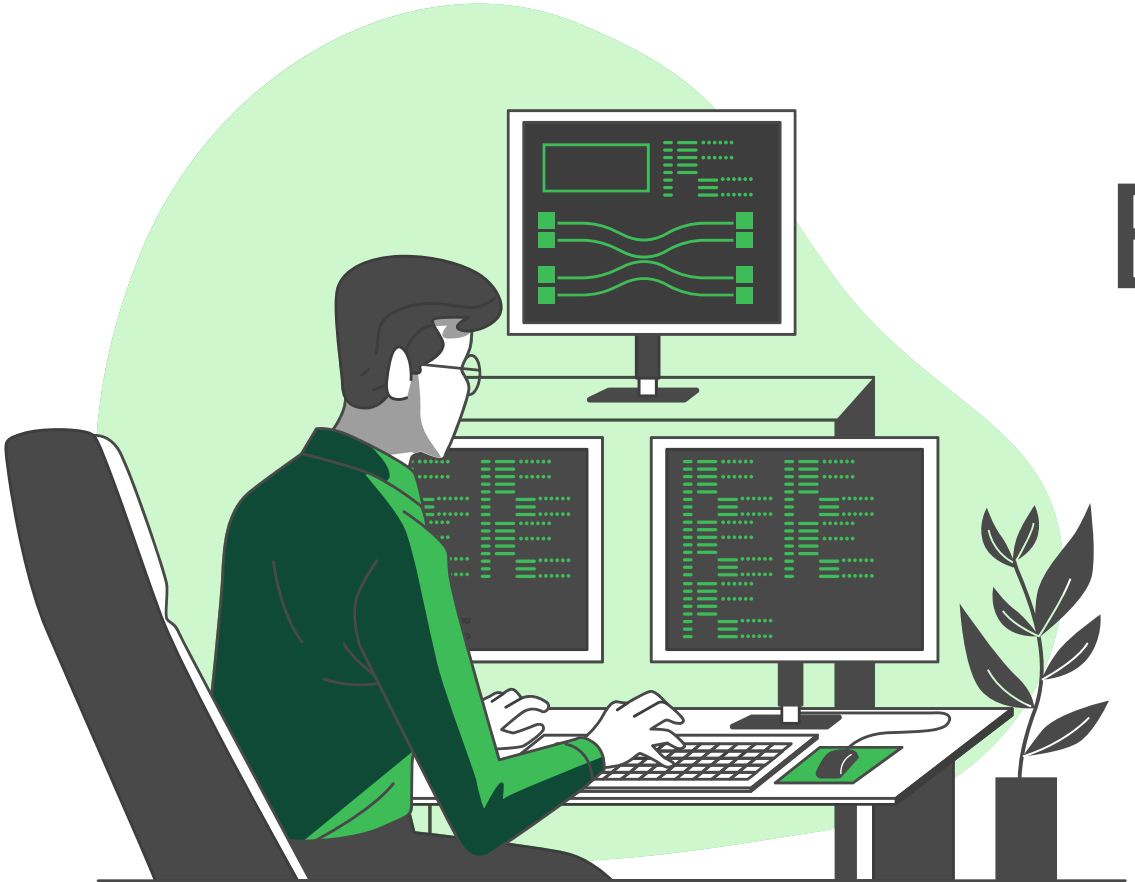
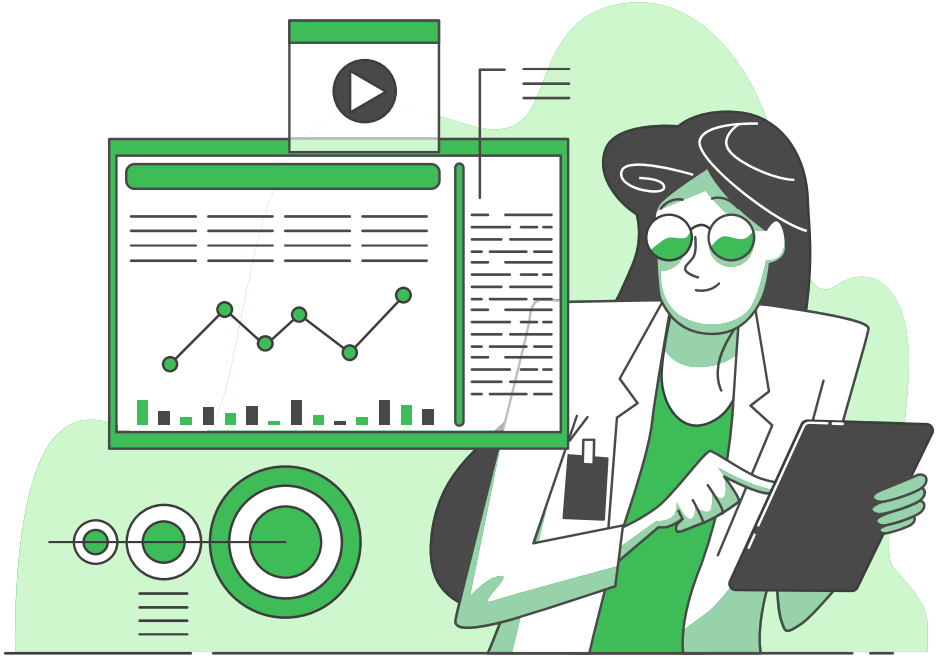


