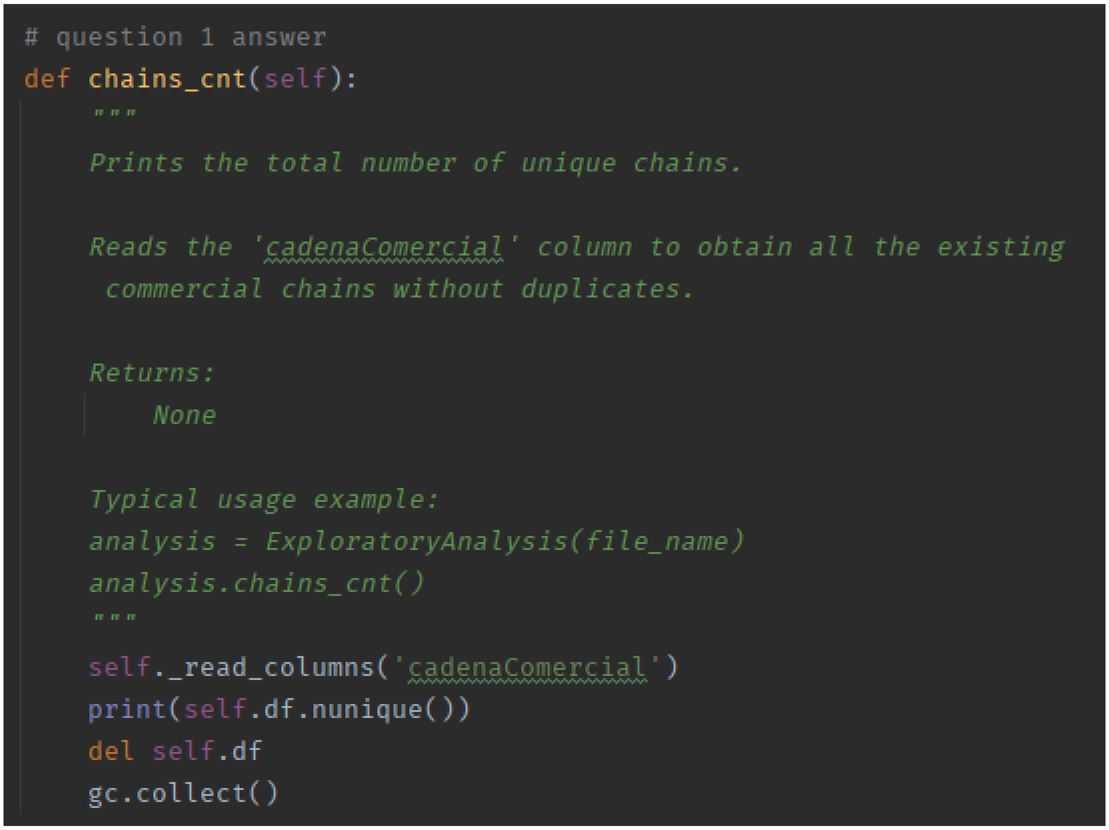
Geovanni Velázquez Medina

Data Engineering Bootcamp Report

This document has the answers for questions 1 through 6 of the data bootcamp. For the script I decided to use OOP since I find it quite comfortable for these types of analysis. I created a method to read in the file every time a method is run. This is because when working with big files in my current job I have to be really efficient with memory use. I do not have the luxury of saving the file in more dataframes.

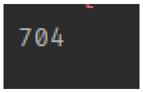
Question 1: How many commercial chains are monitored, and therefore, included in this database?



```
# question 1 answer
def chains_cnt(self):
    """
    Prints the total number of unique chains.

    Reads the 'cadenaComercial' column to obtain all the existing
     commercial chains without duplicates.

    Returns:
        None

    Typical usage example:
    analysis = ExploratoryAnalysis(file_name)
    analysis.chains_cnt()
    """
    self._read_columns('cadenaComercial')
    print(self.df.nunique())
    del self.df
    gc.collect()
```

For this question I used pandas’ “nunique” method which basically returns the total sum of unique values of “cadenaComercial”. This comes after reading in the chain column which is cleaned by removing any null values. I encountered a few rows that had the header instead of any useful info so I removed those as well.

