Cyclistics using PostgreSQL

Zsolnay

# Prepare & initial investigation

## Import data

1. Made copies of all data files
2. Imported data into PostgreSQL. (The BigQuery Sandbox account does not accept large files).
   1. First check files in a text file, to quickly find column names and detect correct data type to import correctly, and to change file names to more easily reflect contents (e.g, 202110-divvy-tripdata > Oct\_2021).
   2. Created database: Cyclistics\_project, schema: fiscal\_year\_2021\_2022, and tables: Oct\_2021, Nov\_2021, Dec\_2021, Jan\_2022, Feb\_2022, Mar\_2022, Apr\_2022, May\_2022, June\_2022, July\_2022, Aug\_2022, and Sept\_2021
      1. Example:

CREATE TABLE fiscal\_year\_2021\_2022."Oct\_2021"

(ride\_id VARCHAR,

rideable\_type VARCHAR,

started\_at timestamp without time zone,

ended\_at timestamp without time zone,

start\_station\_name VARCHAR,

start\_station\_id VARCHAR,

end\_station\_name VARCHAR,

end\_station\_id VARCHAR,

start\_lat double precision,

start\_lng double precision,

end\_lat double precision,

end\_lng double precision,

member\_casual VARCHAR);

* 1. Checked each file for proper import by column names and column and row count (not including header):
     1. Oct\_2021 631226 13
     2. Nov\_2021 359978 13
     3. Dec\_2021 247540 13
     4. Jan\_2022 103770 13
     5. Feb\_2022 115609 13
     6. Mar\_2022 284042 13
     7. Apr\_2022 371249 13
     8. May\_2022 634858 13
     9. Jun\_2022 769204 13
     10. July\_2022 823488 13
     11. Aug\_2022 785932 13
     12. Sept\_2022 701339 13

1. Examined the content and layout of Oct\_2021 to get a feel for the data.
   1. SELECT \* FROM fiscal\_year\_2021\_2022."Oct\_2021"

LIMIT 100

* 1. Looked at variables for 'rideable\_type' and 'member\_casual'

SELECT

DISTINCT rideable\_type

FROM fiscal\_year\_2021\_2022."Oct\_2021";

* + 1. Return is:

"rideable\_type"

"classic\_bike"

"docked\_bike"

"electric\_bike"

And

"member\_casual"

"casual"

"member"

* 1. Looked at number and names of stations in 'end\_station\_name'.

Return is 791 names, some of which are maintenance, temp, and vaccination sites.

## Process (wrangling, cleaning, and transformation)

1. Combined files into quarters and THEN into fiscal year file named: cyclistic\_2021\_2022
   1. E.g., Oct\_2021, Nov\_2021, and Dec\_2021 > 1Q

INSERT INTO fiscal\_year\_2021\_2022."1Q"

SELECT \* FROM fiscal\_year\_2021\_2022."Oct\_2021";

INSERT INTO fiscal\_year\_2021\_2022."1Q"

SELECT \* FROM fiscal\_year\_2021\_2022."Nov\_2021";

INSERT INTO fiscal\_year\_2021\_2022."1Q"

SELECT \* FROM fiscal\_year\_2021\_2022."Dec\_2021"

ON CONFLICT DO NOTHING;

* + 1. fiscal\_year\_2021\_2022."1Q" Total rows: 1238744
    2. fiscal\_year\_2021\_2022."2Q" Total rows: 503421
    3. fiscal\_year\_2021\_2022."3Q" Total rows: 1775311
    4. fiscal\_year\_2021\_2022."4Q" Total rows: 2310759
  1. Created fiscal year file named: Total\_Rides

CREATE TABLE fiscal\_year\_2021\_2022."total\_rides" AS

(SELECT \* FROM fiscal\_year\_2021\_2022."1Q"

UNION ALL

SELECT \* FROM fiscal\_year\_2021\_2022."2Q"

UNION ALL

SELECT \* FROM fiscal\_year\_2021\_2022."3Q"

UNION ALL

SELECT \* FROM fiscal\_year\_2021\_2022."4Q");

