2/2/2021

PALLAVI DALVI

General Assemply Data Analytics Immersive

Divvy BikeShare Project

Process Document

Contents

[Introduction 2](#_Toc63519681)

[Quick Stats About Divvy Bikes 2](#_Toc63519682)

[Posing Questions 3](#_Toc63519683)

[Trends and descriptive analytics 3](#_Toc63519684)

[Easy questions - trends over time: Question 1: 3](#_Toc63519685)

[Moderate questions - trends over time: Question 2 4](#_Toc63519686)

[Harder questions - trends over time: Question 1 8](#_Toc63519687)

[Geospatial analytics 13](#_Toc63519688)

[Easy questions - working with geospatial: Question 1: 13](#_Toc63519689)

[Moderate questions - working with geospatial: Question 2: 20](#_Toc63519690)

[Harder questions - working with geospatial: Question 2: 24](#_Toc63519691)

# Introduction

A bicycle-sharing system, public bicycle scheme, is a service in which bicycles are made available for shared use to individuals on a short-term basis for a price or free. Many bike share systems allow people to borrow a bike from a "dock" and return it at another dock belonging to the same system. Docks are special bike racks that lock the bike, and only release it by computer control. The user enters payment information, and the computer unlocks a bike. The user returns the bike by placing it in the dock, which locks it in place.

Over the past decade, bicycle-sharing systems have been growing in number and popularity in cities across the world. Bicycle-sharing systems allow users to rent bicycles.

# Aim:

Some organizations don’t have dedicated business intelligence tools and Data Analyst will have to do data analysis using Excel, SQL, and the sheer power of intellect and will.

In this project, I will perform descriptive and geospatial analysis on the data provided **for US City - Chicago under Divvy Bikes.**

This is achieved by running SQL queries, downloading the results into Excel and doing analysis in Excel as well.

# Quick Stats About Divvy Bikes

**Data sets present related to Divvy Bikes are**

* Trips for year 2016
* Trips for year 2017
* Trips for year 2018
* Trips for year 2019
* Stations

**Number of bikes used per year:**

|  |  |  |
| --- | --- | --- |
| **Year** | **No of Bikes** | **Query** |
| 2016 | 5748 | SELECT COUNT(DISTINCT bikeid) FROM public.divvybikes\_2016; |
| 2017 | 6243 | SELECT COUNT(DISTINCT bikeid) FROM public.divvybikes\_2017; |
| 2018 | 6133 | SELECT COUNT(DISTINCT bikeid) FROM public.divvybikes\_2018; |
| 2019 | 6017 | SELECT COUNT(DISTINCT bikeid) FROM public.divvybikes\_2019; |

Number of Trips per year:

|  |  |  |
| --- | --- | --- |
| **Year** | **No of Trips** | **Query** |
| 2016 | 3595383 | SELECT COUNT(DISTINCT trip\_id) FROM public.divvybikes\_2016 |
| 2017 | 3829003 | SELECT COUNT(DISTINCT trip\_id) FROM public.divvybikes\_2017 |
| 2018 | 3603082 | SELECT COUNT(DISTINCT trip\_id) FROM public.divvybikes\_2018 |
| 2019 | 3818004 | SELECT COUNT(DISTINCT trip\_id) FROM public.divvybikes\_2019 |

# Posing Questions

## Trends and descriptive analytics

### Trends over time: Question 1:

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

Queries

SELECT date\_part('year',start\_time) as year ,date\_part('month',start\_time)as month, count(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2016

GROUP BY 1,2

ORDER BY 1,2;

SELECT date\_part('year',start\_time) as year ,date\_part('month',start\_time)as month, count(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2017

GROUP BY 1,2

ORDER BY 1,2;

SELECT date\_part('year',start\_time) as year ,date\_part('month',start\_time)as month, count(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2018

GROUP BY 1,2

ORDER BY 1,2;

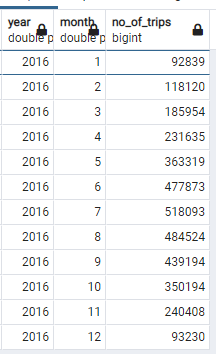
SELECT date\_part('year',start\_time) as year ,date\_part('month',start\_time)as month, count(trip\_id) as no\_of\_trips

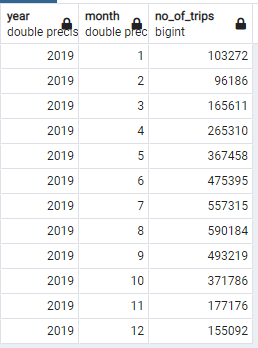
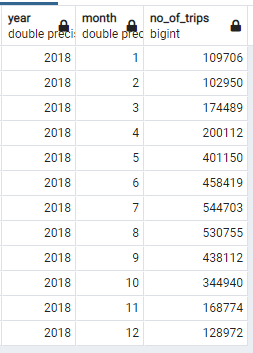
FROM public.divvybikes\_2019

GROUP BY 1,2

ORDER BY 1,2

Results:





Output File: 

Analysis in Excel:

### Trends over time: Question 2

Which stations are showing the greatest growth rates?

Answer:

Here there are 3 aspects to find growth rate by which data is retrieved for years 2016,2017,2018 and 2019.

