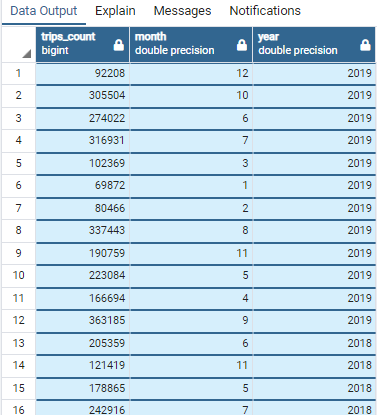
UNION ALL (SELECT \* FROM bluebikes\_2017 UNION ALL (SELECT \* FROM bluebikes\_2018

UNION ALL SELECT \* FROM bluebikes\_2019) ) order by start\_time asc)

select count(\*) as trips\_count, DATE\_PART('month', start\_time) AS month, DATE\_PART('Year', start\_time) AS year

from all\_bluebikes group by month, year

order by year desc



Successfully run. Total query runtime: 47 secs 303 msec 48 rows affected. Data exported to sri\_project2\_sheet 1.

**Which organizations are showing the most growth in bike rentals?**

I am comparing data on bluebikes 2016 – 19 with divvybikes 2016-19

**Key Findings**

**Bluebikes subscribers have been steadily growing past 4 years**

**Bluebikes are 50% low on the number of rides it gets , compared to Divvy bike**

**If bluebikes would find a way to tap the non-subscribers it will definitely improve their long-term revenue and growth for the company**

**SQL using all bluebikes data across years 2016 to 2019**

WITH all\_bluebikes AS (SELECT \*

FROM bluebikes\_2016

UNION ALL (

SELECT \* FROM bluebikes\_2017

UNION ALL

(SELECT \* FROM bluebikes\_2018

UNION ALL

SELECT \* FROM bluebikes\_2019) )order by start\_time asc)

select user\_type, count(\*) as trips\_count, DATE\_PART('month', start\_time) AS month, DATE\_PART('Year', start\_time) AS year

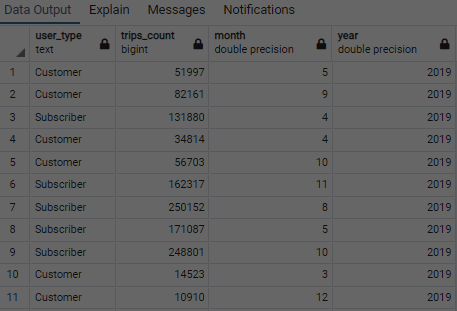
from all\_bluebikes

group by user\_type, month, year

order by year desc

Successfully run. Total query runtime: 49 secs 409 msec. 96 rows affected

Output snapshot for bluebikes



**SQL for all divvybikes data across years 2016 to 2019**

WITH all\_divvybikes AS (

SELECT \*

FROM divvybikes\_2016

UNION ALL (

SELECT \* FROM divvybikes\_2017

UNION ALL

(SELECT \* FROM divvybikes\_2018

UNION ALL

SELECT \* FROM divvybikes\_2019) ) order by start\_time asc

)

select user\_type, count(\*) as trips\_count, DATE\_PART('month', start\_time) AS month, DATE\_PART('Year', start\_time) AS year

from all\_divvybikes

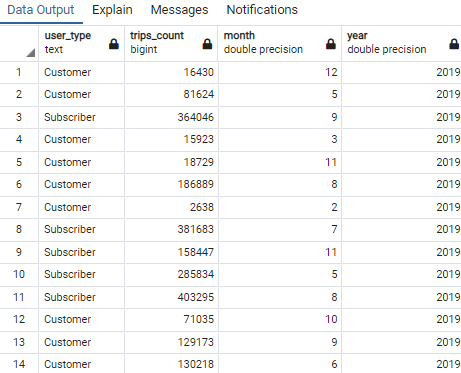
group by user\_type, month, year

order by year desc

Successfully run. Total query runtime: 2 min 45 secs.

106 rows affected.