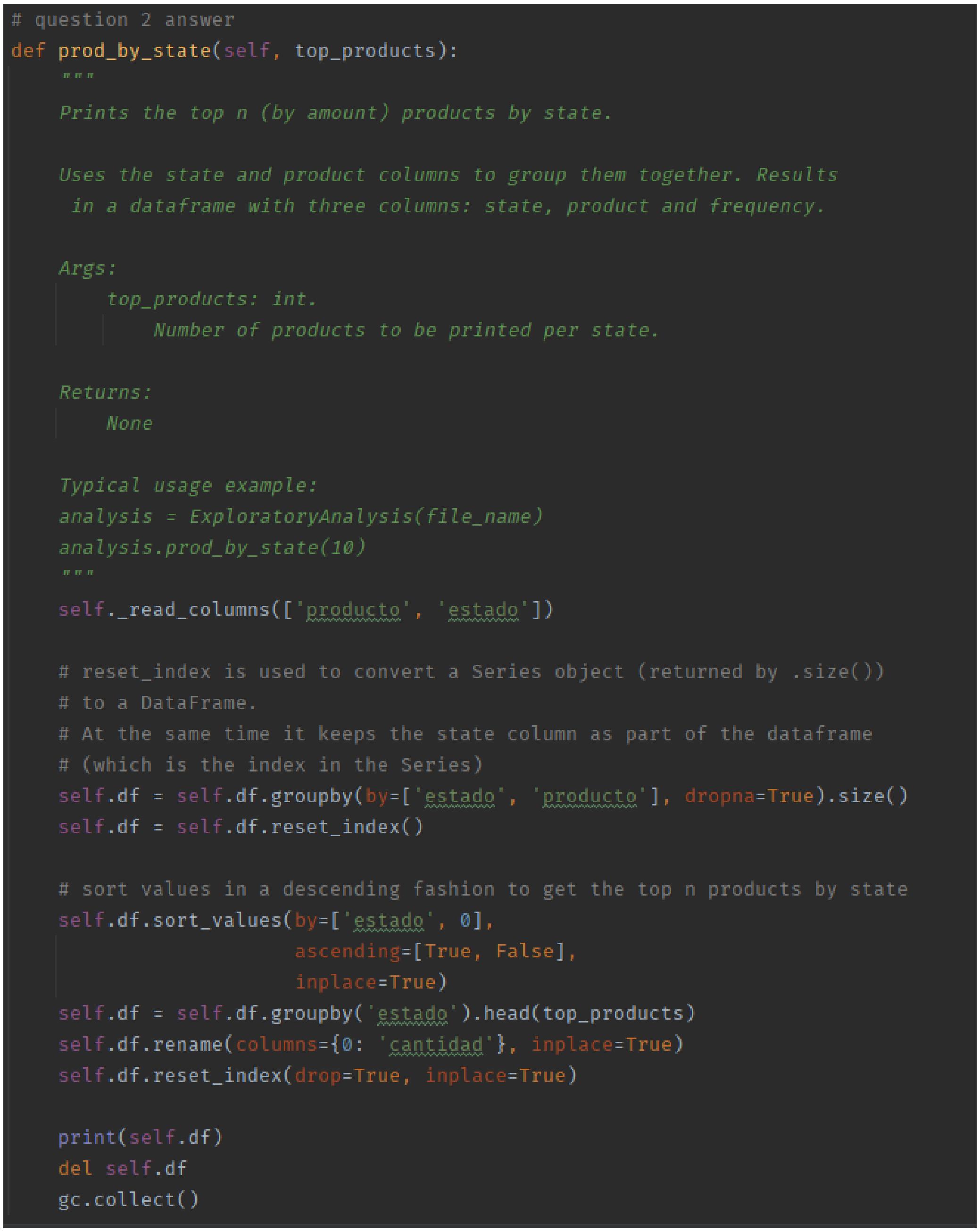
Answer:



```
704
```

Geovanni Velázquez Medina

Question 2: What are the top 10 monitored products by State?

