99 Data Analytics Challenge

Ewerton Silva - www.linkedin.com/in/ewertonsilva

Technical

1. What is the average trip cost of holidays? How does it compare to non-holidays?

```
SELECT
  D.holiday
  ,AVG(trip_fare) AS average_trip_cost
FROM
(
SELECT
  DATE(T.call_time) AS DATE_TRIP
  ,T.trip_fare
  ,C.holiday
FROM trip T
LEFT JOIN calendar C ON DATE(T.call_time) = C.calendar_date
) AS D
GROUP BY 1
```

Non Holiday = 5,25
Holiday = 5,45

2. Find the average call time (the time in which a trip was requested) of the first time passengers make a trip.
I didn't understand the question

3. Find the average number of trips per driver for every week day

```
SELECT
        D.week_day_index
        ,D.week_day
        ,COUNT(DISTINCT D.id)                   AS trips
        ,COUNT(DISTINCT D.driver_id)              AS drivers
        ,COUNT(DISTINCT D.id)/COUNT(DISTINCT D.driver_id) AS avg_trips_per_driver
FROM
(
SELECT
        T.driver_id
        ,T.id
        ,DATE(T.call_time) AS call_time
        ,C.week_day
        ,CASE
                WHEN C.week_day LIKE 'Monday'    THEN 1
                WHEN C.week_day LIKE 'Tuesday'   THEN 2
                WHEN C.week_day LIKE 'Wednesday' THEN 3
                WHEN C.week_day LIKE 'Thursday'  THEN 4
                WHEN C.week_day LIKE 'Friday'    THEN 5
                WHEN C.week_day LIKE 'Saturday'  THEN 6
                WHEN C.week_day LIKE 'Sunday'    THEN 7
        END AS
                week_day_index
FROM trip T
LEFT JOIN calendar C ON DATE(T.call_time) = C.calendar_date
```

```
) AS D
GROUP BY 1, 2
ORDER BY 1
```

	Weekday	Avg Trips per Driver
1	Monday	3,47
2	Tuesday	3,18
3	Wednesday	3,23
4	Thursday	3,29
5	Friday	3,41
6	Saturday	3,78
7	Sunday	4

4. Which day of the week drivers usually drive the most distance on average?

```
SELECT
        D.week_day_index
        ,D.week_day
        ,AVG(D.trip_distance) as avg_distance
FROM
(
SELECT
        T.driver_id
        ,T.id
        ,DATE(T.call_time) AS call_time
        ,C.week_day
        ,CASE
                WHEN C.week_day LIKE 'Monday'    THEN 1
                WHEN C.week_day LIKE 'Tuesday'   THEN 2
                WHEN C.week_day LIKE 'Wednesday' THEN 3
                WHEN C.week_day LIKE 'Thursday'  THEN 4
                WHEN C.week_day LIKE 'Friday'    THEN 5
                WHEN C.week_day LIKE 'Saturday'  THEN 6
                WHEN C.week_day LIKE 'Sunday'    THEN 7
        END AS
                week_day_index
        ,T.trip_distance
FROM trip T
LEFT JOIN calendar C ON DATE(T.call_time) = C.calendar_date
) AS D
GROUP BY 1, 2
ORDER BY 3 DESC
```

On Sunday

5. What was the growth percentage of rides month over month?

```
SELECT
        CONCAT(C.year, '-', C.month) AS YEARMONTH
        ,(C.trips - C.trips_previous_month)/C.trips_previous_month as growth
FROM
(
        SELECT
                D.*
                ,LAG(trips, 1) OVER() AS TRIPS_PREVIOUS_MONTH
        FROM
        (
                SELECT
                        DATE_PART('year', T.call_time)   AS YEAR
                        ,DATE_PART('month', T.call_time) AS MONTH
                        ,COUNT(T.ID)                     AS trips
                FROM TRIP T
                GROUP BY 1, 2
        ) AS D
) AS C
```