Output snapshot for divvybikes



**Is there a difference in growth between holiday activity and commuting activity? comparing bluebikes and divvy**

Key Findings

For both Bluebikes and Divvy bikes

When we look in terms of months, during Dec and Jan, activity is low across all years.

and this period corresponds to holiday period globally and

so yes it is negatively affecting bluebikes in terms of company growth and revenue

Whereas when we look at holiday period in terms of weekends like Saturday and Sunday

we find less number of rides on those days across all years

that means people prefer to ride bikes for commuting mon-fri than weekends

interestingly most subscribers follow this pattern and the casual users appear to indulge in weekends

I am using two ways to answering this Hard Question – firstly by looking at holiday period – by separating days of the week. From this I got data about weekends and weekdays of each year – in terms of number of trips by each user type (customer, subscriber).

here assumption 1 is week end trips = holiday activity ; weekday trips = commuting activity

I extracted data from divvybikes across all years and compared that with bluebikes to give answers to this question.

**-SQL for data from divvybikes**

WITH all\_divvybikes AS (

SELECT \*

FROM divvybikes\_2016

UNION ALL (

SELECT \* FROM divvybikes\_2017

UNION ALL

(SELECT \* FROM divvybikes\_2018

UNION ALL

SELECT \* FROM divvybikes\_2019) ) order by start\_time asc

)

select user\_type, extract(dow from start\_time) AS day, DATE\_PART('Year', start\_time) AS year, count(\*) as trips\_count

from all\_divvybikes

----getting data by day of week per year where sunday or saturday will be treated as holidays -- for analysis

----here DOW calculates the day of week Sunday (0) to Saturday (6)

group by user\_type, day, year

order by year, day desc

**--SQL for data on bluebikes**

WITH all\_bluebikes AS (

SELECT \*

FROM bluebikes\_2016

UNION ALL (

SELECT \* FROM bluebikes\_2017

UNION ALL

(SELECT \* FROM bluebikes\_2018

UNION ALL

SELECT \* FROM bluebikes\_2019) ) order by start\_time asc

)

select user\_type, extract(dow from start\_time) AS day\_of\_week, DATE\_PART('Year', start\_time) AS year, count(\*) as trips\_count

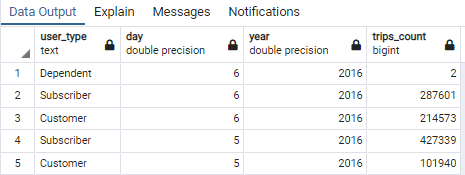
from all\_bluebikes

----getting data by day of week per year where sunday or saturday will be treated as holidays -- for analysis –

--here also DOW calculates the day of week Sunday (0) to Saturday (6)

group by user\_type, day\_of\_week, year

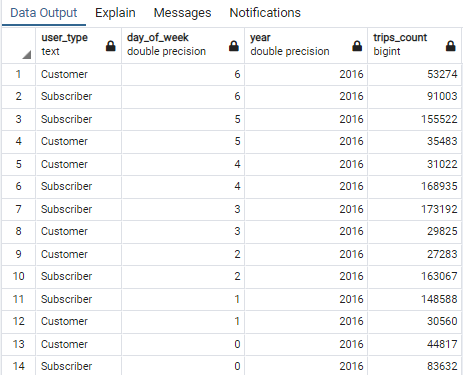
order by year, day\_of\_week desc



Output snapshot for divvybikes – in terms of days of the week -

Successfully run. Total query runtime: 4 min 45 secs.

66 rows affected.



Output snapshot for bluebikes – in terms of days of the week -

Successfully run. Total query runtime: 1 min 54 secs. 56 rows affected.

As a Second approach, I looked at holiday period – dec and jan -- of each year –and extracted the number of trips in relation to each user type (consumer, subscriber).

here assumption 1 is summed trips for dec and jan = holiday activity ;

assumption 2 is the sum of trips for pairs of remainder of months = commuting activity i.e (feb, mar ), (April, may ), (June,July ), (aug,sep) and (oct,nov)

To do this, I re-used the two SQL queries created previously for the moderate question and I extracted more data from that.

