DKD

IRONHACK
PAYMENTS

Exploratory Data Analysis and Data Quality Analysis

Project 1

Group DKDencia

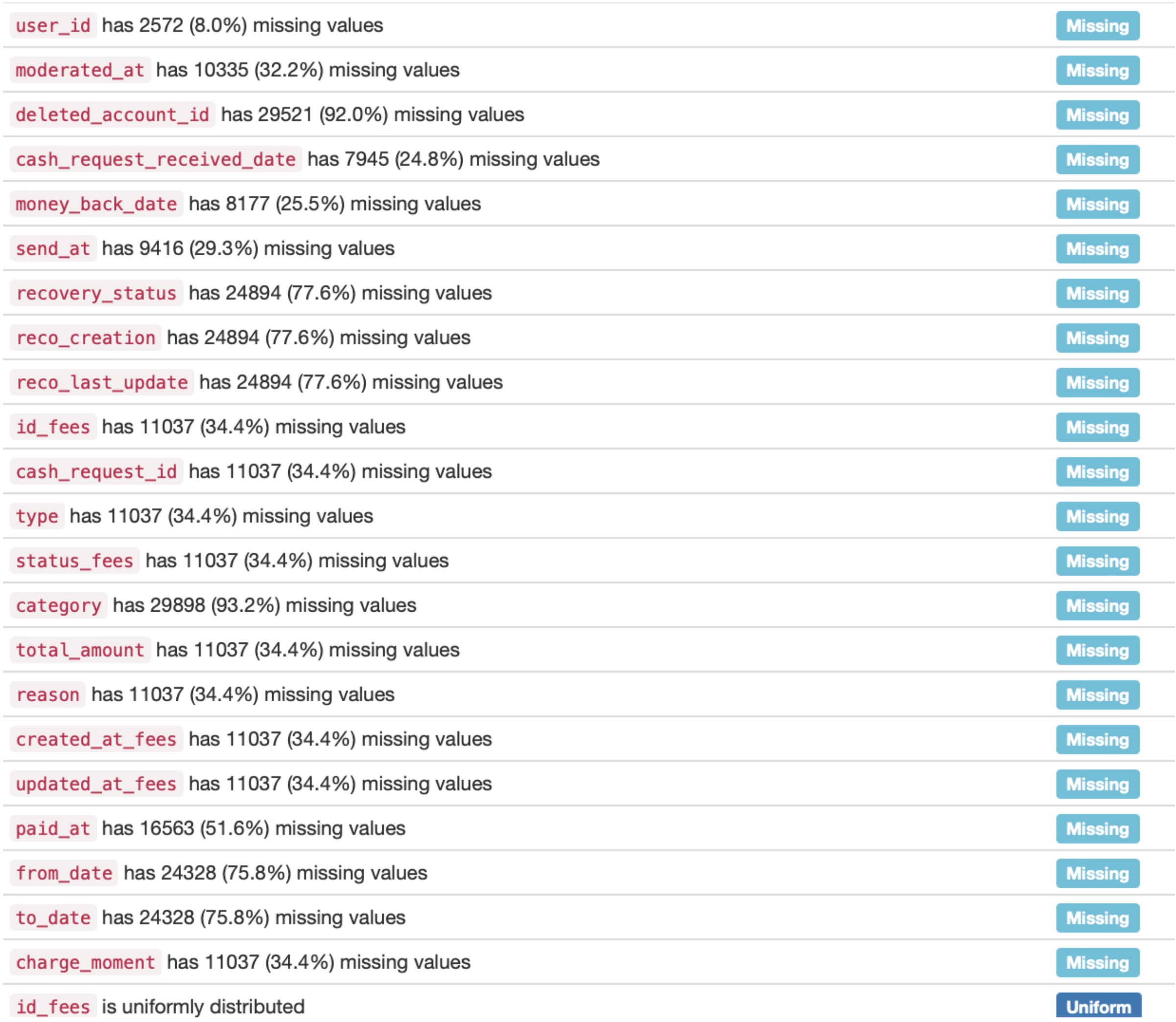
Mónica López Calero
María Pérez Cano
Verónica Santamaría

ÍNDICE

DATA QUALITY ANALYSIS

OVERVIEW

Preliminary visualization of results of the variables included in the database of cash requests received.



