1. Transactions:

- Query on the same users are using the same device fingerprint or not

```sql
-- check if many users are using the same device fingerprint
select userID,
    count(*) as txn_count,
    sum(case when userChargeAmount = 0 then 1 else 0 end) as zero_amount_txns,
    min(reqDate) as first_txn,
    max(reqDate) as last_txn
from transaction
group by userID
having txn_count >= 5 and (zero_amount_txns >= 3);
```

- 8 users made more than 5 actions or made no payment in a short periods
- Next action: we will check whether they used the same IP or device model or not?

```sql
select userID, deviceID, platform, osVer, deviceModel, userIP
from transaction
where userID in ('0446c341a9756cede989fb03a30203a8', 'acd6b370d43843dc12aab97dfa39583e','4219acfb75a28586e9d86f727badefe1',
'e0c979e16e7cf1eedf92846ccc1bc789','9a1256c59ef6c5b0fde46b6024c9d119',
'477db45ec864a1bfb73a7c08572b297b','d48fe5ad11427ce4422316efe57af8bd','db6b865ee81111bbbae46e3233db71f3');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	deviceID	platform	osVer	deviceModel	userIP
0446c341a9756cede989fb03a30203a8					
0446c341a9756cede989fb03a30203a8					
0446c341a9756cede989fb03a30203a8					
0446c341a9756cede989fb03a30203a8					
0446c341a9756cede989fb03a30203a8					

transaction 79

-> 53 rows returns

0446c341a9756cede989fb03a30203a8 - 10 rows

4219acfb75a28586e9d86f727badefe1 - 5rows

477db45ec864a1bfb73a7c08572b297b - 7 rows

9a1256c59ef6c5b0fde46b6024c9d119 - 9 rows

Acd6b370d43843dc12aab97dfa39583e - 5 rows

D48fe5ad11427ce4422316efe57af8bd - 5 rows

Db6b865ee81111bbbae46e3233db71f3 - 5 rows

E0c979e16e7cf1eedf92846ccc1bc789 - 7 rows

The results reveals that: same deviceID (669083EAA6629B91B3932B6E5F12DD59)x2 with the userIP (171.248.159.136) across the multiple users

→ several users logging in or transacting from the same deviceIP

Notice that the userID such as *0446c341a9756cede989fb03a30203a8, 477db45ec864a1bfb73a7c08572b297b, 9a1256c59ef6c5b0fde46b6024c9d119, E0c979e16e7cf1eedf92846ccc1bc789* are having signs of bot-driven abuse because of being repeated many times.

Besides, it is recorded that almost **deviceID column** has missing or empty values across multiple accounts, meaning that:

- the system couldn't track the device
- the bot can be intentionally hiding device data

Next step: we will find deviceFingerprint shared across multiple accounts?

So why deviceFingerprint? – Because bot often reuse the same device environment to perform like creating fake accounts, claiming promotions repeatedly or triggering actions quickly. Otherwise, normal people often have their own unique phone/device or at least different combinations of device properties.

```sql
-- find deviceFingerprints shared across multiple accounts
SELECT
    CONCAT(COALESCE(deviceID, ''), '_', COALESCE(platform, ''), '_', COALESCE(osVer, ''), '_', COALESCE(deviceModel, '')) AS deviceFingerprint,
    COUNT(DISTINCT userID) AS user_count,
    COUNT(*) AS total_txns,
    MIN(cashbackTime) AS first_cashback,
    MAX(cashbackTime) AS last_cashback
FROM transaction
GROUP BY deviceFingerprint
HAVING user_count >= 3
ORDER BY user_count DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

deviceFingerprint	user_count	total_txns	first_cashback	last_cashback
_platform1__	169	252		
__	157	259	6/1/2022	6/1/2022
_platform3__	4	4		

If more than 3 users share the same exact deviceFingerprint and cashback behavior, they are highly likely automated.

From the result, we can see that '*platform 1*' and '*--------*' are mismatching, it can be bot behavior. However, we need to confirm on some information of each platform.