BI Internship Use Case Report

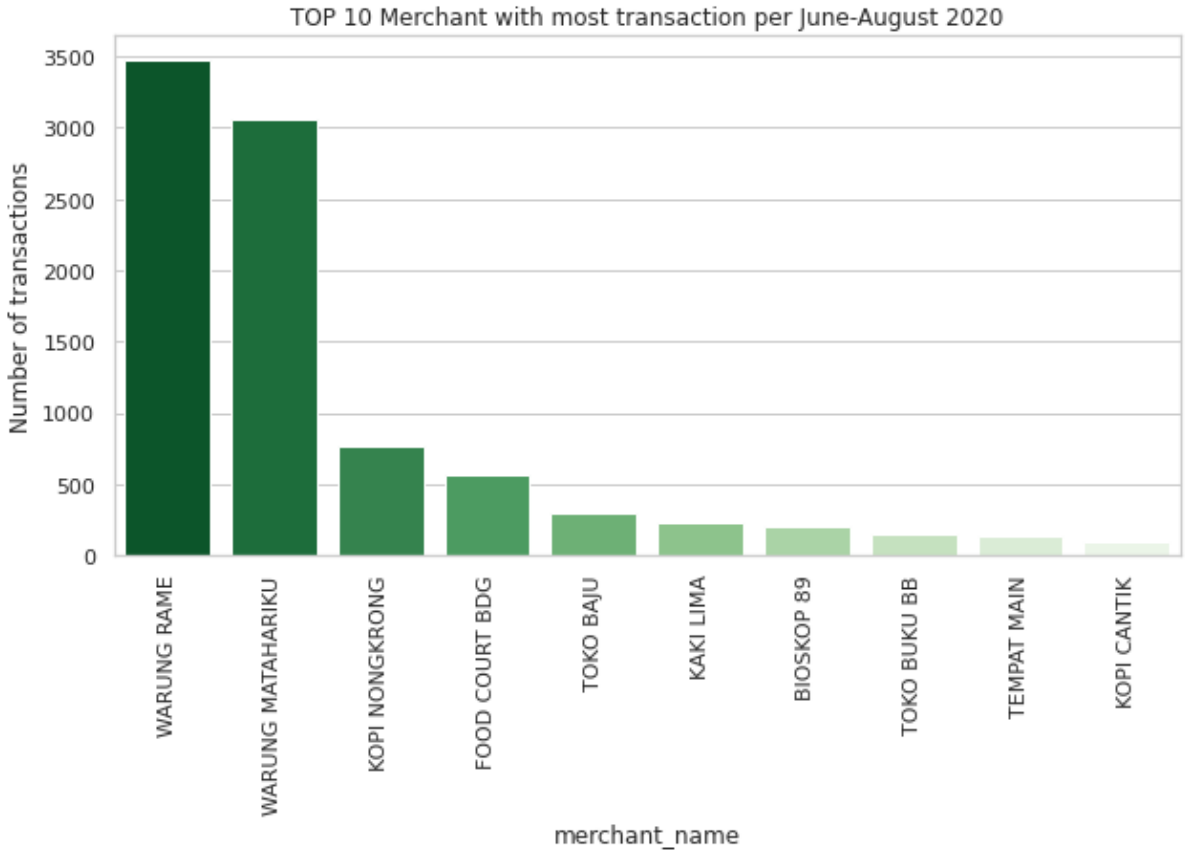
By: Iqrar Agalosi Nureyza

Table of Contents

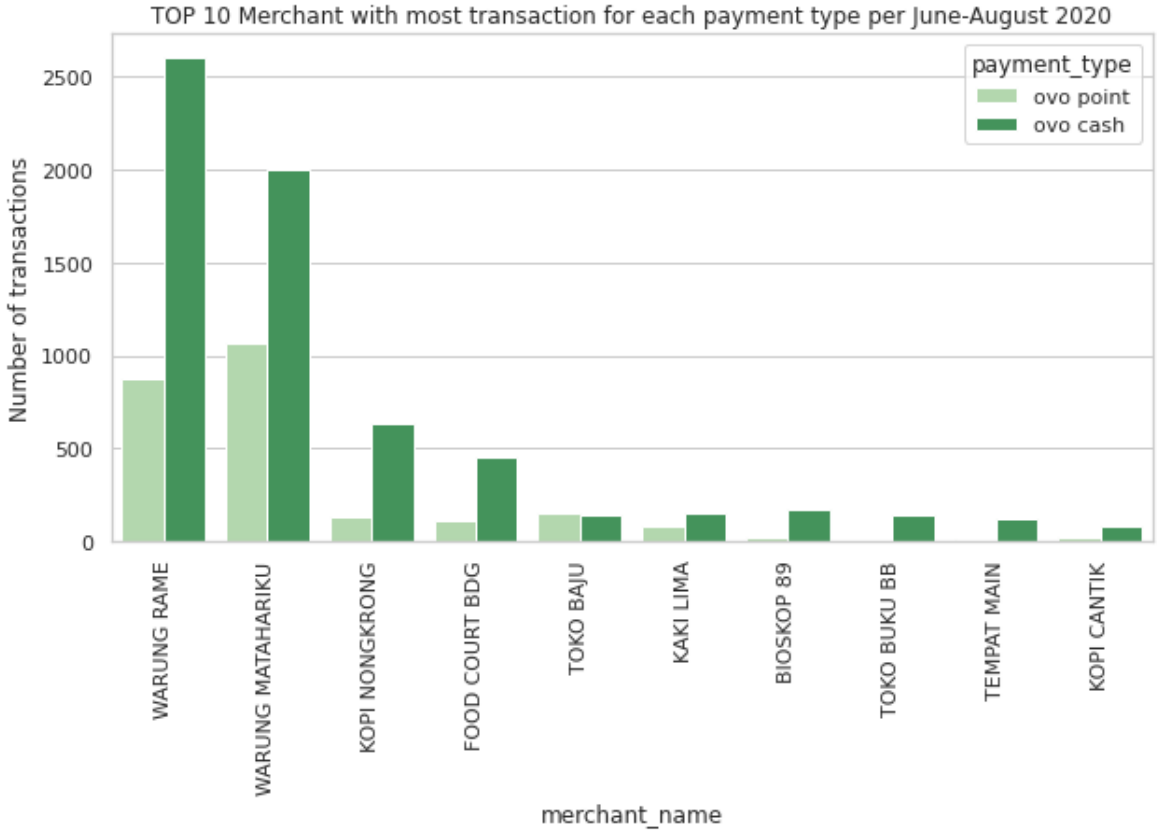


1.1
Section

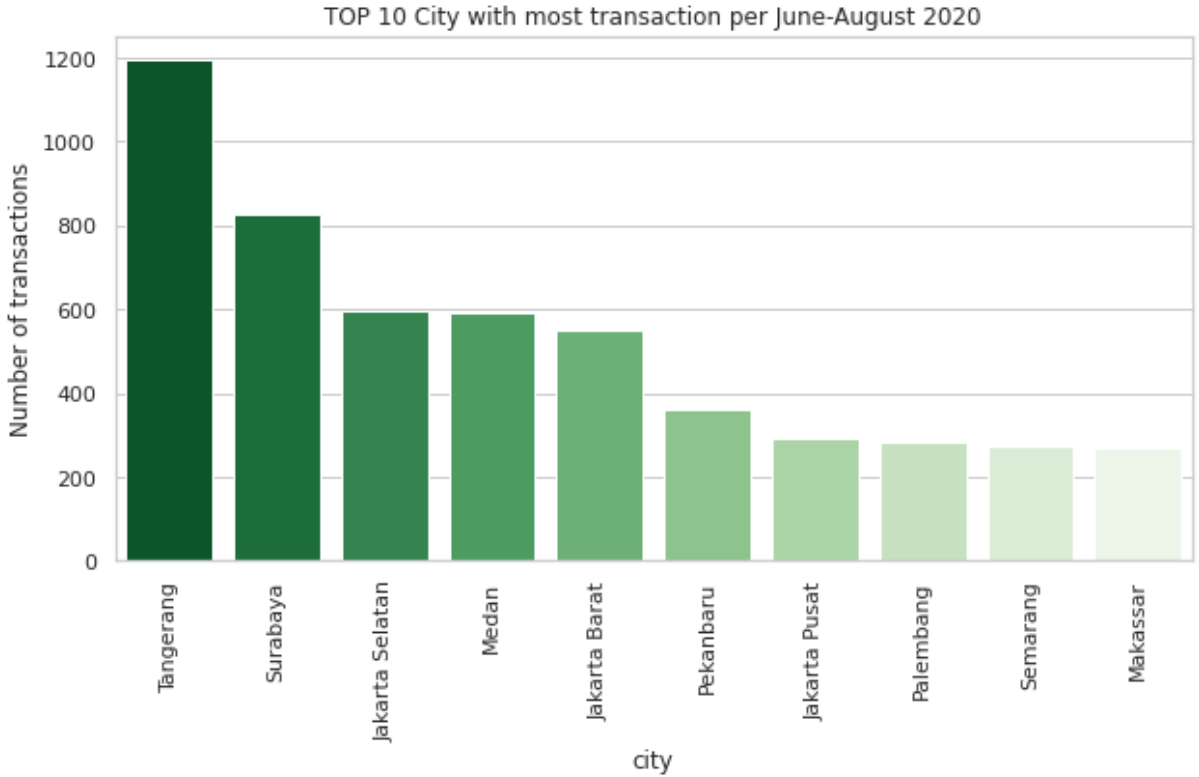
Insights from the data



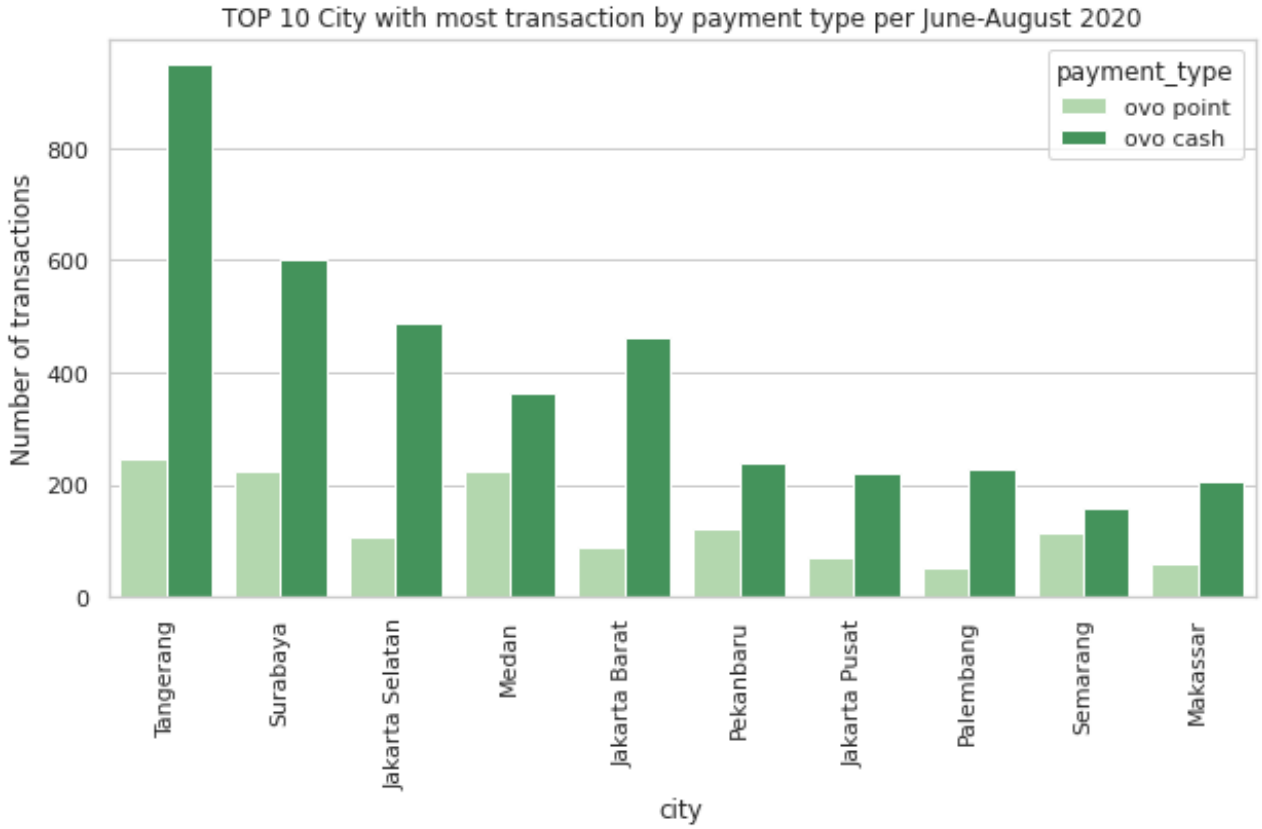
Warung Rame dominates the transactions data following by Warung Matahariku with 3000+ transactions



For each merchant on the TOP 10, payment using OVO Cash is dominating except Toko Baju. OVO Point payment on Toko Baju is slightly higher than OVO Cash.



