Project 2:

Initial Investigation

Database structure			
After examining the database, we determined that there where two types of tables:			
Core/Transaction tables Recording each ride as entry in the tables.		**Lookup/Reference tables** Holding details about each bike station	
bluebikes_2016 bluebikes_2017 bluebikes_2018 bluebikes_2019	divvybikes_2016 divvybikes_2017 divvybikes_2018 divvybikes_2019	bluebikes_stations	divvy_stations
See figure 1: Database tables for Blue bikes and Divvy bikes			

Key Fields	
Table	Column
bluebikes_stations	ID
divvy_stations	ID
bluebikes_2017, bluebikes_2018, bluebikes_2019	start_station_ID
bluebikes_2017, bluebikes_2018, bluebikes_2019	end_station_ID
divvybikes_2017, divvybikes_2018, divvybikes_2019	start_station_ID
divvybikes_2017, divvybikes_2018, divvybikes_2019	end_station_ID

Early identified issues			
The two organisation's table structure differed slightly with column names, column order and in some cases data types. Meaning queries would have to be tailored to combined data from both sets.			
bluebikes_2019		**divvybikes_2019**	
bike_id		bikeid	
user_birth_year		birthyear	
user_gender	integer	gender	text
absent		trip_id	
bluebikes_stations		**divvy_stations**	
longtitude	*setup misspelling*	longitude	
total_docks		docks	
number		*absent*	
district		*absent*	
public		*absent*	

Data cleaning	
Table structures	
bluebikes_stations	
public	found to hold the same values for every entry and thus ignored
longtitude	renamed to "longitude" in query results to prevent further errors
total_docks	renamed to "docks" in query results to prevent confusion

bluebikes_2019	
user_birth_year	renamed to “birthyear” in query results to prevent confusion
user_gender	renamed to “gender” in query results to prevent confusion
The above was also applied to the tables for 2017 and 2018	
divvybikes_2019	
gender	converted to integer during query
The above was also applied to the tables for 2017 and 2018	

Elements: Null values		
See query section for related queries		
Table	**Column**	**Number**
bluebikes_stations	ID	3
bluebikes_2018	birthyear	9529
divvybikes_2017	user_birth_year	836758
divvybikes_2018	user_birth_year	555209
divvybikes_2019	user_birth_year	538751

SQL

Trends and descriptive analytics

Trends over time

Question 1: How many different users were there in each month of each year?

Query 1. – Blues bikes and Divvy bikes trip count

This query unions all data from Blue Bikes and Divvy bikes from 2017-2018.
Date parts are converted into separate measures for month and year.
Seasons are introduced to add a further dimension to our investigation.
Originally the data for Blue Bikes and Divvy were queried separately until it was found that the UNION returned the two results in ample time. As such the Org column was added to distinguish the two datasets

```
-- Trends and descriptive analytics
-- How many different users were there in each month of each year?
-- Number of users over 2017 to 2019 Blue bikes and Divvy bikes
-- Project team: Hunt J, Khaoua H, Leung M, Moody C FEB 2021
-- Author: Leung M 2 Feb 2021
-- Mod: Hunt J, include season, 2 Feb 2021
-- Mod: Hunt J, Union both tables, 3 Feb 2021
-- Date last modified: 3 Feb 2021
WITH
bluebikes_Join as
(SELECT *
FROM bluebikes_2017
UNION ALL
SELECT *
FROM bluebikes_2018
UNION ALL
SELECT *
FROM bluebikes_2019),
divvy_Join as
(SELECT *
FROM divvybikes_2017
UNION ALL
SELECT *
```

```
FROM divvybikes_2018
UNION ALL
SELECT *
FROM divvybikes_2019),
bluebikes_table (year,month,user_type,trips_bluebike) AS
(SELECT
date_part('year', start_time) as year,
date_part('month',start_time) as month,
user_type,
count(*) as trips_bluebike
FROM bluebikes_Join
GROUP BY year,month,user_type),
divvy_table (year,month,user_type,trips_divvy) as
(SELECT
date_part('year', start_time) as year ,
date_part('month', start_time) as month,
user_type,
count(*) as trips_divvy
FROM divvy_Join
GROUP BY year,month,user_type)

SELECT year,
CASE
WHEN month BETWEEN 03 AND 05 THEN 'Spring'
WHEN month BETWEEN 06 AND 08 THEN 'Summer'
WHEN month BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
month,
'Bluebike' as org,
user_type,
trips_bluebike as trips
FROM bluebikes_table
UNION
SELECT year,
CASE
WHEN month BETWEEN 03 AND 05 THEN 'Spring'
WHEN month BETWEEN 06 AND 08 THEN 'Summer'
WHEN month BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
month,
'Divvy' as org,
user_type,
trips_divvy
FROM divvy_table

/*
SELECT *
FROM bluebikes_table
LEFT JOIN divvy_table using(month,year)
ORDER BY year,month asc
*/
```

Query Results Sample

year	season	month	org	user_type	trips
2017	Autumn	9	Bluebike	Customer	22962
2017	Autumn	9	Divvy	Customer	116191
2017	Autumn	10	Bluebike	Customer	21232
2017	Autumn	10	Divvy	Customer	58809

2017	Autumn	11	Bluebike	Customer	8316
2017	Autumn	11	Divvy	Customer	12703
2017	Spring	3	Bluebike	Customer	1905
2017	Spring	3	Divvy	Customer	12495
2017	Spring	4	Bluebike	Customer	22538

Trends over time

Question 2: Is the subscription side of these businesses growing?