<u>****Platform 1:</u>

```sql
select
    userID,
    userIP,
    campaignID,
    userChargeAmount,
    reqDate,
    cashbackTime
from transaction
where concat(coalesce(deviceID,''),'_', coalesce(platform, ''),'_', coalesce(osVer,''),'_',coalesce(deviceModel,'')) = '_platform1__'
order by reqDate;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	userIP	campaignID	userChargeAmount	reqDate	cashbackTime
5bd285dff683393e0e66a1af741c53ee		0	5000	6/1/2022	
55518b7065aaad14678e574f5676d830		0	6316	6/1/2022	
57ace8e5331c4d0a08e7b515903d09a1		0	5000	6/1/2022	
490a57e8e31b87c5bbecff8183fcded9		0	5000	6/1/2022	
635eb346e972d56a5aa528bfddc4ad15		0	6316	6/1/2022	
832380e8ed8ec9772ebf3915752fe6aa		0	5000	6/1/2022	

transaction 37

➔ 252 rows returned

From the result, *userIP, campaignID* and *cashbackTime* are almost blank, implying that information is hidden due to bot or automation.

Normally, useIP is a line of numbers (eg: 14.191.228.77) instead of 0/blank, meaning that there are surely signs of bot-driven abuse.

```sql
select
    userIP,
    count(distinct userID) as num_users,
    sum(userChargeAmount) as total_charged,
    min(reqDate) as first_req,
    max(reqDate) as last_req
from transaction
where concat(coalesce(deviceID,''),'_',coalesce(platform,''),'_',coalesce(osVer,''),'_',coalesce(deviceModel,'')) ='_platform1__'
group by userIP
order by total_charged desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userIP	num_users	total_charged	first_req	last_req
	169	19592395	6/1/2022	6/1/2022

Abnormal behavior in *platform 1* also reveals through the amount deducted from transactions. In this case, unknown *userIP* supports the motion that bot is screaming the device and many campaignID and stole a quite large amount of **19592395 (169/252 users)**.

Moreover, within a day, these accounts were created so that the total charged are huge for these quite new accounts.

****Platform '---------':

```
select
    userID,
    userIP,
    campaignID,
    userChargeAmount,
    reqDate,
    cashbackTime
from transaction
where concat(coalesce(deviceID,''),'_', coalesce(platform, ''),'_', coalesce(osVer,''),'_',coalesce(deviceModel,'')) = '___'
order by reqDate;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	userIP	campaignID	userChargeAmount	reqDate	cashbackTime
6144f675fe4190f53ec47737eeb234a3		9080	0	6/1/2022	6/1/2022
c61762d442ed8ad86a429f1f9a8d51bf		12432	0	6/1/2022	6/1/2022
f0656e613ed22e72d3428c885e9da0f5		12433	0	6/1/2022	6/1/2022
f0656e613ed22e72d3428c885e9da0f5		12432	0	6/1/2022	6/1/2022
a3fc828e6680b29fe2ee29cab85505f4		9080	0	6/1/2022	6/1/2022

➔ 259 rows

Like platform 1, userIP is missing, userChargeAmount is recorded of 0, but campaignID are identical at some transactions.

We can suspect the bot due to:

- Lack of userIP is abnormal
- userChargeAmount = 0, it can be that bot is testing or triggering spam the campaigns
- campaignID is repeatedly, it can be that bot is repeating a campaign to gain more rewards

```
select
    userIP,
    count(distinct userID) as num_users,
    sum(userChargeAmount) as total_charged,
    min(reqDate) as first_req,
    max(reqDate) as last_req
from transaction
where concat(coalesce(deviceID,''),'_',coalesce(platform,''),'_',coalesce(osVer,''),'_',coalesce(deviceModel,'')) ='___'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userIP	num_users	total_charged	first_req	last_req
	157	0	6/1/2022	6/1/2022