* + 1. fiscal\_year\_2021\_2022."total\_tides" 5828235
  1. Checked row and column count:
     1. 1238744 + 503421 + 1775311 + 2310759 = 5828235
     2. 13 columns

1. Checked for NULL values

Example:

SELECT ride\_id, rideable\_type, started\_at, ended\_at, start\_station\_name, start\_station\_id, end\_station\_name, end\_station\_id, start\_lat, start\_lng, end\_lat, end\_lng, member\_casual

FROM fiscal\_year\_2021\_2022.total\_rides

WHERE member\_casual IS NULL;

* 1. Return:

ride\_id IS NULL

rideable\_type IS NULL

started\_at IS NULL

ended\_at IS NULL

start\_station\_name = 895032

start\_station\_id = 895032

end\_station\_name = 958227

end\_station\_id = 958227

start\_lat IS NULL

start\_lng IS NULL

end\_lat is 5844

end\_lng is 5844

member\_casual IS NULL

* 1. (895032 x 2) + (958227 x 2) + (5844 x 2)= ((895032 \* 2) + (958227 \* 2)) + (5844 \* 2)

= 3718206

1. Removed null values (and created new draft)

CREATE TABLE fiscal\_year\_2021\_2022."total\_rides\_V2" AS

(SELECT \*

FROM fiscal\_year\_2021\_2022."total\_rides"

WHERE start\_station\_name NOT LIKE '%NULL%'

AND start\_station\_id NOT LIKE '%NULL%'

AND end\_station\_name NOT LIKE '%NULL%'

AND end\_station\_id NOT LIKE '%NULL%'

AND end\_lat NOT LIKE '%NULL%'

AND end\_lng NOT LIKE '%NULL%');

* 1. 4474141 columns remain; 5828235 - 4474141 = 1354094 were removed.

1. Checked for duplicate ride\_id values: 0

SELECT ride\_id FROM fiscal\_year\_2021\_2022."total\_rides\_V2"

* 1. 4474141 variables, so no duplicates

1. Checked for leading or trailing spaces from start\_station\_name and end\_station\_name

Example:

SELECT \*

FROM fiscal\_year\_2021\_2022."total\_rides\_V2"

WHERE start\_station\_name LIKE ' %' or start\_station\_name LIKE '% ';

* 1. start\_station\_name return: 71
  2. start\_station\_name return: 84

1. Updated start\_station\_name and end\_station\_name

Example:

UPDATE fiscal\_year\_2021\_2022."total\_rides\_V2"

SET start\_station\_name = TRIM(start\_station\_name);

* 1. Check = 0
  2. Check = 0

1. Checked for uniformity of character length in ride\_id
   1. Check length of string:

SELECT LENGTH(ride\_id)

FROM fiscal\_year\_2021\_2022."total\_rides\_V2";

* 1. Return = 16 characters long
  2. Check uniformity:

SELECT ride\_id

FROM fiscal\_year\_2021\_2022."total\_rides\_V2"

WHERE LENGTH(ride\_id) <> 16

* 1. Return = 0

1. Found and removed stations warehouse, repair, and charging from columns start\_station\_name and end\_station\_name:
   1. Warehouse
      1. Base - 2132 W Hubbard = 890/127
      2. Base - 2132 W Hubbard Warehouse = 317/134
      3. Hastings WH 2 = 2/2
   2. Mobile stations
      1. DIVVY CASSETTE REPAIR MOBILE STATION = 0/6
      2. Throop/Hastings Mobile Station = 1/1
   3. Charging stations
      1. Bissell St & Armitage Ave - Charging = 18/20
      2. Lincoln Ave & Roscoe St - Charging = 3/3
      3. Pawel Bialowas - Test- PBSC charging station = 1/1
      4. Wilton Ave & Diversey Pkwy - Charging = 17/1

CREATE TABLE fiscal\_year\_2021\_2022."total\_rides\_V3" AS

(SELECT \*

FROM fiscal\_year\_2021\_2022."total\_rides\_V2"

WHERE start\_station\_name NOT LIKE '%Base - 2132 W Hubbard%'

AND start\_station\_name NOT LIKE '%Base - 2132 W Hubbard Warehouse%'

