Catching Fraud

1. Analysis of the query

```
WITH processed_users -- Creation of a subtable
     AS (SELECT LEFT(u.phone_country, 2) AS short_phone_country, --Selection
of the first two characters of the phone number
                u.id
         FROM   users u)
SELECT t.user_id,
       t.merchant_country,
        Sum(t.amount / fx.rate / Power(10, cd.exponent)) AS amount -- Convert
amount to Euro
FROM   transactions t
       JOIN fx_rates fx
         ON ( fx.ccy = t.currency
                AND fx.base_ccy = 'EUR' ) -- Get the exchange rate info to
convert all amounts in Euros
       JOIN currency_details cd
         ON cd.currency = t.currency
       JOIN processed_users pu
         ON pu.id = t.user_id
WHERE  t.source = 'GAIA'
          AND pu.short_phone_country = t.merchant_country -- Filter to get
transaction in the country of the phone of the user
GROUP  BY t.user_id,
          t.merchant_country
ORDER  BY amount DESC;
```

The goal of this query is to get the amount of money spent, in Euros, by users in the country of their phone. It sorts amounts from the highest to the lowest, filtering only expenses coming from source GAIA.

The query itself is working as it is correctly written. But it give us no results as the condition:

```
AND pu.short_phone_country = t.merchant_country
```

doesn't send us anything as short_phone_country has only two characters, while merchant_country has three.

We fix the query :

```
WITH processed_users -- Creation of a subtable
     AS (SELECT LEFT(u.phone_country, 2) AS short_phone_country,
                u.id
         FROM   users u)
SELECT t.user_id,
       t.merchant_country,
          Sum(t.amount / fx.rate / Power(10, cd.exponent)) AS amount FROM
transactions t
       JOIN fx_rates fx
         ON ( fx.ccy = t.currency
              AND fx.base_ccy = 'EUR' )
       JOIN currency_details cd
```

```
        ON cd.currency = t.currency
      JOIN processed_users pu
        ON pu.id = t.user_id
WHERE  t.source = 'GAIA'
       AND pu.short_phone_country = LEFT(t.merchant_country, 2)
GROUP  BY t.user_id,
          t.merchant_country
ORDER  BY amount DESC;
```

USER_ID	MERCHANT_COUNTRY	amount
f4f81f33-7ae1-45f3-9011-3ef47ae51d38	HUN	299317598.1484371
c649559f-8f5e-4a3e-901f-46aeeccde74d	HUN	81117252.77073133
74fdc60d-ee12-47c1-85f0-1c7dd1dbbf16	HUN	76878611.20641503
33930839-d0f3-478c-a807-8b5c1de9f5dd	JPN	21755975.95664639
47a0a032-fd77-4161-aee7-51b0f804d4b2	HUN	16681469.761387784
3c1aa14d-818a-474f-847f-3d24907dd1c7	HUN	12865527.401201654
65815942-9d63-42d9-a64d-85f8b3bef819	HUN	9683314.26919845
9ff7ad28-2b96-4db2-9980-62d48d6d7b3a	HUN	8351035.744316829
b7496a8d-ffd1-4f56-864d-2ad25bafc243	HUN	5242486.349094559
dd672c5e-0cab-4a94-9a93-98334a6b915(	HUN	5024811.514295287
9faf4b80-4720-49b1-b021-591f8d462591	HUN	4327769.661032245

2. Catching users with first succeed purchase over 10 USD

```
with completed_transaction as (
   select t."USER_ID"
,t."CREATED_DATE"
,(t."AMOUNT" / fx."rate" / Power(10, cd.exponent)) AS amount
FROM   transactions t
      JOIN fx_rates fx
        ON ( fx."ccy" = t."CURRENCY"
             AND fx."base_ccy" = 'USD' )
            JOIN currency_details cd
        ON cd."currency" = t."CURRENCY"
where t."STATE" ='COMPLETED'
and (t."AMOUNT" / fx."rate" / Power(10, cd.exponent))>10)
,
first_date as (
   select "USER_ID"
   ,min("CREATED_DATE") AS first_purchase_date
   from completed_transaction ct
   GROUP BY "USER_ID")

SELECT
fd."USER_ID"
,fd."first_purchase_date"
,ct."amount"
from first_date fd
LEFT JOIN completed_transaction ct on (fd."first_purchase_date"=ct."CREATED_DATE"
and fd."USER_ID"=ct."USER_ID")
ORDER  BY ct."amount";
```