1- To find Start Stations where Number of trips increasing each year

2- To find End Stations where Number of trips increasing each year

3- To find Start and End Station pairs where Number of trips increasing each year

*1- To find Start Stations where Number of trips increasing each year*

Query:

WITH trips\_FromStation\_2016 as (SELECT start\_station\_id, count(\*) as no\_of\_trips\_FromStation\_2016

FROM public.divvybikes\_2016

GROUP BY 1

ORDER BY 2 DESC),

trips\_FromStation\_2017 as (SELECT start\_station\_id, count(\*) as no\_of\_trips\_FromStation\_2017

FROM public.divvybikes\_2017

GROUP BY 1

ORDER BY 2 DESC),

trips\_FromStation\_2018 as (SELECT start\_station\_id, count(\*) as no\_of\_trips\_FromStation\_2018

FROM public.divvybikes\_2018

GROUP BY 1

BY 2 DESC),

trips\_FromStation\_2019 as (SELECT start\_station\_id, count(\*) as no\_of\_trips\_FromStation\_2019

FROM public.divvybikes\_2019

GROUP BY 1

ORDER BY 2 DESC)

SELECT start\_station\_id, name,

no\_of\_trips\_FromStation\_2016,

no\_of\_trips\_FromStation\_2017,

no\_of\_trips\_FromStation\_2018,

no\_of\_trips\_FromStation\_2019

FROM trips\_FromStation\_2016 as a

JOIN trips\_FromStation\_2017 as b USING (start\_station\_id)

JOIN trips\_FromStation\_2018 as c USING (start\_station\_id)

JOIN trips\_FromStation\_2019 as d USING (start\_station\_id)

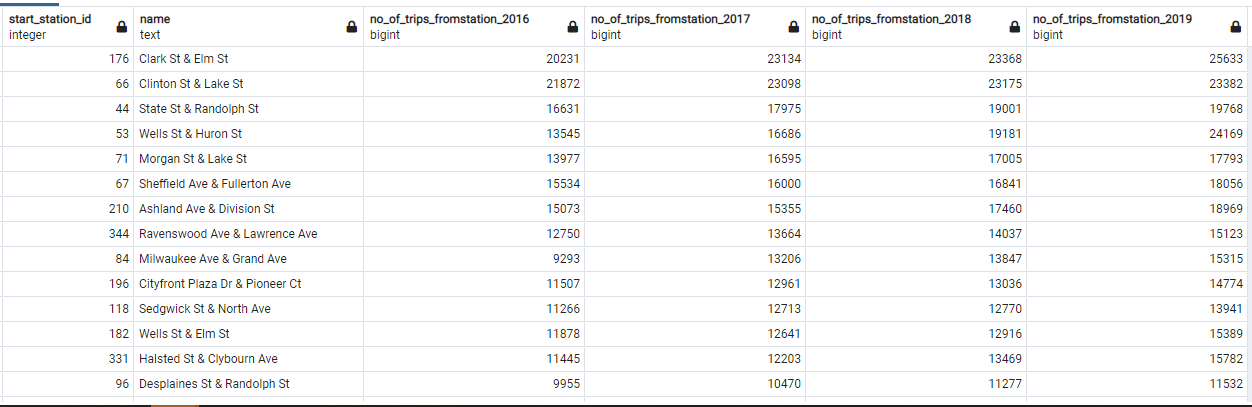
WHERE no\_of\_trips\_FromStation\_2019 >=no\_of\_trips\_FromStation\_2018

AND no\_of\_trips\_FromStation\_2018>=no\_of\_trips\_FromStation\_2017

AND no\_of\_trips\_FromStation\_2017 >=no\_of\_trips\_FromStation\_2016

Query explanation:

Output:



Result Set:



*2- To find End Stations where Number of trips increasing each year:*

Query:

WITH trips\_EndStation\_2016 as (SELECT end\_station\_id, count(\*) as no\_of\_trips\_EndStation\_2016

FROM public.divvybikes\_2016

GROUP BY 1

ORDER BY 2 DESC),

trips\_EndStation\_2017 as (SELECT end\_station\_id, count(\*) as no\_of\_trips\_EndStation\_2017

FROM public.divvybikes\_2017

GROUP BY 1

ORDER BY 2 DESC),

trips\_EndStation\_2018 as (SELECT end\_station\_id, count(\*) as no\_of\_trips\_EndStation\_2018

FROM public.divvybikes\_2018

GROUP BY 1

ORDER BY 2 DESC),

trips\_EndStation\_2019 as (SELECT end\_station\_id, count(\*) as no\_of\_trips\_EndStation\_2019

FROM public.divvybikes\_2019

GROUP BY 1

ORDER BY 2 DESC)

SELECT end\_station\_id,name,

no\_of\_trips\_EndStation\_2016,

no\_of\_trips\_EndStation\_2017,

no\_of\_trips\_EndStation\_2018,

no\_of\_trips\_EndStation\_2019

FROM trips\_EndStation\_2016 as a

JOIN trips\_EndStation\_2017 as b USING (end\_station\_id)

JOIN trips\_EndStation\_2018 as c USING (end\_station\_id)

JOIN trips\_EndStation\_2019 as d USING (end\_station\_id)

JOIN public.divvy\_stations ON end\_station\_id=id

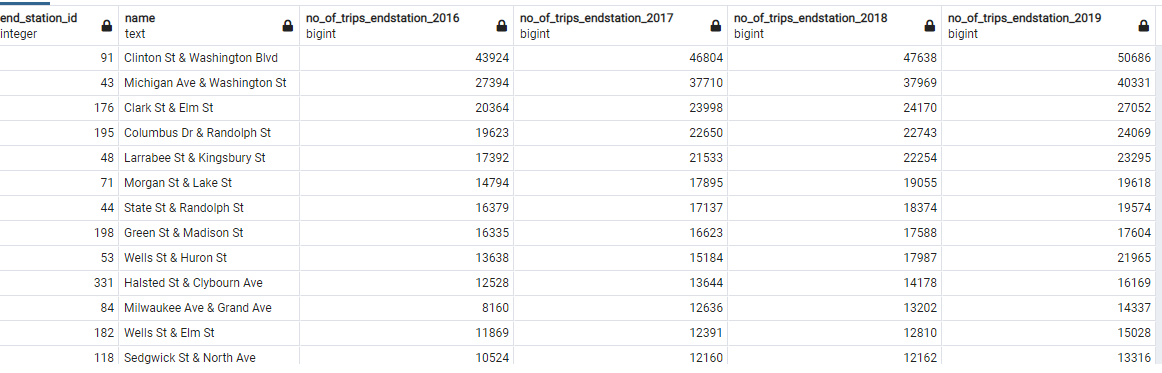
WHERE no\_of\_trips\_EndStation\_2019 >=no\_of\_trips\_EndStation\_2018