2. Map card

Case 1: the same users repeated mapping requests

```sql
-- check if many users are mapping requests
select *
from (select
    userID,
    count(*) as total_requests,
    sum(case when requestStatus = 1 then 1 else 0 end) as successful_requests,
    sum(case when requestStatus <> 1 then 1 else 0 end) as failed_requests
from map_card
group by userID
having count(*) >=5 and sum(case when requestStatus = 1 then 1 else 0 end) = 0 ) as sub
order by failed_requests asc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	total_requests	successful_requests	failed_requests
b73ab84ed75bd3d6cfbd97ce95309b9c	6	0	6
366f97ff2f63a807e96c1b2e4eb406ca	6	0	6
4e5d8ec3c12722a68635597a32854fc4	6	0	6
3daad30162a99d253190d45f7d6fb3f8	6	0	6
8aec31a7d0aa4a570a8393173e155fee	6	0	6

➔ 60 rows

Case 2: many users are requesting mapping in many times per day

```sql
-- check if many users are requesting mapping in many times per day
select
    userID,
    reqDate,
    count(*) as total_requests,
    sum(case when requestStatus = 1 then 1 else 0 end) as failed_requests
from map_card
group by userID, reqDate
having count(*) >= 3 and sum(case when requestStatus = 1 then 1 else 0 end) = 0
order by failed_requests asc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	reqDate	total_requests	failed_requests

➔ No sign

Case 3: Analyze according to bimID

```sql
select
    bimID,
    count(distinct userID) as user_count,
    sum(case when requestStatus = 1 then 1 else 0 end) as success_count
from map_card
group by bimID
having user_count >= 2 and success_count = 0
order by user_count asc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

bimID	user_count	success_count
39dc60c6ee7ca21312444166bd2bec8f	2	0
56d342964a02c90b5b5991fe8f750d3e	2	0
5fdf6cb722c99020f6c49f977c493898	2	0
ae204a5d8a7c48165a5ddf44d92e8a89	2	0
afdd0103949e4cf6f1dfec35638cc607	2	0

Result 34

Many users use the same bimID but failed, we observe that there are 8 cases of being attacked

3. User profile

We will detect shared devices in user profile

```sql
-- check if shared devices in user profile
select phone_provider, count(distinct userID) as user_count
from user_profile
group by phone_provider
having count(distinct userID) >=5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

phone_provider	user_count
7	124
7001	5
7002	5
7020	6
7021	6
7023	6
7025	5

Result 4

- ➔ Bot-driven possibility
- ➔ We will find out which userID? (in file excel)

```sql
with filtered_provider as (
    select phone_provider, count(distinct userID) as user_count
    from user_profile
    group by phone_provider
    having count(distinct userID) >5
),
shared_device as
    (select deviceID
    from transaction
    group by deviceID
    having count(distinct userID)>1
),
```

```sql
user_shared_device as (
    select up.userID, up.phone_provider, count(distinct t.deviceID) as share_device_count
    from user_profile up
    join transaction t on up.userID = t.userID
    join shared_device sd on t.deviceID = sd.deviceID
    join filtered_provider fp on up.phone_provider = fp.phone_provider
    group by up.userID, up.phone_provider
)
select userID, phone_provider, share_device_count
from user_shared_device
order by phone_provider, userID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	phone_provider	share_device_count

Result 26

For deviceID, no sign of sharing.

4. Referral_mapping:

In the referrap_mapping, we will observe the userID and refereeId in 2 aspects of bot-driven abuse signs:

Case 1: Abnormally high daily referral volume

→ to find users sending unusual number of referrals in a single day

```sql
-- check the abnormally high daily referral volumne
select userID, reqDate, count(*) as daily_referral_count
from referral_mapcard
group by userID, reqDate
having daily_referral_count > 10; -- no bot detected
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	reqDate	daily_referral_count