Query 2.1 – Divvy Bikes

This query unions all data from Divvy bikes from 2017-2018.
Date parts are converted into separate measures for month and year.
Seasons are introduced to add a further dimension to our investigation.
Date is converted into Month number.
Dependent users are ignored as there are so few of them and they only appear in 2017

```
-- Trends and descriptive analysis
-- Is the subscription side of these businesses growing?
-- Monthly summary over 2017 to 2019 for Divvy Bikes
-- Project Team: Hunt J, Khaoua H, Leung M, Moody C FEB 2021
-- Authors: Khaoua H, Leung M 2 Feb 2021
-- Mod: Hunt J, Simplified Group By, 2 Feb 2021
-- Mod: Hunt J, Changed to include Season, 2 Feb 2021
-- Mod: Hunt J, Changed where clause restriction to exclude Dependents and
add user_type column, 3 Feb 2021
-- Date last modified: 3 Feb 2021
WITH
bikes_Join as
(select * from divvybikes_2017
UNION
select * from divvybikes_2018
UNION
select * from divvybikes_2019),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
user_type,
count(user_type) as total_users
FROM bikes_Join
WHERE user_type != 'Dependent'
GROUP BY year,season,month,user_type)

SELECT *
FROM bikes_table
ORDER BY year,month ASC
```

Query Results Sample

year	season	month	user_type	total_users
2017	Winter	1	Customer	5315
2017	Winter	1	Subscriber	106626
2017	Winter	2	Customer	23585

2017	Winter	2	Subscriber	142756
2017	Spring	3	Customer	12495
2017	Spring	3	Subscriber	140910
2017	Spring	4	Customer	61247
2017	Spring	4	Subscriber	207407
2017	Spring	5	Customer	82319

Query 2.2 – Blue Bikes

This query unions all data from Blue bikes from 2017-2018 for investigation
Date parts are converted into separate measures for month and year.
Seasons are introduced to add a further dimension to our investigation.
Date is converted into Month number.

```
-- Trends and descriptive analysis
-- Is the subscription side of these businesses growing?
-- Monthly summary over 2017 to 2019 for Blue Bikes
-- Project Team: Hunt J, Khaoua H, Leung M, Moody C FEB 2021
-- Authors: Khaoua H, Leung M
-- Mod: Changed to include Season, Hunt J 3 Feb 2021
-- Date written:  2 Feb 2021
-- Mod: Hunt J, Changed where clause restriction and add user_type column, 3
Feb 2021
-- Date last modified: 3 Feb 2021
WITH bikes_Join as
(SELECT * FROM bluebikes_2017
UNION ALL
SELECT * FROM bluebikes_2018
UNION ALL
SELECT * FROM bluebikes_2019),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
user_type,
count(user_type) as total_users
FROM bikes_Join
GROUP BY year,season,month,user_type)

SELECT *
FROM bikes_table
ORDER BY year,month ASC
```

Query Results Sample

year	season	month	user_type	total_users
2017	Winter	1	Subscriber	18437
2017	Winter	1	Customer	1081
2017	Winter	2	Subscriber	16211
2017	Winter	2	Customer	1061
2017	Spring	3	Subscriber	29348
2017	Spring	3	Customer	1905
2017	Spring	4	Customer	22538
2017	Spring	4	Subscriber	77322
2017	Spring	5	Subscriber	105602

Trends over time

Question 3: Is there a difference in growth between holiday activity and commuting activity?

Query 3.1 – Divvy Bikes

This query unions all data from divvy bikes from 2017-2018 for investigation
Date parts are converted into separate measures for year, month and day.
Seasons are introduced to add a further dimension to our investigation.
Day name and weekend were introduced to help determine the impact of holidays
Looking at all user types
Dependent users are ignored as there are so few of them and they only appear in 2017

```
-- Trends and descriptive analysis
-- Is there a difference in growth between holiday activity and commuting
activity?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Daily rental over 2017 to 2019 Divvy Bikes
-- Author: Hunt J 2 Feb 2021
-- Mod: Added seasons 2 Feb 2021
-- Mod: Added weekend 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH bikes_Join as
(SELECT * FROM divvybikes_2017
UNION ALL
SELECT * FROM divvybikes_2018
UNION ALL
SELECT * FROM divvybikes_2019),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
date_part('day',start_time) as day,
rtrim(to_char(start_time, 'day')) AS day_name,
CASE
 WHEN rtrim(to_char(start_time, 'day')) = 'sunday' THEN TRUE
 WHEN rtrim(to_char(start_time, 'day')) = 'saturday' THEN TRUE
 ELSE FALSE
END AS Weekend,
count(user_type) as total_users
FROM bikes_Join
WHERE user_type != 'Dependent'
GROUP BY year,season,month,day,day_name,weekend)

SELECT *
FROM bikes_table
ORDER BY year,month,day ASC
```

Query Results Sample

year	season	month	day	day_name	weekend	total_users
2017	Winter	1	1	sunday	TRUE	1727
2017	Winter	1	2	monday	FALSE	1960
2017	Winter	1	3	tuesday	FALSE	4537
2017	Winter	1	4	wednesday	FALSE	3269
2017	Winter	1	5	thursday	FALSE	2917
2017	Winter	1	6	friday	FALSE	2516

2017	Winter	1	7	saturday	TRUE	1330
2017	Winter	1	8	sunday	TRUE	1193
2017	Winter	1	9	monday	FALSE	3816

Query 3.2 – Blue Bikes

This query unions all data from blue bikes from 2017-2018 for investigation
Date parts are converted into separate measures for year, month and day.
Seasons are introduced to add a further dimension to our investigation.
Day name and weekend were introduced to help determine the impact of holidays
Looking at all user types

```
-- Trends and descriptive analysis
-- Is there a difference in growth between holiday activity and commuting
activity?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Daily rental over 2017 to 2019 Blue Bikes
-- Author: Hunt J 2 Feb 2021
-- Mod: Added seasons 2 Feb 2021
-- Mod: Added weekend 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH bikes_Join as
(SELECT * FROM bluebikes_2017
UNION ALL
SELECT * FROM bluebikes_2018
UNION ALL
SELECT * FROM bluebikes_2019),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
date_part('day',start_time) as day,
rtrim(to_char(start_time, 'day')) AS day_name,
CASE
 WHEN rtrim(to_char(start_time, 'day')) = 'sunday' THEN TRUE
 WHEN rtrim(to_char(start_time, 'day')) = 'saturday' THEN TRUE
 ELSE FALSE
END AS Weekend,
count(user_type) as total_users
FROM bikes_Join
/* WHERE user_type = 'Subscriber'  */
GROUP BY year,season,month,day,day_name,weekend)

SELECT *
FROM bikes_table
ORDER BY year,month,day ASC
```