AND no\_of\_trips\_EndStation\_2018>=no\_of\_trips\_EndStation\_2017

AND no\_of\_trips\_EndStation\_2017 >=no\_of\_trips\_EndStation\_2016

Query explanation:

Output:



Result Set:



*3- To find Start and End Station pairs where Number of trips increasing each year*

Query:

WITH trips\_StartEndStation\_2016 as (SELECT start\_station\_id,end\_station\_id, count(\*) as no\_of\_trips\_StartEndStation\_2016

FROM public.divvybikes\_2016

GROUP BY 1,2

ORDER BY 3 DESC),

trips\_StartEndStation\_2017 as (SELECT start\_station\_id,end\_station\_id, count(\*) as no\_of\_trips\_StartEndStation\_2017

FROM public.divvybikes\_2017

GROUP BY 1,2

ORDER BY 3 DESC),

trips\_StartEndStation\_2018 as (SELECT start\_station\_id,end\_station\_id, count(\*) as no\_of\_trips\_StartEndStation\_2018

FROM public.divvybikes\_2018

GROUP BY 1,2

ORDER BY 3 DESC),

trips\_StartEndStation\_2019 as (SELECT start\_station\_id,end\_station\_id, count(\*) as no\_of\_trips\_StartEndStation\_2019

FROM public.divvybikes\_2019

GROUP BY 1,2

ORDER BY 3 DESC)

SELECT a.start\_station\_id,

a.end\_station\_id,

no\_of\_trips\_StartEndStation\_2016,

no\_of\_trips\_StartEndStation\_2017,

no\_of\_trips\_StartEndStation\_2018,

no\_of\_trips\_StartEndStation\_2019

FROM trips\_StartEndStation\_2016 as a

JOIN trips\_StartEndStation\_2017 as b ON a.start\_station\_id=b.start\_station\_id AND a.end\_station\_id=b.end\_station\_id

JOIN trips\_StartEndStation\_2018 as c ON b.start\_station\_id=c.start\_station\_id AND b.end\_station\_id=c.end\_station\_id

JOIN trips\_StartEndStation\_2019 as d ON c.start\_station\_id=d.start\_station\_id AND c.end\_station\_id=d.end\_station\_id

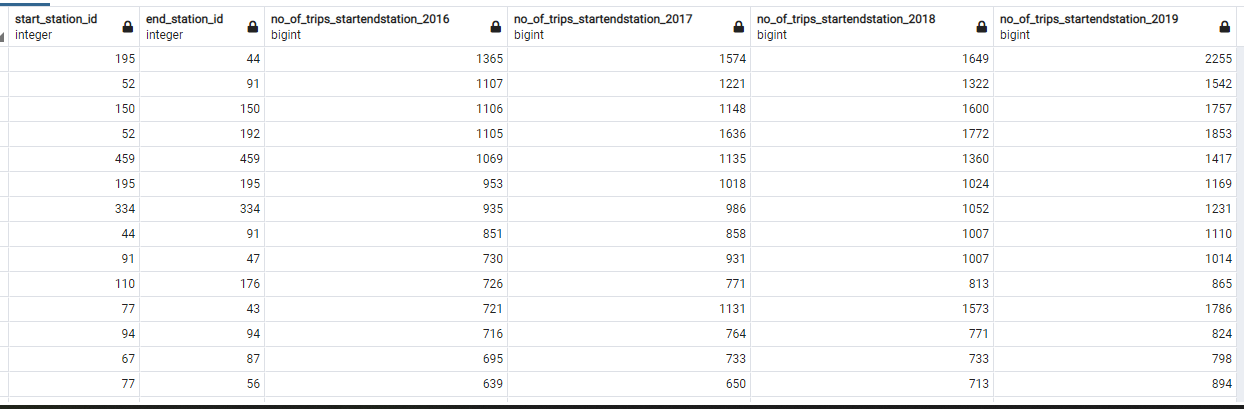
WHERE no\_of\_trips\_StartEndStation\_2019 >=no\_of\_trips\_StartEndStation\_2018

AND no\_of\_trips\_StartEndStation\_2018>=no\_of\_trips\_StartEndStation\_2017

AND no\_of\_trips\_StartEndStation\_2017 >=no\_of\_trips\_StartEndStation\_2016

ORDER BY 3 DESC

**Output**:



**Result:**



**Analysis in Excel:**

To calculate Percent Growth Rate:

Step 1-

Calculate the percent change from one period to another is calculated from the formula:

= (Present Value – Past Value)/Past Value \*100

Example: Station id – 176 (Clark St & Elm St) as starting station, below is the data.

|  |  |
| --- | --- |
| Year | Number of Trips |
| 2016 | 20231 |
| 2017 | 23134 |
| 2018 | 23368 |
| 2019 | 25633 |

Percent Change = (25633-20231)/ 20231 \*100

= 26.70%

Step 2-

The annual percentage growth rate is simply the percent growth divided by N, the number of years.

Here

= Percent Change /No of years

= 26.70%/3

=8.90%

Brief About Analysis:

### Trends over time: Question 1

Is there a difference in growth between holiday activity and commuting activity?

Answer:

Here below points have been taken into consideration for years 2016, 2017, 2018, 2019:

* Holidays in Chicago

- Christmas And New Year Holiday Period:

24 December, 25 December, 31 December, 1 January

- Weekends of every week (Saturday and Sunday)

* Commuting Activity to be considered – Monday to Friday every week

I have extracted data based on above conditions.

