MP : Metrocar Analysis

Project background:

Metrocar management assigned me the responsibility of conducting a customer funnel analysis to pinpoint areas that require improvement and optimisation. During the meeting, I will present my findings and recommendations using slides to illustrate the findings effectively.

Metrocar is a ride-sharing app (similar to Uber/Lyft).

Tools used: SQL (analysis) and Tableau (further analysis and visualisations).

Project:

Data source: The data for this dataset was generated specifically for this project. Inspired by publicly available datasets for Uber/Lyft.

Challenges:

1. Initial challenge was to extract data from Beekeeper. Beekeeper limits resulting dataset downloads to 50,000 rows. Work around was to download the tables (separate feature that is not limited to any number of rows) and upload to local PostgreSQL database and where I could query the tables and also prepare dataset for further analysis and visualisation in Tableau.
2. Data contains a lot of unknown user age range that could have affected the results related to age_range

Summary: The primary conclusion is that a significant proportion of Metrocar users, specifically 60% of the total user base, utilise the iOS platform. Additionally, the majority of these users fall within the age range of 25 to 44 years. Furthermore, the distribution across other platforms includes 30% on Android and 10% accessing via the web.

In my analysis of the user funnel, a significant observation emerged. The most noteworthy aspect is the substantial decline in user engagement, specifically a 50% drop, occurring primarily due to cancellations after a ride has been requested and accepted. This notable reduction raises concerns and prompts the need for a deeper investigation. The available data doesn't currently provide many insights into the underlying reasons for this steep decline, necessitating a more in-depth exploration.

Business questions:

▼ **What steps of the funnel should we research and improve? Are there any specific drop-off points preventing users from completing their first ride?**

The conversion rate stands at only 50.77 percent during the cancellation stage, indicating a need for deeper investigation. The data indicates that cancellations linked to drop-offs are evenly spread across various geolocations, making it challenging to establish any correlation between drop-offs and cancellations.

▼ **Metrocar currently supports 3 different platforms: ios, android, and web. To recommend where to focus our marketing budget for the upcoming year, what insights can we make based on the platform?**

The data unveils that iOS remains the most engaged platform across all stages of the funnel. When considering revenue alone, iOS leads with an impressive $2,721,960, while Android lags behind at $1,307,676, and the web contributes a modest $442,544. Given the cancellation rate of 50%, my initial suggestion is to channel our efforts into further investigation, aiming to pinpoint the underlying reasons behind the subpar conversion rate. Additionally, marketing effort should be focused on engaging android and web platform users to bolster our market presence.

▼ **What age groups perform best at each stage of our funnel? Which age group(s) likely contain our target customers?**

Users within the age range of 25 to 44 years consistently outperforms “younger” 12-24 and older “45-54” age groups. Analysis can be found at:

Code:

```
-- Checking data integrity: missing values, duplicates, outliers, typos
SELECT COUNT(*)
FROM app_downloads; -- 23608 total records

SELECT COUNT(app_download_key)
FROM app_downloads; -- 23608 unique key values as expected

SELECT platform, COUNT(app_downloads)
FROM app_downloads
GROUP BY 1; -- IOS 14290, WEB 2383, android 6935. So we have 3 platforms no typos

SELECT AVG(length(app_download_key))
FROM app_downloads  -- avg 32. Each key has 32 chararacter length

SELECT MIN(download_ts), MAX(download_ts)
FROM app_downloads; -- MIN 2021-01-01 00:05:59, MAX 2021-12-31 23:52:27. Table data interval

SELECT COUNT(*)
FROM signups; -- 17623

SELECT COUNT(user_id)
FROM signups -- 17623


SELECT COUNT(*)
FROM ride_requests; -- 385477

SELECT COUNT(*)
FROM transactions; -- 223652

SELECT COUNT(*)
FROM reviews; -- 156211
```

```
--Full table join.
SELECT a.app_download_key,
    a.platform,
    a.download_ts,
    s.session_id,
    s.user_id,
    s.age_range,
    s.signup_ts,
    r.ride_id,
    r.driver_id,
    r.request_ts,
    r.accept_ts,
    r.pickup_location,
    r.dropoff_location,
    r.pickup_ts,
    r.dropoff_ts,
    r.cancel_ts,
    t.transaction_id,
    t.purchase_amount_usd,
    t.charge_status,
    t.transaction_ts,
    re.review_id,
    re.rating,
    re.review
   FROM app_downloads a
     FULL JOIN signups s ON a.app_download_key = s.session_id
     FULL JOIN ride_requests r ON s.user_id = r.user_id
     FULL JOIN transactions t ON r.ride_id = t.ride_id
     FULL JOIN reviews re ON t.ride_id = re.ride_id;
```

