(1)

SELECT b.start\_station\_id, b2.id, b2.docks, b2.name, COUNT(\*) as count\_rides

FROM baywheels\_2019 b

LEFT JOIN baywheels\_stations b2

ON b.start\_station\_id = b2.id

WHERE b2.docks IS NOT NULL

GROUP BY 1, 2, 3

ORDER BY count\_rides DESC

;

SELECT AVG(docks) as avg\_docks

FROM baywheels\_stations;

SELECT b.start\_station\_id, b2.id, b2.docks, b2.name, COUNT(\*) as count\_rides

FROM baywheels\_2019 b

JOIN baywheels\_stations b2

ON b.start\_station\_id = b2.id

WHERE b2.docks IS NOT NULL

GROUP BY 1, 2, 3

ORDER BY count\_rides DESC

;

--DOESN'T WORK

SELECT bike\_id, COUNT(\*)

FROM baywheels\_2017

GROUP BY 1

UNION

SELECT bike\_id, COUNT(\*)

FROM baywheels\_2018

GROUP BY 1

UNION

SELECT bike\_id COUNT(\*)

GROUP BY 1

FROM baywheels\_2019;

SELECT bike\_id, COUNT(\*) AS count\_rides

FROM baywheels\_2017

GROUP BY 1

ORDER BY 2 DESC;

SELECT bike\_id, COUNT(\*) AS count\_rides

FROM baywheels\_2018

GROUP BY 1

ORDER BY 2 DESC;

SELECT bike\_id, COUNT(\*) AS count\_rides

FROM baywheels\_2019

GROUP BY 1

ORDER BY 2 DESC;

(2)

--number of rides per bike\_id

SELECT DISTINCT bike\_id,COUNT(\*) as count\_user\_type

FROM baywheels\_2019

GROUP BY 1

ORDER BY 2 DESC;

--number of rides per user\_type

SELECT user\_type, COUNT(\*) as count\_user\_type

FROM baywheels\_2019

GROUP BY 1;

SELECT \*

FROM baywheels\_2019;

--create trip duration column + category

SELECT bike\_id, user\_type, end\_time - start\_time AS trip\_duration,

CASE

WHEN (end\_time - start\_time) > '00:00:00' AND (end\_time - start\_time) < '00:30:00' THEN 'Short'

WHEN (end\_time - start\_time) BETWEEN '00:30:00' AND '01:00:00' THEN 'Medium'

WHEN (end\_time - start\_time) > '01:00:00' AND (end\_time - start\_time) < '02:00:00' THEN 'Long'

WHEN (end\_time - start\_time) >= '02:00:00' THEN 'Super long'

ELSE 'n/a'

END AS trip\_duration\_category

FROM baywheels\_2019

ORDER BY CASE WHEN (end\_time - start\_time) > '00:00:00' AND (end\_time - start\_time) < '00:30:00' THEN 1

WHEN (end\_time - start\_time) BETWEEN '00:30:00' AND '01:00:00' THEN 2

WHEN (end\_time - start\_time) > '01:00:00' AND (end\_time - start\_time) < '02:00:00' THEN 3

WHEN (end\_time - start\_time) >= '02:00:00' THEN 4

ELSE 5 END, 3

; --note: Order by customized list by CASE

--create trip duration column + category

SELECT user\_type,

CASE

WHEN (end\_time - start\_time) >= '00:00:00' AND (end\_time - start\_time) < '00:30:00' THEN 'Short'

WHEN (end\_time - start\_time) BETWEEN '00:30:00' AND '01:00:00' THEN 'Medium'

WHEN (end\_time - start\_time) > '01:00:00' AND (end\_time - start\_time) < '02:00:00' THEN 'Long'

WHEN (end\_time - start\_time) >= '02:00:00' THEN 'Super long'

ELSE ''

END AS trip\_duration\_category, COUNT(start\_time) as ride\_count

FROM baywheels\_2019

GROUP BY 1, 2

HAVING COUNT(start\_time) > 10

ORDER BY 2, 3 DESC

;

--SOLUTION:

--Customers taking longer rides than Subscribers.

--Subscribers taking shorter rides.

--number of rides by month 2019(18,17)

--SOLUTION: Perform maintenance and tuning during winter months.

