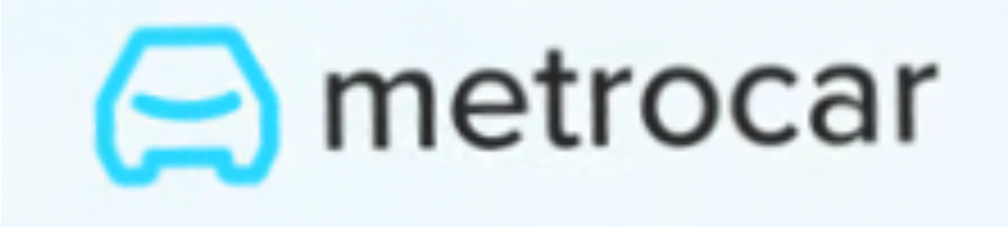

Funnel Analysis:

To identify areas for improvement and optimization

Report by: Mahlet S. Sisay

Data Analyst

Oct 2023

Table of Contents

metrocar

Executive summary

Metrocar is a ride-sharing app that operates similarly to Uber and Lyft, connecting riders with drivers through a mobile application. The main goal of the project is to conduct a funnel analysis to understand drop-off points at various stages of the user journey and identify areas for improvement. Here's a summary of the key findings and recommendations:

User Funnel Drop-off Points:

- *Ride Accepted to Ride Completed:* Approximately 50% of users who had their rides accepted did not complete those rides.
- *Signup to Ride Requested:* Among the 17,623 users who completed the sign-up process, only roughly 70.40% of them proceeded to request a ride. Further analysis, such as A/B testing, is recommended to understand and improve this transition.
- *Download to Signup:* There is an approximate 25% decline in the number of users who download the app compared to those who complete the sign-up process.

Ride Funnel Drop-off Point:

- *Ride Requested to Ride Accepted:* Out of a total of 385,477 ride requests, only around 65% of these requests were accepted by drivers.

Additional Insights:

- *Platform Usage*: IOS users generated majority of ride requests, constituting approximately 60% of all requests.
- *Target Audience*: The primary customer base for Metrocar falls within the age range of 35 to 44, followed by the 25-34 age group.
- *Peak Hour Strategy*: Detailed analysis of ride request distribution throughout the day highlights high demand during specific time slots, such as mornings (8AM - 10AM), afternoons (4PM - 6PM), and early evenings (6PM - 8PM). This suggests the potential effectiveness of implementing a price surging strategy during peak hours.

In conclusion, Metrocar's funnel analysis offers valuable insights into user behavior and ride dynamics, emphasizing critical areas for improvement and optimization. Addressing these drop-off points and catering to peak-hour demand could enhance service efficiency and customer satisfaction.

Introduction

Company Background:

Metrocar is a ride-sharing app that operates similarly to well-known services like Uber and Lyft. It provides transportation services to passengers through a network of drivers. Metrocar's business model is based on a platform that connects riders with drivers through a mobile application. It acts as an intermediary between riders and drivers, providing a user-friendly platform to connect and facilitate the ride-hailing process.

Metrocar customers can access the ride-sharing services through both mobile apps for Android and iOS platforms as well as through the company's website.

Figure 1 metrocar mobile application home screen

Project background

Aim:

The main goal of this project is to conduct a funnel analysis by understanding the drop off at every stage of the funnel and address key business questions to identify areas for improvement and optimization.

Funnel Analysis

Funnel analysis is a method in data analysis used to track and understand the sequential steps or stages that users or customers go through when interacting with a product, service, or website. It's called a "funnel" because the shape of the analysis resembles that of a real-world funnel. However, it could also be easily represented by a bar chart.

Funnel analysis allows businesses and organizations identify where users drop off or convert, helping them to ultimately increase desired outcomes, such as sales, sign-ups, or conversions. It is widely used in e-commerce, marketing, and product development to drive growth and revenue.

Metrocar's funnel:

The customer funnel for Metrocar typically includes the following stages:

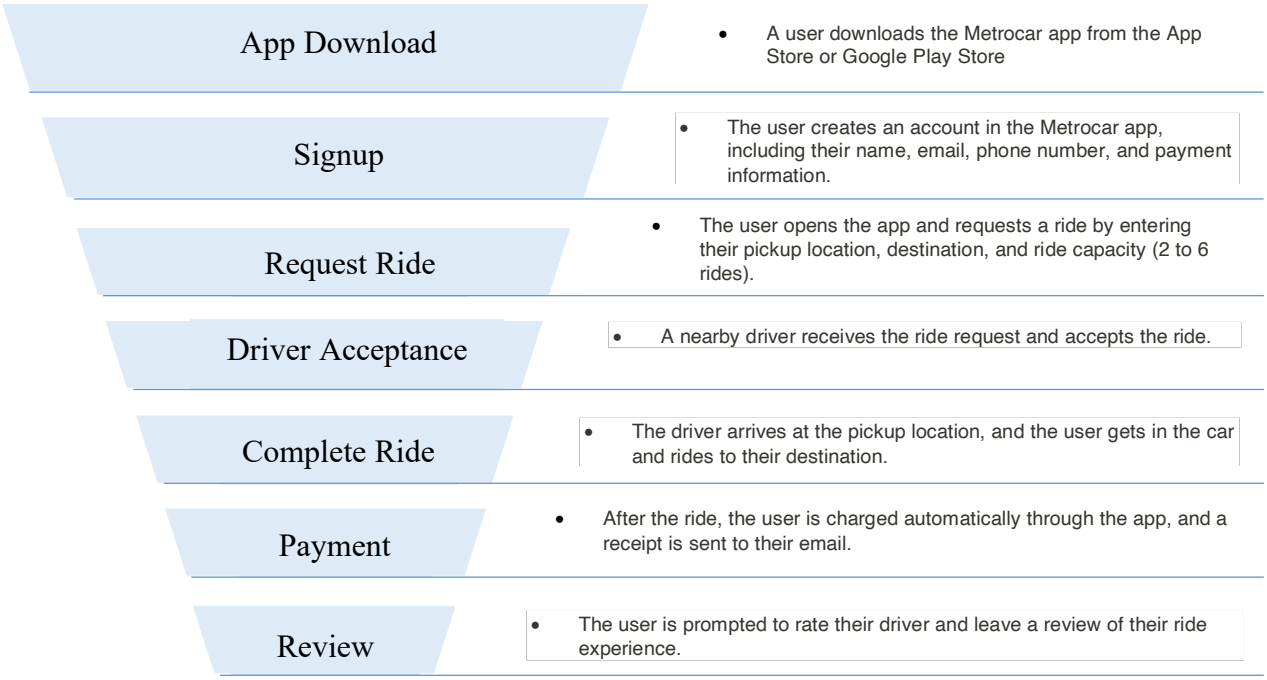

Figure 2 metrocar's customer funnel stages

Key Business questions:

The funnel analysis and recommendations were made based on the following key business questions:

1. What steps of the funnel should we research and improve? Are there any specific drop-off points preventing users from completing their first ride?
2. What part of our funnel has the lowest conversion rate? What can we do to improve this part of the funnel?
3. 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?
4. What age groups perform best at each stage of our funnel? Which age group(s) likely contain our target customers?
5. Surge pricing is the practice of increasing the price of goods or services when there is the greatest demand for them. If we want to adopt a price-surging strategy, what does the distribution of ride requests look like throughout the day?

Funnel Analysis result:

Funnel improvement analysis:

To understand which step of the funnel to improve, we first need to examine and study the specific drop-off points along the funnel. The funnel was constructed both in user-level granularity and ride-level granularity.

1. User level Granularity: The total number of users in each funnel is stated below.

Funnel Order	Funnel Step	User Count	Percent of Previous	Percent of Top
1	Download app	23608	-	100.00%
2	User signup	17623	74.65%	74.65%
3	Request ride	12406	70.40%	52.55%
4	Rides accepted	12278	98.97%	52.01%
5	Rides completed	6233	50.77%	26.40%
6	Payment	6233	100.00%	26.40%
7	Reviews	4348	69.76%	18.42%

(**Refer to: Appendix B, 1**)