**Holiday Data for Year 2016**

Query:

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2016

WHERE TO\_CHAR(start\_time,'D')::NUMERIC IN (1,7)

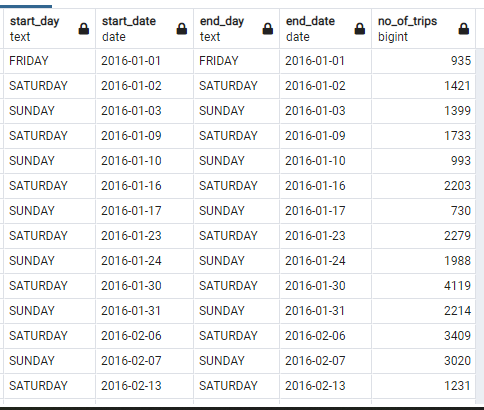
OR (date\_part('MONTH',start\_time) =1 AND date\_part('DAY',start\_time) = 1)

OR (date\_part('MONTH',start\_time) =12 AND date\_part('DAY',start\_time) IN(24,25,31))

GROUP BY 1,2,3,4

Order BY 2

Output sample:



Result:



**Holiday Data for Year 2017**

**Query:**

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2017

WHERE TO\_CHAR(start\_time,'D')::NUMERIC IN (1,7)

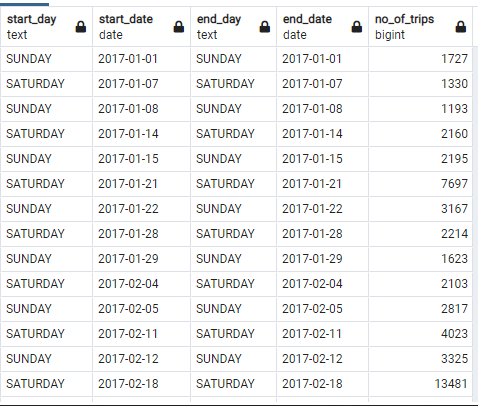
OR (date\_part('MONTH',start\_time) =1 AND date\_part('DAY',start\_time) = 1)

OR (date\_part('MONTH',start\_time) =12 AND date\_part('DAY',start\_time) IN(24,25,31))

GROUP BY 1,2,3,4

Order BY 2

**Output Sample:**



**Result:**

****

**Holiday Data for Year 2018**

**Query:**

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2018

WHERE TO\_CHAR(start\_time,'D')::NUMERIC IN (1,7)

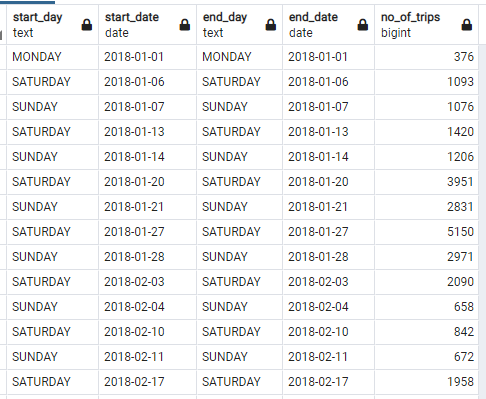
OR (date\_part('MONTH',start\_time) =1 AND date\_part('DAY',start\_time) = 1)

OR (date\_part('MONTH',start\_time) =12 AND date\_part('DAY',start\_time) IN(24,25,31))

GROUP BY 1,2,3,4

Order BY 2

**Output Sample:**



**Result**



**Holiday Data for Year 2019**

**Query:**

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2019

WHERE TO\_CHAR(start\_time,'D')::NUMERIC IN (1,7)

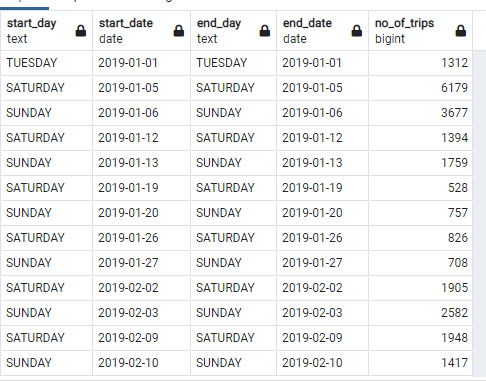
OR (date\_part('MONTH',start\_time) =1 AND date\_part('DAY',start\_time) = 1)

OR (date\_part('MONTH',start\_time) =12 AND date\_part('DAY',start\_time) IN(24,25,31))

GROUP BY 1,2,3,4

Order BY 2

**Output Sample:**



**Result:**

****

**Commuting Data for 2016**

**Query:**

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2016

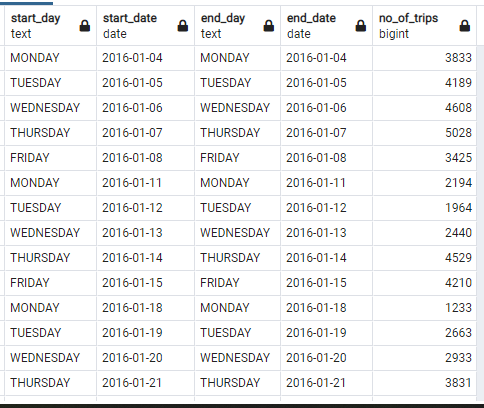
WHERE date(start\_time)::text NOT SIMILAR TO '2016-12-31|2016-12-25|2016-12-24|2016-01-01'

AND TO\_CHAR(start\_time,'D')::NUMERIC IN (2,3,4,5,6)

