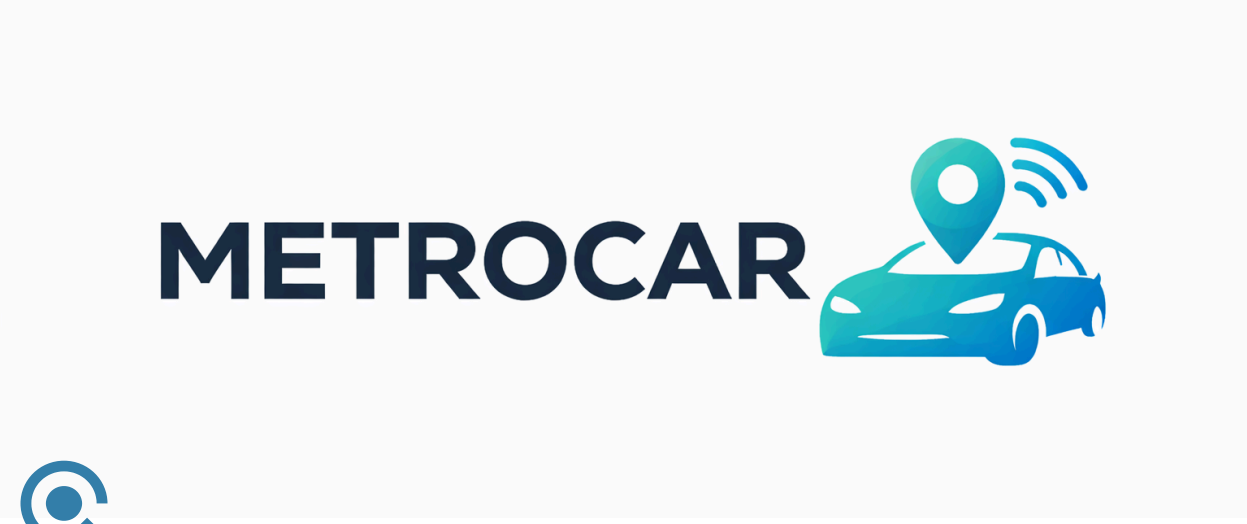


Metrocar — User Journey & Revenue Analysis

Team	
Status	Done

🟦 Executive Summary

17.6K users generated **$4.25M in total revenue** with an average **Revenue per User (ARPU) of $684**.
The analysis shows that the primary business bottleneck is **ride completion**, not user acquisition.

The largest drop-off occurs **after ride acceptance**, indicating operational rather than marketing issues.
Improving ride completion rate during peak hours represents the **highest-impact opportunity for revenue growth** without additional acquisition spend.

📊 **Key KPIs**

- **Users:** 17.6K
- **Total Revenue:** $4.25M
- **Successful Trips:** 223K

- **Revenue per User:** $684
- **Average Rating:** 3.1

🟦 Data & Objective

🎯 **Objective**

Understand user behavior across the full journey — from app download to completed and paid ride — and identify the main drivers and blockers of revenue growth.

▼ 📂 **Data sources**

- App downloads
- User signups
- Ride requests, acceptance, completion
- Transactions and payments
- User ratings and reviews