(positive) – no sign of bot/automation

Case 2: Daily referral streaks

Bot may send referrals every single day – unlike normal users

```sql
-- check if bot may send referral every single day
select userID, count(distinct reqDate) as active_days
from referral_mapcard
group by userID
having active_days > 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	active_days
0446c341a9756cede989fb03a30203a8	8
2cf86e0ca892dc6712b65541e0523f85	5
5b9008b34e646b27d4f839a5806301f8	4
810ab3f18ace731b6d81d6f96323dea8	4
f45574b03d02356238cbe06268144535	4
fe4ce77fee80c0e298faec18c7e3d83a	4

These referrals are possible of being bot-driven abuse, however, based on the *active_days* we can observe more about user *0446** and *2cf8** because of the high possibility.

5. Transfer

First of all, we will observe the transfer among senders/receivers

```sql
select
    sender,
    count(distinct receiver) as referral_count,
    date(reqDate) as transaction_date,
    count(*) as transaction_count,
    sum(amount) as total_amount
from transfer
group by sender, date(reqDate)
having count(distinct receiver) > 20
order by sender, transaction_date desc; -- transfer among  senders/receivers repeatedly
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sender	referral_count	transaction_date	transaction_count	total_amount
116d80d7669176073fbb1a991a6ce958	42	2022-06-06	269	46168011
256f5d267816966c1060f5117d7e06bb	44	2022-06-08	51	542000
256f5d267816966c1060f5117d7e06bb	79	2022-06-07	136	1667500
256f5d267816966c1060f5117d7e06bb	113	2022-06-06	178	1647000
256f5d267816966c1060f5117d7e06bb	84	2022-06-05	150	1028300
256f5d267816966c1060f5117d7e06bb	88	2022-06-04	143	1216000
256f5d267816966c1060f5117d7e06bb	68	2022-06-03	111	1008000
256f5d267816966c1060f5117d7e06bb	105	2022-06-02	214	2170000

Result 62

➔ 40 rows returns

Noticeably sender 256f*, 4ad9*, 583b*, d11e*

```sql
-- count unique mutual pairs and estimate loss from loop patterns
with mutual_referrals as (
  select
    least(a.sender, a.receiver) as user1,
    greatest(a.sender, a.receiver) as user2
  from transfer a
  join transfer b
    on a.sender = b.receiver
   and a.receiver = b.sender
  group by
    least(a.sender, a.receiver),
    greatest(a.sender, a.receiver)
)
```

```sql
select
  count(*) as mutual_pair_count,
  count(*) * 100000 as estimated_loss
from mutual_referrals;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

mutual_pair_count	estimated_loss
3057	305700000

Since we detect the loop patterns, the estimated_loss is 305,700,000.

6. Campaign

Here, we will analyze based on campaign code ZPI_220801_115 and information from transactions:

1- Analyze transactions patterns for abuse

```sql
-- analyze transaction patterns for abuse
with campaign_transactions as (
    select
        t.userID,
        t.deviceID,
        t.userIP,
        t.reqDate,
        t.amount,
        t.discountAmount,
        t.cashbackTime,
        t.campaignID
    from transaction t
    join campaigninfo ci on t.campaignID = ci.campaignID
    where ci.campaignCode = 'ZPI_220801_115'
        and t.transStatus = 1)
    select
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	deviceID	userIP	transaction_count	first_transaction	last_transaction	total_amount	total_discount	cashback_count
43925d969c461257782aa7cd5bab21ff			4	6/1/2022	6/1/2022	200000	200000	1
ec7659ba7e522e3f7e44c62411c96251			4	6/1/2022	6/1/2022	200000	200000	1
20e4b79d8f9c091aa7f89396c87ae22f			3	6/1/2022	6/1/2022	150000	150000	1
95e42a212b759fe73480fa4900f1404e	aa9319...	113.1...	3	6/1/2022	6/1/2022	624290	60000	1
05e3ffb2bc6608534cbc44dbb0e376ee			3	6/1/2022	6/1/2022	150000	150000	1
0b6f34720d5a67b2eee9b22a986684c0			3	6/1/2022	6/1/2022	150000	150000	1
974f839cfdc8219e19ea351e9d7a07b0			3	6/1/2022	6/1/2022	150000	150000	1
9e33664d68a6bdf48bdfc5d769e63828			3	6/1/2022	6/1/2022	150000	150000	1
3771972b52fe4c08d345d61fc383b955	2e2948...	42.11...	3	6/1/2022	6/1/2022	251275	70000	1