GROUP BY 1,2,3,4

Order BY 2

**Output Sample:**



**Result:**

****

**Commuting Data for 2017**

**Query:**

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2017

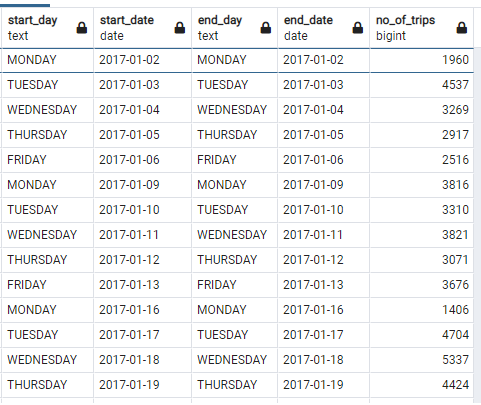
WHERE date(start\_time)::text NOT SIMILAR TO '2017-12-31|2017-12-25|2017-12-24|2017-01-01'

AND TO\_CHAR(start\_time,'D')::NUMERIC IN (2,3,4,5,6)

GROUP BY 1,2,3,4

Order BY 2

**Output Sample:**



**Result:**

****

**Commuting Data for 2018:**

**Query**

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2018

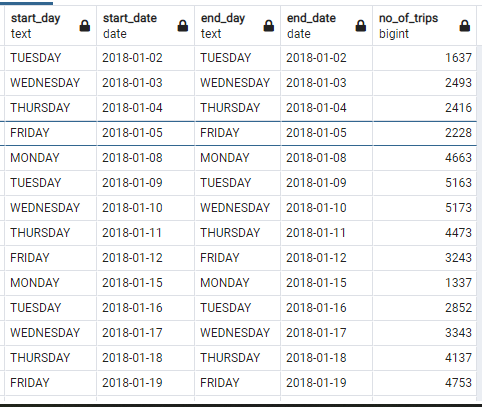
WHERE date(start\_time)::text NOT SIMILAR TO '2018-12-31|2018-12-25|2018-12-24|2018-01-01'

AND TO\_CHAR(start\_time,'D')::NUMERIC IN (2,3,4,5,6)

GROUP BY 1,2,3,4

Order BY 2

**Output Sample:**



**Result:**

****

**Commuting Data for 2019:**

**Query**

SELECT TO\_CHAR(start\_time,'DAY') as Start\_day,

(date\_trunc('DAY',start\_time))::DATE as Start\_Date,

TO\_CHAR(start\_time,'DAY') as End\_day,

(date\_trunc('DAY',start\_time))::DATE as End\_Date,

COUNT(trip\_id) as no\_of\_trips

FROM public.divvybikes\_2019

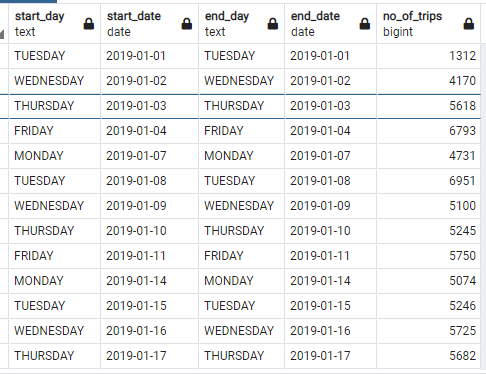
WHERE date(start\_time)::text NOT SIMILAR TO '2019-12-31|2019-12-25|2019-12-24|2019-01-01'

AND TO\_CHAR(start\_time,'D')::NUMERIC IN (2,3,4,5,6)

GROUP BY 1,2,3,4

Order BY 2

**Output Sample:**



**Result**

****

## Geospatial analytics

### orking with geospatial: Question 1:

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

**Answer:**

Steps 1:

Calculate the distance between distinct start\_station and end\_station pairs among all the trips

Steps 2:

Calculate the highest distance covered among stations. (i.e. longest journey in kilometers)

Step 3:

Get the details of the stations involved Step 2 (stations having longest journey in kilometers)

Step 4:

Find the trips which had longest journey.

Step 5:

Repeat Step 1 to Step 4 for all the years 2016,2017,2018,2019

**Year 2016:**

**Step 1: Query to find distance between distinct start\_station and end\_station pairs among all the trips**

SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2016

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

**Data Extracted:**



**Step 2: Query to find highest distance:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

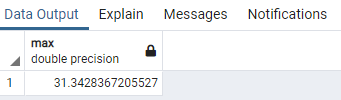
FROM public.divvybikes\_2016

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT MAX(distance)

FROM t1;



**Output**

**Step 3: Details of Stations with longest Journey:**

**Query to find station ids with longest journey:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2016

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

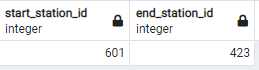
LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT start\_station\_id,end\_station\_id

FROM t1

WHERE distance=(SELECT MAX(distance) FROM t1);

**Output:**





**Step 4: Trips which had longest journey.**

**Query to find trips with longest journey using station ids.**

SELECT divvybikes\_2016.\*,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2016

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

WHERE start\_station\_id=601

AND end\_station\_id=423

**Result:**



**Year 2017:**

**Step 1: Query to find distance between distinct start\_station and end\_station pairs among all the trips**

SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2017

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

**Data Extracted:**



**Step 2: Query to find highest distance:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2017

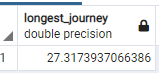
LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT MAX(distance)

FROM t1;

**Result**



**Step 3: Details of Stations with longest Journey:**

**Query to find station ids with longest journey:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2017

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

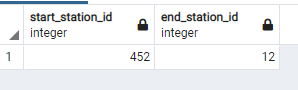
LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT start\_station\_id,end\_station\_id

FROM t1

WHERE distance=(SELECT MAX(distance) FROM t1);

**Output:**



**Step 4: Trips which had longest journey.**

**Query to find trips with longest journey using station ids.**

SELECT divvybikes\_2017.\*,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2017