**Geospatial**

**What was the longest journey? What do we know about it?**

**Key Findings**

**Working with geospatial, the longest journey was found to be a ride from Glendale Square (Ferry St at Broadway) to Belgrade Ave at Walworth St. Ride distance was calculated to be 10 k. it happened in mid-august 2019 and by a man likely to be 51 years in age.**

WITH all\_bluebikes AS (

SELECT \*

FROM bluebikes\_2016

UNION ALL (

SELECT \* FROM bluebikes\_2017

UNION ALL

(SELECT \* FROM bluebikes\_2018

UNION ALL

SELECT \* FROM bluebikes\_2019) ) order by start\_time asc

),

end\_coordinates as (select id as end\_station\_id, name as end\_station\_name, latitude as end\_latitude, longtitude as end\_longitude from bluebikes\_stations),

start\_coordinates as (select id as start\_station\_id, name as start\_station\_name, latitude as start\_latitude, longtitude as start\_longitude from bluebikes\_stations)

select start\_time, end\_time, age(end\_time,start\_time) as trip\_duration, all\_bluebikes.start\_station\_id, all\_bluebikes.end\_station\_id, start\_station\_name, end\_station\_name, s.start\_longitude, e.end\_latitude, e.end\_longitude, calculate\_distance(s.start\_latitude, s.start\_longitude, e.end\_latitude, e.end\_longitude,'k'), user\_type, user\_birth\_year, user\_gender --, extract(month from all\_bluebikes.start\_time) AS month, DATE\_PART('Year', all\_bluebikes\_2016.start\_time) AS year

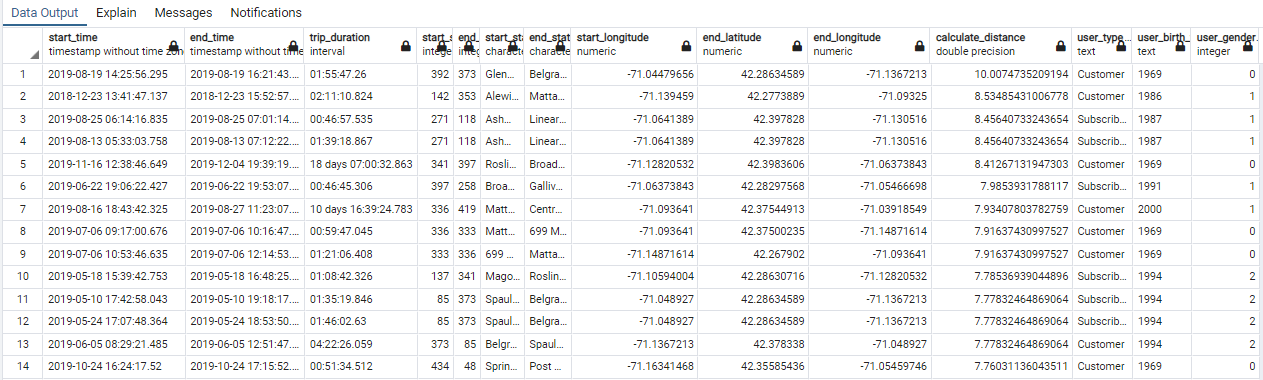
FROM all\_bluebikes

inner join start\_coordinates s on all\_bluebikes.start\_station\_id = s.start\_station\_id

inner join end\_coordinates e on all\_bluebikes.end\_station\_id = e.end\_station\_id

order by calculate\_distance desc

Output successfully run.