jun: +9%
jul: 0%
ago: +7%
set: -21%

6. List the top 5 drivers per number of trips in the top 5 largest cities

```
WITH
T1
AS
(
        SELECT
                d.*
                ,RANK() OVER(ORDER BY trips desc) AS city_rank
        FROM
        (
        SELECT
                city_id
                ,count(id) as trips
        FROM trip T
        GROUP BY 1
        ) AS d
),

T2
AS
(
        SELECT
                T.city_id
                ,T.driver_id
                ,COUNT(T.id) AS trips
        FROM trip T
        GROUP BY 1, 2
)

SELECT
        c.name AS city
        ,d.driver_id
        ,d.trips
FROM
(
        SELECT
                t2.city_id
                ,t2.driver_id
                ,t2.trips
                ,ROW_NUMBER() OVER(PARTITION BY T2.city_id ORDER BY T2.trips DESC) AS rank_
        FROM t1
        LEFT JOIN t2 ON t1.city_id = t2.city_id
        WHERE t1.city_rank <= 5
) AS d
LEFT JOIN city c ON cast(d.city_id as int) = cast(c.id as int)
WHERE rank_ <= 5
```

City	Driver	Trips
Isengard	957ca30b	148
Isengard	7d0c4185	139
Isengard	ee25c3b5	139
Isengard	934bae83	136
Isengard	683bbf62	134
Minas Tirith	e38a1cc4	197
Minas Tirith	0729b626	153
Minas Tirith	ae160405	144
Minas Tirith	6294c442	143
Minas Tirith	db9f74db	143
Rohan	28925239	172
Rohan	b4dde07c	152
Rohan	0d3159ba	148
Rohan	5f17a268	146
Rohan	f343c362	145
Rivendell	a947550b	143
Rivendell	c5d1405e	142
Rivendell	5cfeb4fa	138
Rivendell	aeafa773	130
Rivendell	df233c3c	129
The Shire	5b0e66b0	161
The Shire	66e525c0	156
The Shire	7405b635	155
The Shire	2922568d	151
The Shire	0f2da8f1	146

Analytical

1.

Let's say it's 2019-09-23 and a new Operations manager for The Shire was just hired. She has 5 minutes during the Ops weekly meeting to present an overview of the business in the city, and since she's just arrived, she asked your help to do it. What would you prepare for this 5 minutes presentation? Please provide 1-2 slides with your idea.

The first page of this link

2.

She also mentioned she has a budget to invest in promoting the business. What kind of metrics and performance indicators would you use in order to help her decide if she should invest it into the passenger side or the driver side? Extra point if you provide data-backed recommendations.

The second page of this link

3.

One month later, she comes back, super grateful for all the helpful insights you have given her. And says she is anticipating a driver supply shortage due to a major concert that is going to take place the next day and also a 3 day city holiday that is coming the next month. What would you do to help her analyze the best course of action to either prevent or minimize the problem in each case?

In the same way for passenger side, we could build a RFM model to drivers
The second page of this link

4.

Optional. We want to build up a model to predict “Possible Churn Users” (e.g.: no trips in the past 4 weeks). List all features that you can think about and the data mining or machine learning model or other methods you may use for this case.

The first job for this issue will be the RFM model introduced on 2 point. We can follow passengers classified as At Risk to identify possible churn and plan retention actions. The second page of this link

Queries developed to analysis

1. Passengers and Revenue Growth

```
select
        d2.*
        ,lag(revenue)    over(partition by d2.city) as previous_revenue
        ,lag(passengers) over(partition by d2.city) as previous_passengers
from
(
        select
                d1.city
                ,d1.trip_month
                ,d1.trip_month_index
                ,sum(d1.trip_fare)      as revenue
                ,count(d1.passenger_id) as passengers
        from
        (
                select
                        c.name  as city
                        ,concat(cast(date_part('year', t.call_time) as text),
                                        '-',
                                        case
                                                when date_part('month', t.call_time) = 5 then 'may'
                                                when date_part('month', t.call_time) = 6 then 'jun'
                                                when date_part('month', t.call_time) = 7 then 'jul'
                                                when date_part('month', t.call_time) = 8 then 'aug'
                                                when date_part('month', t.call_time) = 9 then 'sep'
                                        end) as trip_month
                        ,case
                                when date_part('month', t.call_time) = 5 then 1
                                when date_part('month', t.call_time) = 6 then 2
                                when date_part('month', t.call_time) = 7 then 3
                                when date_part('month', t.call_time) = 8 then 4
                                when date_part('month', t.call_time) = 9 then 5
                        end as trip_month_index
                        ,t.trip_fare
                        ,t.passenger_id
                from trip t
                left join city c on cast(t.city_id as int) = c.id
                where date_part('day', t.call_time) between 1 and 23
        ) as d1
        group by 1, 2, 3
        order by 1, 3
) as d2
```