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

WHERE start\_station\_id=452

AND end\_station\_id=12

**Output:**



**Year 2018**

**Step 1: Query to find distance between distinct start\_station and end\_station pairs among all the trips**

SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

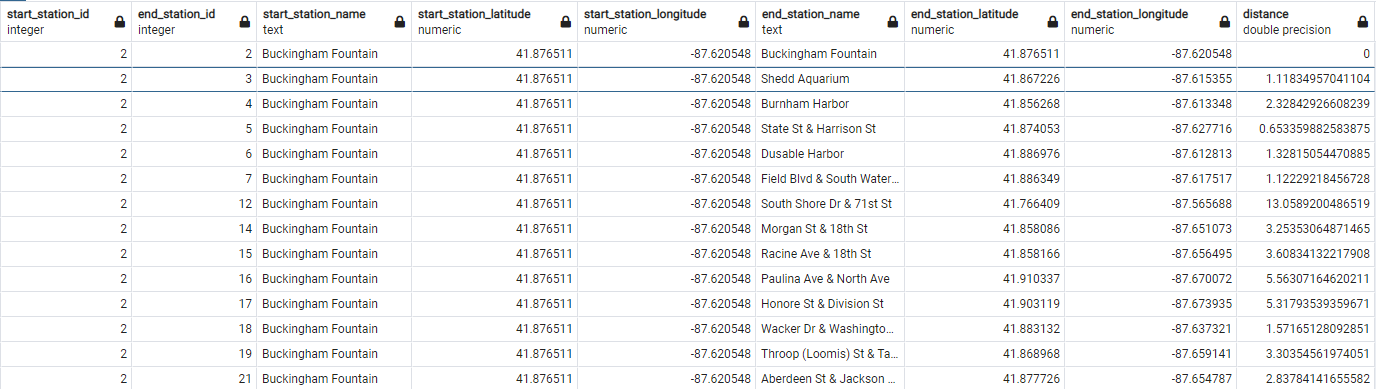
calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2018

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

**Data Extracted:**



****

**Step 2: Query to find highest distance:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2018

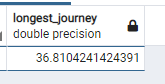
LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT MAX(distance)

FROM t1;

**Result**



**Step 3: Details of Stations with longest Journey:**

**Query to find station ids with longest journey:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2018

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

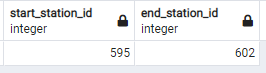
LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT start\_station\_id,end\_station\_id

FROM t1

WHERE distance=(SELECT MAX(distance) FROM t1);

**Output:**



**Step 4: Trips which had longest journey.**

**Query to find trips with longest journey using station ids.**

SELECT divvybikes\_2018.\*,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2018

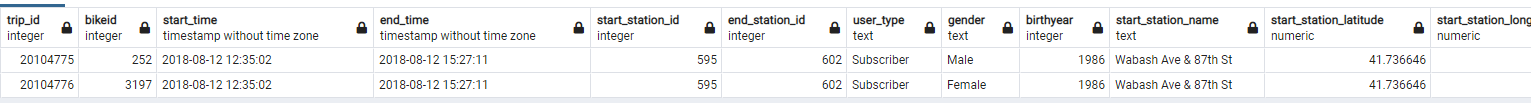
LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

WHERE start\_station\_id=595

AND end\_station\_id=602

**Output**



**Year 2019**

**Step 1: Query to find distance between distinct start\_station and end\_station pairs among all the trips**

SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

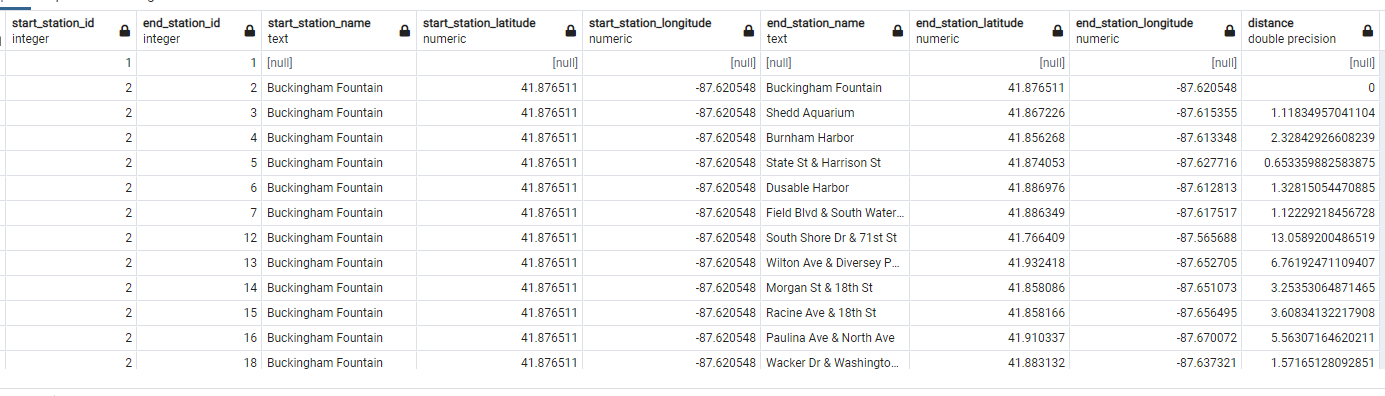
calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2019

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

**Data Extracted:**



**Step 2: Query to find highest distance:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2019

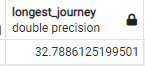
LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT MAX(distance)

FROM t1;

**Result:**



**Step 3: Details of Stations with longest Journey:**

**Query to find station ids with longest journey:**

WITH t1 as (SELECT DISTINCT start\_station\_id, end\_station\_id,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2019

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

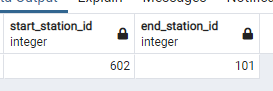
LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id)

SELECT start\_station\_id,end\_station\_id

FROM t1

WHERE distance=(SELECT MAX(distance) FROM t1);

**Result:**



**Step 4: Trips which had longest journey.**

SELECT divvybikes\_2019.\*,

a.name as start\_station\_name,a.latitude as start\_station\_latitude,a.longitude as start\_station\_longitude,

b.name as end\_station\_name,b.latitude as end\_station\_latitude,b.longitude as end\_station\_longitude,

calculate\_distance(a.latitude, a.longitude, b.latitude, b.longitude,'K') as distance

FROM public.divvybikes\_2019

LEFT JOIN public.divvy\_stations a ON start\_station\_id=a.id

LEFT JOIN public.divvy\_stations b ON end\_station\_id=b.id

WHERE start\_station\_id=602

AND end\_station\_id=101

**Result:**



### Working with geospatial: Question 2:

What was the furthest relocation?