Constructing tables for analysis in Tableau:

```
-- Metrocar Funnel
-- Defining download stage
WITH  st1 AS (SELECT 0 AS stage, 'download' AS stage_name,
       platform, age_range, download_ts::date AS "date",
       COUNT(app_download_key) AS user_count, 0 AS ride_count
FROM app_downloads
LEFT JOIN signups
ON app_download_key = session_id
GROUP BY 3,4,5),
-- Defining signup stage
    st2 AS (SELECT 1 AS stage, 'signup' AS stage_name,
       platform, age_range, download_ts::date as "date",
       COUNT(user_id) AS user_count, 0 AS ride_count
FROM app_downloads
LEFT JOIN signups
ON app_download_key = session_id
WHERE session_id IS NOT NULL
GROUP BY 3,4,5),
-- Defining request stage
    st3 AS (SELECT 2 AS stage, 'request' AS stage_name,
           platform, age_range, download_ts::date AS "date",
           COUNT(DISTINCT user_id) AS user_count, COUNT(ride_id) AS ride_count
FROM ride_requests AS r
LEFT JOIN signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE request_ts IS NOT NULL
GROUP BY 3,4,5),
-- Defining accepted stage
    st4 AS (SELECT 3 AS stage, 'accepted' AS stage_name,
           platform, age_range, download_ts::date AS "date",
           COUNT(DISTINCT user_id) AS user_count, COUNT(ride_id) AS ride_count
FROM ride_requests AS r
LEFT JOIN signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE accept_ts IS NOT NULL
GROUP BY 3,4,5),
-- Defining completed stage
    st5 AS (SELECT 4 AS stage, 'completed' AS stage_name,
           platform, age_range, download_ts::date AS "date",
           COUNT(DISTINCT user_id) AS user_count, COUNT(ride_id) AS ride_count
FROM ride_requests AS r
LEFT JOIN signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE dropoff_ts IS NOT NULL
GROUP BY 3,4,5),
-- Defining payment stage
    st6 AS (SELECT 5 AS stage, 'payment' AS stage_name,
           platform, age_range, download_ts::date AS "date",
           COUNT(DISTINCT user_id) AS user_count, COUNT(ride_id) AS ride_count
FROM ride_requests AS r
LEFT JOIN signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
LEFT JOIN transactions AS t
USING(ride_id)
WHERE charge_status = 'Approved'
GROUP BY 3,4,5),
-- Defining review stage
    st7 AS (SELECT 6 AS stage, 'review' AS stage_name,
           platform, age_range, download_ts::date AS "date",
           COUNT(DISTINCT r.user_id) AS user_count, COUNT(ride_id) AS ride_count
FROM ride_requests AS r
LEFT JOIN signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
LEFT JOIN reviews AS t
USING(ride_id)
WHERE review IS NOT NULL
GROUP BY 3,4,5)

-- glueing the stages together
SELECT * FROM st1
UNION
SELECT * FROM st2
UNION
SELECT * FROM st3
UNION
SELECT * FROM st4
UNION
SELECT * FROM st5
UNION
SELECT * FROM st6
UNION
SELECT * FROM st7
ORDER BY 1,3,4,5
```

```
-- examining average time trips get accepted. completed trips of cancelled trips
SELECT  AVG(accept_ts - request_ts) AS avg_accept_time, COUNT(*),
    CASE  WHEN cancel_ts IS NULL THEN 'completed'
        ELSE 'cancelled' END AS cancelled_status FROM ride_requests
GROUP BY cancelled_status
ORDER BY 1
```

```
-- complete funnel by Platform and Age
WITH stage1 AS (SELECT platform, age_range, COUNT(*) AS downloads
FROM app_downloads AS a
LEFT JOIN signups AS s
ON a.app_download_key = s.session_id
GROUP BY 1,2),
stage2 AS (SELECT platform, age_range, COUNT(user_id) AS users
FROM signups AS s
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
GROUP BY 1,2
),
stage3 AS (SELECT platform, age_range, COUNT(DISTINCT user_id) AS requested
FROM ride_requests AS r
LEFT join signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE request_ts IS NOT NULL
GROUP BY 1,2),
stage4 AS (SELECT platform, age_range, COUNT(DISTINCT user_id) AS accepted
FROM ride_requests AS r
LEFT join signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE accept_ts IS NOT NULL
GROUP BY 1,2),
stage5 AS (SELECT platform, age_range, COUNT(DISTINCT user_id) AS completed
FROM ride_requests AS r
LEFT join signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE dropoff_ts IS NOT NULL
GROUP BY 1,2),
stage6 AS (SELECT platform, age_range, COUNT(DISTINCT user_id) AS payment
FROM transactions AS t
LEFT join ride_requests AS r
USING(ride_id)
LEFT JOIN signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE charge_status = 'Approved'
GROUP BY 1,2),
stage7 AS (SELECT platform, age_range, COUNT(DISTINCT user_id) AS review
FROM reviews AS r
LEFT JOIN signups AS s
USING(user_id)
LEFT JOIN app_downloads AS a
ON a.app_download_key = s.session_id
WHERE rating IS NOT NULL
GROUP BY 1,2)
```

```
SELECT *
FROM stage1
JOIN stage2 USING(platform, age_range)
JOIN stage3 USING(platform, age_range)
JOIN stage4 USING(platform, age_range)
JOIN stage5 USING(platform, age_range)
JOIN stage6 USING(platform, age_range)
JOIN stage7 USING(platform, age_range)
```