2. Growth of passengers

```
select
        a.*
        ,lag(accumulated_passengers) over(partition by city) as previous_accumulated_passengers
from
(
        select
                r.*
                ,sum(passengers) over(partition by city order by start_year_month, city) as
accumulated_passengers
        from
        (
                select
                        d.city
                        ,d.start_year_month
                        ,count(distinct passenger_id) as passengers
                from
                (
                        select
                                p.city
                                ,p.passenger_id
                                ,date(concat(date_part('year', p.start_date), '-',
                                        date_part('month', p.start_date), '-01'))   as
start_year_month
                        from
                        (
                                select
                                        p.id              as passenger_id
                                        ,p.first_call_time  as start_date
                                        ,c.name           as city
                                        ,min(t.call_time)   as first_trip_date
                                        ,max(t.call_time)   as last_trip_date
                                        ,count(t.id)        as trips
                                        ,sum(t.trip_fare)   as total_trip_fare
                                from passenger p
                                left join trip t on p.id = t.passenger_id
                                left join city c on cast(t.city_id as int) = c.id
                                where p.first_call_time between '2019-05-02' and '2019-09-23'
                                and c.name is not null
                                group by 1, 3
                        ) as p
                ) as d
                group by 1, 2
        ) as r
)
as a
```

3. Active passengers

```
with
all_passengers
as
(
select
        b.city
        ,b.arrived_month
        ,case
                when b.arrived_month like 'before 2019-May' then 1
                when b.arrived_month like '2019-May'       then 2
                when b.arrived_month like '2019-Jun'       then 3
                when b.arrived_month like '2019-Jul'       then 4
                when b.arrived_month like '2019-Ago'       then 5
                when b.arrived_month like '2019-Sep'       then 6
        else
                null
        end as arrived_month_index
        ,count(passenger_id) as passengers
from
(
        select
                l.city
                ,l.passenger_id
                ,l.arrived_month
                ,case
                        when date_part('day', date('2019-09-23') - last_trip) <= 14 then 'active'
                        else
                                'non-active'
                end as is_active_passenger
        from
        (
                select
                        d.city
                        ,d.passenger_id
                        ,d.arrived_month
                        ,max(d.call_time) as last_trip
                from
                (
                        select
                                c.name as city
                                ,p.id  as passenger_id
                                ,date(concat(date_part('year', p.first_call_time), '-',
                                                                date_part('month', p.first_call_time), '-',
                                                                date_part('day', p.first_call_time))) as
first_call_time
                                ,case
                                        when p.first_call_time < '2019-05-02' then 'before 2019-May'
                                        when p.first_call_time between '2019-05-02' and '2019-05-31' then
'2019-May'
                                        when p.first_call_time between '2019-06-01' and '2019-06-30' then
'2019-Jun'
                                        when p.first_call_time between '2019-07-01' and '2019-07-31' then
'2019-Jul'