**What was the furthest relocation?**

**Furthest relocation was a 9.09 km move, where the bike was relocated from station id 111 at Packard Ave at Powerhouse Blvd in Somerville to station id 340 at Blue Hill Ave at Almont St in Boston.**

**SQL query for top 10 furthest relocations for bluebikes 2016.**

WITH bluebikes2016 as (

SELECT bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id

FROM bluebikes\_2016

order by bike\_id, end\_time asc),

bluebikes2016\_lag as (

Select \*, LAG(end\_station\_id,1) over (partition by bike\_id ) as previous\_end\_station\_id

from bluebikes2016),

relocations as (

SELECT \*,

case

when previous\_end\_station\_id <> start\_station\_id THEN 1 else 0 end as relocated

from bluebikes2016\_lag),

bluebikes2016\_lag\_distance as (

Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, s.latitude as start\_station\_latitude, s.longtitude as start\_station\_longitude,

p.latitude as previous\_latitude, p.longtitude as previous\_longitude,

calculate\_distance(s.latitude, s.longtitude, p.latitude, p.longtitude,'k')

from relocations

inner join bluebikes\_stations s on s.id = start\_station\_id

inner join bluebikes\_stations p on p.id = previous\_end\_station\_id

order by bike\_id, end\_time)

Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, start\_station\_latitude, start\_station\_longitude,

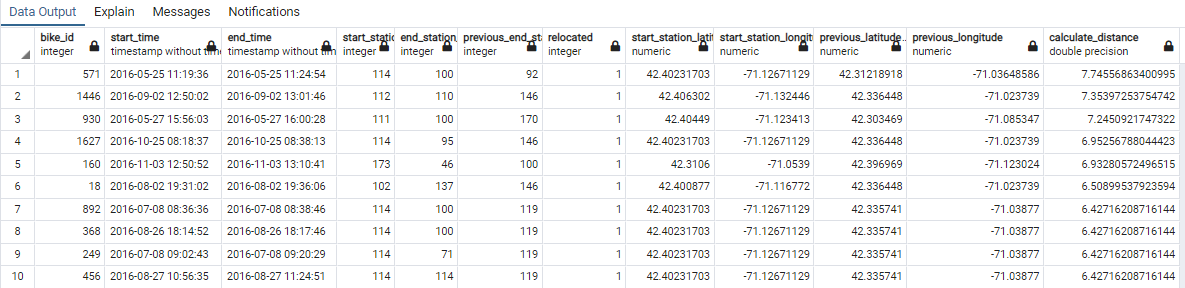
previous\_latitude, previous\_longitude, calculate\_distance

from bluebikes2016\_lag\_distance

order by calculate\_distance desc

limit 10

Output tested successfully prior to limiting rows to 10

Successfully run. Total query runtime: 2 min 45 secs. 10 rows affected 

**SQL query for top 10 furthest relocations for bluebikes\_2017 moderate geospatial**

WITH bluebikes2017 as (

SELECT bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id

FROM bluebikes\_2017

order by bike\_id, end\_time asc),

bluebikes2017\_lag as (

Select \*, LAG(end\_station\_id,1) over (partition by bike\_id ) as previous\_end\_station\_id

from bluebikes2017),

relocations as (

SELECT \*,

case

when previous\_end\_station\_id <> start\_station\_id THEN 1 else 0 end as relocated

from bluebikes2017\_lag),

bluebikes2017\_lag\_distance as (

Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, s.latitude as start\_station\_latitude, s.longtitude as start\_station\_longitude,

p.latitude as previous\_latitude, p.longtitude as previous\_longitude,

calculate\_distance(s.latitude, s.longtitude, p.latitude, p.longtitude,'k')

from relocations

inner join bluebikes\_stations s on s.id = start\_station\_id

inner join bluebikes\_stations p on p.id = previous\_end\_station\_id

order by bike\_id, end\_time)

Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, start\_station\_latitude, start\_station\_longitude,