Query Results Sample

year	season	month	day	day_name	weekend	total_users
2017	Winter	1	1	sunday	TRUE	481
2017	Winter	1	2	monday	FALSE	802
2017	Winter	1	3	tuesday	FALSE	651

2017	Winter	1	4	wednesday	FALSE	1534
2017	Winter	1	5	thursday	FALSE	1330
2017	Winter	1	6	friday	FALSE	836
2017	Winter	1	7	saturday	TRUE	106
2017	Winter	1	8	sunday	TRUE	111
2017	Winter	1	9	monday	FALSE	392

Geospatial

Question 4: What was the longest journey? What do we know about it?

Query 4.1 Divvy bikes

To use the calculate distance function we needed to have the coordinates of both the starting point and the ending point in the same record. To gain this result we first query the coordinates for the starting point via a CTE. We then reference that CTE and query the coordinates for the ending point. Once both coordinates are available, we can calculate the distance.
Initial runs of this query proved difficult as the database was not able to return the results.
We then realised that all we needed to answer the question was the MAX distance overall.
Results proved successful for the MAX distance for each month.
In an endeavour to gain some further insight into where the distance was travelled from the start station id was introduced.
Given issues with running the query it was decided to only investigate 2019
The query below is for the Divvy bikes

```
-- Geospatial
-- What was the longest journey? What do we know about it?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Daily rental over 2019 Divvy Bikes
-- Provide latitude and longitude for both stations
-- Calculate the distance between those stations
-- Author: Hunt J 2 Feb 2021
-- Mod: Stripped the query back compared to earlier versions as other details
can be added in Excel 2 Feb 2021
-- Mod: Added seasons 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH
L1_bikes_2019 AS
(SELECT
b.bikeid,
b.start_time,
b.start_station_id,
b.end_station_id,
s.latitude as ss_latitude,
s.longtitude as ss_longitude
FROM divvybikes_2019 b
JOIN divvybikes_stations s ON b.start_station_id = s.id
WHERE b.start_station_id != b.end_station_id /* AND
(date_part('month',start_time) BETWEEN 06 AND 08)*/),

SELECT
date_part('month',b.start_time) AS month,
b.start_station_id,
/* b.end_station_id, */
MAX(calculate_distance(b.ss_latitude, b.ss_longitude,
                       e.latitude, e.longtitude,
                       'K')) AS longest_ride
FROM L1_bikes_2019 b
```

```
JOIN divvybikes_stations e ON b.end_station_id = e.id
GROUP BY month,b.start_station_id
ORDER BY longest_ride DESC
```

Query Results Sample

month	start_station_id	max
8	392	16.10546747
8	271	13.6092684
11	341	13.53888211
6	397	12.8512446
8	336	12.76866089
7	333	12.7401695
7	336	12.7401695
5	137	12.52933752
6	373	12.5180001

Query 4.2 Blue bikes

See notes above for Query 4.1 Divvy bikes.
The query below is for the Blue bikes

```
-- Geospatial
-- What was the longest journey? What do we know about it?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Daily rental over 2019 Blue Bikes
-- Provide latitude and longitude for both stations
-- Calculate the distance between those stations
-- Author: Hunt J 2 Feb 2021
-- Mod: Stripped the query back compared to earlier versions as other details
can be added in Excel 2 Feb 2021
-- Mod: Added seasons 3 Feb 2021
-- Mod: Added weekend 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH
L1_bbikes_2019 AS
(SELECT
b.start_time,
b.start_station_id,
b.end_station_id,
s.latitude as ss_latitude,
s.longtitude as ss_longitude
FROM bluebikes_2019 b
JOIN bluebikes_stations s ON b.start_station_id = s.id
WHERE b.start_station_id != b.end_station_id /* AND
(date_part('month',start_time) BETWEEN 06 AND 08)*/)

SELECT
date_part('month',b.start_time) AS month,
b.start_station_id,
/* b.end_station_id, */
MAX(calculate_distance(b.ss_latitude, b.ss_longitude,
                       e.latitude, e.longtitude,
                       'K')) as Longest_ride
FROM L1_bbikes_2019 b
JOIN bluebikes_stations e ON b.end_station_id = e.id
GROUP BY month,b.start_station_id
ORDER BY longest_ride DESC
/* LIMIT 10 */
```

Query Results Sample

month	start_station_id	longest_ride	longest
8	595	36.81042414	TRUE
7	417	30.25589412	FALSE
3	596	29.68381653	FALSE
5	598	29.64951753	FALSE
5	247	29.38796744	FALSE
9	596	29.38796744	FALSE
7	121	29.32668054	FALSE

Geospatial

Question 5: How often do bikes need to be relocated?

Query 5.1: Divvy Bikes

The following query uses the LAG Windows function to grab the last end station so that we can compare it to the current start station. If there are different then it is determined that the bike has been moved by a vehicle (possibly overnight). Once this is determined, we can count the number of moves. It was then decided to add the number of all rides so that we could easily compared the difference and produce a percentage of moved bikes.