AND start\_station\_name NOT LIKE '%Hastings WH 2%'

AND start\_station\_name NOT LIKE '%DIVVY CASSETTE REPAIR MOBILE STATION%'

AND start\_station\_name NOT LIKE '%Throop/Hastings Mobile Station%'

AND start\_station\_name NOT LIKE '%Bissell St & Armitage Ave - Charging%'

AND start\_station\_name NOT LIKE '%Lincoln Ave & Roscoe St - Charging%'

AND start\_station\_name NOT LIKE '%Pawel Bialowas - Test- PBSC charging station%'

AND start\_station\_name NOT LIKE '%Wilton Ave & Diversey Pkwy - Charging%');

* 1. Check: row count: 4472892
  2. Removed same stations from column end\_station\_name:
     1. UPDATE fiscal\_year\_2021\_2022."total\_rides\_V3"

SET end\_station\_name =

WHERE end\_station\_name NOT LIKE '%Base - 2132 W Hubbard%'

AND end\_station\_name NOT LIKE '%Base - 2132 W Hubbard Warehouse%'

AND end\_station\_name NOT LIKE '%Hastings WH 2%'

AND end\_station\_name NOT LIKE '%DIVVY CASSETTE REPAIR MOBILE STATION%'

AND end\_station\_name NOT LIKE '%Throop/Hastings Mobile Station%'

AND end\_station\_name NOT LIKE '%Bissell St & Armitage Ave - Charging%'

AND end\_station\_name NOT LIKE '%Lincoln Ave & Roscoe St - Charging%'

AND end\_station\_name NOT LIKE '%Pawel Bialowas - Test- PBSC charging station%'

AND end\_station\_name NOT LIKE '%Wilton Ave & Diversey Pkwy - Charging%';

* 1. Check: row count: 4472599
  2. Investigated how many stations with Temp in the name would need to be deleted:

SELECT COUNT(\*) AS num\_of\_rows\_to\_delete

FROM fiscal\_year\_2021\_2022."total\_rides\_V4"

WHERE start\_station\_name LIKE '%Temp%';

* + 1. Return: 30446 rows
    2. I did not remove these.

1. Standardized column data-type and labels:
   1. Did not need to retype/cast data (see above)
   2. Relabelled columns and check:

Example:

ALTER TABLE fiscal\_year\_2021\_2022."total\_rides\_V3"

RENAME COLUMN ride\_id to trip\_id

* + 1. ride\_id > trip\_id
    2. rideable\_type > bike\_type
    3. member\_casual > user\_type

1. Created new columns: start\_date, start\_time:

Example:

ALTER TABLE fiscal\_year\_2021\_2022."total\_rides\_V4" ADD COLUMN start\_date date;

ALTER TABLE fiscal\_year\_2021\_2022."total\_rides\_V4" ADD COLUMN start\_time time;

UPDATE fiscal\_year\_2021\_2022."total\_rides\_V4"

SET start\_date = started\_at :: date,

start\_time = started\_at :: time;

* 1. Check that columns were created

1. Created new columns: end\_date, end\_time:
   1. Check that columns were created
2. Created columns: month and day
   1. ALTER TABLE fiscal\_year\_2021\_2022."total\_rides\_V5" ADD COLUMN month VARCHAR;

UPDATE fiscal\_year\_2021\_2022."total\_rides\_V4"

SET month = TO\_CHAR (start\_date, 'Month');

* 1. ALTER TABLE fiscal\_year\_2021\_2022."total\_rides\_V4" ADD COLUMN day VARCHAR;

UPDATE fiscal\_year\_2021\_2022."total\_rides\_V4"

SET day = TO\_CHAR (start\_date, 'Day');

* 1. Check that columns were created

1. Created new column trip\_duration for trip in seconds:

ALTER TABLE fiscal\_year\_2021\_2022."total\_rides\_V4" ADD COLUMN trip\_duration INTEGER;

UPDATE fiscal\_year\_2021\_2022."total\_rides\_V4"

SET trip\_duration = EXTRACT(EPOCH FROM (ended\_at - started\_at));

1. Checked trip\_duration column for outliers

SELECT

MIN(trip\_duration),

MAX(trip\_duration)

FROM fiscal\_year\_2021\_2022."total\_rides\_V4";

* 1. Noted that trip\_duration has trips under 60 seconds long (-7621 seconds).
  2. Noted that trip\_duration has trips over 86400 seconds long (over 24hrs [2442301 seconds]) as they are likely stolen.