SELECT

CASE

WHEN DATE\_PART('month', start\_time) = 1 THEN 'January'

WHEN DATE\_PART('month', start\_time) = 2 THEN 'February'

WHEN DATE\_PART('month', start\_time) = 3 THEN 'March'

WHEN DATE\_PART('month', start\_time) = 4 THEN 'April'

WHEN DATE\_PART('month', start\_time) = 5 THEN 'May'

WHEN DATE\_PART('month', start\_time) = 6 THEN 'June'

WHEN DATE\_PART('month', start\_time) = 7 THEN 'July'

WHEN DATE\_PART('month', start\_time) = 8 THEN 'August'

WHEN DATE\_PART('month', start\_time) = 9 THEN 'September'

WHEN DATE\_PART('month', start\_time) = 10 THEN 'October'

WHEN DATE\_PART('month', start\_time) = 11 THEN 'November'

WHEN DATE\_PART('month', start\_time) = 12 THEN 'December'

ELSE '' END AS month, COUNT(\*) as number\_of\_rides

FROM baywheels\_2019

GROUP BY 1

ORDER BY 2 DESC

--Q. WHY CAN YOU NOT ORDER BY THE CASE CLAUSE AFTER GROUPING?

--number of rides going from start\_station to end\_station

SELECT b9.start\_station\_id, bs.name, COUNT(\*) AS ride\_count, b9.end\_station\_id

FROM baywheels\_2019 b9

JOIN baywheels\_stations bs

ON b9.start\_station\_id = bs.id

GROUP BY 1, 2, 4

ORDER BY 3 DESC;

--Q. NEED TO FIGURE OUT HOW TO MATCH name WITH end\_station\_id. As it stands, it matches with the start\_station\_id.

--ride\_counts of start&end stations with names for both

SELECT b9.start\_station\_id, bs.name, b9.end\_station\_id, bs2.name, COUNT(\*) AS count\_rides

FROM baywheels\_2019 b9

JOIN baywheels\_stations bs

ON b9.start\_station\_id = bs.id

JOIN baywheels\_stations bs2

ON b9.end\_station\_id = bs2.id

GROUP BY 1,2,3,4

HAVING COUNT(\*) > 2000

ORDER BY 5 DESC, 1;

--number of rides starting at this station

SELECT b9.start\_station\_id, COUNT(\*) AS ride\_count

FROM baywheels\_2019 b9

JOIN baywheels\_stations bs

ON b9.start\_station\_id = bs.id

GROUP BY 1

ORDER BY 2 DESC

LIMIT 15;

--number of rides ending at this station

SELECT b9.end\_station\_id, COUNT(\*) AS ride\_count

FROM baywheels\_2019 b9

JOIN baywheels\_stations bs

ON b9.start\_station\_id = bs.id

GROUP BY 1

ORDER BY 2 DESC

LIMIT 15;

--Q. IS THERE A WAY TO UNITE THESE TWO ABOVE QUERIES AND GET A DIFFERENCE OF COUNTS BY STATION\_ID

--number of rides starting at this station + docks

SELECT b9.start\_station\_id, COUNT(\*) AS ride\_count, bs.docks

FROM baywheels\_2019 b9

JOIN baywheels\_stations bs

ON b9.start\_station\_id = bs.id

GROUP BY 1,3

ORDER BY 2 DESC

LIMIT 15;

--number of rides ending at this station + docks

SELECT b9.end\_station\_id, COUNT(\*) AS ride\_count, bs.docks

FROM baywheels\_2019 b9

JOIN baywheels\_stations bs

ON b9.end\_station\_id = bs.id

GROUP BY 1, 3

ORDER BY 2 DESC

LIMIT 15;

--Q. IS THERE A WAY TO UNITE THESE TWO ABOVE QUERIES AND GET A DIFFERENCE OF COUNTS BY STATION\_ID

--start&end stations with names and count > 2000

SELECT b9.start\_station\_id, bs.name, b9.end\_station\_id, bs2.name, COUNT(\*) AS count\_rides

FROM baywheels\_2019 b9

JOIN baywheels\_stations bs

ON b9.start\_station\_id = bs.id

JOIN baywheels\_stations bs2

ON b9.end\_station\_id = bs2.id

GROUP BY 1,2,3,4

HAVING COUNT(\*) > 2000

ORDER BY 5 DESC, 1;

--time spent in activity for each bike\_id

SELECT bike\_id, COUNT(\*) as count\_rides

FROM baywheels\_2019

GROUP BY 1

ORDER BY 2 DESC

LIMIT 10;

--number of bicycles

SELECT COUNT(DISTINCT bike\_id)

FROM

(SELECT \*

FROM baywheels\_2017

UNION

SELECT \*

FROM baywheels\_2018

UNION

SELECT \*

FROM baywheels\_2019) as full\_years

ORDER BY 1;

--time spent in activity for each bike\_id

SELECT bike\_id, end\_time - start\_time as trip\_duration

FROM baywheels\_2019

ORDER BY 2 DESC

LIMIT 10;