Note: The query could simply be adjusted to run separately for the years 2017 and 2018

```
-- Geospatial
-- How often do bikes need to be relocated?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Daily rental Divvy Bikes 2019
-- Was the bike moved by transport vechicle or not
-- Author: Hunt J 2 Feb 2021
-- Mod: Add the number of rides to be able to compare, Hunt J 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH
bike_cte AS        /* Grab the rides in order */
(SELECT Distinct bikeid,
start_time,
start_station_id as start,
end_station_id as stop
FROM divvybikes_2019
ORDER BY bikeid,start_time
),
delay_cte AS      /* Grab the start position and the previous end position */
(SELECT
bikeid,
start_time,
start,
LAG(stop, 1) OVER( Partition BY 1) as previous_stop
FROM bike_cte
),
moved_bike AS     /* Was the bike moved or not */
(SELECT
bikeid,
start_time,
start,
previous_stop,
start !=previous_stop AS Moved
FROM delay_cte
),
total_rides AS      /* All the rides */
(SELECT
date_part('month',start_time) AS month,
count(*) AS number_of_rides
```

```
FROM divvybikes_2019
GROUP BY month
ORDER BY month
),
-- Count the moves
total_moves AS (SELECT
date_part('month',start_time) AS month,
count(CASE WHEN moved THEN 1 END) AS number_of_moves
FROM moved_bike
GROUP BY month
ORDER BY month
)
-- Grab the monthly moves and total rides

SELECT
m.month,
number_of_moves,
number_of_rides
FROM total_rides r
JOIN total_moves m USING(month)
```

Query Results Sample

month	number_of_moves	number_of_rides
1	12056	103272
2	8737	96186
3	14439	165611
4	19731	265310
5	27840	367458
6	32136	475395
7	37510	557315

Query 5.2: Blue bikes

See notes for Query 5.1: Divvy bikes

```
-- Geospatial
-- How often do bikes need to be relocated?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Daily rental Blue Bikes 2019
-- Was the bike moved by transport vechicle or not
-- Author: Hunt J 2 Feb 2021
-- Mod: Add the number of rides to be able to compare, Hunt J 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH
bike_cte AS        /* Grab the rides in order */
(SELECT Distinct bike_id,
start_time,
start_station_id as start,
end_station_id as stop
FROM bluebikes_2019
ORDER BY bike_id,start_time
),
delay_cte AS      /* Grab the start position and the previous end position */
(SELECT
bike_id,
start_time,
start,
LAG(stop, 1) OVER( Partition BY 1) as previous_stop
FROM bike_cte
```

```
),
moved_bike AS     /* Was the bike moved or not */
(SELECT
bike_id,
start_time,
start,
previous_stop,
start!=previous_stop AS Moved
FROM delay_cte
),
total_rides AS      /* All the rides */
(SELECT
date_part('month',start_time) AS month,
count(*) AS number_of_rides
FROM bluebikes_2019
GROUP BY month
ORDER BY month
),
-- Count the moves
total_moves AS (SELECT
date_part('month',start_time) AS month,
count(CASE WHEN moved THEN 1 END) AS number_of_moves
FROM moved_bike
GROUP BY month
ORDER BY month
)
-- Grab the monthly moves and total rides

SELECT
m.month,
number_of_moves,
number_of_rides
FROM total_rides r
JOIN total_moves m USING(month)
```

Query Results Sample

month	number_of_moves	number_of_rides
1	8520	69872
2	7053	80466
3	9388	102369
4	12290	166694
5	14303	223084
6	16610	274022
7	21274	316931
8	21950	337443
9	22528	363185

Geospatial

Question 6: How far is a typical journey?

Query 6.1 – Divvy Bikes

Building on the query generated for Question 4 where the longest distance was calculated. The query was to produce the average distance of rides with the notion that a typical ride is considered the most common ride.

Note: Query is simply changed to run for each year 2017-2019

```
-- Geospatial
-- How far is a typical journey?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Provide latitude and longitude for both stations
-- Calculate the distance between those stations for Divvy Bikes 2019
-- Stripped the query back compared to earlier versions as other details can
be added in Excel
-- Author: Hunt J 2 Feb 2021
-- Mod: Moody C Stripped back to produce average for each month 3 Feb 2021
-- Mod: Hunt J Selection order rearraged 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH
L1_bikes_2019 AS
(SELECT
b.bikeid,
b.start_time,
b.start_station_id,
b.end_station_id,
s.latitude as ss_latitude,
s.longtitude as ss_longitude
FROM divvybikes_2019 b
JOIN divvybikes_stations s ON b.start_station_id = s.id
WHERE b.start_station_id != b.end_station_id)

SELECT
date_part('month',start_time) As month,
AVG(calculate_distance(b.ss_latitude, b.ss_longitude,
                       e.latitude, e.longtitude,
                       'K'))
FROM L1_bikes_2019 b
JOIN divvybikes_stations e ON b.end_station_id = e.id
Group by month
Order by month
```

Query Results Sample

month	avg
1	1.773641131
2	1.729070231
3	1.932267125
4	2.179283264

Query 6.2 – Blue Bikes

See notes for Query 6.1 Divvy bikes

Note: Query simply changed to run for each year 2017-2019

```
-- Geospatial
-- How far is a typical journey?
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Provide latitude and longitude for both stations
```

```
-- Calculate the distance between those stations for Blue Bikes 2019
-- Stripped the query back compared to earlier versions as other details can
be added in Excel
-- Author: Hunt J 2 Feb 2021
-- Mod: Moody C Stripped back to produce average for each month 3 Feb 2021
-- Mod: Hunt J Selection order rearraged 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH
L1_bikes_2019 AS
(SELECT
b.bike_id,
b.start_time,
b.start_station_id,
b.end_station_id,
s.latitude as ss_latitude,
s.longtitude as ss_longitude
FROM bluebikes_2019 b
JOIN bluebikes_stations s ON b.start_station_id = s.id
WHERE b.start_station_id != b.end_station_id)

SELECT
date_part('month',start_time) As month,
AVG(calculate_distance(b.ss_latitude, b.ss_longitude,
                       e.latitude, e.longtitude,
                       'K'))
FROM L1_bikes_2019 b
JOIN bluebikes_stations e ON b.end_station_id = e.id
Group by month
Order by month
```

Query Results Sample

month	avg
1	1.85021983
2	1.839804044
3	1.962101523
4	2.024402455

Business and Commercial

Question 7: What sort of people use these bikes, and when do they use them?