USER_ID	first_purchase_date	amount
330b0904-0271-43c1-80e6-e3395f6d888	2016-02-18	10.000711361205937
93dd18e3-c631-4d29-bd41-bf67525bc037	2017-09-11	10.000711361205937
0ad6e671-7346-44c1-8216-6a78eed73a5d	2018-07-17	10.018601078143435
8b810569-016f-4a08-9094-a1d49068ae4e	2016-03-22	10.038592843634747
8053054c-a220-49d2-8c80-6de5984eb26	2017-02-07	10.061052777627093
d8760af4-7e45-4f72-9c6c-0964d0f33157	2018-06-14	10.07647432606356
01589ac2-05bd-4414-b9fb-f9793e1171c2	2018-06-15	10.078033457420558
397965b5-03c6-4627-afd8-faac3fb54202	2018-07-13	10.103504477110754
9818f617-fb92-42f9-9862-e471ac890a17	2017-01-08	10.129508401463893
714661e3-b09b-42ef-a7b8-14a90bd455f5	2016-08-26	10.15981358740694

3. Fraudster

First we will analyze what are the characteristics of a Fraudster

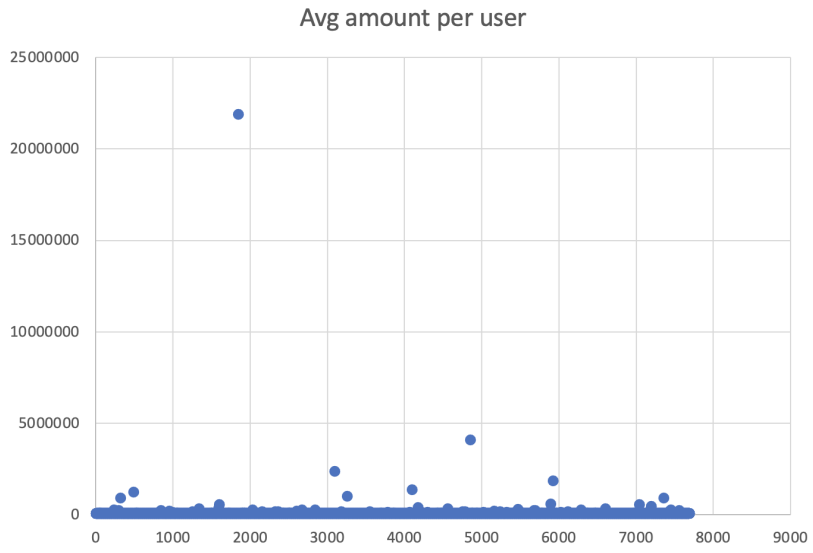
a) Avg amount of purchase per day and per type

	Avg Action movment per day		
TYPE	FRAUD = NO	FRAUD = YES	Diference
ATM	1,296	2,364	45%
BANK_TRANSFER	1,126	1,417	21%
CARD_PAYMENT	2,172	2,475	12%
P2P	1,776	1,546	-15%
TOPUP	1,481	1,815	18%

Frauders use to have 50% more ATM movement than normal people

b) Avg amount per movement per type in USD

We have some outliers in the No fraudsters list that could drive us to uncorrect numbers for average.
We will eliminate those outliers from the analisis



But we are not sure that outliers could be fraudulent people.