```

```
                                        when p.first_call_time between '2019-08-01' and '2019-08-31' then
'2019-Ago'
                                        when p.first_call_time between '2019-09-01' and '2019-09-23' then
'2019-Sep'
                                else
                                        null
                                end as arrived_month
                                ,t.call_time
                        from passenger p
                        left join trip t on p.id = t.passenger_id
                        left join city c on cast(t.city_id as int) = c.id
                        where p.first_call_time is not null
                ) as d
                group by 1, 2, 3
        )
        as l
        where l.arrived_month is not null
        and l.city is not null
) as b
group by 1, 2, 3
order by 1, 3
),

active_passengers
as
(
select
        b.city
        ,b.arrived_month
        ,case
                when b.arrived_month like 'before 2019-May' then 1
                when b.arrived_month like '2019-May'       then 2
                when b.arrived_month like '2019-Jun'       then 3
                when b.arrived_month like '2019-Jul'       then 4
                when b.arrived_month like '2019-Ago'       then 5
                when b.arrived_month like '2019-Sep'       then 6
        else
                null
        end as arrived_month_index
        ,count(passenger_id) as active_passengers
from
(
        select
                l.city
                ,l.passenger_id
                ,l.arrived_month
                ,case
                        when date_part('day', date('2019-09-23') - last_trip) <= 14 then 'active'
                        else
                                'non-active'
                end as is_active_passenger
        from
        (
                select
                        d.city
                        ,d.passenger_id
                        ,d.arrived_month
                        ,max(d.call_time) as last_trip
```

```
                from
                (
                        select
                                c.name as city
                                ,p.id  as passenger_id
                                ,date(concat(date_part('year', p.first_call_time), '-',
                                                date_part('month', p.first_call_time), '-',
                                                date_part('day', p.first_call_time))) as
first_call_time
                                ,case
                                        when p.first_call_time < '2019-05-02' then 'before 2019-May'
                                        when p.first_call_time between '2019-05-02' and '2019-05-31' then
'2019-May'
                                        when p.first_call_time between '2019-06-01' and '2019-06-30' then
'2019-Jun'
                                        when p.first_call_time between '2019-07-01' and '2019-07-31' then
'2019-Jul'
                                        when p.first_call_time between '2019-08-01' and '2019-08-31' then
'2019-Ago'
                                        when p.first_call_time between '2019-09-01' and '2019-09-23' then
'2019-Sep'
                                else
                                        null
                                end as arrived_month
                                ,t.call_time
                        from passenger p
                        left join trip t on p.id = t.passenger_id
                        left join city c on cast(t.city_id as int) = c.id
                        where p.first_call_time is not null
                ) as d
                group by 1, 2, 3
        )
        as l
        where l.arrived_month is not null
        and l.city is not null
) as b
where b.is_active_passenger like 'active'
group by 1, 2, 3
order by 1, 3
)

select
        ap.city
        ,ap.arrived_month
        ,ap.arrived_month_index
        ,ap.passengers
        ,tp.active_passengers
from all_passengers ap
left join active_passengers tp on ap.city = tp.city and ap.arrived_month = tp.arrived_month
```