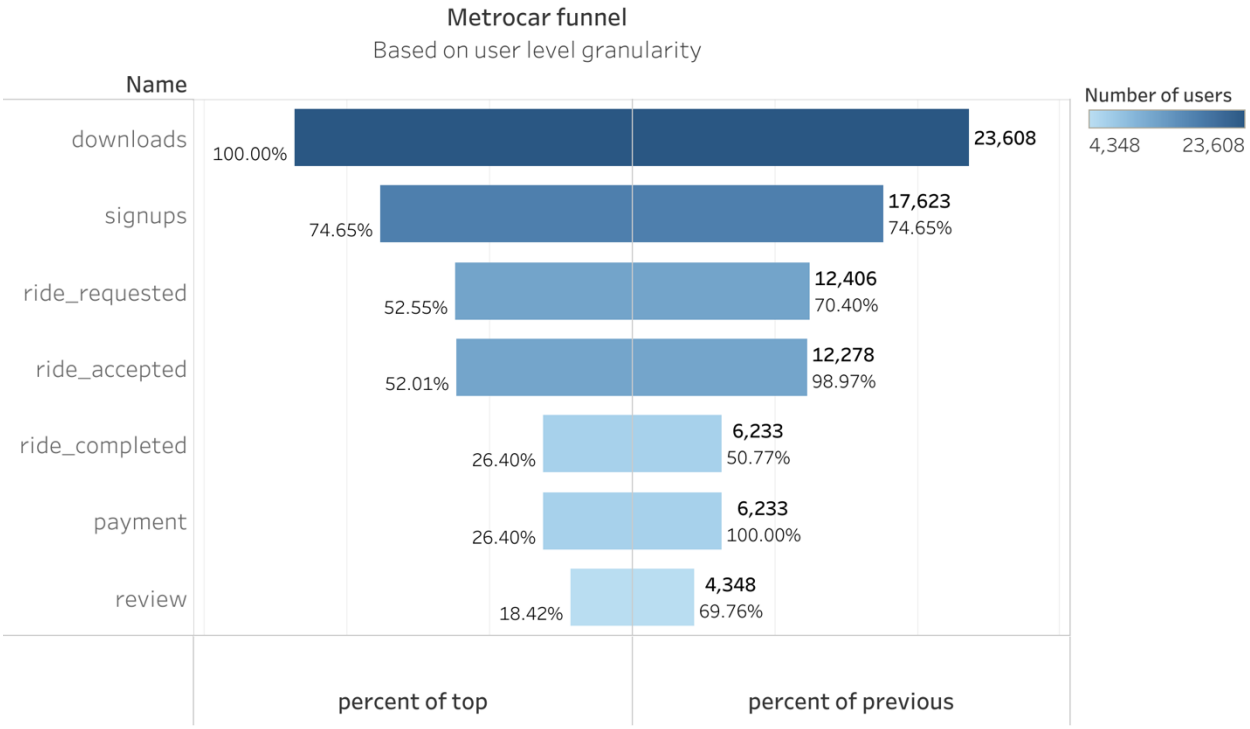

Figure 3 metrocar funnel calculated using number of users.

There are three critical user drop-off points along the funnel.

1. **From Ride Accepted to Ride Completed:**
    - About 50% of users who had their rides accepted didn't complete those rides. It's a major drop-off that needs attention.
2. **Signup to Ride Requested**:
    - Of 17,623 users who signed up, only around 70.40% of users went on to request a ride. This is a significant drop-off, and further analysis, such as A/B testing, is recommended to understand and improve this transition.
3. **Download to Signup**:
    - A drop of around 25% occurred from users downloading the app to signing up. To enhance the conversion rate from app download to signup, further analysis, like A/B testing should be considered.

Ride level granularity: The total number of rides in each funnel is stated below.

Funnel Order	Funnel Step	Ride Count	Percent of Previous	Percent of Top
1	Download app	Null	-	-
2	User signup	Null	-	-
3	Request ride	385477	-	100.00%
4	Rides accepted	248379	64.43%	64.43%
5	Rides completed	223652	90.04%	58.02%
6	Payment	212628	95.07%	55.16%
7	Reviews	156211	73.47%	40.52%

(**Refer to Appendix B, 2**)