TYPE	FRAUD = NO	FRAUD = YES	Diference
ATM	280	246	-14%
BANK_TRANSFER	440	658	33%
CARD_PAYMENT	122	66	-85%
P2P	177	453	61%
TOPUP	274	448	39%

c) Standard deviation between movements

TYPE	FRAUD = NO	FRAUD = YES	Diference
ATM	248	112	-121%
BANK_TRANSFER	480	546	12%
CARD_PAYMENT	194	108	-79%
P2P	187	153	-23%
TOPUP	338	688	51%

d) Are movements done in the same country than the phone number ?

fraudsters	avg_count_transaction	share_transaction_same_country
FALSE	83.0402943237561317	0.29657237755747143182
TRUE	38.9656652360515021	0.81847624260093001213

This is the first highly relevant information.
80% of transactions from frauders come from the same country as their mobile phone. For non Fraudsters it's 20%

e) Amount of the first purchase

fraudsters	avg_first_transaction
FALSE	531.9783087918091
TRUE	248.89446650190567

Now to detect eventual fraudulent people we will create a query with the following condition:

First purchase < 248
Share of transaction same country > 80%
AVG ATM Extraccion > 2,4

Finally we got the following query

```
with full_info as (
select tr."USER_ID"
,tr."CURRENCY"
```

```
,tr."CREATED_DATE"
,tr."TYPE"
,CASE when tr."USER_ID" = f."user_id" then 'TRUE'
          else 'FALSE'
          end as fraudster
,(tr."AMOUNT" / fx."rate" / Power(10, cd.exponent)) AS amount
from transactions tr
      JOIN fx_rates fx
        ON ( fx."ccy" = tr."CURRENCY"
             AND fx."base_ccy" = 'USD' )
            JOIN currency_details cd
        ON cd."currency" = tr."CURRENCY"
      FULL JOIN fraudsters f on tr."USER_ID"=f."user_id")
,
non_fraudulent as (
select
"USER_ID"
,"amount"
,"CREATED_DATE"
,"TYPE"
from full_info
where "fraudster" = 'FALSE'),

average_per_user_per_day as (
select
"USER_ID"
,"CREATED_DATE"
,"TYPE"
,count("amount") as event_day
from non_fraudulent
GROUP BY "USER_ID", "CREATED_DATE","TYPE"),

average_per_usertype as (
select
"USER_ID",
"TYPE",
avg("event_day") as avg_event
from average_per_user_per_day
GROUP BY ("USER_ID","TYPE")),

processed_users -- Creation of a subtable
     AS (SELECT LEFT(u."PHONE_COUNTRY", 2) AS short_phone_country, --Selection of
the first two characters of the phone number
              u."ID" -- Selection of the User ID
        FROM   users u ),
```

```
full_information as (
select tr."USER_ID"
,tr."CURRENCY"
,tr."AMOUNT"
,tr."CREATED_DATE"
,tr."TYPE"
,CASE when tr."USER_ID" = f."user_id" then 'TRUE'
          else 'FALSE'
          end as fraudsters
,  CASE  when  LEFT(tr."MERCHANT_COUNTRY",  2)=pu."short_phone_country"  THEN  1  ELSE  0
end as same_country
from transactions tr
FULL JOIN fraudsters f on tr."USER_ID"=f."user_id"
JOIN processed_users pu on pu."ID"= tr."USER_ID"
where tr."MERCHANT_COUNTRY" is not NULL),

country_share as (
select
"USER_ID"
,"fraudsters"
,count("same_country") as avg_count_transaction
,avg("same_country") as share_transaction_same_country
from full_information
group by ("USER_ID", "fraudsters")),

completed_transaction as (
   select t."USER_ID"
,t."CREATED_DATE"
,CASE when t."USER_ID" = f."user_id" then 'TRUE'
          else 'FALSE'
          end as fraudsters
,(t."AMOUNT" / fx."rate" / Power(10, cd.exponent)) AS amount
FROM   transactions t
      JOIN fx_rates fx
        ON ( fx."ccy" = t."CURRENCY"
             AND fx."base_ccy" = 'USD' )
           JOIN currency_details cd
        ON cd."currency" = t."CURRENCY"
            FULL JOIN fraudsters f on t."USER_ID"=f."user_id"
where t."STATE" ='COMPLETED')
,
first_date as (
   select "USER_ID"
   ,"fraudsters"
```

```
   ,min("CREATED_DATE") AS first_purchase_date
   from completed_transaction ct
   GROUP BY "USER_ID","fraudsters")
,
first_transaction_amount as (
SELECT
fd."USER_ID"
,fd."fraudsters"
,fd."first_purchase_date"
,ct."amount" as "first_transaction_amount"
from first_date fd
LEFT JOIN completed_transaction ct on (fd."first_purchase_date"=ct."CREATED_DATE"
and fd."USER_ID"=ct."USER_ID")
ORDER BY ct."amount"),


final_table as (SELECT
nf."USER_ID"
,fa."first_transaction_amount"
,cs."share_transaction_same_country"
, au.avg_event as ATM_event_per_day
from  non_fraudulent nf
join first_transaction_amount fa on nf."USER_ID"=fa."USER_ID"
join country_share cs on  nf."USER_ID"=cs."USER_ID"
join average_per_usertype au on(nf."USER_ID"=au."USER_ID" and au."TYPE"='ATM'))

select distinct
"USER_ID"
,"first_transaction_amount"
,"share_transaction_same_country"
,"atm_event_per_day"
from final_table
where "first_transaction_amount" < 248
and "share_transaction_same_country" > 0.8
and "atm_event_per_day" > 2.4
```