Query 7.1 – Divvy Bikes

To find out more about our riders I added age groups that were sourced from US population statistical sites, thus being able to break up the riders into Age group, season rides and weekend or weekday rides. Hoping that these segments would help me to see stand out clusters within the rides being taken.
Given the structure of the SQL it was important to exclude the records where the riders birth year was not null or 0 or less. In previous queries I did not need to exclude records like this as the birth year was not being taken into consideration. Had I done that then a great deal of Customer users would have been excluded from the analysis.
I also decided that I would only take on those riders older than 13 as the number of riders before this age were quite low, whilst there where records with ages higher than 75, I felt that this was a good cut off for the amount of data I wanted. It was also observed that when looking at ages higher than 75 the data displayed that quite a few people most likely put in a false age, as it was difficult to believe we have riders who are over 100 years old.

```
-- Business and commercial
-- What sort of people use these bikes, and when do they use them?
-- Number of users over 2017 to 2019 Divvy bikes
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Author: Hunt J 3 Feb 2021
-- Date last modified: 3 Feb 2021

WITH bikes_Join as
(SELECT bikeid as
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,birthye
ar,gender FROM divvybikes_2017
 WHERE (birthyear IS NOT NULL AND birthyear > 0) AND
(date_part('year',start_time)-birthyear) BETWEEN 13 AND 75
UNION ALL
SELECT bikeid as
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,birthye
ar,gender FROM divvybikes_2018
WHERE (birthyear IS NOT NULL AND birthyear > 0) AND
(date_part('year',start_time)-birthyear) BETWEEN 13 AND 75
 UNION ALL
SELECT bikeid as
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,birthye
ar,gender FROM divvybikes_2019
WHERE (birthyear IS NOT NULL AND birthyear > 0) AND
(date_part('year',start_time)-birthyear) BETWEEN 13 AND 75),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
user_type,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
date_part('day',start_time) as day,
rtrim(to_char(start_time, 'day')) AS day_name,
CASE
 WHEN rtrim(to_char(start_time, 'day')) = 'sunday' THEN TRUE
 WHEN rtrim(to_char(start_time, 'day')) = 'saturday' THEN TRUE
 ELSE FALSE
END AS Weekend,
CASE
```

```
 WHEN date_part('year',start_time)-birthyear BETWEEN 13 AND 24 THEN 'Gen Z'
 WHEN date_part('year',start_time)-birthyear BETWEEN 25 AND 28 THEN 'Gen Y.1'
 WHEN date_part('year',start_time)-birthyear BETWEEN 29 AND 40 THEN 'Gen Y.2'
 WHEN date_part('year',start_time)-birthyear BETWEEN 41 AND 56 THEN 'Gen X'
 WHEN date_part('year',start_time)-birthyear BETWEEN 57 AND 75 THEN 'Baby
Boomers'
END age_class,
count(user_type) as total_users
FROM bikes_Join
GROUP BY year,user_type,season,month,day,day_name,weekend,age_class)

SELECT *
FROM bikes_table
ORDER BY year,month,day ASC
```

Query Results Sample

year	user_type	season	month	day	day_name	weekend	age_class	total_users
2017	Customer	Winter	1	14	saturday	TRUE	Gen X	11
2017	Customer	Winter	1	16	monday	FALSE	Gen X	6
2017	Customer	Winter	1	18	wednesday	FALSE	Gen Y.2	4
2017	Customer	Winter	1	19	thursday	FALSE	Gen Y.2	1
2017	Customer	Winter	1	19	thursday	FALSE	Gen X	1

Query 7.2 – Blue Bikes

See notes for Query 7.1 Divvy bikes

A main difference in this query is the required treatment of the birth year field as it is stored as a text and needed to be converted to an integer to calculate the age of the rider. It was also discovered that the age field held values of '\n' (a new line escape character) and so it was chosen to exclude those records from the analysis.

```
-- Business and commercial
-- What sort of people use these bikes, and when do they use them?
-- Number of users over 2017 to 2019 Blue Bikes
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Author: Hunt J 3 Feb 2021
-- Date last modified: 4 Feb 2021

WITH bikes_Join as
(SELECT
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,substri
ng(user_birth_year,1,4)::INTEGER as birthyear,user_gender AS gender FROM
bluebikes_2017
 WHERE (user_birth_year IS NOT NULL AND user_birth_year != '\N') AND
(date_part('year',start_time)-(substring(user_birth_year,1,4)::INTEGER))
BETWEEN 13 AND 75
UNION ALL
SELECT
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,substri
ng(user_birth_year,1,4)::INTEGER as birthyear,user_gender AS gender FROM
bluebikes_2018
WHERE (user_birth_year IS NOT NULL AND user_birth_year != '\N') AND
(date_part('year',start_time)-(substring(user_birth_year,1,4)::INTEGER))
BETWEEN 13 AND 75
 UNION ALL
SELECT
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,substri
ng(user_birth_year,1,4)::INTEGER as birthyear,user_gender AS gender FROM
bluebikes_2019
```

```
WHERE (user_birth_year IS NOT NULL AND user_birth_year != '\N') AND
(date_part('year',start_time)-(substring(user_birth_year,1,4)::INTEGER))
BETWEEN 13 AND 75),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
user_type,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
date_part('day',start_time) as day,
rtrim(to_char(start_time, 'day')) AS day_name,
CASE
 WHEN rtrim(to_char(start_time, 'day')) = 'sunday' THEN TRUE
 WHEN rtrim(to_char(start_time, 'day')) = 'saturday' THEN TRUE
 ELSE FALSE
END AS Weekend,
CASE
 WHEN date_part('year',start_time)-birthyear BETWEEN 13 AND 24 THEN 'Gen Z'
 WHEN date_part('year',start_time)-birthyear BETWEEN 25 AND 28 THEN 'Gen Y.1'
 WHEN date_part('year',start_time)-birthyear BETWEEN 29 AND 40 THEN 'Gen Y.2'
 WHEN date_part('year',start_time)-birthyear BETWEEN 41 AND 56 THEN 'Gen X'
 WHEN date_part('year',start_time)-birthyear BETWEEN 57 AND 75 THEN 'Baby
Boomers'
END age_class,
count(user_type) as total_users
FROM bikes_Join
GROUP BY year,user_type,season,month,day,day_name,weekend,age_class)

SELECT *
FROM bikes_table
ORDER BY year,month,day ASC
```

Query Results Sample

year	user_type	season	month	day	day_name	weekend	age_class	total_users
2017	Subscriber	Winter	1	1	sunday	TRUE	Gen Z	39
2017	Subscriber	Winter	1	1	sunday	TRUE	Gen Y.1	76
2017	Subscriber	Winter	1	1	sunday	TRUE	Gen Y.2	158
2017	Subscriber	Winter	1	1	sunday	TRUE	Gen X	68
2017	Subscriber	Winter	1	1	sunday	TRUE	Baby Boomers	38

Business and Commercial

Q8 Own Question: Which day of the week are we seeing the most riders?