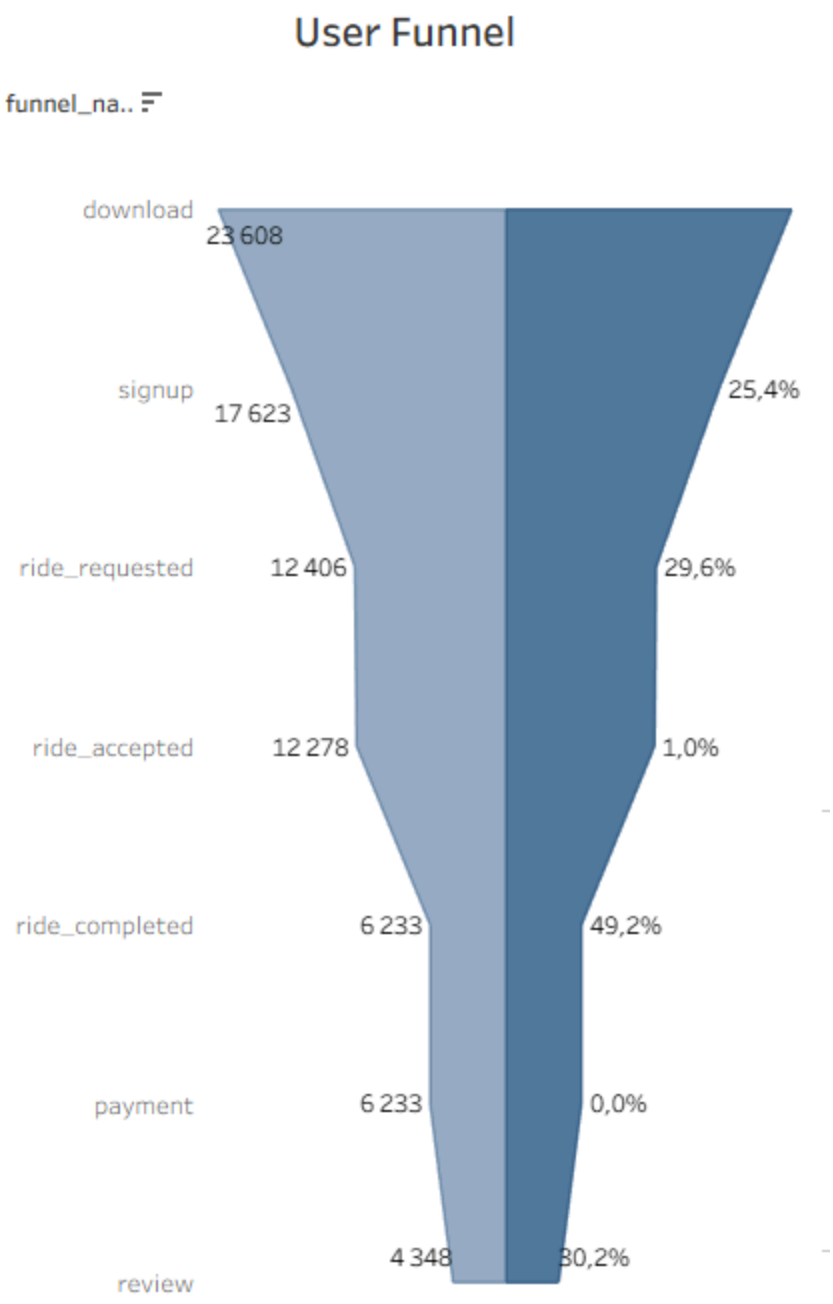
🟦 User Funnel Analysis

What we see

- ~25% drop from **download → signup**
- **~49% drop from ride accepted → ride completed** (largest loss)
- ~70% of completed rides receive a review

Why it matters

User acquisition performs reasonably well, while **operational execution after ride acceptance** causes the largest revenue leakage.



🟦 Rides & Time Patterns

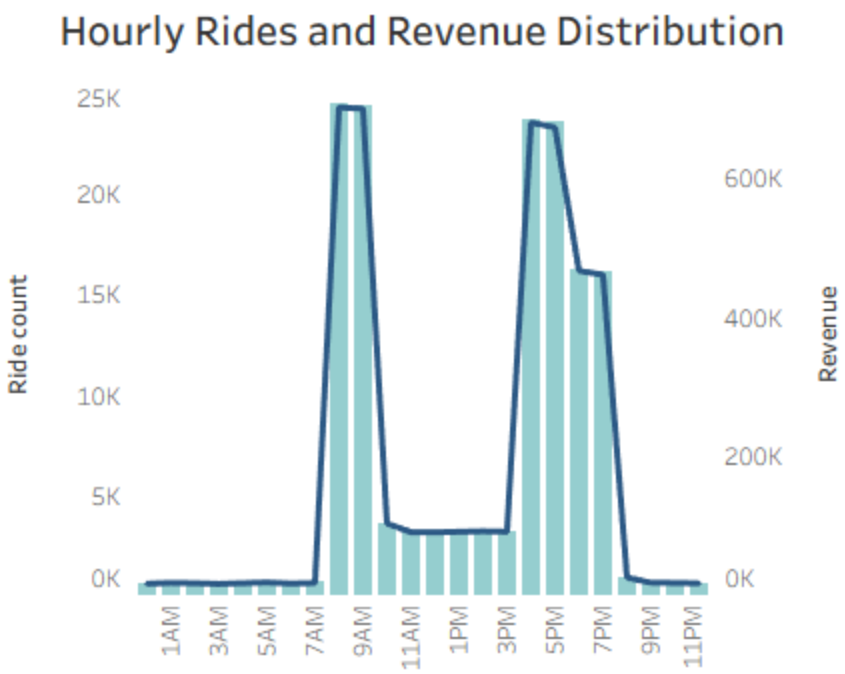
What we see

- Clear usage peaks:
  - **Morning commute (8–10 AM)**
  - **Evening commute (5–8 PM)**

• Revenue closely follows ride volume

Why it matters

Metrocar usage is strongly **commute-driven**, making operational reliability during peak hours critical.



