Ironhack Payments

Kerem Senler and Carlos R. Vidondo

Part 1: EDA

Step 1: Merging two files into 1 (outer join)

File 1

extract - cash request - data analyst.csv

File 2

extract - fees - data analyst - .csv

```python
import pandas as pd

# Load the files
file1_path = 'extract - cash request - data analyst.csv'
file2_path = 'extract - fees - data analyst - .csv'

# Reading the CSV files
df1 = pd.read_csv(file1_path)
df2 = pd.read_csv(file2_path)

# Performing a full outer join on the 'id' column from df1 and df2
merged_df = pd.merge(df1, df2, on='id', how='outer', suffixes=('_cash_request', '_fees'))

# Saving the merged DataFrame to a CSV file
merged_file_path = 'merged_data.csv'
merged_df.to_csv(merged_file_path, index=False)

# Provide the path to the saved merged file
merged_file_path
```

Merged_data.csv

Step 2 (EDA): Data set structure and values

```python
# Summary statistics of the dataset
data_summary = data.describe(include='all')

# Checking the shape of the dataset
data_shape = data.shape

# Checking for missing values
print ('missing values')
missing_values = data.isnull().sum()

# Display the data summary, shape, and missing values information
data_summary, data_shape, missing_values
```

```
[11 rows x 28 columns],
(26598, 28),
id                              0
amount                       2628
status_cash_request          2628
created_at_cash_request      2628
updated_at_cash_request      2628
user_id                      4731
moderated_at                10563
deleted_account_id          24494
reimbursement_date           2628
cash_request_received_date  10309
money_back_date             10055
transfer_type                2628
send_at                      9957
recovery_status             23268
reco_creation               23268
reco_last_update            23268
cash_request_id              5541
type                         5537
status_fees                  5537
category                    24402
total_amount                 5537
reason                       5537
created_at_fees              5537
updated_at_fees              5537
paid_at                     11067
from_date                   18832
to_date                     18832
charge_moment                5537
dtype: int64)
```

Dataset Structure:
- The dataset has 26,598 rows and 28 columns.
- Columns include details on transactions, user IDs, statuses, dates, amounts, and etc.

Missing Values:
Several columns have missing values.
Notably:
- *deleted_account_id has the most missing values (24,494).
- moderated_at, cash_request_received_date, money_back_date, and others also have significant amounts of missing data.

***Data cleaning needed**

Step 3 (EDA): Data Cleaning

Find out which metrics is most import

→ ‘Amount’ is the most important

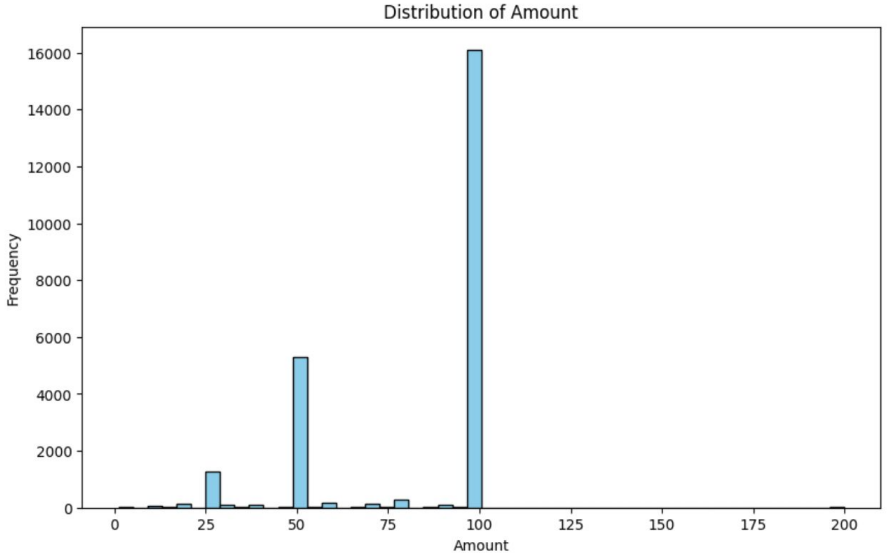
→

```
data_cleaned = data.dropna(subset=['amount']).copy()
data_cleaned.loc[:, 'status_cash_request'] = data_cleaned['status_cash_request'].fillna('unknown')
data_cleaned.to_csv('data_cleaned.csv', index=False)
```