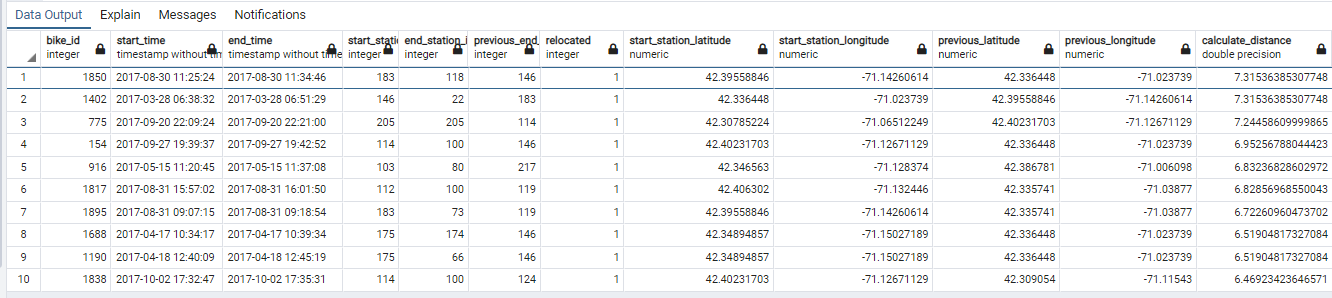
previous\_latitude, previous\_longitude, calculate\_distance

from bluebikes2017\_lag\_distance

order by calculate\_distance desc

limit 10

Successfully run. Total query runtime: 1 min 47 secs. 10 rows affected.



**SQL query for top 10 furthest relocations for bluebikes 2018 – moderate geospatial**

WITH bluebikes2018 as (

SELECT bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id

FROM bluebikes\_2018

order by bike\_id, end\_time asc),

bluebikes2018\_lag as (

Select \*, LAG(end\_station\_id,1) over (partition by bike\_id ) as previous\_end\_station\_id

from bluebikes2018),

relocations as (

SELECT \*,

case

when previous\_end\_station\_id <> start\_station\_id THEN 1 else 0 end as relocated

from bluebikes2018\_lag),

bluebikes2018\_lag\_distance as (

Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, s.latitude as start\_station\_latitude, s.longtitude as start\_station\_longitude,

p.latitude as previous\_latitude, p.longtitude as previous\_longitude,

calculate\_distance(s.latitude, s.longtitude, p.latitude, p.longtitude,'k')

from relocations

inner join bluebikes\_stations s on s.id = start\_station\_id

inner join bluebikes\_stations p on p.id = previous\_end\_station\_id

order by bike\_id, end\_time)

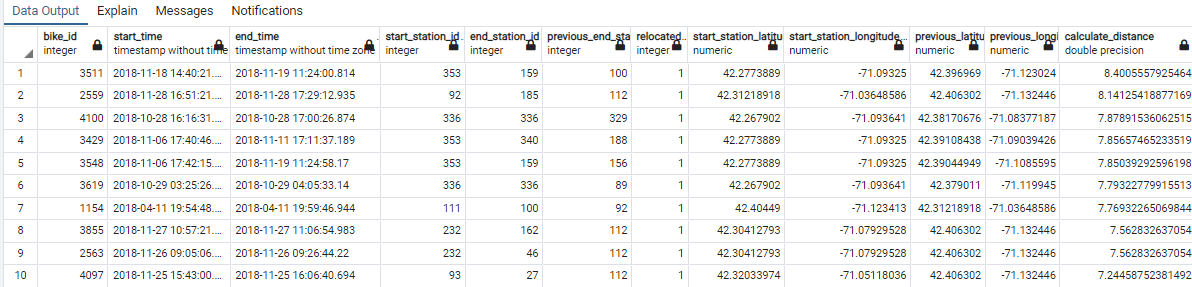
Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, start\_station\_latitude, start\_station\_longitude,

previous\_latitude, previous\_longitude, calculate\_distance

from bluebikes2018\_lag\_distance

order by calculate\_distance desc

limit 10

Successfully run. Total query runtime: 2 min 8 secs.10 rows affected. 