user_id has 2572 (8.0%) missing values	Missing
moderated_at has 10335 (32.2%) missing values	Missing
deleted_account_id has 29521 (92.0%) missing values	Missing
cash_request_received_date has 7945 (24.8%) missing values	Missing
money_back_date has 8177 (25.5%) missing values	Missing
send_at has 9416 (29.3%) missing values	Missing
recovery_status has 24894 (77.6%) missing values	Missing
reco_creation has 24894 (77.6%) missing values	Missing
reco_last_update has 24894 (77.6%) missing values	Missing
id_fees has 11037 (34.4%) missing values	Missing
cash_request_id has 11037 (34.4%) missing values	Missing
type has 11037 (34.4%) missing values	Missing
status_fees has 11037 (34.4%) missing values	Missing
category has 29898 (93.2%) missing values	Missing
total_amount has 11037 (34.4%) missing values	Missing
reason has 11037 (34.4%) missing values	Missing
created_at_fees has 11037 (34.4%) missing values	Missing
updated_at_fees has 11037 (34.4%) missing values	Missing
paid_at has 16563 (51.6%) missing values	Missing
from_date has 24328 (75.8%) missing values	Missing
to_date has 24328 (75.8%) missing values	Missing
charge_moment has 11037 (34.4%) missing values	Missing
id_fees is uniformly distributed	Uniform

For **df_cr** we have identified the following NaN values:

- Missing cells 116.261, about (%) 30.3% of our data for this function.
- They were missing in:
    - ***cash_request_received_date***: There are 7.681 transactions without "*cash_request_received_date*", nevertheless, we have decided not to remove these. NaNs are mainly found in status like "*canceled*", "*rejected*", "*transaction_declined*" and "*direct_debit_rejected*". However, there are 3 transactions where status is "*active*" (with "*id*" 15603, 25015, 26297) and 986 transactions as status "*money_back*", which shouldn't

occur and might signal there was an error in inputting the date within the system.

  - ***money_back_date***: we observed this variable is missing 7.427 values (31%) and determined these do not need to be removed from our data analysis. We have identified there are 3 transactions with status "*money_back*" but without "*money_back_date*", which we believe might be an error. We could remove just the ones with status “money_back” with no timestamp for money_back_date

  - ***user_id***: there are 2.103 transactions without user_id. These won’t be removed since they are directly included in the “*deleted_account_id”* variable.

  - ***send_at***: there are 7.329 transactions without “*sent_at*”. Out of these, 4.403 transactions have status as (“*active”, “direct_debit_sent”, “money_back”*) which seems to be contradicting in and of itself since the definition for all these variables implies that a cash request has been successful and thus, the money sent.

On a different hand, for **df_fees** we have identified the following NaN values:

- Missing cells 50.989, about (%) 18.6%% of our data for this function
- They were missing in: “*category*”, “*paid_at*”, “*from_date*”, “*to_date”*

This makes sense since:

- ***cash_request_id***: there are 4 missing . These will be removed since it is the key to merge with cr.

- ***category***: Only happens when there are incidents. We will know if this is correct when merging with cr dataset

- ***paid_at***: NaN happens when the type is "postpone" (⅓ has paid at) or "incident" (2)

  - 208 NaNs with status “*accepted*” (type postpone)
  - 278 NaNs with status “*rejected*” (type postpone)
  - 4778 NaNs with status “*canceled*” – this is expected, since they cancel the fee, meaning no payment exists.
  - In general, if its is type *postpone*, normally it does not have ***paid_at***.

- ***from_date***: Apply only to postpone fees. – Okay

- ***to_date***: Apply only to postpone fees. – Okay

*NOTE: ALL of the **updated_at** (in cr and fees, dfs are highly concentrated in just one day, which does not make much sense).*

DUPLICATES

No duplicates in **df_cr** neither in **df_fees**.

ZERO OR NEGATIVE VALUES

No Zero or negative values in any of the amounts for **df_cr** or **df_fees**.

INCONSISTENCIES

We found out that there was one cash request with a reimbursement date prior to the cash request date. This is an error which we don't believe can greatly affect our analysis since we are not using the date of reimbursement as a reference,, so we have decided to ignore that transaction (id_cr = 8626).