1. Checked how many rows have outliers

SELECT \*

FROM fiscal\_year\_2021\_2022."total\_rides\_V4"

WHERE trip\_duration <= 60 or trip\_duration >= 86400;

* 1. Return: 74712 rows

1. Removed trip\_duration outliers (CREATE new draft of dataframe):

CREATE TABLE fiscal\_year\_2021\_2022."total\_rides\_CLEAN" AS

SELECT \*

FROM fiscal\_year\_2021\_2022."total\_rides\_V4"

WHERE trip\_duration > 60 AND trip\_duration < 86400

ORDER BY trip\_duration DESC;

* 1. Return: 4397887 (4472599-74712= 4397887)

1. Rechecked for outliers:
   1. Return: 0
2. Exported clean CSV file: total\_rides\_2021-2022\_CLEAN

# Aggregate and Analyze

### Counted rides by user type and percentage of total

1. Divided total rides by user type:

SELECT user\_type, COUNT(\*) AS number\_of\_rides

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY user\_type

ORDER BY user\_type DESC

* 1. Return:

user\_type total

"member" 2618743

"casual" 1779144

1. Calculated percentage of rides by user:

SELECT user\_type,

COUNT(user\_type) AS total,

ROUND (COUNT(user\_type) \* 100.0 /

(SELECT COUNT(\*)

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN")) AS percent

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY user\_type

* 1. Return:

user\_type total percent

"casual" 1779144 40

"member" 2618743 60

1. Totaled monthly ride count and percentage of monthly rides by user type:

SELECT month, user\_type,

COUNT(\*) AS total,

ROUND(COUNT(\*) \* 100.0 / (SELECT COUNT(\*)

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN")) AS percent

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY month, user\_type

ORDER BY month

* 1. Return: months are out of order
     1. Example:

"month" "user\_type" "total" "percent"

"April" "casual" 90747 2

"April" "member" 177666 4

"August" "casual" 265563 6

"August" "member" 328365 7

"December" "casual" 44644 1

"December" "member" 129282 3

* 1. Put months in order

SELECT month, user\_type,

COUNT(\*) AS total,

ROUND(COUNT(\*) \* 100.0 / (SELECT COUNT(\*)

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN")) AS percent

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY month, user\_type

ORDER BY CASE WHEN month='January' THEN 1

WHEN month='February' THEN 2

WHEN month='March' THEN 3

WHEN month='April' THEN 4

WHEN month='May' THEN 5

WHEN month='June' THEN 6

WHEN month='July' THEN 7

WHEN month='August' THEN 8

WHEN month='September' THEN 9

WHEN month='October' THEN 10

WHEN month='November' THEN 11

ELSE 12

END;

* + 1. Return: this did nothing
  1. Checked length of string for month and day:

SELECT LENGTH (month)

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY month;

* + 1. Return: all months have 9 characters
  1. Trimmed newly created month and day columns

UPDATE fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

SET month = TRIM(month), day = TRIM (day);

* + 1. Return for month (days also returned trimmed):

"length"

5

6

8

8

7

4

4

5

3

8

7

9

* 1. Did the same for day column
  2. Reran original query (3b):
     1. Returned with months in order:

"month" "user\_type" "total" "percent"

"January" "member" 66554 2

"January" "casual" 12461 0

"February" "casual" 14950 0

"February" "member" 72656 2

"March" "casual" 66329 2

"March" "member" 146390 3

"April" "casual" 90747 2

"April" "member" 177666 4

"May" "casual" 216860 5

"May" "member" 277063 6

"June" "casual" 287406 7

"June" "member" 322107 7

"July" "casual" 306378 7

"July" "member" 324096 7

"August" "member" 328365 7

"August" "casual" 265563 6

"September" "member" 307658 7

"September" "casual" 217375 5

"October" "casual" 187206 4

"October" "member" 284038 6

"November" "member" 182868 4

"November" "casual" 69225 2

"December" "member" 129282 3

"December" "casual" 44644 1

* 1. Put days in order using the same script.