Query 8.1 – Divvy Bikes

In this query I wanted to extend the query in Query 7.1 to see if we could see any standout results by comparing the day the rides were taken by the different age groups. I also included more details of the starting station to try find a possible map of where different ages groups where riding from the most. The sample size was reduced to 2019 for the convenience of running the query.

```
-- Business and commercial
-- Which day of the week are we seeing the most riders?
-- Number of users over 2019 Divvy bikes
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Author: Hunt J 4 Feb 2021
-- Mod: Removed count aggregate and group by statement. Added
start_station_id, Hunt J 4 Feb 2021
-- Mod: Commented  out 2018 and 2017 for smaller sample,Hunt J, 4 Feb 2021
-- Date last modified: 4 Feb 2021

WITH bikes_Join as
(SELECT bikeid as
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,birthye
ar,gender FROM divvybikes_2019
 WHERE (birthyear IS NOT NULL AND birthyear > 0) AND
(date_part('year',start_time)-birthyear) BETWEEN 13 AND 75
/* UNION ALL
SELECT bikeid as
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,birthye
ar,gender FROM divvybikes_2018
WHERE (birthyear IS NOT NULL AND birthyear > 0) AND
(date_part('year',start_time)-birthyear) BETWEEN 13 AND 75
 UNION ALL
SELECT bikeid as
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,birthye
ar,gender FROM divvybikes_2017
WHERE (birthyear IS NOT NULL AND birthyear > 0) AND
(date_part('year',start_time)-birthyear) BETWEEN 13 AND 75*/ ),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
user_type,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
date_part('day',start_time) as day,
rtrim(to_char(start_time, 'day')) AS day_name,
CASE
 WHEN rtrim(to_char(start_time, 'day')) = 'sunday' THEN TRUE
 WHEN rtrim(to_char(start_time, 'day')) = 'saturday' THEN TRUE
 ELSE FALSE
END AS Weekend,
CASE
 WHEN date_part('year',start_time)-birthyear BETWEEN 13 AND 24 THEN 'Gen Z'
 WHEN date_part('year',start_time)-birthyear BETWEEN 25 AND 28 THEN 'Gen Y.1'
 WHEN date_part('year',start_time)-birthyear BETWEEN 29 AND 40 THEN 'Gen Y.2'
 WHEN date_part('year',start_time)-birthyear BETWEEN 41 AND 56 THEN 'Gen X'
```

```
 WHEN date_part('year',start_time)-birthyear BETWEEN 57 AND 75 THEN 'Baby
Boomers'
END age_class,
start_station_id
FROM bikes_Join)

SELECT
year,
user_type,
season,
month,
day,
day_name,
weekend,
age_class,
start_station_id,
latitude,
longitude,
name,
docks
FROM bikes_table b
JOIN divvy_stations d ON d.id = b.start_station_id
ORDER BY year,month,day ASC
```

Query Results Sample

Please see Excel workbooks due to width of results

Query 8.2 – Blue Bikes

See notes for Query 8.1 Divvy bikes

```
-- Business and commercial
-- Which day of the week are we seeing the most riders?
-- Number of users over 2019 Divvy bikes
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Author: Hunt J 4 Feb 2021
-- Mod: Removed count aggregate and group by statement. Added
start_station_id, Hunt J 4 Feb 2021
-- Mod: Commented  out 2018 and 2017 for smaller sample,Hunt J, 4 Feb 2021
-- Date last modified: 4 Feb 2021

WITH bikes_Join as
(SELECT
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,substri
ng(user_birth_year,1,4)::INTEGER as birthyear,user_gender AS gender FROM
bluebikes_2017
 WHERE (user_birth_year IS NOT NULL AND user_birth_year != '\N') AND
(date_part('year',start_time)-(substring(user_birth_year,1,4)::INTEGER))
BETWEEN 13 AND 75
/*UNION ALL
SELECT
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,substri
ng(user_birth_year,1,4)::INTEGER as birthyear,user_gender AS gender FROM
bluebikes_2018
WHERE (user_birth_year IS NOT NULL AND user_birth_year != '\N') AND
(date_part('year',start_time)-(substring(user_birth_year,1,4)::INTEGER))
BETWEEN 13 AND 75
 UNION ALL
SELECT
bike_id,start_time,end_time,start_station_id,end_station_id,user_type,substri
ng(user_birth_year,1,4)::INTEGER as birthyear,user_gender AS gender FROM
bluebikes_2019
```

```
WHERE (user_birth_year IS NOT NULL AND user_birth_year != '\N') AND
(date_part('year',start_time)-(substring(user_birth_year,1,4)::INTEGER))
BETWEEN 13 AND 75*/),
bikes_table as
(SELECT DISTINCT
date_part('year', start_time) as year,
user_type,
CASE
WHEN date_part('month',start_time) BETWEEN 03 AND 05 THEN 'Spring'
WHEN date_part('month',start_time) BETWEEN 06 AND 08 THEN 'Summer'
WHEN date_part('month',start_time) BETWEEN 09 AND 11 THEN 'Autumn'
ELSE 'Winter'
END AS season,
date_part('month', start_time) as month,
date_part('day',start_time) as day,
rtrim(to_char(start_time, 'day')) AS day_name,
CASE
 WHEN rtrim(to_char(start_time, 'day')) = 'sunday' THEN TRUE
 WHEN rtrim(to_char(start_time, 'day')) = 'saturday' THEN TRUE
 ELSE FALSE
END AS Weekend,
CASE
 WHEN date_part('year',start_time)-birthyear BETWEEN 13 AND 24 THEN 'Gen Z'
 WHEN date_part('year',start_time)-birthyear BETWEEN 25 AND 28 THEN 'Gen Y.1'
 WHEN date_part('year',start_time)-birthyear BETWEEN 29 AND 40 THEN 'Gen Y.2'
 WHEN date_part('year',start_time)-birthyear BETWEEN 41 AND 56 THEN 'Gen X'
 WHEN date_part('year',start_time)-birthyear BETWEEN 57 AND 75 THEN 'Baby
Boomers'
END age_class,
start_station_id
FROM bikes_Join)

SELECT
year,
user_type,
season,
month,
day,
day_name,
weekend,
age_class,
start_station_id,
latitude,
longtitude AS longitude,
name,
district,
total_docks AS docks
FROM bikes_table b
JOIN bluebikes_stations s ON s.id = b.start_station_id
ORDER BY year,month,day ASC
```