🟦 Platforms & Age Groups

What we see

• **iOS users** generate higher revenue per user

• **Android users** drive volume but lower ARPU

• Most valuable age segment: **25–44**

Why it matters

Revenue optimization should prioritize **high-value segments**, not just scale.

🟦 Key Business Answers

Where do we lose the most users?

→ Between **ride accepted and ride completed**

When are users most active?

→ Morning and evening commute hours

Which segments generate the most value?

→ iOS users, age 25–44

How effective is monetization?

→ ARPU is high; revenue growth depends on improving completion rate

🟦 Recommendations

✅ **Short-term (high impact)**

- Investigate ride cancellations after acceptance
- Improve driver availability during peak hours

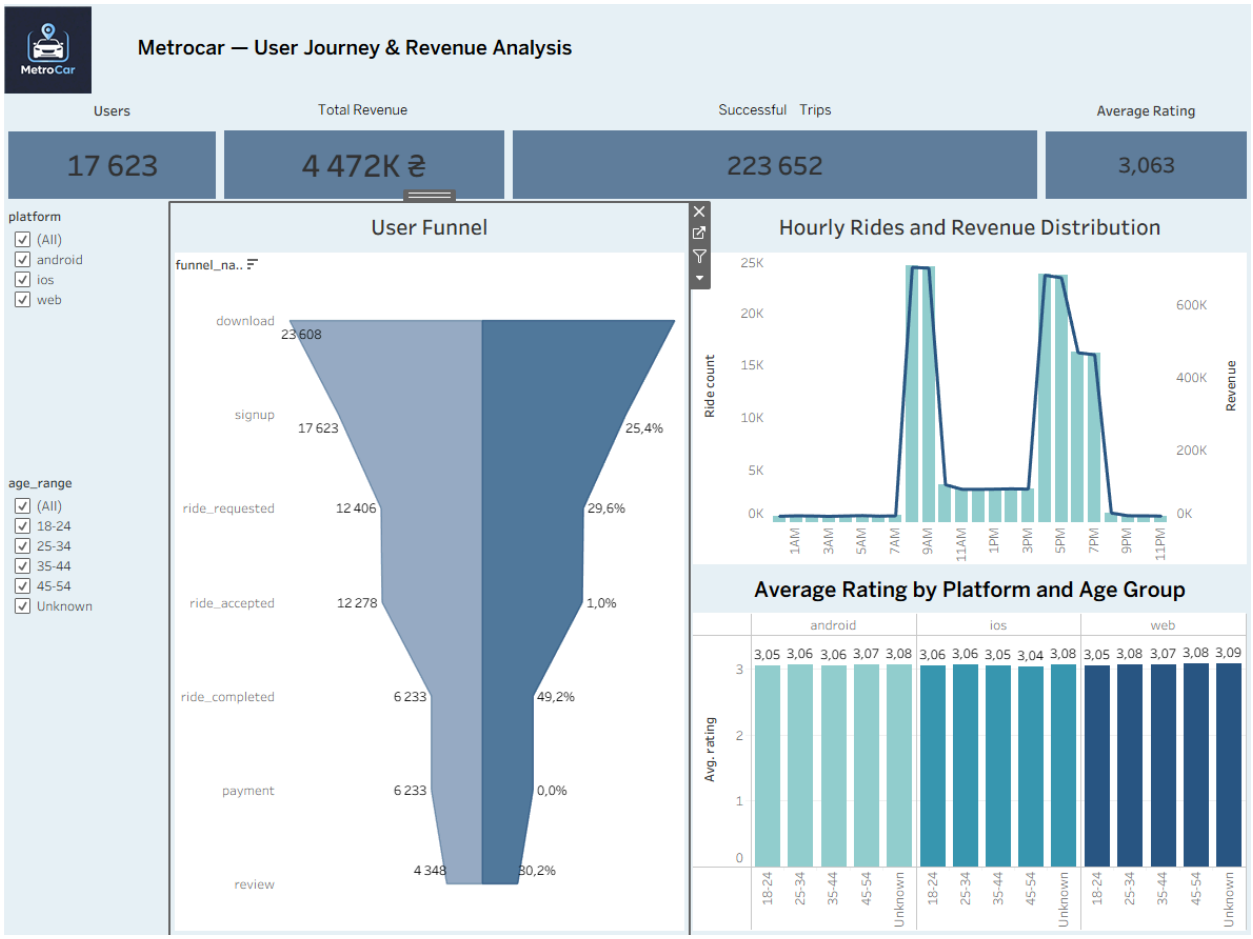
📈 **Mid-term**

- Prioritize iOS users for premium features
- Run product experiments for the 25–44 age group

🎯 **KPIs to monitor**

- Ride completion rate
- Cancellation rate after acceptance
- ARPU by platform
- Peak-hour completion performance

🟦 Dashboard Preview



🟦 Appendix (Optional)

▼ Technical Appendix (SQL & Calculations)

```
SELECT
    s.age_range,
    COUNT(DISTINCT s.user_id) AS users,
    COUNT(DISTINCT rr.ride_id) AS completed_rides
FROM signups s
LEFT JOIN ride_requests rr
    ON s.user_id = rr.user_id
    AND rr.dropoff_ts IS NOT NULL
GROUP BY s.age_range
ORDER BY completed_rides DESC;
```

🔍 Explanation

- **signups** — source of user demographic data
- **ride_requests** — source of ride lifecycle data
- `dropoff_ts IS NOT NULL` → filters only **successfully completed rides**
- `LEFT JOIN` ensures users without completed rides are still included

📌 Used to analyze **ride completion distribution across age groups**.

```
SELECT
    ad.platform,
    COUNT(DISTINCT s.user_id) AS users,
    COUNT(DISTINCT rr.ride_id) AS rides,
    COUNT(DISTINCT CASE
        WHEN rr.dropoff_ts IS NOT NULL THEN rr.ride_id
    END) AS completed_rides
FROM app_downloads ad
LEFT JOIN signups s
