```sql
select funnel_step, funnel_name, platform, age_range, number_of_users, number_of_rides,
        case when funnel_step = 1 then 'downloaded'
        when funnel_step = 2 then 'signed up'
        when funnel_step = 3 then 'ride requested'
        when funnel_step = 4 then 'ride accepted'
        when funnel_step = 5 then 'ride completed'
        when funnel_step = 6 then 'ride paid'
        else 'ride reviewed'
        end as Funnel_step_name
from funnel_analysis fa
```

```sql
with
downl_sign_review as
(select
        s.user_id,
        ad.download_ts,
        ad.download_ts::date as download_date,
        extract(hour from ad.download_ts) as download_hour,
        s.signup_ts,
        s.signup_ts::date as signup_date,
        extract(hour from s.signup_ts) as signup_hour,
        ad.platform,
        s.age_range,
        Count (rr.ride_id) as quantity_of_rides
        from app_downloads as ad
        left join signups as s on ad.app_download_key = s.session_id
        left join ride_requests as rr on s.user_id = rr.user_id
        group by s.user_id, ad.download_ts, s.signup_ts, ad.platform, s.age_range),
first_request as
(select distinct
        user_id,
        FIRST_VALUE (request_ts)
                OVER (PARTITION BY user_id ORDER BY request_ts)
                AS first_requested_ride_ts
                from ride_requests rr)
select
        dsr.user_id,
        dsr.download_ts,
        dsr.download_date,
        dsr.download_hour,
        dsr.signup_ts,
        dsr.signup_date,
        dsr.signup_hour,
        dsr.platform,
        dsr.age_range,
        dsr.quantity_of_rides,
        fr.first_requested_ride_ts,
        first_requested_ride_ts::date as first_requested_ride_date,
        case when dsr.signup_date is null then 'not_signed_up'
        else 'signed_up'
        end as map_sign_up,
        case when fr.first_requested_ride_ts is null then 'not_requested'
        else 'requested'
        end as map_requested
from downl_sign_review as dsr left join first_request as fr
on dsr.user_id=fr.user_id
```

```sql
SELECT
rr.ride_id,
rr.user_id,
rr.driver_id,
rr.request_ts,
rr.request_ts::date as request_date,
extract(hour from rr.request_ts) as request_hour,
rr.accept_ts,
rr.accept_ts::date as accept_date,
extract(hour from rr.accept_ts) as accept_hour,
rr.pickup_ts,
rr.pickup_ts::date as pickup_date,
extract(hour from rr.pickup_ts) as pickup_hour,
rr.dropoff_ts,
rr.dropoff_ts::date as dropoff_date,
extract(hour from rr.dropoff_ts) as dropoff_hour,
rr.cancel_ts,
rr.cancel_ts::date as cancel_date,
extract(hour from rr.cancel_ts) as cancel_hour,
t.purchase_amount_usd,
t.charge_status,
s.age_range,
ad.platform,
case when rr.request_ts is null then 'not_requested'
        else 'requested'
        end as map_requested2,
case when rr.accept_ts is null then 'not_accepted'
        else 'accepted'
        end as map_accepted,
case when rr.pickup_ts is null then 'not_picked_up'
        else 'picked_up'
        end as map_picked_up,
case when rr.dropoff_ts is null then 'not_dropped_off'
        else 'dropped_off'
        end as map_dropped_off,
case when rr.cancel_ts is null then 'not_cancelled'
        else 'cancelled'
        end as map_cancelled
FROM ride_requests as rr
left join reviews as r on rr.ride_id = r.ride_id
left join transactions as t on t.ride_id =rr.ride_id
left join signups as s on s.user_id=rr.user_id
left join app_downloads as ad on ad.app_download_key = s.session_id
```

```sql
with drivers_rating as(
        select distinct driver_id,
        round(AVG(rating),2) as average_rating
        from reviews
        group by driver_id)
select
        case when average_rating >0 and average_rating <= 1 then '>0 and <=1'
        when average_rating >1 and average_rating <= 2 then '>1 and <=2'
        when average_rating >2 and average_rating <= 3 then '>2 and <=3'
        when average_rating >3 and average_rating <= 4 then '>3 and <=4'
        when average_rating >4 and average_rating <= 5 then '>4 and <=5'
else 'Unknown'
end as average_rating_buckets_drivers,
count (distinct driver_id)
from drivers_rating
group by average_rating_buckets_drivers
```

```sql
with rides_rating as(
        select ride_id, rating
        from reviews)
select
        case when rating =0 then '0'
        when rating =1 then '1'
        when rating =2 then '2'
        when rating =3 then '3'
        when rating =4 then '4'
        when rating =5 then '5'
else 'Unknown'
end as rating_buckets_rides,
count (distinct ride_id) as quantity_of_rides
from rides_rating
group by rating_buckets_rides
```