**SQL query for top 10 furthest relocations for bluebikes 2019 – moderate geospatial**

WITH bluebikes2019 as (SELECT bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id

FROM bluebikes\_2019

order by bike\_id, end\_time asc),

bluebikes2019\_lag as ( Select \*, LAG(end\_station\_id,1) over (partition by bike\_id ) as previous\_end\_station\_id

from bluebikes2019),

relocations as ( SELECT \*, case when previous\_end\_station\_id <> start\_station\_id THEN 1 else 0 end as relocated from bluebikes2019\_lag),

bluebikes2019\_lag\_distance as (

Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, s.latitude as start\_station\_latitude, s.longtitude as start\_station\_longitude,

p.latitude as previous\_latitude, p.longtitude as previous\_longitude,

calculate\_distance(s.latitude, s.longtitude, p.latitude, p.longtitude,'k')

from relocations

inner join bluebikes\_stations s on s.id = start\_station\_id

inner join bluebikes\_stations p on p.id = previous\_end\_station\_id

order by bike\_id, end\_time)

Select bike\_id, start\_time, end\_time, start\_station\_id, end\_station\_id, previous\_end\_station\_id, relocated, start\_station\_latitude, start\_station\_longitude,

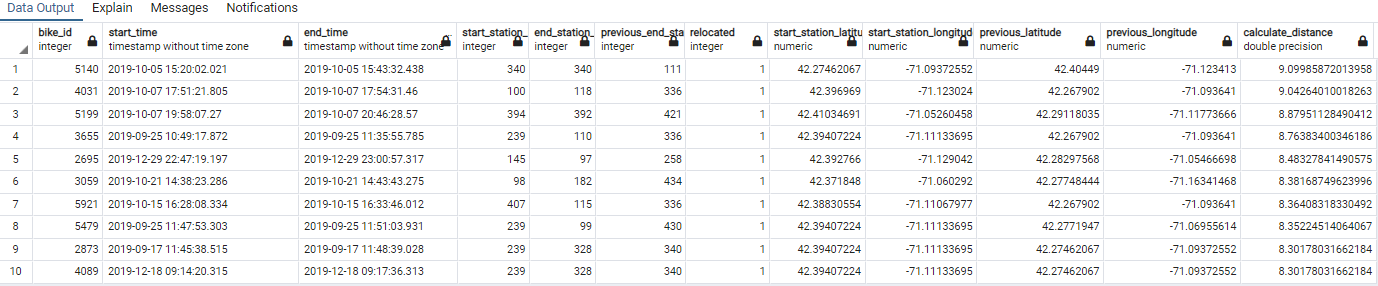
previous\_latitude, previous\_longitude, calculate\_distance

from bluebikes2019\_lag\_distance

order by calculate\_distance desc

limit 10

Successfully run. Total query runtime: 2 min 46 secs. 10 rows affected.



**How far is a typical journey?**

Extracting and Comparing data between bluebikes 2016 vs bluebikes 2019 in terms of average trip distance per month along with an overview of monthly trip counts, gender and user type variations for the two years.

Key Findings

A typical journey irrespective of gender for bluebike riders has been about 1.15 k distance. This was calculated from analyzing distances between the start and end stations of each ride, and compared rides data between genders from 2016 and 2019.

In boston , Typical bike ride distances based on gender from bluebikes data are



In chicago , Typical bike ride distances based on gender from Divvy bikes data are



**SQL query for top 10 furthest relocations for bluebikes\_2019**

**SQL for bluebikes2016**

With blue2016 AS (

Select bike\_id, user\_type, user\_gender, s.latitude, s.longtitude, e.latitude, e.longtitude, calculate\_distance(s.latitude, s.longtitude, e.latitude, e.longtitude,'k') as trip\_distance,

DATE\_PART('month', start\_time) AS months, DATE\_PART('Year', start\_time) AS years

from bluebikes\_2016

inner join bluebikes\_stations s on s.id = bluebikes\_2016.start\_station\_id

inner join bluebikes\_stations e on e.id = bluebikes\_2016.end\_station\_id)

Select count(bike\_id) as trip\_counts, months, avg(trip\_distance) as average\_distance, user\_gender, user\_type

From blue2016

group by blue2016.months, user\_gender, user\_type

**SQL for bluebikes2019**

With blue2019 AS(

Select bike\_id, user\_type, user\_gender, s.latitude, s.longtitude, e.latitude, e.longtitude, calculate\_distance(s.latitude, s.longtitude, e.latitude, e.longtitude,'k') as trip\_distance,