SQL analysis:

▼ **How many times was the app downloaded?**

```
-- total downloads
SELECT COUNT(*)
FROM app_downloads;
```

count: 23,608

▼ **How many users signed up on the app?**

```
-- code logic: check whether any user:
-- did not sign up (signup_ts) = 0
-- signed up once (signup_ts) = 1
-- sign up more than once (signup_ts) > 0
WITH users_signup_frequency AS(
  SELECT user_id, COUNT(signup_ts) AS signup_instances
  FROM signups
  GROUP BY 1
  HAVING COUNT(signup_ts) = 1)
SELECT COUNT(*) AS unique_signups
FROM users_signup_frequency;
```

count: 17,623

▼ **How many unique users requested a ride through the Metrocar?**

```
SELECT COUNT(DISTINCT user_id)
FROM ride_requests;
```

count: 12406

▼ **How many unique users completed a ride through the Metrocar app?**

```
SELECT COUNT(DISTINCT user_id)
FROM ride_requests
WHERE cancel_ts IS NULL;
```

count: 6233

▼ **How many rides were requested through the app?**

```
SELECT COUNT(request_ts)
FROM ride_requests;
```

count: 385,477

▼ How many rides were requested and completed through the app?

```
-- trips that where requested and journey completed
SELECT COUNT (*) - COUNT(cancel_ts) AS journeys_completed
FROM ride_requests;
```

count: 223,652

▼ What is the average time of a ride from pick up to drop off? Shortest? Longest?

```
SELECT  AVG(dropoff_ts - pickup_ts) AS average,
        MIN(dropoff_ts - pickup_ts) AS shortest,
        MAX(dropoff_ts - pickup_ts) AS longest
FROM ride_requests
WHERE pickup_ts IS NOT NULL;
```