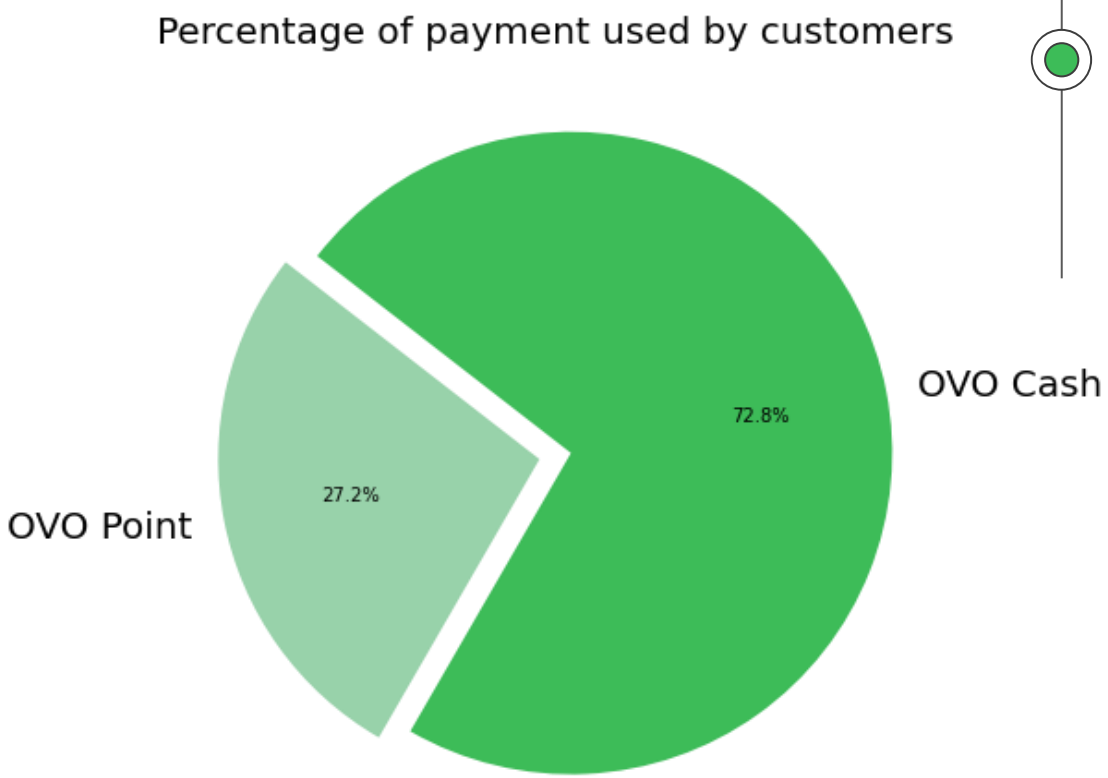
Tangerang is the city where customers and merchants have 1100+ total transactions. Following by Surabaya with 800+ transactions and Jakarta Selatan with almost 600 transactions during June-August 2020.