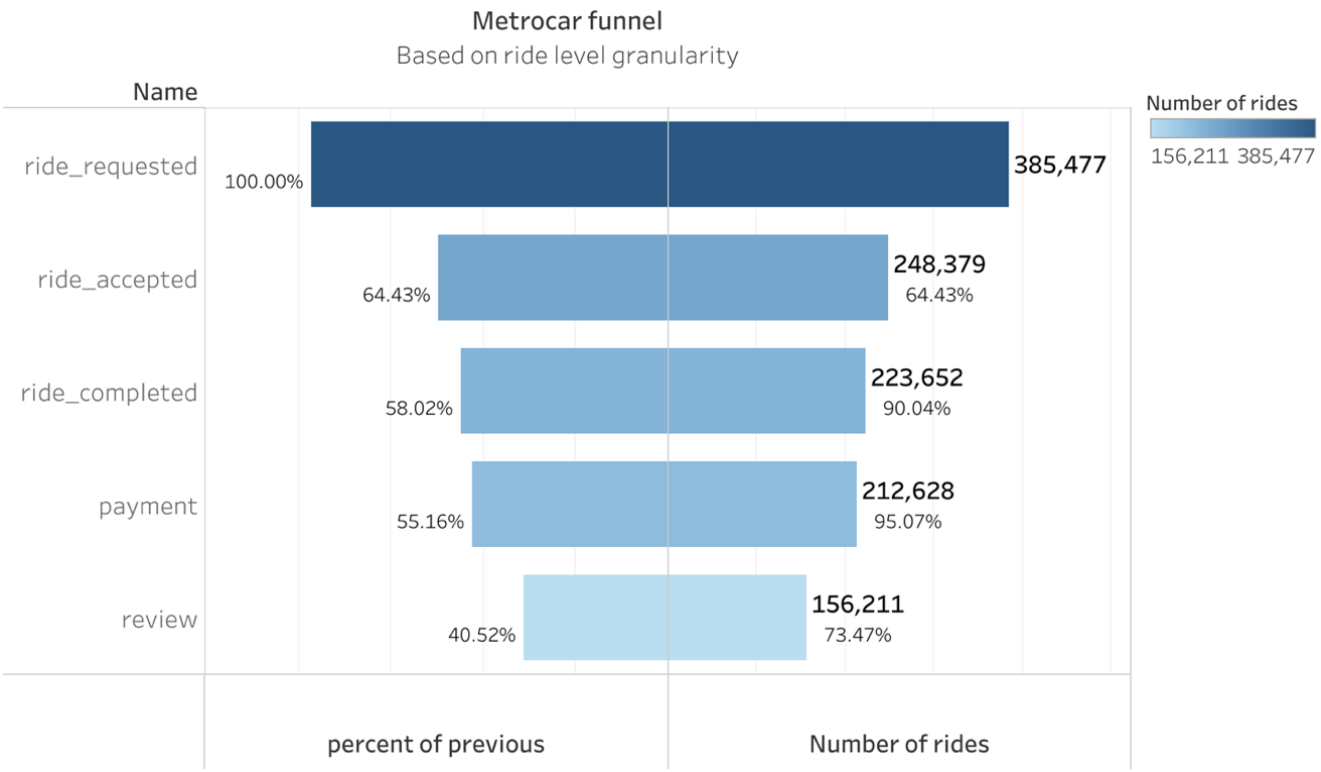

Figure 4 metrocar funnel calculated using the number of ride requests.

When considering the number of rides requested, only one major drop-off point occurs.

1. **From ride requested to ride accepted.**
    - From 385,477 rides requested, only around 65% of rides were accepted. This is a significant drop-off, and further analysis and investigation on the driver side of the application are needed.

Conversion rate investigation:

By using the funnel result from user level granularity and percent of previous approach, the lowest conversion rate, approximately 50%, occurs from ride accepted to ride completed.

Improving the conversion rate from "ride accepted" to "ride completed" is crucial for increasing the efficiency of metrocar's service. Here are some strategies and recommendations to help improve this part of the funnel:

- **Incentivize drivers:** Offering incentives to drivers who consistently complete rides, such as bonuses or rewards for high completion rates can motivate drivers to accept complete rides.
- **Enhance Rider Communication:** Improve communication between drivers and riders. Consider implementing in-app messaging or call features to make it easier for riders to communicate with drivers.
- **Reduce Cancellations:** Implement policies or fees to discourage ride cancellations by both riders and drivers.

Platform-based marketing insight:

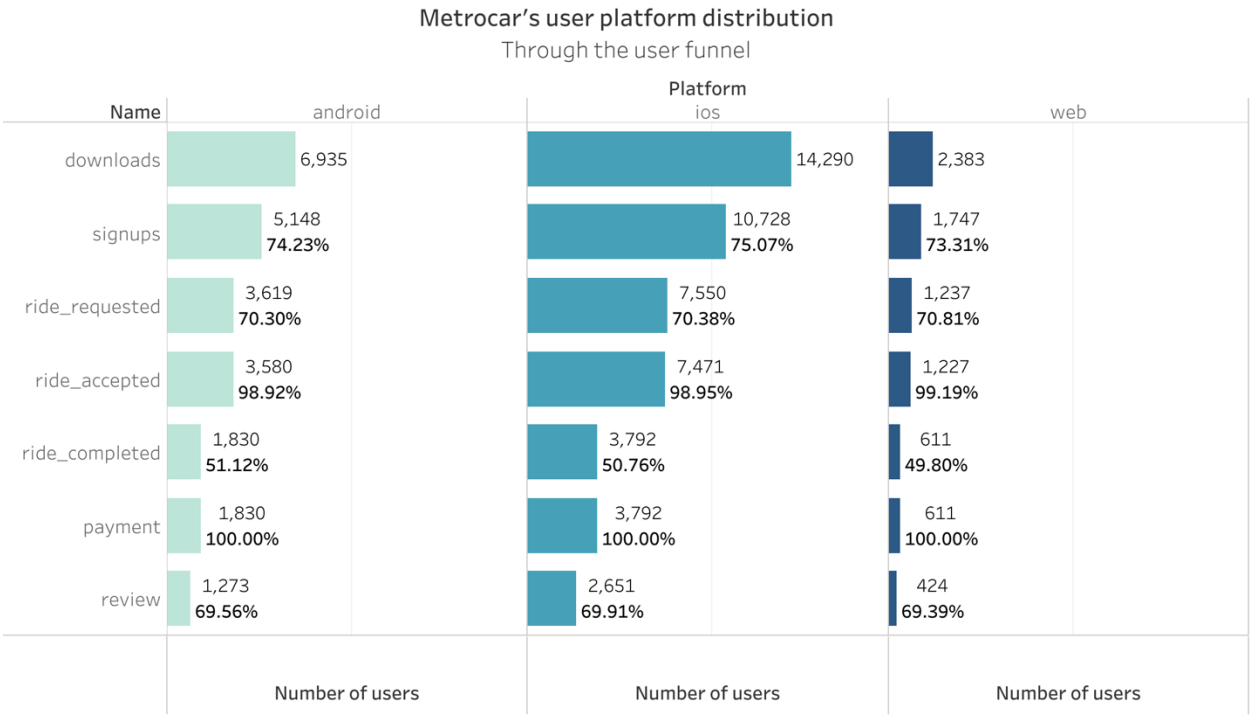

Figure 5 ride request distribution based on platforms used.

- The iOS platform generated the highest number of ride requests, which, in turn, led to the highest purchase amount, accounting for approximately 60% of all ride requests.

Age group performance insights:

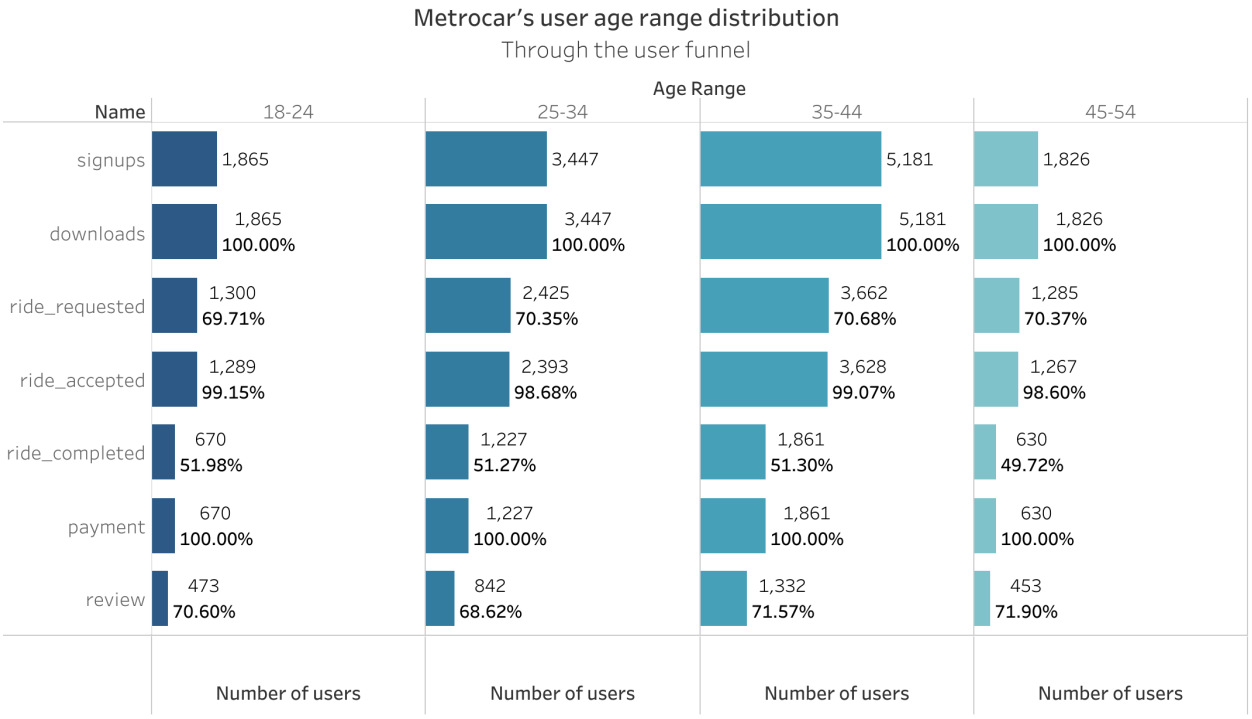

Figure 6 users age range distribution through the user funnel.

- Null and unknown values have been omitted from the visualization to provide more precise insight into the age groups targeted.
- According to the findings, Metrocar's primary target customers are individuals aged 35 to 44, followed by those in the 25-34 age group.