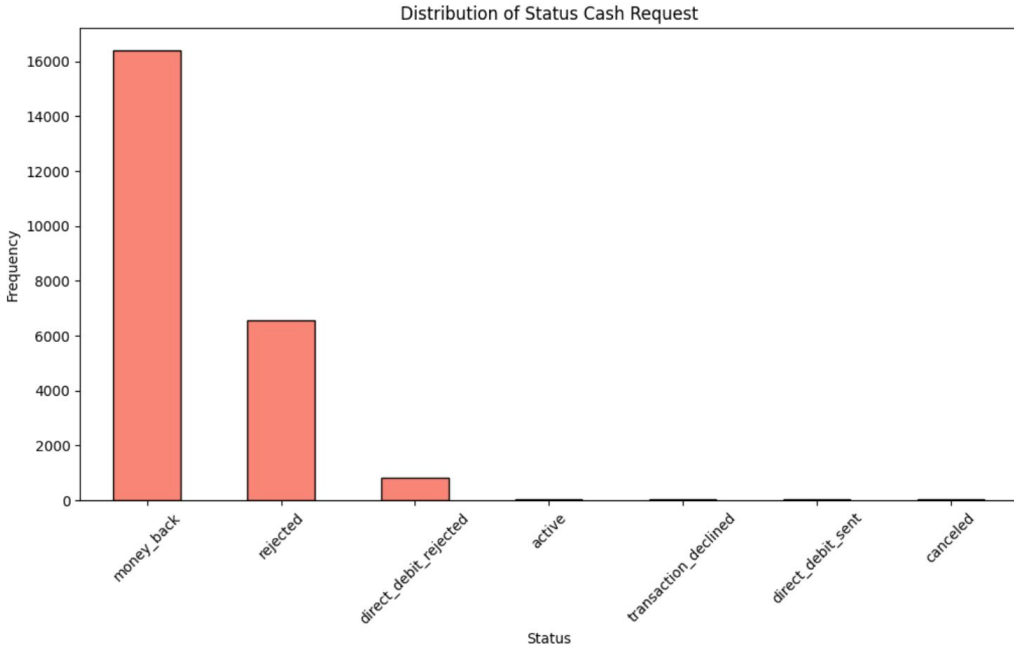
→ data_cleaned.csv

Step 4 (EDA): Visualizations and Distribution Analysis

- Students mainly asked for the full amount (100)



- Most of the money lended are paid back (reimbursed)



Step 5 (EDA) : Creating Cohort(s)

Create cohorts based on **'created_at_cash_request'** metric (group the ones in the same month in **Year+Month** format.

→

Add this data as a **new parameter (cohort_month)** to data_cleaned.csv

→

from_date	to_date	charge_moment	cohort_month
2020-06-26 22:00:00+00	2020-07-26 22:00:00+00	before	2019-12
2020-06-06 22:00:00+00	2020-07-05 21:44:16.477+00	after	2019-12
2020-05-27 22:00:00+00	2020-06-27 03:49:38.153+00	after	2019-12
2020-06-12 22:00:00+00	2020-07-09 22:00:00+00	before	2019-12
NaN	NaN	after	2020-05

Part 2: Metrics to analyze

1. Frequency of Service Usage

Purpose: Understand how often users from each cohort utilize IronHack Payments' cash advance services over time.

Group by

cohort_month
user_id

→ Calculate **average_usage_per_user** based on **cohort_month** →

	cohort_month	average_usage_per_user
0	2019-11	1.000000
1	2019-12	1.121951
2	2020-01	1.106918
3	2020-02	1.068027
4	2020-03	1.131148

Outcome: The table shows the average number of times users in each cohort used the service. For example, users from the cohort of December 2019 used the service on average around 1.12 times.

2. Incident rate

Purpose: Determine the incident rate, specifically focusing on payment incidents, for each cohort. Identify if there are variations in incident rates among different cohorts.