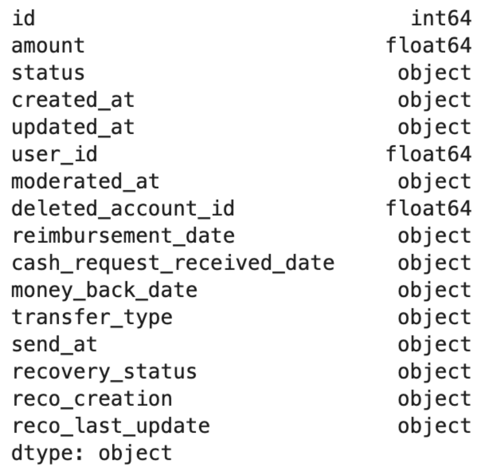
Additionally, We also found that there is inconsistency between user_id and deleted_account_id in one case. We will just ignore it since it is just one instance (id_cr = 280).

DATATYPES

We decided to convert all the *ids* into Integers for consistency, since some of them were automatically transformed into float64 by pandas.

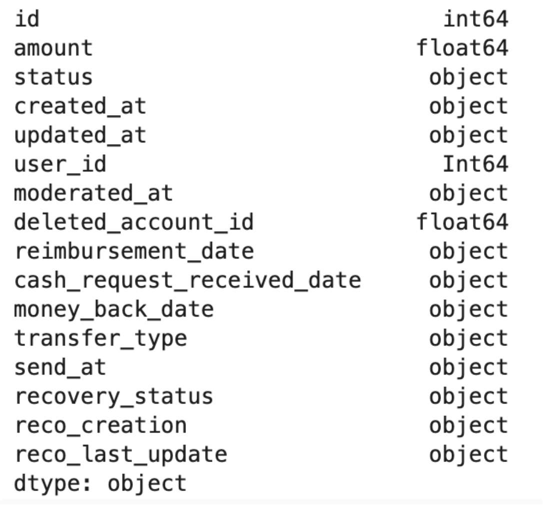
This is what it means for our data:

From



```
id                           int64
amount                     float64
status                      object
created_at                  object
updated_at                  object
user_id                    float64
moderated_at                object
deleted_account_id         float64
reimbursement_date          object
cash_request_received_date  object
money_back_date             object
transfer_type               object
send_at                     object
recovery_status             object
reco_creation               object
reco_last_update            object
dtype: object
```

To



```
id                            int64
amount                      float64
status                       object
created_at                   object
updated_at                   object
user_id                       Int64
moderated_at                 object
deleted_account_id          float64
reimbursement_date           object
cash_request_received_date   object
money_back_date              object
transfer_type                object
send_at                      object
recovery_status              object
reco_creation                object
reco_last_update             object
dtype: object
```

Merging both DataFrames (cash requests and fees)

df_cr	**df_fees**
● Number of rows = 23.970 ● Key variable is ***id_cr***.	● Number of rows = 21.061 ● Key variable is ***id_fees***. ● Count of ***cash_request_id*** = 21.057 ● Number of unique ***cash_request_id*** = 12.933

Since we are doing a left merge on ***id_cr*** = ***cash_request_id_fees***, we would end up with the same number of rows in our new df as in **df_cr** (23.970). Out of these, 21.057 IDs would contain **df_fees** information. However, the length of the new df is 32.094 transactions. This is due since df_fees can have multiple entries for the same cash_request_id_fees. These additional entries have caused 8.124 extra rows to be generated in the new df.

EXPLORATORY DATA ANALYSIS

With this analysis, we seek to explore key statistics, distributions and visualizations to identify patterns and outliers.

During the EDA we found:

- ***amount***: The most frequent cash request amount issued by Ironhack Payments, representing 68.5% of cases, is 100€. Followed by 50, with 22% of cases and 25 with 4,5%.

- ***status***: In 72.5% of the cases studied have approved the cash request and successfully recovered the funds. However, this does not necessarily mean that the company has already received the money, as this status also includes cases where the risk of payment rejection is very low. On the other hand, in 20.5% of cases, the status is "*Rejected*," meaning that the cash request has not been approved.

- ***created_at_cr***: We can observe that the company's early months were slower, with a significant increase in cash requests starting at the end of May 2020, reaching a peak on the 23rd and 24th of October.

- ***reimbursement_date***: Refunds for cash requests are generally processed between the 5th and 7th of each month. This suggests that Ironhack Payments initiates the withdrawal of the amount from the customer's bank account during this particular period.

- ***updated_at_cr:*** Highly concentrated in just one day, on the 18th to 19th of December 2020. This may indicate an automatic control check or update for that data in the system. Unfortunately, we lack further information on this to be able to give a proper explanation.