Surge pricing strategy:

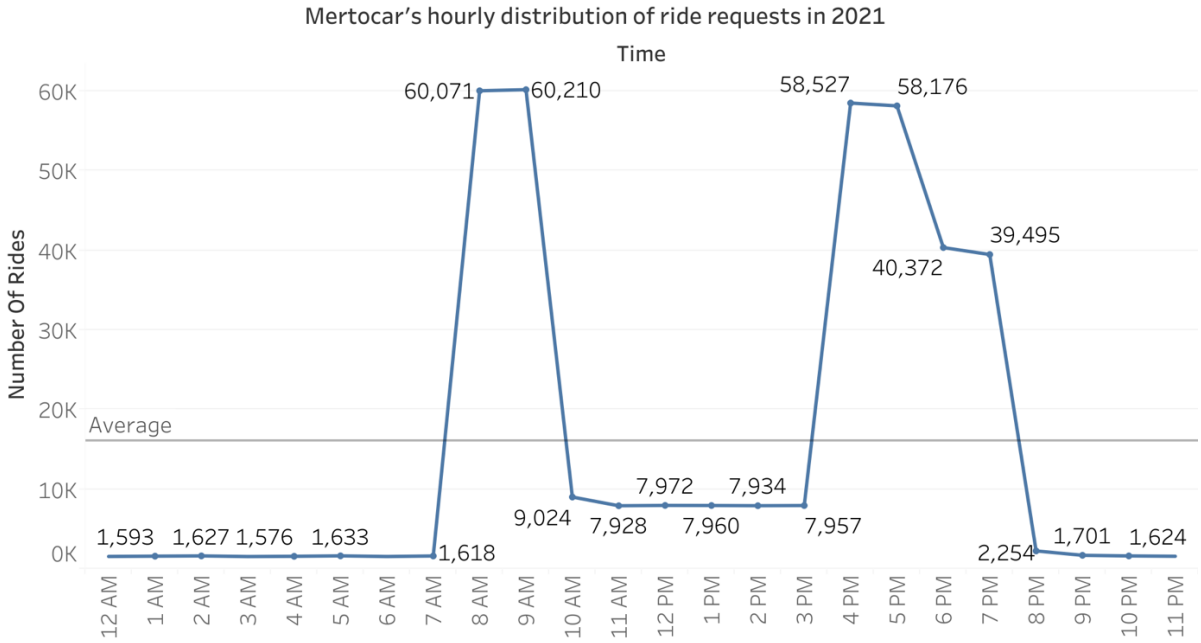

Figure 7 Hourly distribution of ride requests

To adopt a price surging strategy during peak hours, important to investigate the distribution of ride requests throughout the day. Based on the above chart, the distribution can be described as follows:

1. **Morning (8AM – 10AM):**
   - Very high demand during these hours. This is likely due to people commuting to work or other daytime activities.
2. **Afternoon (4 PM – 6 PM):**
   - There is very high demand during these hours as well. This is often referred to as the evening rush hour when people are finishing work and looking for rides home.
3. **Early Evening (6 PM – 8 PM):**
   - High demand during this period, although it's slightly lower than the afternoon peak. Some people are still commuting home, while others may be going out for the evening.

To implement surge pricing effectively, Metrocar can typically increase the prices during this very high demand (morning and afternoon peak hours). The goal is to encourage more drivers to be available during these periods and to help balance the supply and demand for rides.

Appendix

Appendix A:

1. Data source:

Metrocar stores its data in a relational database and the data can be accessed through Beekeeper studio. Database link: -
`postgres://Test:bQNxVzJL4g6u@ep-noisy-flower-846766-pooler.us-east-2.aws.neon.tech/Metrocar`