All payment method for all city in TOP 10 is dominating by OVO Cash

6,700
Customers using **OVO Cash**

2,500
Customers using **OVO Point**



Number of transactions happen every day

~13,014,430 Average

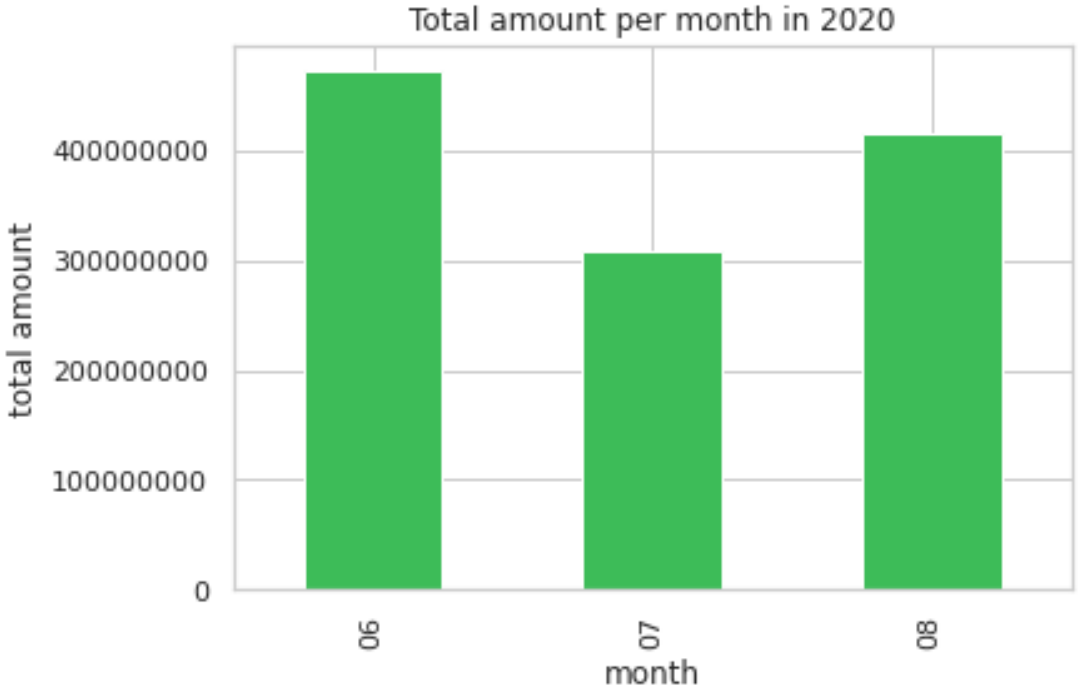
Total amount every day

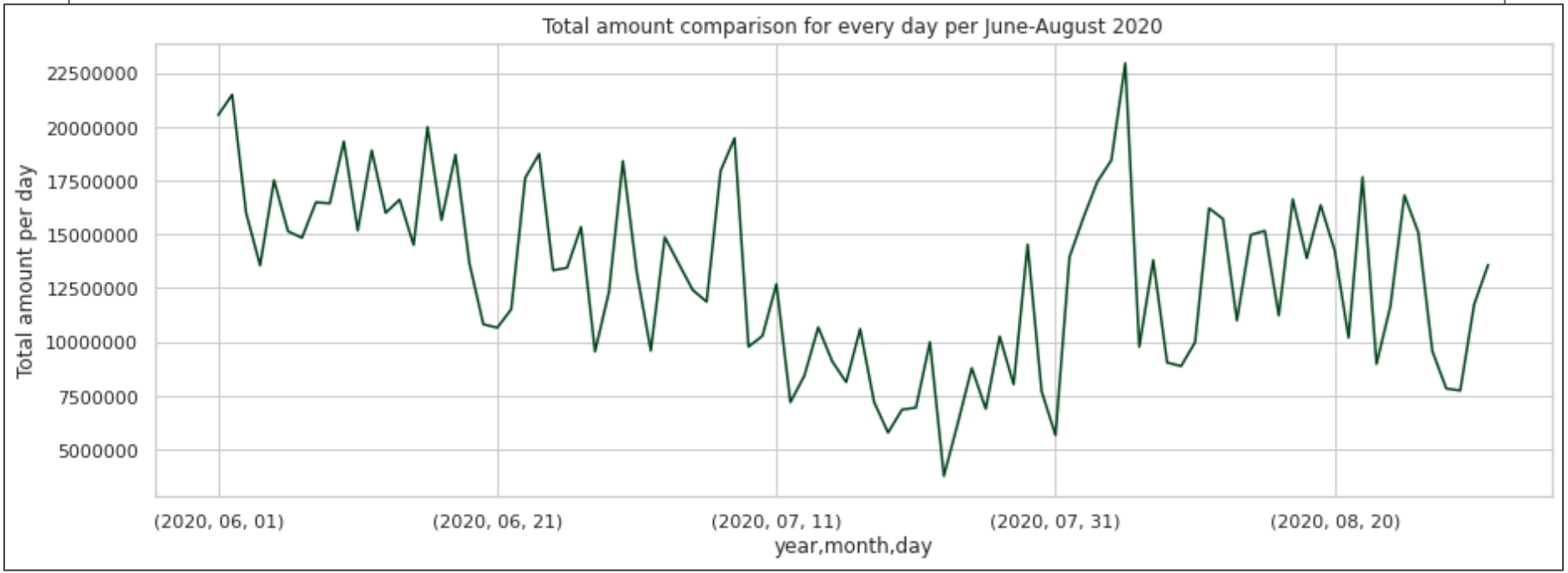
84,828.5 Average

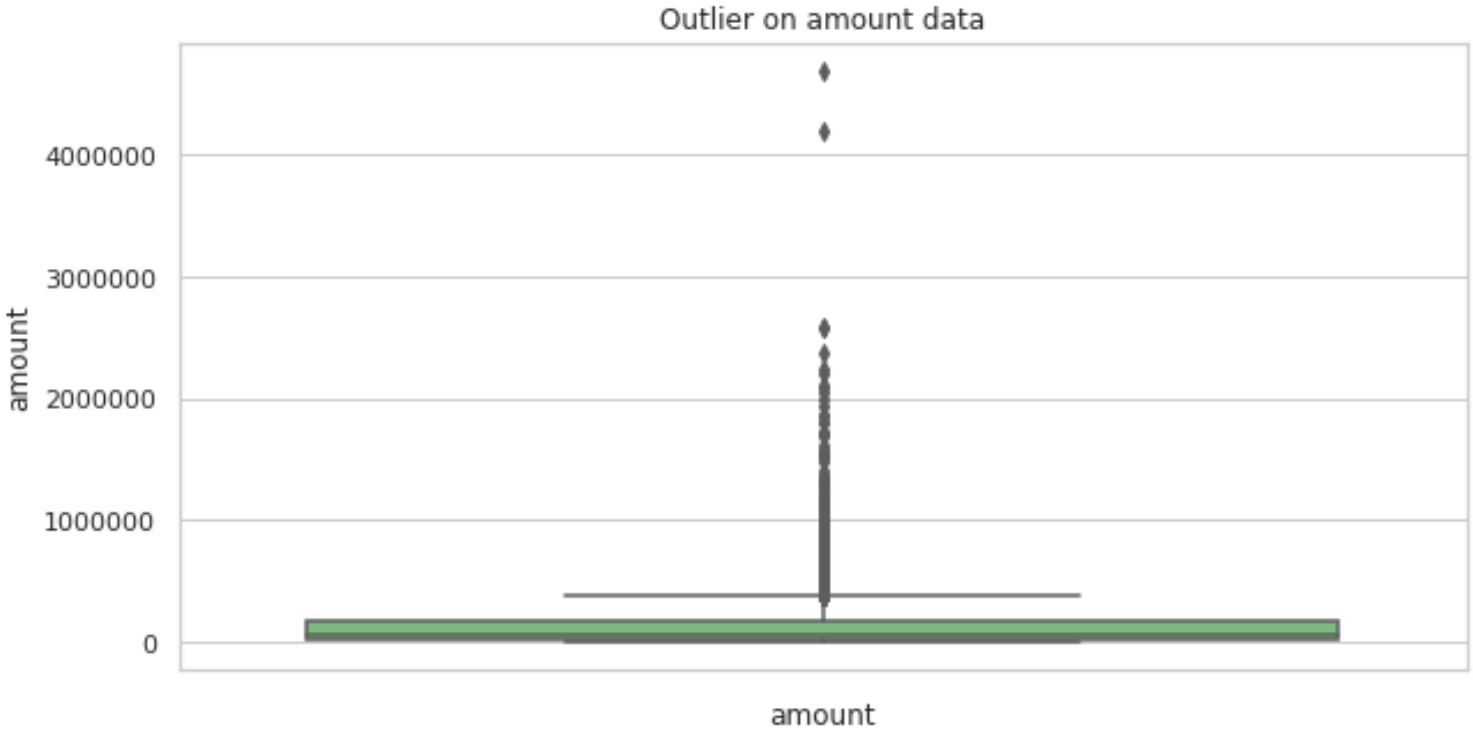
Cashback every day

472,349,090

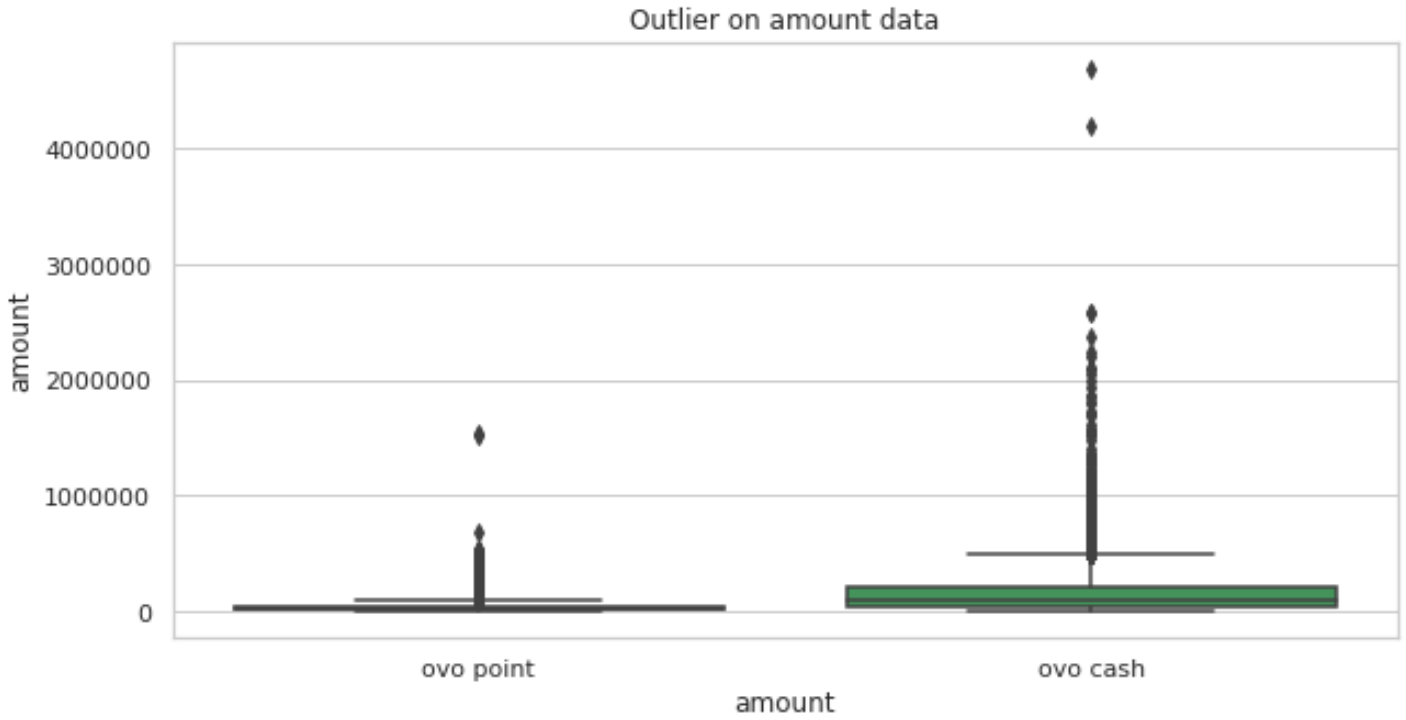
The highest amount is in June 2020



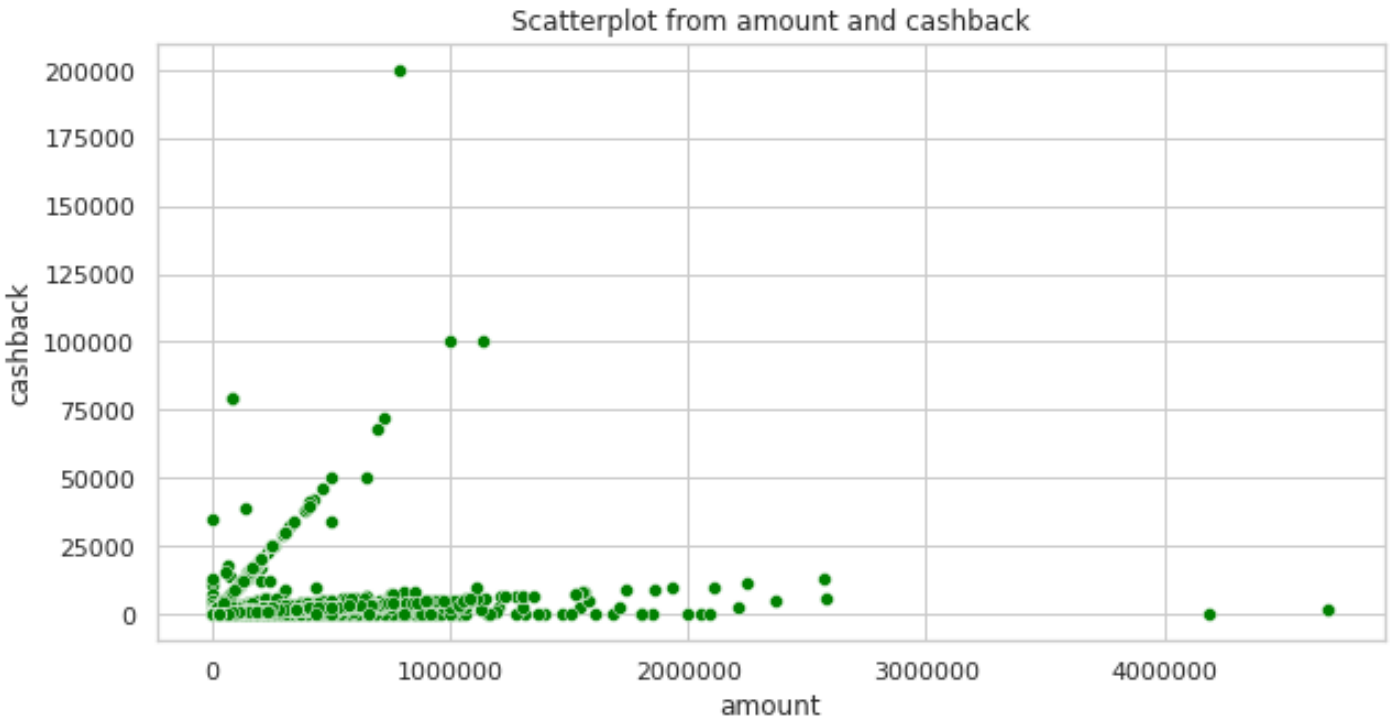

Total amount comparison for every day per June-August 2020
Total amount per day
22500000
20000000
17500000
15000000
12500000
10000000
7500000
5000000
(2020, 06, 01)
(2020, 06, 21)
(2020, 07, 11)
(2020, 07, 31)
(2020, 08, 20)
year,month,day



By using box plot, we can see that amount in the data has a quite big outlier there. This is natural since there could be a big nominal in a single transaction from rich customer or from expensive stuff.



Outlier in ovo point is smaller dan ovo cash. This is happened because the majority of customers are using ovo cash. So there are more variants in the data.