Query Results Sample

Please see Excel workbooks due to width of results

Business and Commercial

Q8 Own Question: Distance calculation experimentation

This was more of a personal experiment that a bonus question as I wanted to compare the different methods of being able to calculate the amount of distance travelled during a ride and the possibility of being able to calculate (or obtain) the average duration of a ride for that distance. If in the future I would like to be able to compare this to the recorded rental time to see if we can determine whether a rider either stopped during their rental period, rode slowly or rode at steady pace.
This query is simply using an earlier query to bring back the starting station coordinates and the end station coordinates for the initial calculations. I limited the data set to the month of July as this often showed the most rides of all months and provided a good sample set.

```
-- Geospatial
-- What was the longest journey? What do we know about it?
-- Experimentation:  Testing the distance calculation
-- Project team: Hunt J,Khaoua H,Leung M,Moody C 2021
-- Daily rental over 2019 Divy Bikes
-- Provide latitude and longitude for both stations
-- Calculate the distance between those stations
-- Author: Hunt J 2 Feb 2021
-- Mod: Stripped the query back compared to earlier versions as other details
can be added in Excel 2 Feb 2021
-- Mod: Added seasons 2 Feb 2021
-- Mod: Reduced to July, Added both station's detail for external calculation
-- Date last modified: 4 Feb 2021

WITH
L1_bikes_2019 AS
(SELECT
b.bikeid,
b.start_time,
b.start_station_id,
b.end_station_id,
s.latitude as ss_latitude,
s.longitude as ss_longitude
FROM divvybikes_2019 b
JOIN divvy_stations s ON b.start_station_id = s.id
WHERE b.start_station_id != b.end_station_id AND
(date_part('month',start_time) = 07))

SELECT
date_part('month',b.start_time) AS month,
b.start_station_id,
b.ss_latitude AS ss_latitude,
b.ss_longitude AS ss_longitude,
b.end_station_id,
e.latitude AS es_latitude,
e.longitude AS es_logitude,
(calculate_distance(b.ss_latitude, b.ss_longitude,
                          e.latitude, e.longitude,
                          'K')) AS longest_ride
FROM L1_bikes_2019 b
JOIN divvy_stations e ON b.end_station_id = e.id
GROUP BY
month,b.start_station_id,ss_latitude,ss_longitude,end_station_id,es_latitude,
es_logitude
```

Query Results Sample

month	start_station_id	ss_latitude	ss_longitude	end_station_id	es_latitude	es_logitude	distance

7	69	41.909396	-87.677692	68	41.875885	-87.640795	4.82
7	21	41.877726	-87.654787	290	41.921525	-87.707322	6.53
7	333	41.907066	-87.667252	115	41.936266	-87.652662	3.46
7	117	41.94018	-87.65304	311	41.968885	-87.684001	4.09

GENERAL QUERIES

G_Query 01.1: Obtain all stations for Blue bikes

```
-- Obtain a copy of all the stations for further reference
-- Blue bikes stations
SELECT number,name,latitude,longtitude,district,total_docks,id
FROM bluebikes_stations
```

Result: 339 Rows

Query Results Sample

number	name	latitude	longtitude	district	total_docks	id
B32006	Colleges of the Fenway - Fenway at Avenue Louis Pasteur	42.34011512	-71.10061884	Boston	15	3
C32000	Tremont St at E Berkeley St	42.345392	-71.069616	Boston	19	4
B32012	Northeastern University - North Parking Lot	42.341814	-71.090179	Boston	15	5
D32000	Cambridge St at Joy St	42.36125722	-71.06528744	Boston	15	6
A32000	Fan Pier	42.35339051	-71.0445714	Boston	15	7
A32001	Union Square - Brighton Ave at Cambridge St	42.353334	-71.137313	Boston	19	8
A32002	Commonwealth Ave at Agganis Way	42.35169202	-71.11903489	Boston	15	9
A32003	B.U. Central - 725 Comm. Ave.	42.350406	-71.108279	Boston	11	10
A32004	Longwood Ave at Binney St	42.338629	-71.1065	Boston	15	11

G_Query 01.2: Exclude all stations with NULL IDs

```
-- Excluding stations with NULL ID entries
SELECT COUNT(*)
FROM bluebikes_stations
WHERE ID IS NOT NULL
ORDER BY ID
```

Result: 336 Rows | 3 rows excluded

G_Query 02.1: Obtain all stations for Divvy bikes

```
-- Obtain a copy of all the stations for further reference
-- Divvy bikes stations
-- No NULL ID entries
SELECT *
FROM divvy_stations
/* WHERE ID IS NOT NULL */
ORDER BY ID
```