Result 70

```
234        select
235            userID,
236            deviceID,
237            userIP,
238            count(*) as transaction_count,
239            min(reqDate) as first_transaction,
240            max(reqDate) as last_transaction,
241            sum(amount) as total_amount,
242            sum(discountAmount) as total_discount,
243            count(distinct cashbackTime) as cashback_count
244        from campaign_transactions
245        group by userID, deviceID, userIP
246        having count(*) >1
247        order by transaction_count desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

userID	deviceID	userIP	transaction_count	first_transaction	last_transaction	total_amount	total_discount	cashback_count
43925d969c461257782aa7cd5bab21ff			4	6/1/2022	6/1/2022	200000	200000	1
ec7659ba7e522e3f7e44c62411c96251			4	6/1/2022	6/1/2022	200000	200000	1
20e4b79d8f9c091aa7f89396c87ae22f			3	6/1/2022	6/1/2022	150000	150000	1
95e42a212b759fe73480fa4900f1404e	aa9319...	113.1...	3	6/1/2022	6/1/2022	624290	60000	1
05e3ffb2bc6608534cbc44dbb0e376ee			3	6/1/2022	6/1/2022	150000	150000	1
0b6f34720d5a67b2eee9b22a986684c0			3	6/1/2022	6/1/2022	150000	150000	1
974f839cfdc8219e19ea351e9d7a07b0			3	6/1/2022	6/1/2022	150000	150000	1
9e33664d68a6bdf48bdfc5d769e63828			3	6/1/2022	6/1/2022	150000	150000	1

Result 70

Output

Action Output

#	Time	Action	Message
84	16:44:51	SELECT ci.campaignCode -- Test if this column is accessible FROM campaign_info ci WHERE ci.campaig...	Error Code: 1146. Table 'mapping_user.campaign_info' doesn't exist
85	16:45:55	with campaign_transactions as (select t.userID, t.deviceID, t.userIP, t.reqDate, t.amount, ...	56 row(s) returned

Then we will calculate total amount for suspended abusive transactions

```
250 • with campaign_transactions as (
251         select
252             t.userID,
253             t.deviceID,
254             t.userIP,
255             t.reqDate,
256             t.amount,
257             t.discountAmount,
258             t.cashbackTime,
259             t.campaignID
260         from transaction t
261         join campaigninfo ci on t.campaignID = ci.campaignID
262         where ci.campaignCode = 'ZPI_220802_115'
263         and t.transStatus = 1
```

```sql
),
suspected_abuse as (
    select
        userID,
        count(*) as transaction_amount,
        coalesce(sum(amount),0) as total_amount,
        coalesce(sum(discountAmount),0) as total_discount
    from transaction
    group by userID
    having count(*) > 10)
select
    sum(transaction_amount) as total_abusive_transactions,
    sum(total_amount) as total_abusive_amount,
    sum(total_discount) as total_abusive_discount
from suspected_abuse;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_abusive_transactions	total_abusive_amount	total_abusive_discount
28	472000	0

While running the campaign, there are 28 transactions having sign of being abusive, the estimated loss is 472,000.

```sql
-- 3. check deviceID and IP patterns
select
    deviceID,
    userIP,
    count(distinct userID) as unique_users,
    count(*) as transaction_count
from transaction t
join campaigninfo ci on t.campaignID = ci.campaignID
where ci.campaignCode = 'ZPI_220801_115'
    and transStatus = 1
group by deviceID, userIP
having count(distinct userID) > 1
order by transaction_count desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

deviceID	userIP	unique_users	transaction_count
		127	211