AVG time: 00:52:36.738773

▼ How many rides were accepted by a driver?

```
SELECT COUNT(accept_ts)
FROM ride_requests;
```

count: 248,379

▼ How many rides did we successfully collect payments and how much was collected?

```
SELECT  charge_status,
        COUNT(purchase_amount_usd) AS trips,
        ROUND(SUM(purchase_amount_usd)::dec, 2) AS amount
FROM transactions
WHERE charge_status = 'Approved' -- where payment taken
GROUP BY 1;
```

trips: 212,628; sum: 4,251,667.61

▼ How many ride requests happened on each platform?

```
SELECT platform, COUNT(request_ts)
FROM app_downloads AS a
LEFT JOIN signups AS s
ON a.app_download_key = s.session_id
JOIN ride_requests AS r
ON s.user_id = r.user_id
GROUP BY 1;
```

android: 112,317; ios: 234,693; web: 38467

▼ **What is the drop-off from users signing up to users requesting a ride?**

```
WITH  signup AS (SELECT user_id AS users_s FROM signups
                ), -- total signups
      rides AS (SELECT DISTINCT user_id AS users_r FROM ride_requests
                WHERE request_ts IS NOT NULL) -- total unique riders that made a booking

-- % of users that did not make a booking
SELECT ROUND((1 - COUNT(users_r)::dec / COUNT(users_s)::dec) * 100, 3) AS user_drop_off_pct
FROM signup AS s
LEFT JOIN rides AS r
ON (s.users_s = r.users_r);
```

User drop-off % 29.603

▼ **Of the users that signed up on the app, what percentage these users requested a ride?**

```
WITH requests AS (SELECT user_id, COUNT(request_ts)
                  FROM ride_requests
                  GROUP BY 1)
SELECT ROUND(COUNT(r.user_id)::dec /
             COUNT(s.user_id)::dec * 100, 4) AS signedups_request_pct
FROM signups AS s
LEFT JOIN requests AS r
ON s.user_id = r.user_id;
```

pct: 70.3966

▼ **Of the users that signed up on the app, what percentage these users completed a ride?**

```
WITH requests AS (SELECT user_id, COUNT('completed')
                  FROM ride_requests
                  WHERE cancel_ts IS NULL
                  GROUP BY 1)
SELECT ROUND(COUNT(r.user_id)::dec /
             COUNT(s.user_id)::dec * 100, 4) AS signedups_completed_pct
FROM signups AS s
LEFT JOIN requests AS r
ON s.user_id = r.user_id;
```

pct: 35.3686

▼ **PP1 Using the percent of previous approach, what are the user-level conversion rates for the first 3 stages of the funnel (app download to signup and signup to ride requested)?**

```
-- Extracting app downloads, users and download count:
WITH  stage1_dloads AS (SELECT app_download_key, user_id, COUNT(user_id)
                        FROM app_downloads AS a
                        LEFT JOIN signups AS s
                        ON a.app_download_key = s.session_id
                        GROUP BY 1,2),
-- Extracting users who signup_ts (and session_id to connect to downloads):
      stage2_signup AS (SELECT user_id, signup_ts, session_id
                        FROM signups),
-- Defining unique users who requested a trip and their trip request count:
      stage3_rqst AS (SELECT user_id, COUNT(request_ts) AS request_count
                       FROM ride_requests
                        GROUP BY 1)
-- User funnel flow pct:
SELECT  ROUND(COUNT(st2.user_id)::dec / COUNT(app_download_key) * 100, 1) AS stage1_to_2_pct,
        ROUND(COUNT(st3.user_id)::dec / COUNT(st2.user_id) * 100, 1) AS stage2_to_3_pct
FROM stage1_dloads AS st1
LEFT JOIN stage2_signup AS st2
ON  st1.app_download_key = st2.session_id
LEFT JOIN stage3_rqst AS st3
ON st1.user_id = st3.user_id;
```