Result: 611 Rows

Query Results Sample

id	latitude	longitude	name	docks
2	41.876511	-87.620548	Buckingham Fountain	38
3	41.867226	-87.615355	Shedd Aquarium	54
4	41.856268	-87.613348	Burnham Harbor	22
5	41.874053	-87.627716	State St & Harrison St	23
6	41.886976	-87.612813	Dusable Harbor	39
7	41.886349	-87.617517	Field Blvd & South Water St	18

9	41.828792	-87.680604	Leavitt St & Archer Ave	15
11	41.766638	-87.57645	Jeffery Blvd & 71st St	11
12	41.766409	-87.565688	South Shore Dr & 71st St	15

G_Query 02.2: Exclude all stations with NULL IDs

```
-- Excluding stations with NULL ID entries
SELECT COUNT(*)
FROM divvy_stations
WHERE ID IS NOT NULL
```

Result: 611 | 0 rows excluded

G_Query 03.1: Check blue bikes station name field

```
SELECT COUNT(*)
FROM bluebikes_stations
WHERE name IS NOT NULL
```

Result: 339 | 0 rows excluded

G_Query 03.2: Check divvy bikes station name field

```
SELECT COUNT(*)
FROM divvy_stations
WHERE name IS NOT NULL
```

Result: 611 | 0 rows excluded

G_Queries 04.: Check birth year entries of riders

```
SELECT count(*)
FROM divvybikes_2017
WHERE birthyear IS NULL
```

Result: 836758

```
SELECT COUNT(*)
FROM bluebikes_2017
WHERE user_birth_year IS NULL
```

Result: 0

```
SELECT count(*)
FROM divvybikes_2018
WHERE birthyear IS NULL
```

Result: 555209

```
SELECT COUNT(*)
FROM bluebikes_2018
WHERE user_birth_year IS NULL
```

Result: 9529

```
SELECT count(*)
FROM divvybikes_2019
WHERE birthyear IS NULL
```

Result: 538751

```
SELECT COUNT(*)
FROM bluebikes_2019
WHERE user_birth_year IS NULL
```

Result: 0

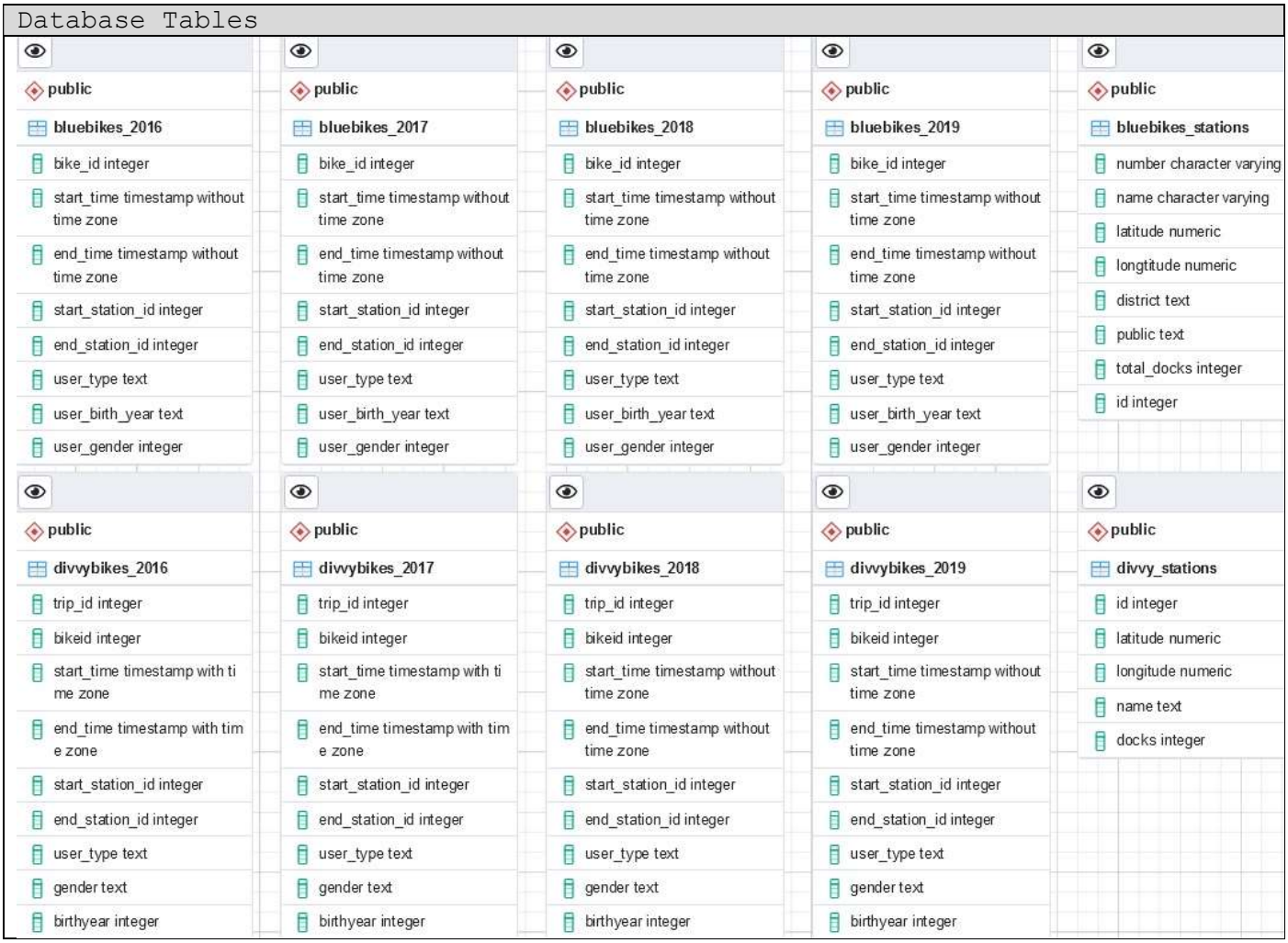

Figure 1: Database tables for Blue bikes and Divvy bikes