1. Totaled ride count and percentage of rides by user type by day of the week:

SELECT day, user\_type,

COUNT(\*) AS total,

ROUND(COUNT(\*) \* 100.0 / (SELECT COUNT(\*)

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN")) AS percent

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY day, user\_type

ORDER BY CASE WHEN day = 'Sunday' THEN 1

WHEN day = 'Monday' THEN 2

WHEN day = 'Tuesday' THEN 3

WHEN day = 'Wednesday' THEN 4

WHEN day = 'Thursday' THEN 5

WHEN day = 'Friday' THEN 6

ELSE 7

END;

* 1. Return:

"day" "user\_type" "total" "percent"

"Sunday" "casual" 309192 7

"Sunday" "member" 296518 7

"Monday" "casual" 206822 5

"Monday" "member" 366622 8

"Tuesday" "member" 420848 10

"Tuesday" "casual" 199917 5

"Wednesday" "member" 415557 9

"Wednesday" "casual" 203274 5

"Thursday" "casual" 221681 5

"Thursday" "member" 404778 9

"Friday" "member" 370376 8

"Friday" "casual" 255838 6

"Saturday" "casual" 382420 9

"Saturday" "member" 344044 8

### Aggregated trip durations by user type and bike type

1. Aggregated column trip duration:

SELECT ROUND(AVG(trip\_duration/60)) AS average\_ride\_duration

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

* 1. Return:

"average\_ride\_duration"

17

1. Aggregated trip\_duration by user type. Note that the average duration of a casual user's ride is ~twice as long as a member's ride.

SELECT user\_type,

ROUND(AVG(trip\_duration/60)) AS average\_ride\_duration,

MIN(trip\_duration/60) AS MIN\_ride\_duration,

MAX(trip\_duration/60) AS MAX\_ride\_duration

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY user\_type

* 1. Return:

"user\_type" "average\_ride\_duration" "min\_ride\_duration" "max\_ride\_duration"

"casual" 24 1 1439

"member" 12 1 1435

1. Compared number of rides, trip duration, and user type by month:

SELECT month, user\_type,

COUNT(\*) AS total,

ROUND(AVG(trip\_duration/60)) AS average\_ride\_duration

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY month, user\_type

ORDER BY CASE WHEN month='January' THEN 1

WHEN month='February' THEN 2

WHEN month='March' THEN 3

WHEN month='April' THEN 4

WHEN month='May' THEN 5

WHEN month='June' THEN 6

WHEN month='July' THEN 7

WHEN month='August' THEN 8

WHEN month='September' THEN 9

WHEN month='October' THEN 10

WHEN month='November' THEN 11

ELSE 12

END;

* 1. Return:

"month" "user\_type" "total" "average\_ride\_duration"

"January" "member" 66554 10

"January" "casual" 12461 18

"February" "casual" 14950 21

"February" "member" 72656 10

"March" "casual" 66329 26

"March" "member" 146390 11

"April" "casual" 90747 25

"April" "member" 177666 11

"May" "casual" 216860 27

"May" "member" 277063 13

"June" "casual" 287406 25

"June" "member" 322107 13

"July" "casual" 306378 25

"July" "member" 324096 13

"August" "member" 328365 13

"August" "casual" 265563 23

"September" "member" 307658 12

"September" "casual" 217375 22

"October" "casual" 187206 24

"October" "member" 284038 12

"November" "member" 182868 11

"November" "casual" 69225 20

"December" "member" 129282 10

"December" "casual" 44644 20

1. Compared number of rides, trip duration, and user type by day of the week:

SELECT day, user\_type,

COUNT(\*) AS total,

ROUND(AVG(trip\_duration/60)) AS average\_ride\_duration

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY day, user\_type

ORDER BY CASE WHEN day = 'Sunday' THEN 1

WHEN day = 'Monday' THEN 2

WHEN day = 'Tuesday' THEN 3

WHEN day = 'Wednesday' THEN 4

WHEN day = 'Thursday' THEN 5

WHEN day = 'Friday' THEN 6

ELSE 7

END;