pct: 74.6; pct: 70.4

▼ **PT1 Using the percent of top approach, what are the user-level conversion rates for the first 3 stages of the funnel (app download to signup and signup to ride requested)?**

```
-- Extracting app downloads, users and download count:
WITH  stage1_dloads AS (SELECT app_download_key, user_id, COUNT(user_id)
                        FROM app_downloads AS a
                        LEFT JOIN signups AS s
                        ON a.app_download_key = s.session_id
                        GROUP BY 1,2),
-- Extracting users who signup_ts (and session_id to connect to downloads):
      stage2_signup AS (SELECT user_id, signup_ts, session_id
                        FROM signups),
-- Defining unique users who requested a trip and their trip request count:
      stage3_rqst AS (SELECT user_id, COUNT(request_ts) AS request_count
                       FROM ride_requests
                        GROUP BY 1)
-- User funnel flow pct:
SELECT  ROUND(COUNT(st2.user_id)::dec / COUNT(app_download_key) * 100, 1) AS stage1_to_2_pct,
        ROUND(COUNT(st3.user_id)::dec / COUNT(app_download_key) * 100, 1) AS stage1_to_3_pct
FROM stage1_dloads AS st1
LEFT JOIN stage2_signup AS st2
ON  st1.app_download_key = st2.session_id
LEFT JOIN stage3_rqst AS st3
ON st1.user_id = st3.user_id;
```

pct: 74.6; pct: 52.5

▼ **PP2 Using the percent of previous approach, what are the user-level conversion rates for the following 3 stages of the funnel? 1. signup, 2. ride requested, 3. ride completed**

▼ code 1

```
-- Defining unique signed users on signup level:
WITH  stage1_signup   AS (SELECT user_id, signup_ts
                          FROM signups
                          ),
-- Defining unique users who requested trips and their trips count:
      stage2_request  AS (SELECT user_id, COUNT(request_ts) AS trip_count
                          FROM ride_requests
                          GROUP BY 1),
-- Defining unique users who completed at least 1 trip and their trip completion count :
      stage3_final    AS (SELECT user_id, COUNT(dropoff_ts) AS dropoff_count
                           FROM ride_requests
                           WHERE dropoff_ts IS NOT NULL
                           GROUP BY 1)
-- User funnel flow pct:
SELECT ROUND(COUNT(trip_count)::dec/ COUNT(st1.user_id)* 100, 1) AS stage1_to_2_pct,
       ROUND(COUNT(dropoff_count)::dec/ COUNT(trip_count) * 100, 1) AS stage2_to_3_pct
FROM stage1_signup AS st1
LEFT JOIN stage2_request AS st2
USING (user_id)
LEFT JOIN stage3_final AS st3
USING (user_id);
```

▼ code 2

```
-- Extracting app downloads, users and download count:
WITH  stage1_dloads AS (SELECT app_download_key, user_id, COUNT(user_id)
                        FROM app_downloads AS a
                        LEFT JOIN signups AS s
                        ON a.app_download_key = s.session_id
                        GROUP BY 1,2),
-- Extracting users who signup_ts (and session_id to connect to downloads):
      stage2_signup AS (SELECT user_id, signup_ts, session_id
                          FROM signups),
-- Defining unique users who requested a trip and their trip request count:
      stage3_rqst AS (SELECT user_id, COUNT(request_ts) AS request_count
                         FROM ride_requests
                          GROUP BY 1),
-- Defining unique users who completed at least 1 trip and their trip completion count :
      stage4_final    AS (SELECT user_id, COUNT(dropoff_ts) AS dropoff_count
                          FROM ride_requests
                          WHERE dropoff_ts IS NOT NULL
                          GROUP BY 1)
-- User funnel flow pct:
SELECT  ROUND(COUNT(st2.user_id)::dec / COUNT(app_download_key) * 100, 1) AS stage1_to_2_pct,
        ROUND(COUNT(st3.user_id)::dec / COUNT(st2.user_id) * 100, 1) AS stage2_to_3_pct,
        ROUND(COUNT(st4.dropoff_count)::dec / COUNT(st3.user_id)* 100, 1) AS stage3_to_4_pct
FROM stage1_dloads AS st1
LEFT JOIN stage2_signup AS st2
ON  st1.app_download_key = st2.session_id
LEFT JOIN stage3_rqst AS st3
ON st1.user_id = st3.user_id
LEFT JOIN stage4_final AS st4
ON st1.user_id = st4.user_id;
```

pct: 70.4; pct: 50.2

▼ PT2 Using the percent of top approach, what are the user-level conversion rates for the following 3 stages of the funnel? 1. signup, 2. ride requested, 3. ride completed (hint: signup is the top of this funnel)