2. Database schema

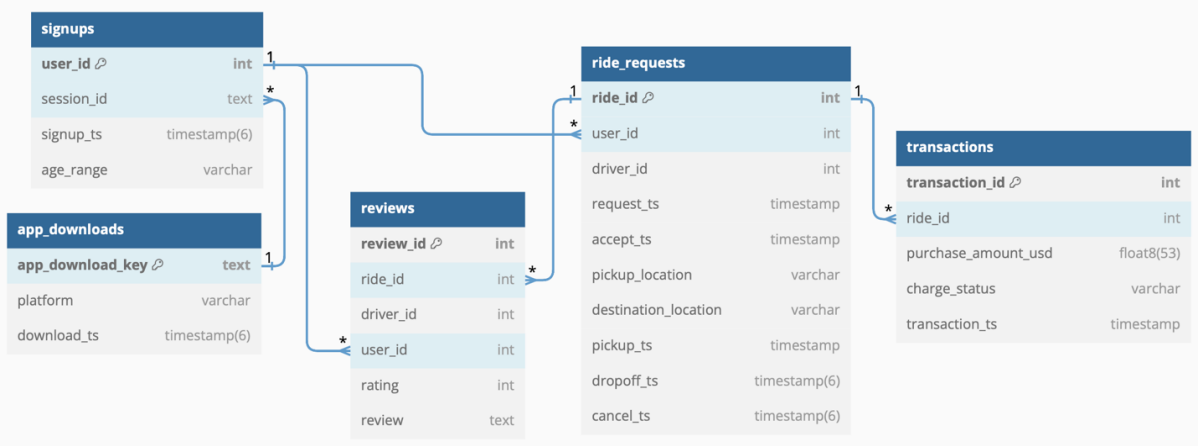

Figure 8 Database schema: visualized using dbdiagram

Appendix B: SQL query results

1. Funnel analysis using user-level granularity.

```
WITH users AS (
  SELECT
      1 funnel_step, 'App download' funnel_name, COUNT(*) user_count
  FROM app_downloads
UNION ALL
  SELECT
      2 funnel_step, 'user signup' funnel_name, COUNT(*) user_count
  FROM signups
UNION ALL
  SELECT
      3 funnel_step, 'Ride requested' funnel_name, COUNT(DISTINCT
user_id) user_count
  FROM ride_requests
UNION ALL
  SELECT
    4 funnel_step, 'Ride accepted' funnel_name, COUNT(DISTINCT
user_id) user_count
FROM ride_requests WHERE accept_ts IS NOT NULL
UNION ALL
  SELECT
```

```
      5 funnel_step, 'Ride completed' funnel_name, COUNT(DISTINCT
user_id) user_count
FROM ride_requests
WHERE dropoff_ts IS NOT NULL
UNION ALL
  SELECT
           6 funnel_step, 'Payment' funnel_name, COUNT(DISTINCT
user_id) user_count
FROM ride_requests
JOIN transactions USING (ride_id)
WHERE transactions.charge_status ='Approved'
UNION ALL
  SELECT
    7 funnel_step, 'Reviews' as funnel_name, COUNT (DISTINCT user_id)
user_count
FROM reviews )
SELECT funnel_step, funnel_name, user_count,
       --lag(user_count,1)over(order by funnel_step) as previous_num,
       ROUND((user_count::numeric/lag(user_count, 1) OVER (ORDER BY
funnel_step))*100,2) percent_of_previous,
       ROUND(user_count::NUMERIC / FIRST_VALUE(user_count) OVER (ORDER
BY funnel_step)*100,2) AS percent_of_top
FROM users ORDER BY 1;
```

2. Funnel analysis using ride-level granularity

```
WITH users AS (
  /*SELECT
      1 funnel_step, 'App download' funnel_name, NULL::INTEGER
ride_count
  FROM app_downloads
UNION distinct
  SELECT
      2 funnel_step, 'user signup' funnel_name, NULL::INTEGER
ride_count
  FROM signups
UNION distinct*/
  SELECT
      3 funnel_step, 'Ride requested' funnel_name, COUNT(ride_id)
ride_count
  FROM ride_requests
UNION ALL
  SELECT
    4 funnel_step, 'Ride accepted' funnel_name, COUNT(ride_id)
ride_count
FROM ride_requests WHERE accept_ts IS NOT NULL
UNION ALL
  SELECT
      5 funnel_step, 'Ride completed' funnel_name, COUNT(ride_id)
ride_count
FROM ride_requests
```

```
WHERE dropoff_ts IS NOT NULL
UNION ALL
  SELECT
           6 funnel_step, 'Payment' funnel_name, COUNT(ride_id)
ride_count
FROM ride_requests
INNER JOIN transactions USING (ride_id)
WHERE transactions.charge_status ='Approved'
UNION ALL
  SELECT
    7 funnel_step, 'Reviews' as funnel_name, COUNT (ride_id)
ride_count
FROM reviews )
SELECT *,
       --funnel_step, funnel_name, ride_count,
       --lag(ride_count,1)over(order by funnel_step) as previous_num,
       ROUND((ride_count::numeric/lag(ride_count, 1) OVER (ORDER BY
funnel_step))*100,2) percent_of_previous,
       ROUND(ride_count::NUMERIC / FIRST_VALUE(ride_count) OVER (ORDER
BY funnel_step)*100,2) AS percent_of_top
FROM users ORDER BY 1;
```

