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).
    a. 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).
    b. 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
        i. 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);
```

    c. Checked each file for proper import by column names and column and row count (not including header):

i.	Oct_2021	631226	13
ii.	Nov_2021	359978	13
iii.	Dec_2021	247540	13
iv.	Jan_2022	103770	13
v.	Feb_2022	115609	13
vi.	Mar_2022	284042	13
vii.	Apr_2022	371249	13

viii.	May_2022	634858	13
ix.	Jun_2022	769204	13
x.	July_2022	823488	13
xi.	Aug_2022	785932	13
xii.	Sept_2022	701339	13

3. Examined the content and layout of Oct_2021 to get a feel for the data.
   a.
      ```
      SELECT * FROM fiscal_year_2021_2022."Oct_2021"
      LIMIT 100
      ```
   b. Looked at variables for 'rideable_type' and 'member_casual'
      ```
      SELECT
         DISTINCT rideable_type
      FROM fiscal_year_2021_2022."Oct_2021";
      ```
      i. Return is:
         ```
         "rideable_type"
         "classic_bike"
         "docked_bike"
         "electric_bike"
         And
         "member_casual"
         "casual"
         "member"
         ```
   c. 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
   a. 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;
      ```
      i. fiscal_year_2021_2022."1Q" Total rows: 1238744
      ii. fiscal_year_2021_2022."2Q" Total rows: 503421
      iii. fiscal_year_2021_2022."3Q" Total rows: 1775311
      iv. fiscal_year_2021_2022."4Q" Total rows: 2310759
   b. 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");
```

   i. fiscal_year_2021_2022."total_tides" 5828235

c. Checked row and column count:
   i. 1238744 + 503421 + 1775311 + 2310759 = 5828235
   ii. 13 columns

2. 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;
```

   a. 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
```

   b. (895032 x 2) + (958227 x 2) + (5844 x 2)= ((895032 * 2) + (958227 * 2)) + (5844 * 2) = 3718206

3. 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%');
```

   a. 4474141 columns remain; 5828235 - 4474141 = 1354094 were removed.

4. Checked for duplicate ride_id values: 0

```
SELECT ride_id FROM fiscal_year_2021_2022."total_rides_V2"
```

   a. 4474141 variables, so no duplicates

5. 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 '% ';
```

   a. start_station_name return: 71
   b. start_station_name return: 84

6. 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);
```

   a. Check = 0
   b. Check = 0

7. Checked for uniformity of character length in ride_id
   a. Check length of string:

```
SELECT LENGTH(ride_id)
FROM fiscal_year_2021_2022."total_rides_V2";
```

   b. Return = 16 characters long
   c. Check uniformity:

```
SELECT ride_id
FROM fiscal_year_2021_2022."total_rides_V2"
WHERE LENGTH(ride_id) <> 16
```

   d. Return = 0

8. Found and removed stations warehouse, repair, and charging from columns start_station_name and end_station_name:
   a. Warehouse
      i. Base - 2132 W Hubbard = 890/127
      ii. Base - 2132 W Hubbard Warehouse = 317/134
      iii. Hastings WH 2 = 2/2
   b. Mobile stations
      i. DIVVY CASSETTE REPAIR MOBILE STATION = 0/6
      ii. Throop/Hastings Mobile Station = 1/1
   c. Charging stations
      i. Bissell St & Armitage Ave - Charging = 18/20
      ii. Lincoln Ave & Roscoe St - Charging = 3/3
      iii. Pawel Bialowas - Test- PBSC charging station = 1/1
      iv. 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%');
```

d. Check: row count: 4472892

e. Removed same stations from column end_station_name:

   i.
   ```
   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%';
   ```

f. Check: row count: 4472599

g. 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%';
   ```

   i. Return: 30446 rows

   ii. I did not remove these.

9. Standardized column data-type and labels:

   a. Did not need to retype/cast data (see above)

   b. Relabelled columns and check:
      Example:

      ```
      ALTER TABLE fiscal_year_2021_2022."total_rides_V3"
      RENAME COLUMN ride_id to trip_id
      ```

   i. ride_id > trip_id
   ii. rideable_type > bike_type
   iii. member_casual > user_type

10. 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;
    ```

    a. Check that columns were created
11. Created new columns: end_date, end_time:
    a. Check that columns were created
12. Created columns: month and day
    a. ```
       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');
       ```
    b. ```
       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');
       ```
    c. Check that columns were created
13. 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));
    ```

14. Checked trip_duration column for outliers

    ```
    SELECT
    MIN(trip_duration),
    MAX(trip_duration)
    FROM fiscal_year_2021_2022."total_rides_V4";
    ```

    a. Noted that trip_duration has trips under 60 seconds long (-7621 seconds).
    b. Noted that trip_duration has trips over 86400 seconds long (over 24hrs [2442301 seconds]) as they are likely stolen.
15. Checked how many rows have outliers

    ```
    SELECT *
    FROM fiscal_year_2021_2022."total_rides_V4"
    WHERE trip_duration <= 60 or trip_duration >= 86400;
    ```

    a. Return: 74712 rows
16. 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;
```

   a. Return: 4397887 (4472599-74712= 4397887)

17. Rechecked for outliers:
    a. Return: 0
18. 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
```

   a. Return:

```
user_type       total
"member"        2618743
"casual"        1779144
```

2. 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
```

   a. Return:

```
user_type       total           percent
"casual"        1779144         40
"member"        2618743         60
```

3. 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
```

   a. Return: months are out of order

i. 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
```

b. 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;
```

i. Return: this did nothing

c. Checked length of string for month and day:

```
SELECT LENGTH (month)
FROM fiscal_year_2021_2022."total_rides_CLEAN"
GROUP BY month;
```

i. Return: all months have 9 characters

d. Trimmed newly created month and day columns

```
UPDATE fiscal_year_2021_2022."total_rides_CLEAN"
  SET month = TRIM(month), day = TRIM (day);
```

i. Return for month (days also returned trimmed):

```
"length"
5
6
8
```

```
8
7
4
4
5
3
8
7
9
```

e. Did the same for day column

f. Reran original query (3b):

   i. 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
```

g. Put days in order using the same script.

4. 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;
```

a. 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

5. Aggregated column trip duration:

```
SELECT ROUND(AVG(trip_duration/60)) AS average_ride_duration
FROM fiscal_year_2021_2022."total_rides_CLEAN"
```

a. Return:

```
"average_ride_duration"
17
```

6. 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
```

a. Return:

```
"user_type"   "average_ride_duration"   "min_ride_duration"   "max_ride_duration"
"casual"                 24                       1                      1439
"member"                 12                       1                      1435
```

7. 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;
```

a. 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
```

8. 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;
```

   a. 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
```

9. 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
```

a. Return:

```
"bike_type"      "total_rides"   "average_ride_duration"   "percent"
"classic_bike"    2695565                  17                    61
"docked_bike"     188124                   48                    4
"electric_bike"   1514198                  14                    34
```

10. 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
```

a. 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

11. 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
```

a. Return:

```
"user_type"   "number_of_round_trips"
"casual"       154171
"member"       72341
```

12. 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
```

a. 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
```

b. 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')
```

c.
```
GROUP BY start_station_id, start_station_name;
```

i. 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"
```

13. 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
```

a. 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
```

14. 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
```

a. 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
```

15. 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
```

a. 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
```

16. Calculated least used stations of casual:
    a. 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
```

17. Calculated least used stations of member:
    a. 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
```