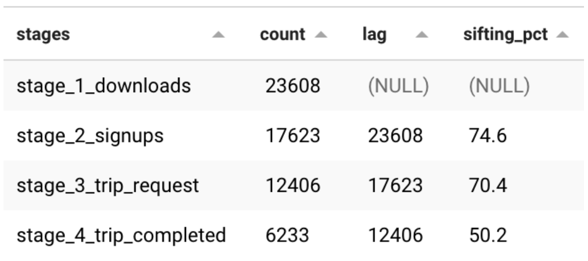
```
-- Extracting app downloads, users and download count:
WITH  stage1_dloads AS (SELECT app_download_key, user_id, COUNT(user_id)
                        FROM app_downloads AS a
                        LEFT JOIN signups AS s
                        ON a.app_download_key = s.session_id
                        GROUP BY 1,2),
-- Extracting users who signup_ts (and session_id to connect to downloads):
      stage2_signup AS (SELECT user_id, signup_ts, session_id
                          FROM signups),
-- Defining unique users who requested a trip and their trip request count:
      stage3_rqst AS (SELECT user_id, COUNT(request_ts) AS request_count
                         FROM ride_requests
                          GROUP BY 1),
-- Defining unique users who completed at least 1 trip and their trip completion count :
      stage4_final    AS (SELECT user_id, COUNT(dropoff_ts) AS dropoff_count
                          FROM ride_requests
                          WHERE dropoff_ts IS NOT NULL
                          GROUP BY 1)
-- User funnel flow pct:
SELECT  --ROUND(COUNT(st2.user_id)::dec / COUNT(app_download_key) * 100, 1) AS stage1_to_2_pct,
        ROUND(COUNT(st3.user_id)::dec / COUNT(st2.user_id) * 100, 1) AS stage2_to_3_pct,
        ROUND(COUNT(st4.dropoff_count)::dec / COUNT(st2.user_id)* 100, 1) AS stage3_to_4_pct
FROM stage1_dloads AS st1
LEFT JOIN stage2_signup AS st2
ON  st1.app_download_key = st2.session_id
LEFT JOIN stage3_rqst AS st3
ON st1.user_id = st3.user_id
LEFT JOIN stage4_final AS st4
ON st1.user_id = st4.user_id;
```

pct: 70.4; pct: 35.4

LAG() method

▼ PP1, PP2 percent of previous

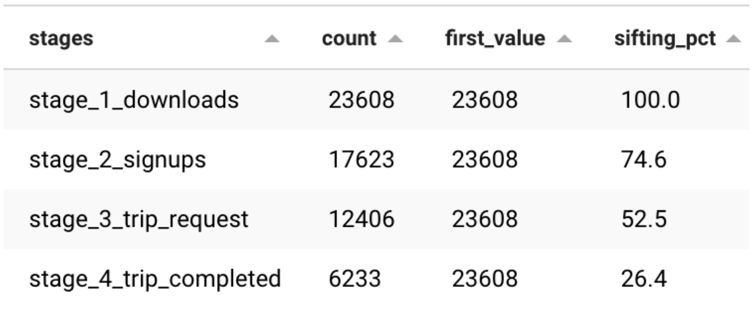
```
-- Constructing user_flow funnel:
With  user_flow AS
(
  SELECT COUNT(app_download_key),
  'stage_1_downloads' AS stages
  FROM app_downloads
 UNION
  SELECT COUNT(user_id),
  'stage_2_signups' AS stages
  FROM signups
 UNION
  SELECT COUNT(DISTINCT user_id),
  'stage_3_trip_request' AS stages
  FROM ride_requests
 UNION
  SELECT COUNT(DISTINCT user_id),
  'stage_4_trip_completed' AS stages
  FROM ride_requests
  WHERE dropoff_ts IS NOT NULL
)
--Calculating sifting through rate (percent of previous):
SELECT stages, count, lag(count) OVER(ORDER BY stages),
ROUND(count::dec / lag(count) OVER(ORDER BY stages) * 100, 1) AS sifting_pct
FROM user_flow
ORDER BY 1
```