Group by

cohort_month
status_cash_request

Create a new column

Incident (if **status_cash_request = 'rejected'**)

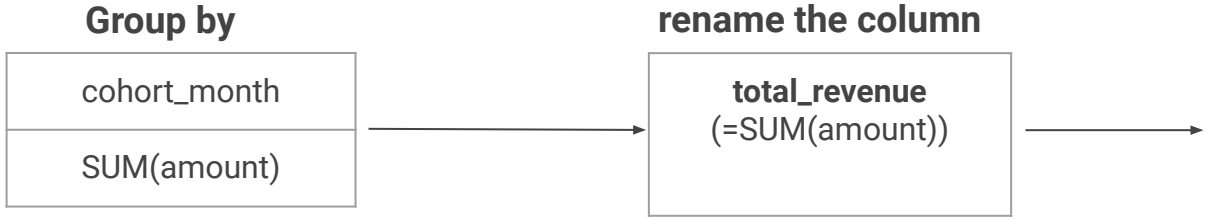
Note: The reason only 'rejected' is counted that other values such as direct_debit_rejected and canceled are way too low in terms of numbers. We prefer to ignore them.

	cohort_month	incident_rate
0	2019-11	0.000000
1	2019-12	0.380623
2	2020-01	0.502242
3	2020-02	0.391304
4	2020-03	0.307377

Outcome: Incident_rate indicates the rate of incidents (e.g., rejected requests) for each cohort. For example, the cohort from December 2019 has an incident rate of approximately 38%.

3. Revenue generated by cohort

Purpose: Calculate the total revenue generated by each cohort over months to assess the financial impact of user behavior.



	cohort_month	total_revenue
0	2019-11	1.0
1	2019-12	27297.0
2	2020-01	21587.0
3	2020-02	16653.0
4	2020-03	23549.0

Outcome: The table shows the total revenue generated by each cohort. For example, the cohort from December 2019 generated a total revenue of 27,297 units (Question: What are these units? USD? EUR?).

4. New Relevant Metric - Retention rate

Purpose: Propose and calculate a new relevant metric that provides additional insights into user behavior or the performance of IronHack Payments' services.

Group by

cohort_month

Create a new metric

Usage_count to identify users who made more than 1 transaction

	cohort_month	retention_rate
0	2019-11	0.000000
1	2019-12	0.117073
2	2020-01	0.081761
3	2020-02	0.061224
4	2020-03	0.125683

Outcome: The table shows the percentage of users from each cohort who made more than one transaction. For instance, in the December 2019 cohort, approximately 11.7% of users made additional transactions after their initial request.

Part 3: Visualizations

Building the code for visualizations

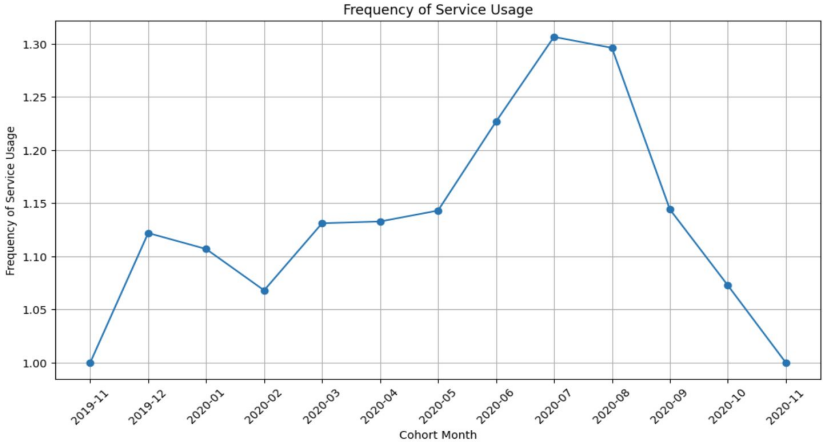
Import matplotlib and use this code structure for each metric.

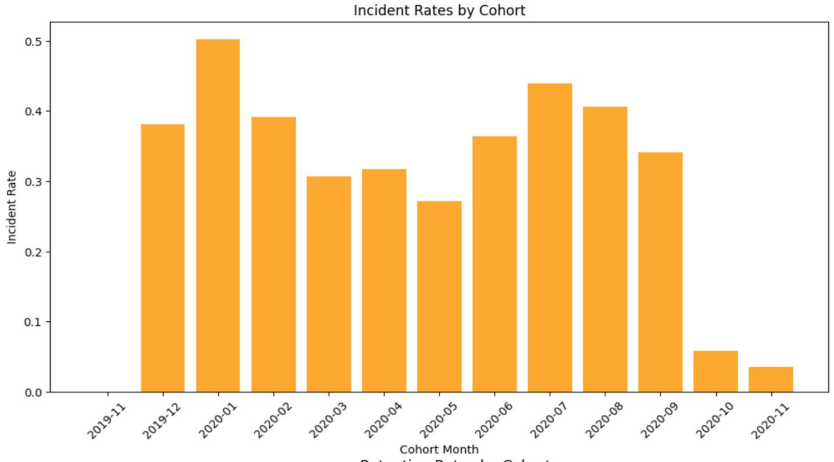
```
# Visualize the four metrics analyzed
import matplotlib.pyplot as plt
```