Churn

```
with
all_passengers
as
(
select
        b.city
        ,b.arrived_month
        ,case
                when b.arrived_month like 'before 2019-May' then 1
                when b.arrived_month like '2019-May'       then 2
                when b.arrived_month like '2019-Jun'       then 3
                when b.arrived_month like '2019-Jul'       then 4
                when b.arrived_month like '2019-Ago'       then 5
                when b.arrived_month like '2019-Sep'       then 6
        else
                null
        end as arrived_month_index
        ,count(passenger_id) as passengers
from
(
        select
                l.city
                ,l.passenger_id
                ,l.arrived_month
                ,case
                        when date_part('day', date('2019-09-23') - last_trip) > 30 then 'possible_churn'
                        else
                                'non-possible_churn'
                end as is_possible_churn_passenger
        from
        (
                select
                        d.city
                        ,d.passenger_id
                        ,d.arrived_month
                        ,max(d.call_time) as last_trip
                from
                (
                        select
                                c.name as city
                                ,p.id  as passenger_id
                                ,date(concat(date_part('year', p.first_call_time), '-',
                                                                        date_part('month', p.first_call_time), '-',
                                                                        date_part('day', p.first_call_time))) as
first_call_time
                                ,case
                                        when p.first_call_time < '2019-05-02' then 'before 2019-May'
                                        when p.first_call_time between '2019-05-02' and '2019-05-31' then
'2019-May'
                                        when p.first_call_time between '2019-06-01' and '2019-06-30' then
'2019-Jun'
                                        when p.first_call_time between '2019-07-01' and '2019-07-31' then
'2019-Jul'
                                        when p.first_call_time between '2019-08-01' and '2019-08-31' then
'2019-Ago'
```

```
                                        when p.first_call_time between '2019-09-01' and '2019-09-23' then
'2019-Sep'
                                    else
                                        null
                                    end as arrived_month
                                    ,t.call_time
                                from passenger p
                                left join trip t on p.id = t.passenger_id
                                left join city c on cast(t.city_id as int) = c.id
                                where p.first_call_time is not null
                        ) as d
                        group by 1, 2, 3
                )
                as l
                where l.arrived_month is not null
                and l.city is not null
) as b
group by 1, 2, 3
order by 1, 3
),

possible_churn_passengers
as
(
select
                b.city
                ,b.arrived_month
                ,case
                        when b.arrived_month like 'before 2019-May' then 1
                        when b.arrived_month like '2019-May'        then 2
                        when b.arrived_month like '2019-Jun'        then 3
                        when b.arrived_month like '2019-Jul'        then 4
                        when b.arrived_month like '2019-Ago'        then 5
                        when b.arrived_month like '2019-Sep'        then 6
                else
                        null
                end as arrived_month_index
                ,count(passenger_id) as possible_churn_passengers
from
(
                select
                        l.city
                        ,l.passenger_id
                        ,l.arrived_month
                        ,case
                                when date_part('day', date('2019-09-23') - last_trip) > 30 then 'possible-churn'
                                else
                                        'non-possible-churn'
                        end as is_possible_churn
                from
                (
                        select
                                d.city
                                ,d.passenger_id
                                ,d.arrived_month
                                ,max(d.call_time) as last_trip
                        from
                        (
```

```
                        select
                                c.name as city
                                ,p.id  as passenger_id
                                ,date(concat(date_part('year', p.first_call_time), '-',
                                                date_part('month', p.first_call_time), '-',
                                                date_part('day', p.first_call_time))) as
first_call_time
                                ,case
                                        when p.first_call_time < '2019-05-02' then 'before 2019-May'
                                        when p.first_call_time between '2019-05-02' and '2019-05-31' then
'2019-May'
                                        when p.first_call_time between '2019-06-01' and '2019-06-30' then
'2019-Jun'
                                        when p.first_call_time between '2019-07-01' and '2019-07-31' then
'2019-Jul'
                                        when p.first_call_time between '2019-08-01' and '2019-08-31' then
'2019-Ago'
                                        when p.first_call_time between '2019-09-01' and '2019-09-23' then
'2019-Sep'
                                else
                                        null
                                end as arrived_month
                                ,t.call_time
                        from passenger p
                        left join trip t on p.id = t.passenger_id
                        left join city c on cast(t.city_id as int) = c.id
                        where p.first_call_time is not null
                ) as d
                group by 1, 2, 3
        )
        as l
        where l.arrived_month is not null
        and l.city is not null
) as b
where b.is_possible_churn like 'possible-churn'
group by 1, 2, 3
order by 1, 3
)

select
        ap.city
        ,ap.arrived_month
        ,ap.arrived_month_index
        ,ap.passengers
        ,coalesce(pc.possible_churn_passengers, 0) as possible_churn_passengers
from all_passengers ap
left join possible_churn_passengers pc on ap.city = pc.city and ap.arrived_month = pc.arrived_month
```