    ON ad.app_download_key = s.session_id
LEFT JOIN ride_requests rr
    ON s.user_id = rr.user_id
GROUP BY ad.platform;
```

🔍 Explanation

- Links **app downloads → signups → rides** to attribute ride activity to the originating platform
- Completed rides are identified using `dropoff_ts IS NOT NULL`

📌 Used to compare **platform-level performance and conversion to completed rides**.

```
SELECT
    EXTRACT(HOUR FROM request_ts) AS hour_of_day,
    COUNT(*) AS total_requests,
    COUNT(accept_ts) AS accepted_rides
```

```
FROM ride_requests
GROUP BY hour_of_day
ORDER BY hour_of_day;
```

🔍 Explanation

- `EXTRACT(HOUR FROM request_ts)` groups rides by hour (0–23)
- `COUNT(*)` counts all ride requests
- `COUNT(accept_ts)` counts only **accepted rides** (non-null values)

📌 Used to identify **peak demand hours** and **acceptance efficiency by time of day**.

```
SELECT
    COUNT(DISTINCT rr.user_id) AS requested,
    COUNT(DISTINCT CASE
        WHEN rr.accept_ts IS NOT NULL THEN rr.user_id
    END) AS accepted,
    COUNT(DISTINCT CASE
        WHEN rr.dropoff_ts IS NOT NULL THEN rr.user_id
    END) AS completed
FROM ride_requests rr;
```

🔍 Explanation

- Counts **unique users** at each key ride stage
- `requested` — users who requested at least one ride
- `accepted` — users with at least one accepted ride
- `completed` — users with at least one completed ride

📌 Used to evaluate **user conversion rates across the ride funnel**.

```
SELECT
    COUNT(DISTINCT rr.user_id) AS requested,
    COUNT(DISTINCT CASE
```

```
    WHEN rr.accept_ts IS NOT NULL THEN rr.user_id
  END) AS accepted,
  COUNT(DISTINCT CASE
    WHEN rr.dropoff_ts IS NOT NULL THEN rr.user_id
  END) AS completed
FROM ride_requests rr;
```

🔍 Explanation

- `requested` — users who requested at least one ride
- `accepted` — users with at least one accepted ride
- `completed` — users with at least one completed ride

📌 Used to evaluate **user drop-offs across the ride funnel** and identify the most critical conversion loss stage.

Documents