* 1. Return:

"day" "user\_type" "total" "average\_ride\_duration"

"Sunday" "casual" 309192 28

"Sunday" "member" 296518 14

"Monday" "casual" 206822 25

"Monday" "member" 366622 12

"Tuesday" "member" 420848 12

"Tuesday" "casual" 199917 22

"Wednesday" "member" 415557 12

"Wednesday" "casual" 203274 21

"Thursday" "casual" 221681 21

"Thursday" "member" 404778 12

"Friday" "member" 370376 12

"Friday" "casual" 255838 22

"Saturday" "casual" 382420 27

"Saturday" "member" 344044 14

1. Checked percentage of use by bike type by total rides and average duration:

SELECT bike\_type,

COUNT(bike\_type) AS total\_rides,

ROUND(AVG(trip\_duration/60)) AS average\_ride\_duration,

ROUND (COUNT(bike\_type) \* 100.0 /

(SELECT COUNT(\*)

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN")) AS percent

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY bike\_type

* 1. Return:

"bike\_type" "total\_rides" "average\_ride\_duration" "percent"

"classic\_bike" 2695565 17 61

"docked\_bike" 188124 48 4

"electric\_bike" 1514198 14 34

1. Checked percentage of use of bike types by user

SELECT user\_type, bike\_type,

COUNT(user\_type) AS total,

ROUND(AVG(trip\_duration/60)) AS average\_ride\_duration,

ROUND(COUNT(user\_type) \* 100.0 / (SELECT COUNT(\*) FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN")) AS percent

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY user\_type, bike\_type

* 1. Return:

"user\_type" "bike\_type" "total" "average\_ride\_duration" "percent"

"casual" "classic\_bike" 925549 24 21

"casual" "docked\_bike" 188124 48 4

"casual" "electric\_bike" 665471 17 15

"member" "classic\_bike" 1770016 13 40

"member" "electric\_bike" 848727 11 19

### Investigated types of trips taken by user type

1. Compared number of round trips and their average duration of each user type by bike type

(used start\_station\_id and end\_station\_id columns since they are numbers, they are more reliable than names - which could be alternatively typed.)

SELECT user\_type, COUNT (\*) AS number\_of\_round\_trips

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

WHERE start\_station\_id = end\_station\_id

GROUP BY user\_type

* 1. Return:

"user\_type" "number\_of\_round\_trips"

"casual" 154171

"member" 72341

1. Calculated most used stations of user types:

SELECT start\_station\_id AS most\_used\_station,

COUNT(\*) AS num\_trips,

ROUND(AVG(trip\_duration)/60) AS duration\_in\_mins

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY

start\_station\_id

ORDER BY

num\_trips DESC

LIMIT 6

* 1. Return:

"most\_used\_station" "num\_trips" "duration\_in\_mins "

"13022" 70719 33

"13300" 39310 34

"LF-005" 37616 25

"13042" 37111 31

"TA1308000050" 35990 14

"13008" 33958 35

* 1. Checked names of most used stations:

SELECT start\_station\_id, start\_station\_name

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

WHERE start\_station\_id IN ('13022', '13300', 'LF-005', '13042', 'TA1308000050', '13008')

* 1. GROUP BY start\_station\_id, start\_station\_name;
     1. Return

"start\_station\_id" "start\_station\_name"

"13008" "Millennium Park"

"13022" "Streeter Dr & Grand Ave"

"13042" "Michigan Ave & Oak St"

"13300" "DuSable Lake Shore Dr & Monroe St"

"LF-005" "DuSable Lake Shore Dr & North Blvd"

"TA1308000050" "Wells St & Concord Ln"

1. Calculated most used stations of casual:

SELECT user\_type, start\_station\_id AS most\_used\_station, start\_station\_name,

COUNT(start\_station\_id) AS num\_trips,

ROUND(AVG(trip\_duration)/60) AS duration\_in\_mins

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

WHERE user\_type = 'casual'