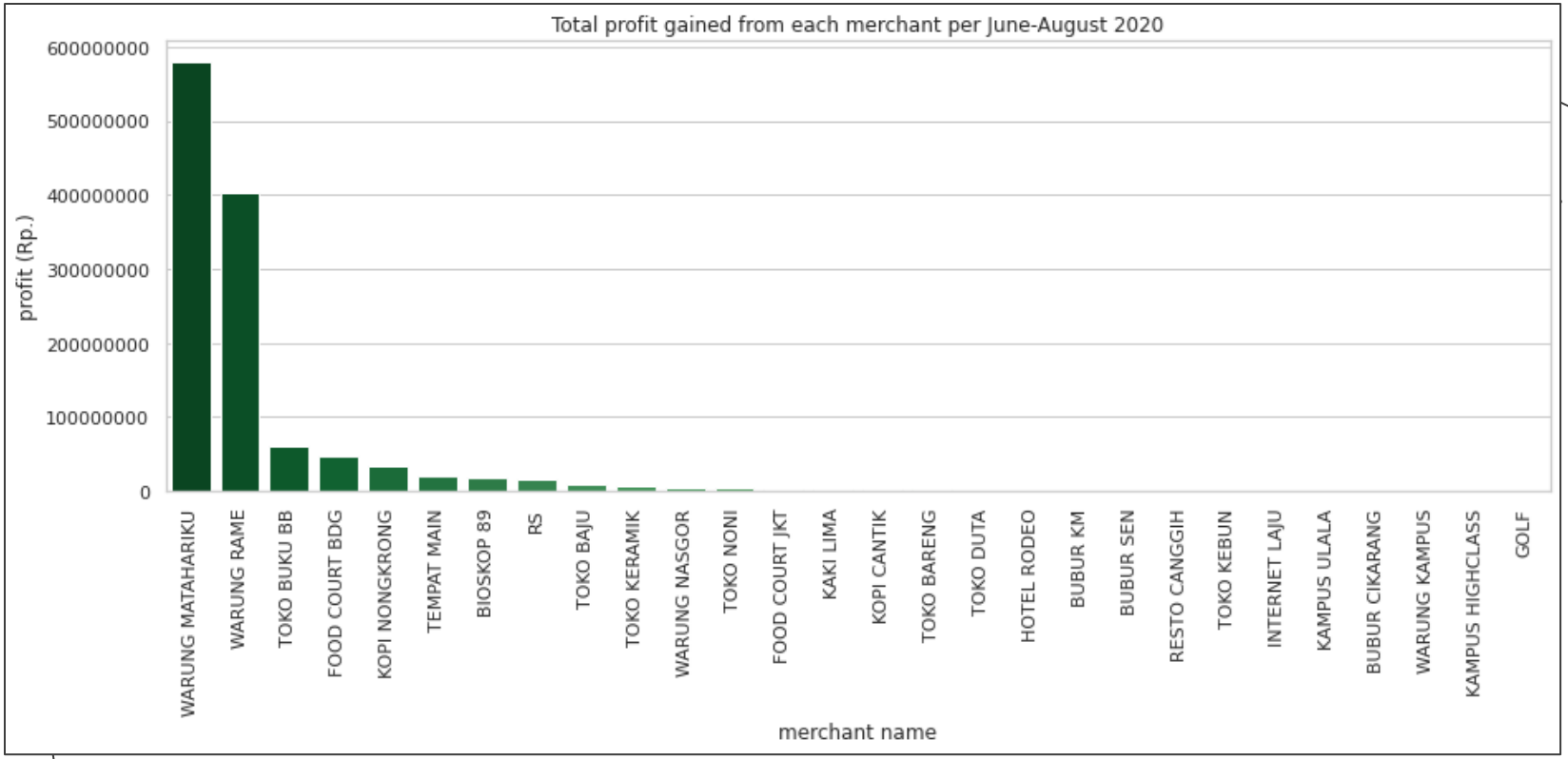
0.5%

Cashback Ratio Avg.

From the scatterplot we see how amount and cashback has a relation with cashback ratio is about 0.5% by average. And there are some data where the cashback has almost 100%. This is possible because OVO could gives high cashback for some customers as a promo.

Rp. 1,220,621,876

Is a total profit in all data by
subtracting amount by cashback



If we see from profit perspective, Warung Matahariku merchant gained more than 550 million rupiah, following by Warung Rame with 400+ million rupiah during June-August 2020. Even Warung Rame has the most transaction happened, Warung Matahariku has more profit than Warung Rame.

Issues to be highlighted

Issue #1

Store Code in ovo transaction data seems inconsistent because it mixed up with long and short code. For example, there is a store code named TCPLJKTLMPU0001 and on the other hand there is code named 490. One is using alphanumeric and one is number only. Would be better if it named with the same format. The data engineer or people who process the data will be more comfortable.

Issue #2

By seeing the big gap between ovo cash and ovo point payment usage. As a senior management, it's time to discuss and find any ideas how to make the usage of ovo point higher.

1.2
Section

Task 1

List total unique customers that had transactions in JABODETABEK between 4 August 2020 and 17 August 2020.

My Solution

```sql
SELECT COUNT(ovo_id)
FROM public.ovo_transaction AS ovo_t
INNER JOIN public.ovo_ref_merchant AS ovo_r
ON ovo_t.merchant_id = ovo_r.merchant_id AND ovo_t.store_code = ovo_r.store_code
-- Use pattern matching for Jakarta, and array for the others to easier the conditional search
WHERE (city LIKE 'Jakarta%' OR city = ANY(ARRAY['Bogor', 'Depok', 'Tangerang', 'Bekasi']))
AND (txndate >= '2020-08-04' AND txndate <= '2020-08-17');
```

Result

total_unique_customer
555

The .csv file for this submission is attached along with this submission

Code for Copy-Paste

The .sql file is available along with this submission, See task1.sql

Task 2

List only the Top 10 cities who had the highest transaction amount, per month.

My Solution

```sql
SELECT * FROM (
    SELECT city,
        SUM(amount) AS total_amount,
        EXTRACT(month FROM txndate::date) AS month_txn,
        -- Use rank() function to give rank for each row so it can be easier to filter
        rank() OVER (PARTITION BY EXTRACT(month FROM txndate::date) ORDER BY SUM(amount) DESC)
    FROM public.ovo_transaction AS ovo_t
    INNER JOIN public.ovo_ref_merchant AS ovo_r
    ON ovo_t.store_code = ovo_r.store_code AND ovo_t.merchant_id = ovo_r.merchant_id
    GROUP BY city, EXTRACT(month FROM txndate::date)
    ORDER BY month_txn ASC, total_amount DESC
-- With this, we can pick all top 10 for each month
) top10_per_month WHERE rank <= 10;
```

Result

city	total_amount	month_txn	rank
Tangerang	51794450	6	1
Surabaya	38026418	6	2
Jakarta Barat	32552603	6	3
Medan	31708128	6	4
Pekanbaru	21507070	6	5
Jakarta Selatan	19285370	6	6
Jakarta Utara	16143553	6	7
Makassar	15902976	6	8
Palembang	15395640	6	9
Jakarta Pusat	13218880	6	10

The .csv file for this submission is attached along with this submission

Result (Cont.)

city	total_amount	month_txn	rank
Tangerang	31940425	7	1
Surabaya	30877885	7	2
Jakarta Barat	27848050	7	3
Medan	23307320	7	4
Pekanbaru	15242526	7	5
Palembang	12934867	7	6
Jakarta Utara	12639128	7	7
Jakarta Selatan	10132617	7	8
Makassar	9031086	7	9
Bekasi	7858880	7	10

The .csv file for this submission is attached along with this submission

Result (Cont.)

city	total_amount	month_txn	rank
Tangerang	46943078	8	1
Surabaya	35705402	8	2
Jakarta Barat	31085902	8	3
Pekanbaru	27145565	8	4
Jakarta Selatan	24149182	8	5
Medan	20500527	8	6
Palembang	18565554	8	7
Jakarta Utara	14725477	8	8
Makassar	11951637	8	9
Jakarta Pusat	11625691	8	10

The .csv file for this submission is attached along with this submission

Code for Copy-Paste

The .sql file is available along with this submission, See task2.sql

Task 3

List total unique customers, total number of transactions, total amount paid, and total cashback per merchant, sort it by total number of transactions from biggest to smallest, then remove merchants who have cash back ratio (total cashback / total amount) below 5%.