DATE\_PART('month', start\_time) AS months, DATE\_PART('Year', start\_time) AS years

from bluebikes\_2019

inner join bluebikes\_stations s on s.id = bluebikes\_2019.start\_station\_id

inner join bluebikes\_stations e on e.id = bluebikes\_2019.end\_station\_id)

Select count(bike\_id) as trip\_counts, months, avg(trip\_distance) as average\_distance, user\_gender, user\_type

From blue2019

group by blue2019.months, user\_gender, user\_type

With blue2019 as(

Select bike\_id, user\_type, user\_gender, s.latitude, s.longtitude, e.latitude, e.longtitude, calculate\_distance(s.latitude, s.longtitude, e.latitude, e.longtitude,'k') as trip\_distance,

DATE\_PART('month', start\_time) AS months, DATE\_PART('Year', start\_time) AS years

from bluebikes\_2019

inner join bluebikes\_stations s on s.id = bluebikes\_2019.start\_station\_id

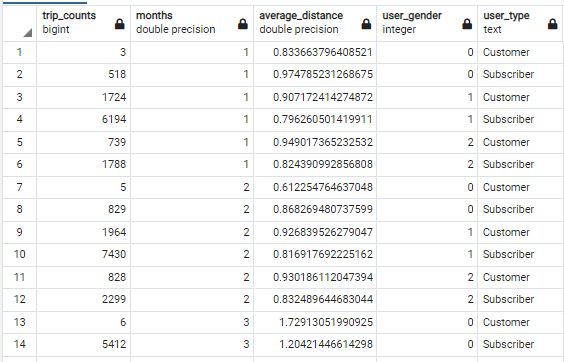
inner join bluebikes\_stations e on e.id = bluebikes\_2019.end\_station\_id)

Select count(bike\_id) as trip\_counts, months, avg(trip\_distance) as average\_distance, user\_gender, user\_type

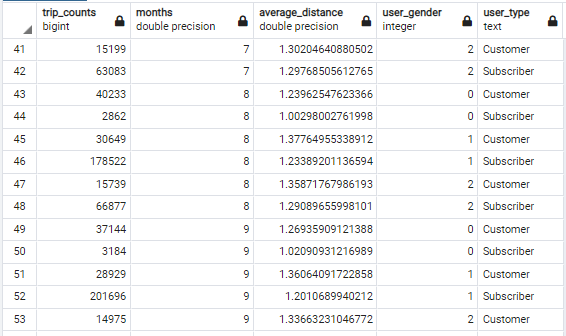
From blue2019

group by blue2019.months, user\_gender, user\_type

Output for Bluebikes2016 Successfully run. Total query runtime: 5 min 22 secs. 56 rows affected.



Output for Bluebikes2019 Successfully run. Total query runtime: 7 min 41 secs. 72 rows affected.



**Blue bikes record the gender of the hirer, but just uses 0, 1 and 2. If the gender proportion of riders in Boston (Blue Bikes) is similar to Chicago (Divvy bikes), can I come up with better labels for those numbers?**

**Key Findings**

**Gender column values as 0, 1, 2 does not convey direct information on what does each value mean. So, from comparison to divvybikes database, we can safely apply the assumptions that 0 = Female, 1 = Male, and 2 = Not revealed/ private. This gives direct understanding of what gender information is retained for each record in the database. Also, User birth year can easily be converted to reflect user age as of current date. Together these would give a quick idea of the demographics of the bike riders community and could be useful info for future analysis.**

SQL to pull up data from divvybikes database

WITH all\_divvybikes AS (

SELECT \*

FROM divvybikes\_2016

UNION ALL (

SELECT \* FROM divvybikes\_2017

UNION ALL

(SELECT \* FROM divvybikes\_2018

UNION ALL

SELECT \* FROM divvybikes\_2019) ) order by start\_time asc

)

select user\_type, gender, count(\*) as trips\_count, DATE\_PART('month', start\_time) AS month, DATE\_PART('Year', start\_time) AS year

from all\_divvybikes

group by user\_type, month, year

order by year desc

Based on this survey – we could arrange to insert new ‘updated gender’ definition columns in blue bikes public data tables