- ***cash_request_received_date*:** the day with the highest frequency/number of cash requests was on the 27th of October 2020. We see they start to increase by the beginning of June 2020.

- ***money_back_date***: 25% of cash requests do not have mone_back_date. The most frequent ***money_back_date*** corresponds to the beginning of November (1st to 8th) of 2020 with around 4.851 cash requests. From our merged data frame that contains ***money_back_date***, in 23.079 instances, the ***money_back_date*** has ***status_cr*** = *'money_back'*. Nevertheless, in 783 cases, ***status_cr*** = *'direct_debit_rejected'*. This is possible since ***money_back_date*** is either the ***paid_by_card*** date or the date where they considered that the direct debit has low odds to be rejected, yet the possibility of rejection still exists.

  Interestingly, for those records without ***money_back_date***,

  - 6.568 transactions have *'rejected'* status.
  - 1.158 have *'direct_debit_rejected'* status.
  - 191 has *'money_back*". We believe this is specially interesting since the status is money back but we don't have the date.

- ***transfer_type***: 60.7% are *'instant'* while 39.3% are *'regular'*. Meaning, for most users for IronHack Payments, there is a certain urgency in the reception of the money.

- ***send_at:*** we identified that 29.3% timestamps are missing. The most frequent send_at is around 6th of Oct 2020. From the not missing values we have the majority with status 'money_back' (17290), rejected (3862), and direct_debit_rejected (1294). From the null values, we have the majority with 'money_back' status (5980), followed by 'rejected' (2706) and 'direct_debit_rejected' (647).

- ***recovery_status***: There are 77.6% of cash requests without ***recovery_status***, meaning that they probably weren't affected by any incident. The most typical ***recovery_status*** is *'completed'* (5.167 transactions, about 16.1% of the cases), followed by *'pending'* (1.996 transactions, about 6.2%)

- ***reco_creation:*** The most frequent time for the ***reco_creation*** was around the 8th and 9th of November 2020 with 356 cases.

- ***reco_last_update***: nothing remarkable.

— Fees df —

- ***id_fees***: There are 21.057 distinct *id fees*, missing values 34.4%, meaning that 34.4% (11037) of cash requests did not have additional fees. FYI, We know that one cash_request can have several fees, thus several id_fees.

- ***cash_request_id***: There are 12933 distinct values, this is how we know that one cash_request_id can have several id_fees. Nothing else remarkable since it was merged with id_cr.

- ***type***: most frequent type of fee is 'postpone': (4987 - 90,18%) 'incident' (442 - 7,99%), 'instant_payment' (101 - 1,82%)

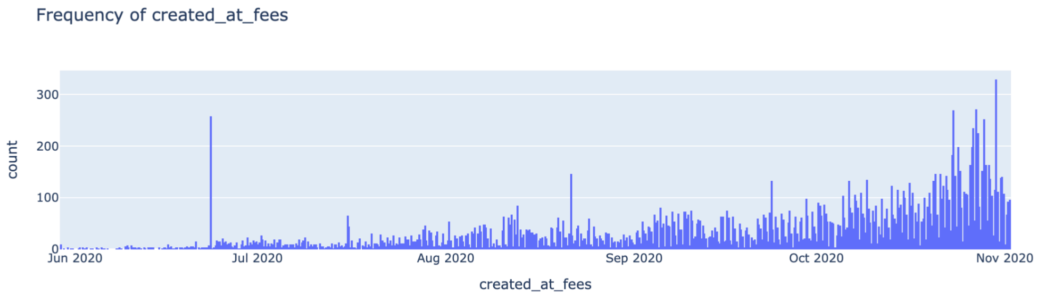
- ***status_fees:*** most frequent 'accepted: 308 - 5.57%' 'cancelled: 4848 - 87.58%' 'confirmed: 86 - 1.55%', 'rejected: 278 - 5.03%'

- ***category***: This is just for the 'incident' types of fees. There are two. The most frequent is rejected_direct_debit (1599), and 'month_delay_on_payment' (597)

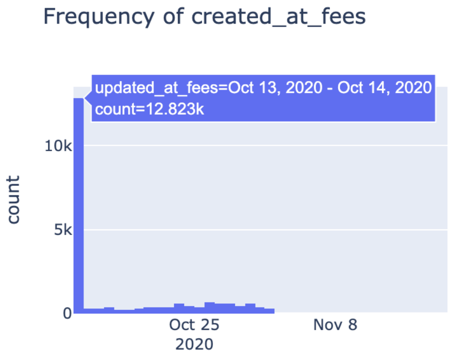
- ***total_amount***: All of the fees (21056) are 5€. (Just except for one of 10€)