My Solution

```sql
SELECT merchant_id,
    -- Simply use aggregate functions
    COUNT(DISTINCT ovo_id) AS total_unique_customer,
    COUNT(txn_id) AS total_transaction,
    SUM(amount) AS total_amount,
    SUM(cashback) AS total_cashback,
    (SUM(cashback)/SUM(amount)) AS cashback_ratio
FROM public.ovo_transaction
GROUP BY merchant_id
HAVING (SUM(cashback)/SUM(amount)) >= 0.05
ORDER BY total_transaction DESC;
```

Result

merchant_id	total_unique_customer	total_transaction	total_amount	total_cashback	cashback_ratio
275	133	133	22300000	1764000	0.07910313901345291
2858	100	102	3194000	194000	0.06073888541014402
17	46	46	5641904	470000	0.08330521043959628
108	10	10	4288053	411000	0.09584769591234063
2016	4	4	1356165	127000	0.09364642207990916

The .csv file for this submission is attached along with this submission

Result (Cont.)

merchant_id	total_unique_customer	total_transaction	total_amount	total_cashback	cashback_ratio
107	4	4	332409	31000	0.09325860611475621
5110	2	2	110000	15000	0.13636363636363635
2831	1	1	75000	14000	0.18666666666666668
155	1	1	426000	42000	0.09859154929577464
2916	1	1	135000	39000	0.28888888888888886
5129	1	1	790000	200000	0.25316455696202533

The .csv file for this submission is attached along with this submission

Code for Copy-Paste

The .sql file is available along with this submission, See task3.sql

Task 4

Show the distribution of customers based on the number of unique merchants s/he had transacted with. (Distribution of customer that have done transaction in 1 unique merchant, 2 unique merchants, 3 unique merchants, etc)

My Solution

```sql
-- Count how many customer that used certain merchants
SELECT total_merchant AS total_unique_merchant_transacted_with, COUNT(ovo_id) AS total_customer
FROM (
    -- Count how many unique merchant per customer
    SELECT ovo_id, COUNT(merchant_id) AS total_merchant
    FROM (
        -- Make sure we do not have duplicate record
        SELECT DISTINCT ovo_id, merchant_id
        FROM public.ovo_transaction
    ) AS unique_record
    GROUP BY ovo_id
    ORDER BY total_merchant
) AS task4_table
GROUP BY total_unique_merchant_transacted_with;
```

Result

total_unique_merchant_transacted_with	total_customer
1	9022
2	38
3	2

From the table above we know that there are 9022 customers that using only one merchant. Following by 38 customers using 2 merchants and 2 customers using 3 different merchants.

The .csv file for this submission is attached along with this submission

Code for Copy-Paste

The .sql file is available along with this submission, See task4.sql

Task 5

List customers who have transactions in the exact following order of merchants: TOKO BAJU, KOPI NONGKRONG, then KAKI LIMA. (If customer have transactions in the same merchant multiple times in sequential order then it counts as one appearance, for example: TOKO BAJU, KOPI NONGKRONG, KOPI NONGKRONG, KAKI LIMA, KOPI NONGKRONG will be TOKO_BAJU, KOPI NONGKRONG, KAKI LIMA, KOPI NONGKRONG)

My Solution

```sql
-- First we need to eliminate all customer that transacted with less than 3 merchants
WITH more2merchant AS (
    SELECT ovo_id
    FROM (
        SELECT DISTINCT ovo_id, merchant_id
        FROM public.ovo_transaction
    ) AS unique_record
    GROUP BY ovo_id
    HAVING COUNT(merchant_id) >= 3
-- Then we sort the previous table and group them by customer ID and transaction date
), grouped_more2merchant AS (
    SELECT txndate, merchant_name, ovo_id
    FROM public.ovo_transaction
    WHERE ovo_id IN (SELECT * FROM more2merchant)
    ORDER BY ovo_id, txndate
-- We will ignore the consecutive duplicates. This can makes the searching easier.
), removed_consecutive_duplicate AS (
    SELECT * FROM (
        SELECT *, lag(merchant_name) OVER (ORDER BY ovo_id, txndate) AS prev_merchant
        FROM grouped_more2merchant
    ) new_table
    -- Ignore the duplicate by checking the previous row
    WHERE prev_merchant IS DISTINCT FROM merchant_name
)
```

My Solution (Cont.)

```sql
-- In the end, we save the next 2 row on new columns to check the order
SELECT *
FROM (
    SELECT
        ovo_id,
        merchant_name,
        lead(merchant_name) OVER (ORDER BY ovo_id, txndate) AS next_merchant,
        lead(merchant_name, 2) OVER (ORDER BY ovo_id, txndate) AS next2_merchant
    FROM removed_consecutive_duplicate
) next_table
-- Check the order if it matches
WHERE merchant_name = 'TOKO BAJU'
AND next_merchant = 'KOPI NONGKRONG'
AND next2_merchant = 'KAKI LIMA';
```

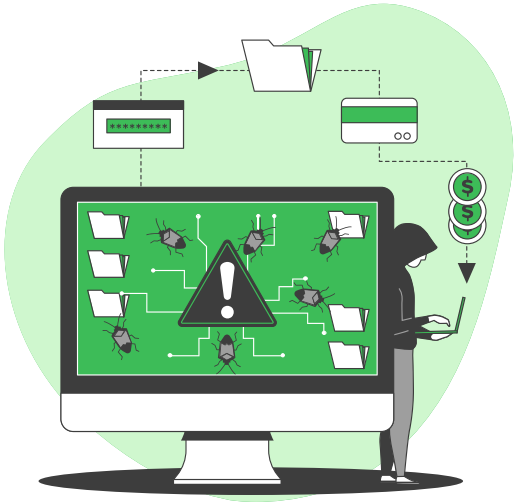
Result

ovo_id	merchant_name	next_merchant	next2_merchant
0001018008303948	TOKO BAJU	KOPI NONGKRONG	KAKI LIMA

The .csv file for this submission is attached along with this submission

Code for Copy-Paste

The .sql file is available along with this submission, See task5.sql



Thanks!

That's all my submission for OVO
Business Intelligence Technical Test

Iqrar Agalosi Nureyza
misteriqrar@gmail.com
+62 823 8506 5800