The query could have been simplified a lot.
And the result for this query identify the following user ID

USER_ID	first_transaction_amount	share_transaction_same_country	atm_event_per_day
11443c92-b76c-4e2a-bf2e-c56fe9f015e4	8.490339896731726	0.84615384615384615385	2.6666666666666667
27c7e90f-9bb5-4051-9210-ed721a67cab1	8.490339896731726	1.00000000000000000000	3.0000000000000000
418c5c05-ccc0-40e5-bcd0-e7e66371f4b8	8.490339896731726	1.00000000000000000000	3.0000000000000000
445b059a-f2a8-41df-8d99-0bff4747e30b	9.812451051323864	1.00000000000000000000	5.0000000000000000
6096f0bb-2271-45bb-bc8c-545c05d5dce2	8.490339896731726	1.00000000000000000000	3.0000000000000000
89608875-0d0d-4fa2-86dd-34087e10f5bd	7.576296485762074	0.86842105263157894737	2.5000000000000000
946fc2b9-d7dc-4402-b4f1-04e5b3023f75	5.95172826760894	1.00000000000000000000	3.0000000000000000
bb6fe4e9-ef5c-4133-acf9-f180843b9195	34.093334185929336	0.81250000000000000000	3.0000000000000000
bb6fe4e9-ef5c-4133-acf9-f180843b9195	37.881482428810365	0.81250000000000000000	3.0000000000000000
d1bbd4b5-6bad-4849-8743-30006daf531b	84.90339896731724	1.00000000000000000000	2.6666666666666667
d1bbd4b5-6bad-4849-8743-30006daf531b	169.8067979346345	1.00000000000000000000	2.6666666666666667

List of user ID probably fraudulent:

11443c92-b76c-4e2a-bf2e-c56fe9f015e4
27c7e90f-9bb5-4051-9210-ed721a67cab1
418c5c05-ccc0-40e5-bcd0-e7e66371f4b8
445b059a-f2a8-41df-8d99-0bff4747e30b
6096f0bb-2271-45bb-bc8c-545c05d5dce2

Conclusion

I have used a really basic approach to try to catch fraudulent people, using SQL.
I use a sensitive approach trying to understand what could describe a Fraudulent person, based on specific behaviours.

It would have been more precise to use a Machine learning approach looking for similar people, as we already have a list of fraudulent people (Labelized Data).
I didn't have enough time to do the exercise. But in a more professional approach I would definitely use a logistic regression to get more accurate results on Fraudulent people.