- ***reason***: This information we have it on type and category

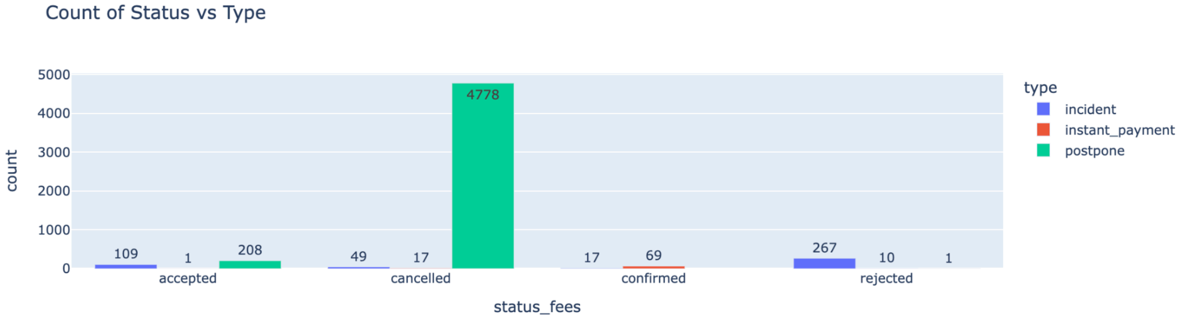
- ***created_at_fees*:** On June 23rd, we can observe there was a peak of 257 created fees. Overall, the fees are evenly distributed until the end of the year, when the number of created fees skyrockets once again.



- ***updated_at_fees*:** Seems that the majority of fees updates are concentrated between the 13th and 14th of October 2020, which may indicate a change in fee policies of automatic control implemented, possibly reflecting a strategic decision.



- **paid_at:** There are 5.526 records identified with missing ***paid_at*** date.

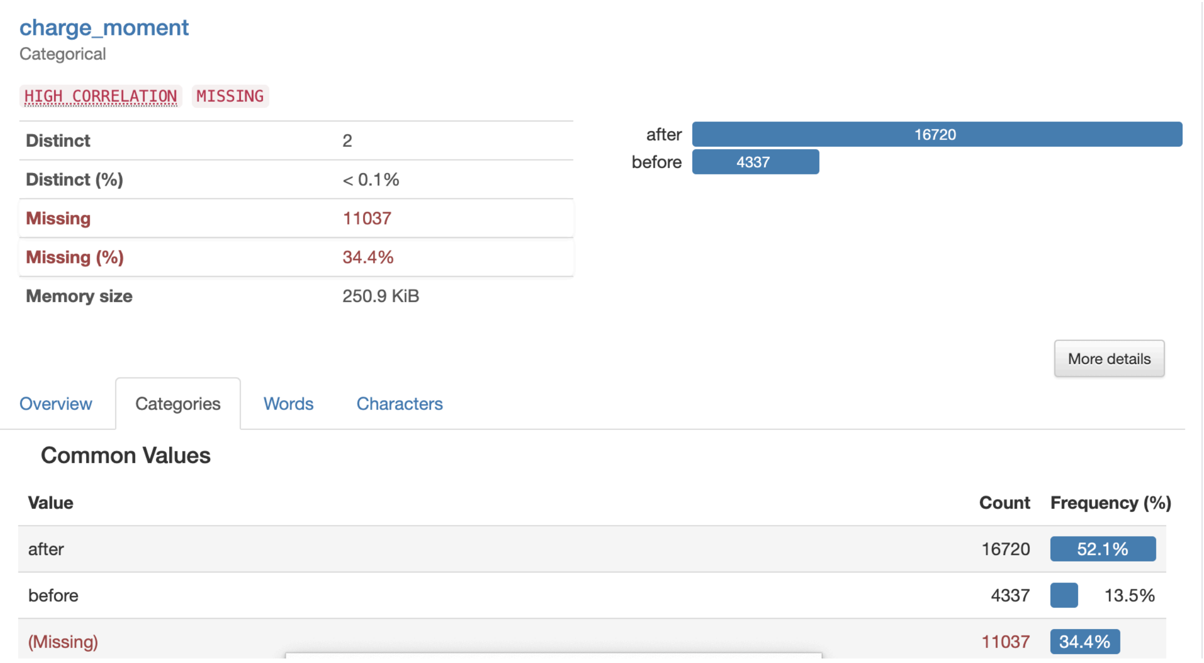


Firstly, we observe that the vast majority of empty ***paid_at*** fields (5.526 cases) correspond to cash request transactions that have been *cancelled* (87.65% of cases). More specifically, these are cash request transactions that attempted to postpone payment and failed in doing so (98.63% of cancelled transactions), which resulted in the cash request reimbursement date remaining unchanged. On the other hand, it is interesting that 208 successfully postponed transactions have not been paid, which may be a log error.