5. LTV

```
select
        d.city
        ,d.passenger_id
        ,cast(sum(d.trip_fare) as int) as ltv
from
(
        select
                c.name as city
                ,p.id  as passenger_id
                ,t.trip_fare
        from passenger p
        left join trip t on p.id = t.passenger_id
        left join city c on cast(t.city_id as int) = c.id
        where p.first_call_time is not null
        and t.city_id is not null
) as d
group by 1, 2
```

6. Drivers churn

```
select
        r.city
        ,r.first_trip_yearmonth
        ,r.first_trip_yearmonth_index
        ,sum(r.is_active_driver) as active_drivers
        ,sum(r.is_driver_churn)  as drivers_churn
from
(
        select
                d.city
                ,d.driver_id
                ,case
                        when d.first_trip between '2019-05-02' and '2019-05-31' then '2019-May'
                        when d.first_trip between '2019-06-01' and '2019-06-30' then '2019-Jun'
                        when d.first_trip between '2019-07-01' and '2019-07-31' then '2019-Jul'
                        when d.first_trip between '2019-08-01' and '2019-08-31' then '2019-Ago'
                        when d.first_trip between '2019-09-01' and '2019-09-23' then '2019-Set'
                end as first_trip_yearmonth
                ,case
                        when d.first_trip between '2019-05-02' and '2019-05-31' then 1
                        when d.first_trip between '2019-06-01' and '2019-06-30' then 2
                        when d.first_trip between '2019-07-01' and '2019-07-31' then 3
                        when d.first_trip between '2019-08-01' and '2019-08-31' then 4
                        when d.first_trip between '2019-09-01' and '2019-09-23' then 5
                end as first_trip_yearmonth_index
                ,d.is_active_driver
                ,d.is_driver_churn
        from
        (
                select
                        t.city
                        ,t.driver_id
                        ,date(concat(date_part('year', t.first_trip), '-',
                                                date_part('month', t.first_trip), '-',
                                                date_part('day', t.first_trip))) as first_trip
                        ,case
                                when date_part('day', date('2019-09-23') - last_trip) > 30 then 1
                                else
                                        0
                                end as is_driver_churn
                        ,case
                                when date_part('day', date('2019-09-23') - last_trip) <= 30 then 1
                                else
                                        0
                                end as is_active_driver
                from
                (
                        select
                                c.name as city
                                ,t.driver_id
                                ,min(t.call_time) as first_trip
                                ,max(t.call_time) as last_trip
                        from trip t
                        left join city c on cast(t.city_id as int) = c.id
```

```
                        where t.city_id is not null
                        and t.call_time is not null
                        group by 1, 2
                    ) as t
            ) as d
) as r
where r.first_trip_yearmonth is not null
group by 1, 2, 3
```

7. RFM

```
select
        presentation.*
        ,case
                when rfm_class like 'Neutral'          then 0
                when rfm_class like 'Can not lose them' then 1
                when rfm_class like 'At Risk Customer'  then 2
                when rfm_class like 'New Customers'     then 3
                when rfm_class like 'Potential Loyalts' then 4
                when rfm_class like 'Champions'         then 5
        end as rfm_class_index
from
(

select
        summary.rfm_class
        ,count(summary.passenger_id) as passengers
from
(
        select
                rfm.*
                ,case
                        when
                                rfm.recency_score = 3
                                and rfm.frequency_score = 3
                                and rfm.monetary_score between 2 and 3
                        then
                                'Champions'
                        when
                                rfm.recency_score = 3
                                and rfm.frequency_score = 3
                                and rfm.monetary_score = 1
                        then
                                'Potential Loyalts'
                        when
                                rfm.recency_score = 3
                                and rfm.frequency_score = 2
                                and rfm.monetary_score = 3
                        then
                                'Potential Loyalts'
                        when
                                rfm.recency_score = 3
                                and rfm.frequency_score between 1 and 3
                                and rfm.monetary_score  between 1 and 3
                        then
                                'New Customers'
                        when
                                rfm.recency_score = 2
                        then
                                'At Risk Customer'
                        when
                                rfm.recency_score = 1
                                and rfm.frequency_score between 2 and 3
                                and rfm.monetary_score  between 2 and 3
```

```
                then
                        'Can not lose them'
                when
                        rfm.recency_score = 1
                        and rfm.frequency_score between 1 and 3
                        and rfm.monetary_score = 1
                then
                        'Neutral'
                when
                        rfm.recency_score = 1
                        and rfm.frequency_score = 1
                        and rfm.monetary_score = 2
                then
                        'Neutral'
                else
                        'Neutral'
        end as rfm_class
from
(
        select
                t.*
                ,case
                        when t.recency <= 30            then 3
                        when t.recency between 31 and 90 then 2
                        when t.recency > 90             then 1
                end as recency_score
                ,case
                        when t.frequency >= 3 then 3
                        when t.frequency = 2  then 2
                        when t.frequency = 1  then 1
                end as frequency_score
                ,case
                        when t.monetary > 20            then 3
                        when t.monetary between 10 and 20 then 2
                        when t.monetary < 10            then 1
                end as monetary_score
        from
        (
                select
                        distinct d.*
                from
                (
                        select
                                t.passenger_id
                                ,date_part('day', date('2019-09-23') - max(t.call_time)
over(partition by t.passenger_id)) as recency
                                ,count(t.id) over(partition by t.passenger_id)
as frequency
                                ,cast(sum(t.trip_fare) over(partition by t.passenger_id) as int)
as monetary
                        from trip t
                        left join city c on cast(t.city_id as int) = c.id
                        where c.name like 'The Shire'
                ) as d
        ) as t
) as rfm
) as summary
group by 1
```

```
order by 2 desc
) as presentation
order by rfm_class_index desc
```