stages	count	lag	sifting_pct
stage_1_downloads	23608	(NULL)	(NULL)
stage_2_signups	17623	23608	74.6
stage_3_trip_request	12406	17623	70.4
stage_4_trip_completed	6233	12406	50.2

▼ PT1 percent of top

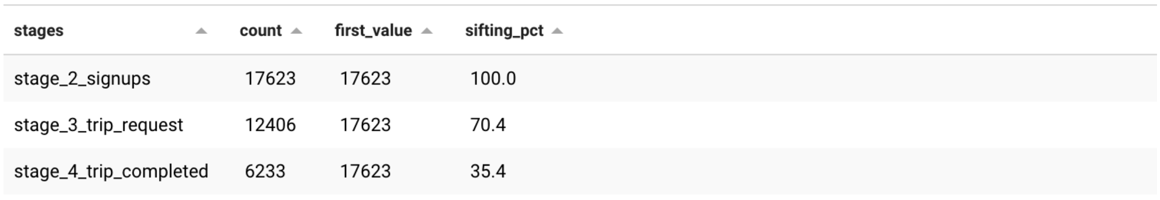
```
-- Constructing user_flow funnel:
With  user_flow AS
(
  SELECT COUNT(app_download_key),
  'stage_1_downloads' AS stages
  FROM app_downloads
 UNION
  SELECT COUNT(user_id),
  'stage_2_signups' AS stages
  FROM signups
 UNION
  SELECT COUNT(DISTINCT user_id),
  'stage_3_trip_request' AS stages
  FROM ride_requests
 UNION
  SELECT COUNT(DISTINCT user_id),
  'stage_4_trip_completed' AS stages
  FROM ride_requests
  WHERE dropoff_ts IS NOT NULL
)
--Calculating sifting through rate (percent of top):
SELECT stages, count, first_value(count) OVER(ORDER BY stages),
ROUND(count::dec / first_value(count) OVER(ORDER BY stages) * 100, 1) AS sifting_pct
FROM user_flow
ORDER BY 1
```



stages	count	first_value	sifting_pct
stage_1_downloads	23608	23608	100.0
stage_2_signups	17623	23608	74.6
stage_3_trip_request	12406	23608	52.5
stage_4_trip_completed	6233	23608	26.4

▼ PT2 percent of top

```
-- Constructing user_flow funnel:
With  user_flow AS
(/*
  SELECT COUNT(app_download_key),
  'stage_1_downloads' AS stages
  FROM app_downloads
 UNION */
  SELECT COUNT(user_id),
  'stage_2_signups' AS stages
  FROM signups
 UNION
  SELECT COUNT(DISTINCT user_id),
  'stage_3_trip_request' AS stages
  FROM ride_requests
 UNION
  SELECT COUNT(DISTINCT user_id),
  'stage_4_trip_completed' AS stages
  FROM ride_requests
  WHERE dropoff_ts IS NOT NULL
)
--Calculating sifting through rate (percent of top):
SELECT stages, count, first_value(count) OVER(ORDER BY stages),
ROUND(count::dec / first_value(count) OVER(ORDER BY stages) * 100, 1) AS sifting_pct
FROM user_flow
ORDER BY 1
```



stages	count	first_value	sifting_pct
stage_2_signups	17623	17623	100.0
stage_3_trip_request	12406	17623	70.4
stage_4_trip_completed	6233	17623	35.4

Tableau links:

https://public.tableau.com/app/profile/rolas/viz/Metrocar_funnel_16914131650960/Metrocar_funnel

https://public.tableau.com/app/profile/rolas/viz/Metrocar_platform/THESTORY