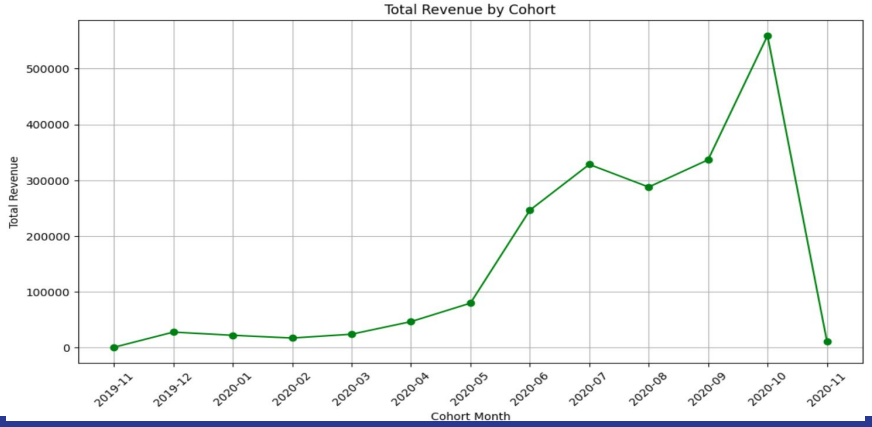
✓ 0.0s

```
# 1. Frequency of service usage
plt.figure(figsize=(12, 6))
plt.plot(cohort_frequency["cohort_month"].astype(str), cohort_frequency["average_usage_per_user"], marker='o')
plt.title("Average Usage per User Over Time")
plt.xlabel("Cohort month")
plt.ylabel("Average User per User")
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

Plotting 4 metrics from the code







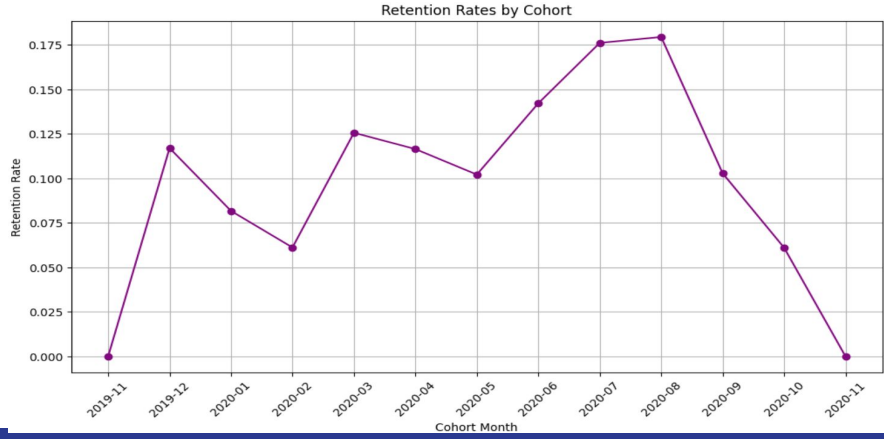
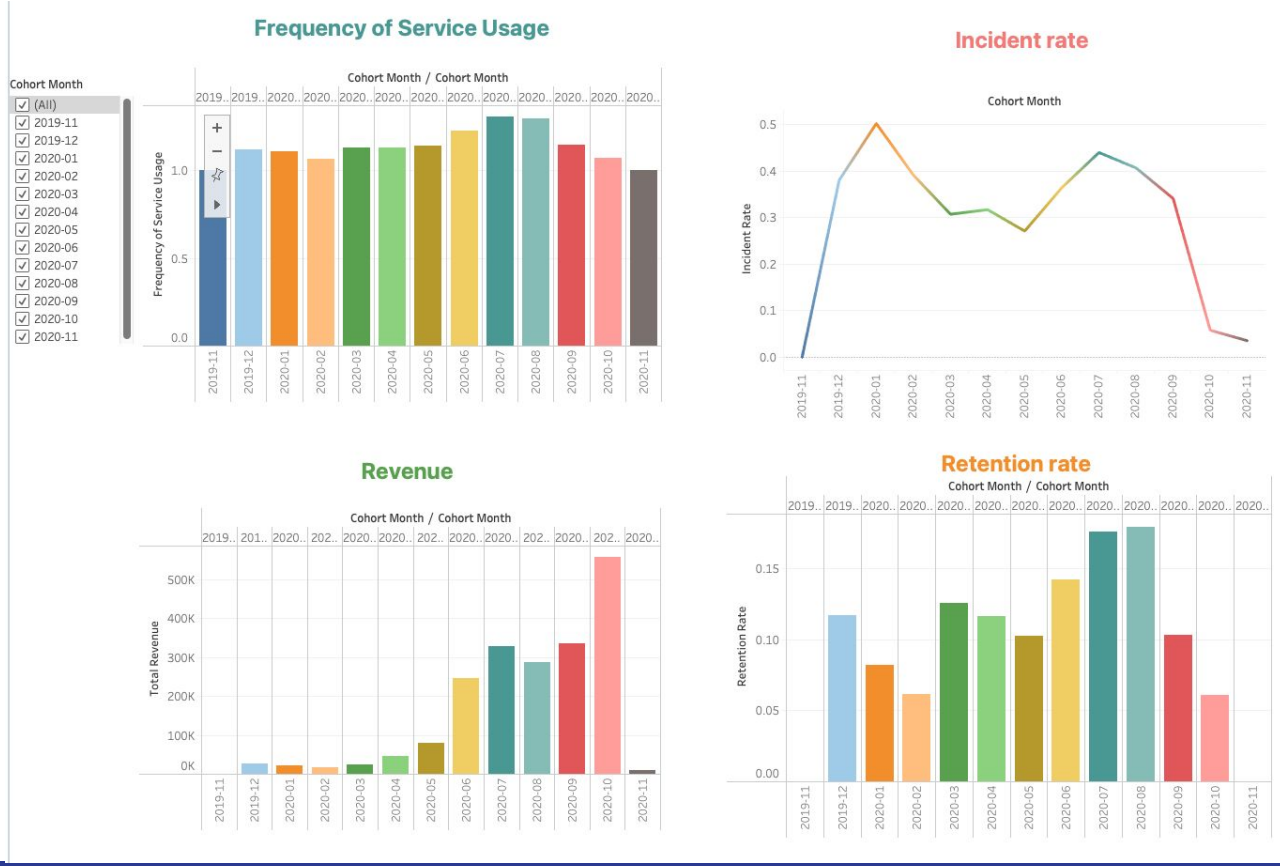


Tableau Dashboard



Part 4: Other deliverables

EDA and Data quality reports

1. Exploratory Data Analysis (EDA) Report

Overview of the Dataset:

- **Total Records:** 26,598
- **Columns:** 28, including transaction amounts, statuses, user IDs, dates, and more.

Key Findings:

- **Distribution of Transaction Amounts:**
  - The distribution is right-skewed, with the majority of transactions being smaller amounts.
  - Most common transaction amounts are concentrated in the lower range (e.g., around $50 to $100).
  - Indicates a preference for smaller cash advances, possibly due to user cautiousness or short-term financial needs.
- **Status of Cash Requests:**
  - Several categories exist for cash request statuses, including `rejected`, `accepted`, and others.
  - The frequency of these statuses varies, with certain statuses being more prevalent, such as `money_back`.
- **Time-Based Patterns:**
  - Users' activity is spread across different months, with some months showing higher user engagement.

2. Data Quality Analysis Report

Data Quality Issues Identified:

- **Missing Values:**
  - Several columns have missing values. Notably:
    - `deleted_account_id` has 24,494 missing values.
    - `moderated_at`, `cash_request_received_date`, `money_back_date` have a significant number of missing entries.
  - **Resolution:** For critical columns like `amount`, rows with missing values were dropped. For categorical columns like `status_cash_request`, missing values were filled with 'unknown' to maintain data integrity.
- **Data Consistency:**
  - The date fields contained time zone information, which was dropped to ensure consistency in date processing.

Actions Taken:

- Rows with missing `amount` were dropped to ensure the reliability of revenue-related analysis.
- Missing values in `status_cash_request` were filled with 'unknown'.
- Time zone information was removed from datetime fields to simplify date analysis.