Bikeshare Analysis (2016-2019)– Shaily S.

Trends and descriptive analytics

*** Q1:How many trips were there in each month of each year?*/**

-- Solving query for Divvybikes (2016-2019) and Union them

```
SELECT
	COUNT(trip_id) AS Monthly_Trip_Count,
	TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY') AS Trip_Month_Year
FROM divvybikes_2019
GROUP BY TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY')
UNION
SELECT
	COUNT(trip_id) AS Monthly_Trip_Count,
	TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY') AS Trip_Month_Year
FROM divvybikes_2018
GROUP BY TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY')
UNION
SELECT
	COUNT(trip_id) AS Monthly_Trip_Count,
	TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY') AS Trip_Month_Year
FROM divvybikes_2017
GROUP BY TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY')
UNION
SELECT
	COUNT(trip_id) AS Monthly_Trip_Count,
	TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY') AS Trip_Month_Year
FROM divvybikes_2016
GROUP BY TO_CHAR(DATE_TRUNC('month',start_time),'MM-YYYY')
ORDER BY Trip_Month_Year;
```

monthly_trip_count	trip_month_year
92839	Jan-16
111942	Jan-17
109706	Jan-18
103272	Jan-19
118120	Feb-16
166343	Feb-17
102950	Feb-18
96186	Feb-19
185954	Mar-16
153406	Mar-17
174489	Mar-18
165611	Mar-19
231635	Apr-16
268662	Apr-17
200112	Apr-18
265310	Apr-19
363319	May-16
345988	May-17

●Q2 Which stations are showing the greatest growth rates?*/

```
-- Using Ctes to find yearly growth rate and joining them in main query

With Trips19 AS

   (SELECT

     ds.id AS Station_Id, (ds.name) AS Station_Name,  Count(trip_id) AS Total_Trips_2019

FROM divvy_stations ds LEFT JOIN divvybikes_2019 db19 ON db19.start_station_id = ds.id

GROUP BY ds.id ),

Trips18 AS

   (SELECT

     ds.id AS Station_Id, (ds.name) AS Station_Name,  Count(trip_id) AS Total_Trips_2018

FROM divvy_stations ds LEFT JOIN divvybikes_2018 db18 ON db18.start_station_id = ds.id

GROUP BY ds.id ),


Trips17 AS

   (SELECT

     ds.id AS Station_Id, (ds.name) AS Station_Name, Count(trip_id) AS Total_Trips_2017

FROM divvy_stations ds LEFT JOIN divvybikes_2017 db17 ON db17.start_station_id = ds.id

GROUP BY ds.id ),
Trips16 AS
    (SELECT  ds.id AS Station_Id,  (ds.name) AS Station_Name, Count(trip_id) AS Total_Trips_2016
FROM divvy_stations ds LEFT JOIN divvybikes_2016 db16 ON db16.start_station_id = ds.id
GROUP BY ds.id )

SELECT
   Station_Id, Trips19.Station_Name AS Station_Name,  Total_Trips_2019,  Total_Trips_2018,
   Total_Trips_2017,  Total_Trips_2016   FROM Trips19 INNER JOIN Trips18  USING (Station_Id)
INNER JOIN Trips17 USING (Station_Id) INNER JOIN Trips16 USING (Station_Id)
ORDER BY station_id ;
```

station_id	station_name	total_trips_2019	total_trips_2018	total_trips_2017	total_trips_2016
456	2112 W Peterson Ave	506	456	562	500
101	63rd St Beach	1211	956	1045	1068
109	900 W Harrison St	6643	6187	6230	4813
21	Aberdeen St & Jackson Blvd	12140	12056	9835	9425
80	Aberdeen St & Monroe St	9460	9875	10540	10577
621	Aberdeen St & Randolph St	10428	6602	0	0
346	Ada St & Washington Blvd	7549	7811	8146	8480
341	Adler Planetarium	16735	14953	23615	21552
444	Albany Ave & 26th St	235	195	146	146

● Q3: Is there a difference in growth between holiday activity and commuting activity?

--ANS: Analysing it 2 ways 1)Holiday season being Dec Jan 2)Weekend vs Weekday

--Take 1 When we look in terms of months, during Dec and Feb, activity is low across all years.

--and this period corresponds to holiday period globally

-- Using Query used in Q1 to answer the Holiday activity and commuting activity

monthly_trip_count	trip_month_year
92839	Jan-16
93230	Dec-16
96186	Feb-19
102950	Feb-18
103272	Jan-19
109706	Jan-18
111942	Jan-17
118120	Feb-16
125396	Dec-17
128972	Dec-18

--Take 2 Weekend -->(Holiday), Weekday -->Commuting

```
SELECT  --start_time AS StartTime,
      COUNT(trip_id),
      TO_char(start_time,'day')
FROM divvybikes_2016
GROUP BY TO_char(start_time,'day')
LIMIT 50;
SELECT
   COUNT(trip_id) AS Day_Trip_Count,
   CASE
      WHEN TO_CHAR(start_time,'day') ILIKE ('%Saturday%') THEN 'WeekEnd'
      WHEN TO_CHAR(start_time,'day') ILIKE ('%Sunday%') THEN 'WeekEnd'
      ELSE 'WeekDay'
   END AS Week_Day_End
FROM divvybikes_2016
GROUP BY Week_Day_End
```

Geospatial

● Q4: What was the longest journey? What do we know about it?