Overall, transactions in ***paid_at*** are distributed the following way: *cancelled* – 87,65%; *accepted* – 5,75%; *rejected* – 5,03%; *confirmed* – 1,55%.

- **from_date** and **to_date**: The ***from_date*** and ***to_date*** parameters are applicable exclusively to postponed fees. Considering the initial reimbursement date of the cash request and the new reimbursement date, it can be observed that out of the 7.766 postponed cases, 2.780 (35.80%) have been paid after 30 days. The most common frequency is followed by 655 cases (8.43%) in which the payment was made after 15 days. The minimum is 0 days and the maximum is 31 days, indicating that postponement is allowed for a maximum of one month only.

- **charge_moment:** A total of 79.40% of the fees are charged *after* (meaning at the time of reimbursement), while 20.60% of the fees are charged *before* (at the time of their creation). This pattern is logical, as an analysis of the types of these fees reveals that both ***instant_payment*** and incident fees always have a ***charge_moment*** *after* and, only 4.337 (55.84%) out of the 7.766 postponed cases are charged *before*.



charge_moment
Categorical

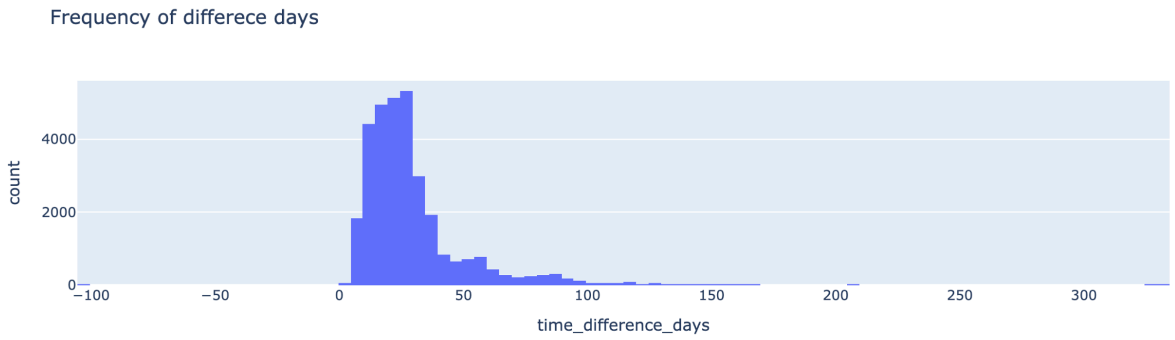
HIGH CORRELATION MISSING

Distinct	2
Distinct (%)	< 0.1%
Missing	11037
Missing (%)	34.4%
Memory size	250.9 KiB

Common Values

Value	Count	Frequency (%)
after	16720	52.1%
before	4337	13.5%
(Missing)	11037	34.4%

We have identified that the ***created_at_cr*** only go up to the end of 2020, but there are ***reimbursement_dates*** up to March 2021. From here, we checked the average payment time between the ***created_at_cr*** and the ***reimbursement_date*** by subtracting the two dates and calculating the average. In this case, we obtained that it usually takes 29 days, on average, to receive the reimbursement, with a median of 24 days. In this way, we determined that the ***reimbursement_dates*** corresponding to the beginning of 2021 correspond to outliers.



On the other hand, analyzing we have identified an error. The id 8626, its ***created_at*** is on the 17.07.2020, its ***reimbursement_date*** is on the 03.04.2020 and the ***cash_request_received_date*** is 21.07.2020. Additionally, we must take into account that it has fees and ***moderated_at***, as well as postponed fees paid on 24.07.2020..

At first, we assume that it may be due to a system error or even, that when having the moderation, as it may have involved manual changes, after paying to postpone the payment of the cash request, the person who moderated the case made a mistake and manually entered the wrong reimbursement date.