GROUP BY

user\_type, start\_station\_id, start\_station\_name

ORDER BY

num\_trips DESC

LIMIT 6

* 1. Return:

"user\_type" "most\_used\_station" "start\_station\_name" "num\_trips" "duration\_in\_mins "

"casual" "13022" "Streeter Dr & Grand Ave" 54792 37

"casual" "13300" "DuSable Lake Shore Dr & Monroe St" 30270 37

"casual" "13008" "Millennium Park" 25080 41

"casual" "13042" "Michigan Ave & Oak St" 23659 37

"casual" "LF-005" "DuSable Lake Shore Dr & North Blvd" 22130 30

"casual" "15544" "Shedd Aquarium" 19293 31

1. Calculated most used stations of member:

SELECT user\_type, start\_station\_id AS most\_used\_station, start\_station\_name,

COUNT(start\_station\_id) AS num\_trips,

ROUND(AVG(trip\_duration)/60) AS duration\_in\_mins

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

WHERE user\_type = 'member'

GROUP BY

user\_type, start\_station\_id, start\_station\_name

ORDER BY

num\_trips DESC

LIMIT 6

* 1. Return:

"user\_type" "most\_used\_station" "start\_station\_name" "num\_trips" "duration\_in\_mins"

"member" "KA1503000043" "Kingsbury St & Kinzie St" 24567 9

"member" "TA1307000039" "Clark St & Elm St" 21451 12

"member" "TA1308000050" "Wells St & Concord Ln" 20645 12

"member" "WL-012" "Clinton St & Washington Blvd" 18654 11

"member" "TA1305000032" "Clinton St & Madison St" 18483 11

"member" "KA1504000135" "Wells St & Elm St" 18242 11

1. Calculated least used stations of user types:

SELECT start\_station\_id AS most\_used\_station, start\_station\_name,

COUNT(start\_station\_id) AS num\_trips,

ROUND(AVG(trip\_duration)/60) AS duration\_in\_mins

FROM fiscal\_year\_2021\_2022."total\_rides\_CLEAN"

GROUP BY

start\_station\_id, start\_station\_name

ORDER BY

num\_trips ASC

LIMIT 6

* 1. Return:

"most\_used\_station" "start\_station\_name" "num\_trips" "duration\_in\_mins"

"1032" "Public Rack - Kedvale Ave & 63rd St" 1 5

"1033" "Public Rack - Pulaski Rd &amp; 65th St" 1 11

"1018" "Public Rack - Kostner Ave & Wrightwood Ave" 1 102

"1030" "Public Rack - Lawndale & 63rd St" 1 37

"1015" "Public Rack - Peterson Ave & Drake Ave" 1 25

"1034" "Public Rack - Kenneth Ave & 63rd St E" 1 8

1. Calculated least used stations of casual:
   1. Return:

"user\_type" "most\_used\_station" "start\_station\_name" "num\_trips" "duration\_in\_mins"

"casual" "1036" "Public Rack - Kedzie Ave & 60th St" 1 3

"casual" "1038" "Public Rack - Kedzie Ave &amp; 62nd Pl" 1 6

"casual" "1032" "Public Rack - Kedvale Ave & 63rd St" 1 5

"casual" "1030" "Public Rack - Lawndale & 63rd St" 1 37

"casual" "1018" "Public Rack - Kostner Ave & Wrightwood Ave" 1 102

"casual" "1040" "Public Rack - Talman Ave & Pershing Rd" 1 40

1. Calculated least used stations of member:
   1. Return:

"user\_type" "most\_used\_station" "start\_station\_name" "num\_trips" "duration\_in\_mins" "member" "1034" "Public Rack - Kenneth Ave & 63rd St E" 1 8

"member" "1036" "Public Rack - Kedzie Ave & 60th St" 1 6

"member" "1033" "Public Rack - Pulaski Rd &amp; 65th St" 1 11

"member" "1015" "Public Rack - Peterson Ave & Drake Ave" 1 25

"member" "1016" "Public Rack - Peterson Ave & Bernard Ave" 1 9

"member" "1039" "Public Rack - Kedzie Ave & 61st Pl W" 1 4

##### 