/*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. */

```
SELECT  bike_id AS Bike_Id, start_time AS Start_Time, end_time AS End_Time,
        st.latitude AS StartLat, st.longtitude AS StartLon, ed.latitude as EndLat,
        ed.longtitude as EndLon,
        calculate_distance(st.latitude, st.longtitude, ed.latitude, ed.longtitude,'k') AS distance,
        st.name AS Start_Station, ed.name AS End_Station


FROM bluebikes_2019 bb
INNER JOIN bluebikes_stations st ON bb.start_station_id = st.id
INNER JOIN bluebikes_stations ed on bb.end_station_id = ed.id
ORDER BY distance DESC;
```

--Q5: ● What was the furthest relocation?

---2019 Bluebikes data using lag function to find previous_end_station

--Using calculate_distance to find distance

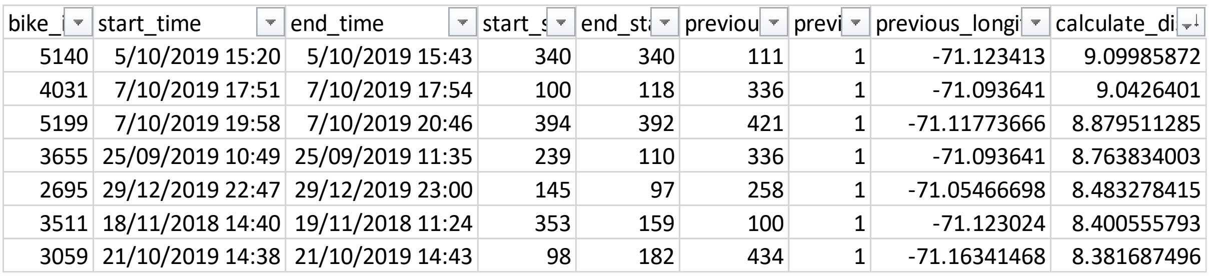
Furthest relocation was a 9.09 km move, where the bike was relocated from station id 111 at Packard Ave at Powerhouse Blvd in Somerville to station id 340 at Blue Hill Ave at Almont St in Boston.

--top 10 furthest relocations for bluebikes 2019

```
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 ,previous_latitude, previous_longitude, calculate_distance from
Bluebikes2019_lag_distance
order by calculate_distance desc
LIMIT 10;
```



bike_i	start_time	end_time	start_s	end_st	previou	previ	previous_longi	calculate_di
5140	5/10/2019 15:20	5/10/2019 15:43	340	340	111	1	-71.123413	9.09985872
4031	7/10/2019 17:51	7/10/2019 17:54	100	118	336	1	-71.093641	9.0426401
5199	7/10/2019 19:58	7/10/2019 20:46	394	392	421	1	-71.11773666	8.879511285
3655	25/09/2019 10:49	25/09/2019 11:35	239	110	336	1	-71.093641	8.763834003
2695	29/12/2019 22:47	29/12/2019 23:00	145	97	258	1	-71.05466698	8.483278415
3511	18/11/2018 14:40	19/11/2018 11:24	353	159	100	1	-71.123024	8.400555793
3059	21/10/2019 14:38	21/10/2019 14:43	98	182	434	1	-71.16341468	8.381687496

```
--top 10 furthest relocations for bluebikes 2018
--Using the same query as above for all the years from 2016-2018 to get the furthest relocations.
-- 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 ,previous_latitude, previous_longitude, calculate_distance from
Bluebikes2018_lag_distance

order by calculate_distance desc

LIMIT 10;
```

bike_i	start_time	end_time	start_s	end_st	previou	previ	previous_longi	calculate_di
3511	18/11/2018 14:40	19/11/2018 11:24	353	159	100	1	-71.123024	8.400555793
2559	28/11/2018 16:51	28/11/2018 17:29	92	185	112	1	-71.132446	8.141254189
4100	28/10/2018 16:16	28/10/2018 17:00	336	336	329	1	-71.08377187	7.878915361
3429	6/11/2018 17:40	11/11/2018 17:11	353	340	188	1	-71.09039426	7.856574652
3548	6/11/2018 17:42	19/11/2018 11:24	353	159	156	1	-71.1085595	7.850392926
3619	29/10/2018 3:25	29/10/2018 4:05	336	336	89	1	-71.119945	7.793227799
1154	11/04/2018 19:54	11/04/2018 19:59	111	100	92	1	-71.03648586	7.769322651
3855	27/11/2018 10:57	27/11/2018 11:06	232	162	112	1	-71.132446	7.562832637
2563	26/11/2018 9:05	26/11/2018 9:26	232	46	112	1	-71.132446	7.562832637
4097	25/11/2018 15:43	25/11/2018 16:06	93	27	112	1	-71.132446	7.244587524

--Q6 :● 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.

```
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
```

----BONUS (Optional)-----

/*Q: 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 you come up with better labels for those numbers? */

/*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