Answer:

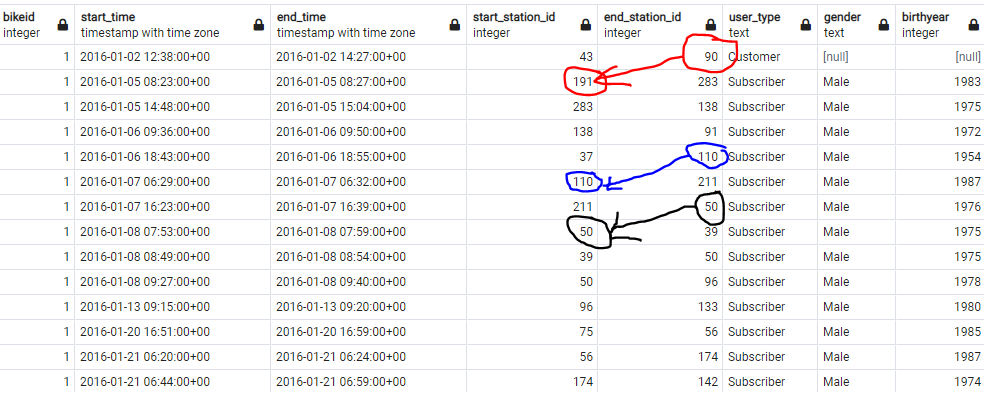
*Understanding the Data set - Divvy Bikes (Chicago)*

As per the problem statement, person ABC hires Bike with id 1234 from Station A and ends the journey at Station B. But next use of bike 1234 starts when person XYZ hires same bike from Station C. Here, we can see bike 1234 has been moved from last end station B to next start station C.

Movement from Station B to Station C is referred as relocation.

To find this data below approach has been followed.

1. Data of Years 2016, 2017,2018,2019 is analysed separately.
2. As there is data of more than 3 Million trips every year and more than 5000 bikes used, data is divided is subsets (based on bike ids) due to system limitations.
3. Illustration to explain the scenario:



For bikeid 1 , there is relocation of bike highlighted by red.

Whereas, stations highlighted with blue and black are not relocations.

1. Distance between end\_station and next start\_station will be calculated to find how far has the relocation happened?
2. Data extraction requires having to perform a calculation in a query that involved a value in the current row and a value in the previous row. The problem is that SQL queries perform operations on a row-by-row basis
3. To implement this solution, we’ll use the Row\_Number() function, the OVER() clause of a SELECT statement, a CTE and an inner join.

**Year 2016:**

**Query for bikeid<=1500**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2016

WHERE bikeid <=1500)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

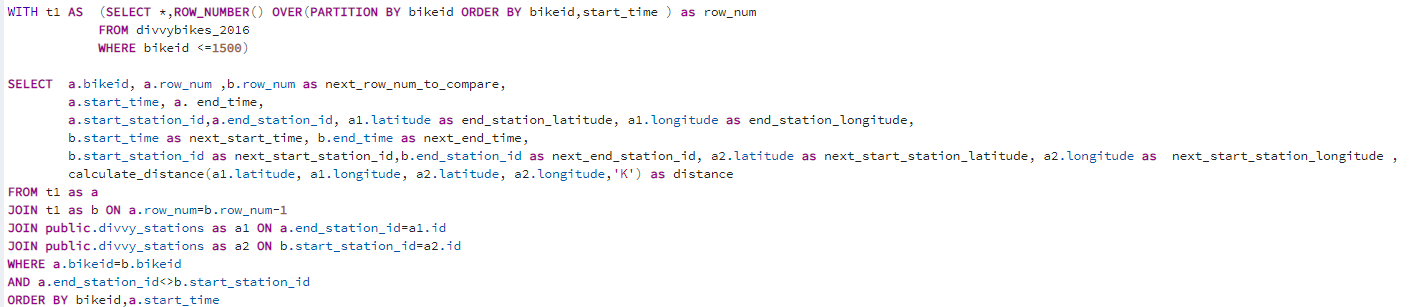
JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

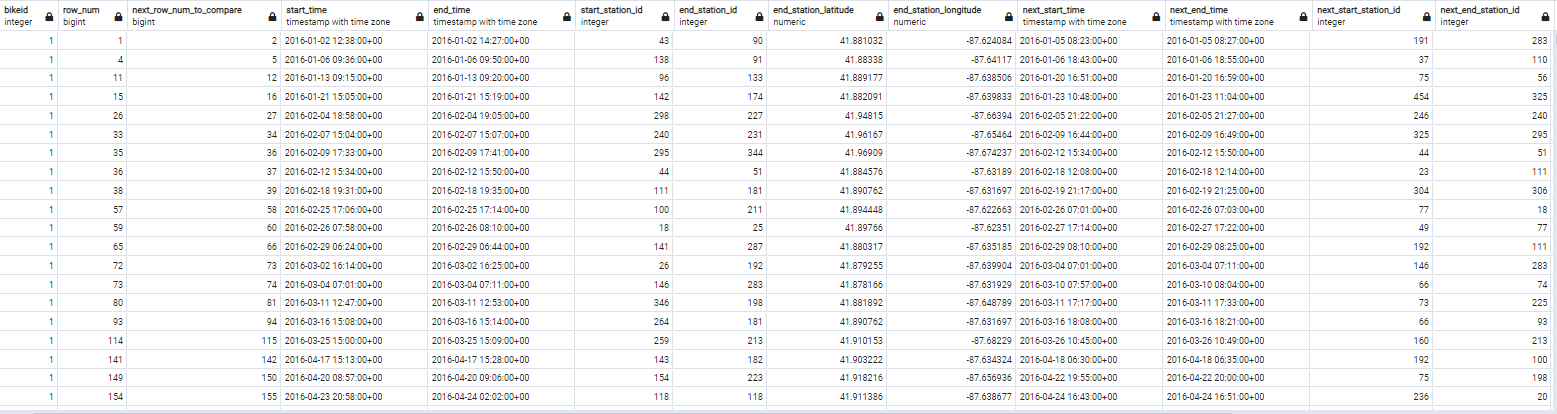
WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time