3. SQL query for funnel summary

```
      WITH users AS (
  SELECT
      app_download_key,
      user_id,
      platform,
  age_range,
date (download_ts) as download_dt
  FROM app_downloads
  LEFT JOIN signups
  on app_downloads.app_download_key = signups.session_id),
  downloads as
  (SELECT 0 as step,
    'downloads' as name,
    platform,
    age_range,
    download_dt,
    COUNT(DISTINCT app_download_key) as users_count,
    0 as ride_counts
    FROM users
    GROUP BY platform, age_range, download_dt),
    signup as
    (SELECT 1 as step,
      'signups' as name,
      users.platform,
      users.age_range,
      users.download_dt,
```

```
 COUNT(DISTINCT user_id) as users_count,
 0 as ride_counts
 FROM signups
 JOIN users USING (user_id)
 WHERE signup_ts IS NOT NULL
 GROUP BY users.platform, users.age_range, users.download_dt),
 requested as
 (SELECT 2 as step,
   'ride_requested' as name,
 users.platform,
 users.age_range,
 users.download_dt,
 COUNT(DISTINCT user_id) as users_count,
 COUNT(DISTINCT ride_id) as ride_counts
 FROM ride_requests
 JOIN users using (user_id)
 WHERE request_ts IS NOT NULL
 GROUP BY users.platform, users.age_range, users.download_dt),
   accepted as
   (SELECT 3 as step,
     'ride_accepted' as name,
 users.platform,
 users.age_range,
 users.download_dt,
 COUNT(DISTINCT user_id) as users_count,
COUNT(DISTINCT ride_id) as ride_counts
 FROM ride_requests
 JOIN users using (user_id)
 WHERE accept_ts IS NOT NULL
 GROUP BY users.platform, users.age_range, users.download_dt),
   completed as
   (SELECT 4 as step,
     'ride_completed' as name,
 users.platform,
 users.age_range,
 users.download_dt,
 COUNT(DISTINCT user_id) as users_count,
 COUNT(DISTINCT ride_id) as ride_counts
 FROM ride_requests
 JOIN users using (user_id)
 WHERE cancel_ts IS NULL
 GROUP BY users.platform, users.age_range, users.download_dt),
     payment as
   (select 5 as step,
     'payment' as name,
 users.platform,
 users.age_range,
 users.download_dt,
COUNT(DISTINCT user_id) as users_count,
COUNT(DISTINCT ride_id) as ride_counts
FROM transactions
```

```
      JOIN ride_requests using (ride_id)
      JOIN users using (user_id)
      WHERE charge_status = 'Approved'
      GROUP BY users.platform, users.age_range, users.download_dt),
           review as
         (SELECT 6 as step,
           'review' as name,
       users.platform,
       users.age_range,
       users.download_dt,
       COUNT(DISTINCT user_id) as users_count,
       COUNT(DISTINCT ride_id) as ride_counts
       FROM reviews
       JOIN users using (user_id)
       GROUP BY users.platform, users.age_range, users.download_dt
         )
SELECT step, name, platform, age_range, download_dt,
       users_count,
       ride_counts
       FROM (
  SELECT * FROM downloads
  UNION ALL
  SELECT * FROM signup
  UNION ALL
  SELECT * FROM requested
  UNION ALL
  SELECT * FROM accepted
  UNION ALL
  SELECT * FROM completed
  UNION ALL
  SELECT * FROM payment
  UNION ALL
  SELECT * FROM review
) AS cd
ORDER BY step,name, platform, age_range, download_dt,
       users_count,
       ride_counts;
```

4. Query for hourly distribution of ride requests

```
SELECT
  count(ride_id) as number_of_rides,
  extract(HOUR from request_ts) as hours,
  CASE
    WHEN EXTRACT(HOUR FROM request_ts) = 0 THEN '12 AM'
    WHEN EXTRACT(HOUR FROM request_ts) < 12 THEN CONCAT(
         EXTRACT(HOUR FROM request_ts), ' AM')
    WHEN EXTRACT(HOUR FROM request_ts) = 12 THEN '12 PM'
    ELSE CONCAT(EXTRACT(HOUR FROM request_ts) - 12,' PM')
  END AS pick_hour_12_hr
FROM
```

```
  ride_requests
WHERE
  request_ts IS NOT null
GROUP BY
  pick_hour_12_hr,
  hours
ORDER BY
  hours;
```

Appendix C: Tableau visualization results

1. User level funnel
   https://public.tableau.com/authoring/metrocar_userlevelgranularity/Sheet1#1
2. Ride level funnel
   https://public.tableau.com/authoring/Metrocarfunnel_ridelevelgranularity/Sheet12#1
3. Platform
   https://public.tableau.com/authoring/metrocarsfunnelsummary/Agegroupbasedinsight2/Platform%20based%20insight#1
4. Age range
   https://public.tableau.com/authoring/metrocarsfunnelsummary/Agegroupbasedinsight2/Age%20group%20based%20insight#1
5. Hourly distribution
   https://public.tableau.com/authoring/metrocarsfunnelsummary/Hourlyriderequestdistribution#1
6. Summary
   https://public.tableau.com/authoring/metrocarsfunnelsummary/Metrocarsfunnelsummary#3

Appendix D: Loom Presentation

https://www.loom.com/share/79a37b885fd748afa663171fc9b74d03?sid=24856dba-cac7-4b5b-9758-4b844ce4290d