**Result:**



**Query for bikeid Between 1501 and 3000**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2016

WHERE bikeid BETWEEN 1501 AND 3000)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Query for bikeid Between 3001 and 4500**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2016

WHERE bikeid BETWEEN 3001 AND 4500)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Query for bikeid >4500**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2016

WHERE bikeid >4500)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Year 2017**

**Query for bikeid <4000**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2017

WHERE bikeid < 4000)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Query for bikeid >=4000**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2017

WHERE bikeid >= 4000)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Year 2018**

**Query for bikeid <=4000**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2018

WHERE bikeid <=4000)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Query for bikeid > 4000**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2018

WHERE bikeid >4000)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Year 2019**

**Query for bikeid <=4000**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2019

WHERE bikeid <=4000)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

**Query for bikeid >4000**

WITH t1 AS (SELECT \*,ROW\_NUMBER() OVER(PARTITION BY bikeid ORDER BY bikeid,start\_time ) as row\_num

FROM divvybikes\_2019

WHERE bikeid >4000)

SELECT a.bikeid, a.row\_num ,b.row\_num as next\_row\_num\_to\_compare,

a.start\_time, a. end\_time,

a.start\_station\_id,a.end\_station\_id, a1.latitude as end\_station\_latitude, a1.longitude as end\_station\_longitude,

b.start\_time as next\_start\_time, b.end\_time as next\_end\_time,

b.start\_station\_id as next\_start\_station\_id,b.end\_station\_id as next\_end\_station\_id, a2.latitude as next\_start\_station\_latitude, a2.longitude as next\_start\_station\_longitude ,

calculate\_distance(a1.latitude, a1.longitude, a2.latitude, a2.longitude,'K') as distance

FROM t1 as a

JOIN t1 as b ON a.row\_num=b.row\_num-1

JOIN public.divvy\_stations as a1 ON a.end\_station\_id=a1.id

JOIN public.divvy\_stations as a2 ON b.start\_station\_id=a2.id

WHERE a.bikeid=b.bikeid

AND a.end\_station\_id<>b.start\_station\_id

ORDER BY bikeid,a.start\_time

### Working with geospatial: Question 2:

How fast do people cycle?

Answer:

To find and analyse the data below approach has been followed.

1. Data of Years 2016, 2017,2018,2019 is analysed separately.
2. As there is data of more than 3 Million trips every year, data is retrieved for each month of each year
3. Seasons considered as below
   1. Winter - JAN FEB MAR
   2. Summer – JUL AUG SEPT
   3. Spring – APR MAY JUN
   4. Fall – OCT NOV DEC

Query:

**Year 2016**

SELECT DATE\_PART('Month',start\_time) as Month, COUNT(\*),

ROUND(AVG(calculate\_distance(b1.latitude, b1.longitude, b2.latitude, b2.longitude,'K')::numeric /

((EXTRACT(EPOCH FROM (end\_time-start\_time))::numeric)/3600)),2) AS average\_speed\_2016

FROM public.divvybikes\_2016

LEFT JOIN public.divvy\_stations b1 ON start\_station\_id=b1.id

LEFT JOIN public.divvy\_stations b2 ON end\_station\_id=b2.id

GROUP BY 1

**Year 2017**

SELECT DATE\_PART('Month',start\_time) as Month, COUNT(\*),

ROUND(AVG(calculate\_distance(b1.latitude, b1.longitude, b2.latitude, b2.longitude,'K')::numeric /

((EXTRACT(EPOCH FROM (end\_time-start\_time))::numeric)/3600)),2) AS average\_speed\_2017

FROM public.divvybikes\_2017

LEFT JOIN public.divvy\_stations b1 ON start\_station\_id=b1.id

LEFT JOIN public.divvy\_stations b2 ON end\_station\_id=b2.id

GROUP BY 1

**Year 2018**

SELECT DATE\_PART('Month',start\_time) as Month, COUNT(\*),

ROUND(AVG(calculate\_distance(b1.latitude, b1.longitude, b2.latitude, b2.longitude,'K')::numeric /

((EXTRACT(EPOCH FROM (end\_time-start\_time))::numeric)/3600)),2) AS average\_speed\_2018

FROM public.divvybikes\_2018

LEFT JOIN public.divvy\_stations b1 ON start\_station\_id=b1.id

LEFT JOIN public.divvy\_stations b2 ON end\_station\_id=b2.id

GROUP BY 1

**Year 2019**

SELECT DATE\_PART('Month',start\_time) as Month, COUNT(\*),

ROUND(AVG(calculate\_distance(b1.latitude, b1.longitude, b2.latitude, b2.longitude,'K')::numeric /

((EXTRACT(EPOCH FROM (end\_time-start\_time))::numeric)/3600)),2) AS average\_speed\_2019

FROM public.divvybikes\_2019

LEFT JOIN public.divvy\_stations b1 ON start\_station\_id=b1.id

LEFT JOIN public.divvy\_stations b2 ON end\_station\_id=b2.id

GROUP BY 1