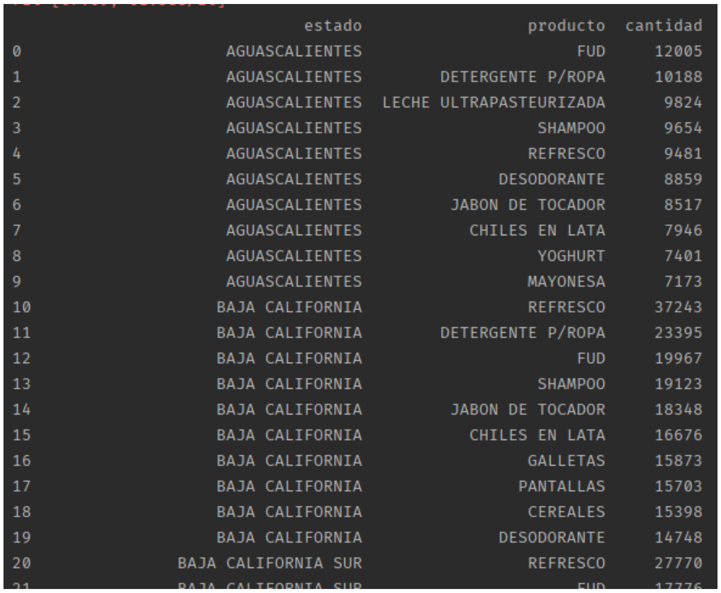


```
# question 2 answer
def prod_by_state(self, top_products):
    """
    Prints the top n (by amount) products by state.

    Uses the state and product columns to group them together. Results
     in a dataframe with three columns: state, product and frequency.

    Args:
        top_products: int.
            Number of products to be printed per state.

    Returns:
        None

    Typical usage example:
    analysis = ExploratoryAnalysis(file_name)
    analysis.prod_by_state(10)
    """
    self._read_columns(['producto', 'estado'])

    # reset_index is used to convert a Series object (returned by .size())
    # to a DataFrame.
    # At the same time it keeps the state column as part of the dataframe
    # (which is the index in the Series)
    self.df = self.df.groupby(by=['estado', 'producto'], dropna=True).size()
    self.df = self.df.reset_index()

    # sort values in a descending fashion to get the top n products by state
    self.df.sort_values(by=['estado', 0],
                        ascending=[True, False],
                        inplace=True)
    self.df = self.df.groupby('estado').head(top_products)
    self.df.rename(columns={0: 'cantidad'}, inplace=True)
    self.df.reset_index(drop=True, inplace=True)

    print(self.df)
    del self.df
    gc.collect()
```

I decided to use “groupby” twice since I had to sort the values after the first use of “groupby” to get the top 10 products.

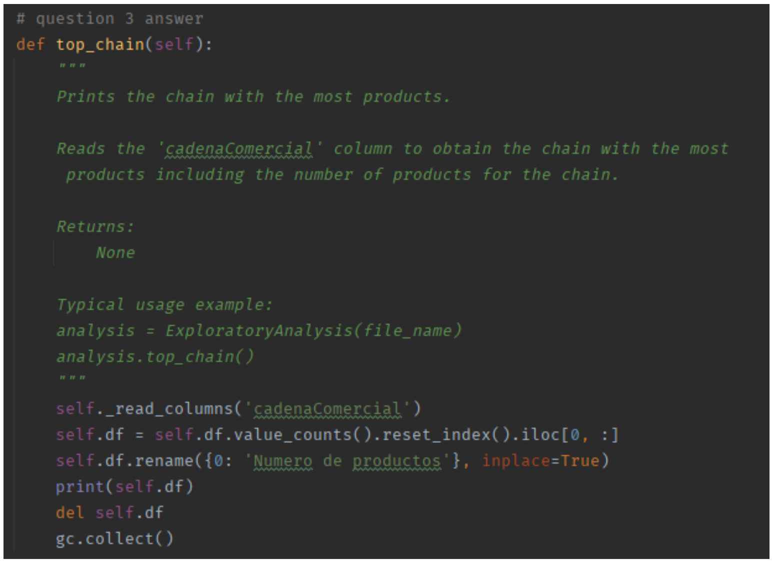
Geovanni Velázquez Medina

Partial answer to question 2:



```
                      estado                   producto  cantidad
0              AGUASCALIENTES                       FUD     12005
1              AGUASCALIENTES          DETERGENTE P/ROPA     10188
2              AGUASCALIENTES  LECHE ULTRAPASTEURIZADA      9824
3              AGUASCALIENTES                   SHAMPOO      9654
4              AGUASCALIENTES                  REFRESCO      9481
5              AGUASCALIENTES               DESODORANTE      8859
6              AGUASCALIENTES          JABON DE TOCADOR      8517
7              AGUASCALIENTES             CHILES EN LATA      7946
8              AGUASCALIENTES                   YOGHURT      7401
9              AGUASCALIENTES                  MAYONESA      7173
10            BAJA CALIFORNIA                  REFRESCO     37243
11            BAJA CALIFORNIA          DETERGENTE P/ROPA     23395
12            BAJA CALIFORNIA                       FUD     19967
13            BAJA CALIFORNIA                   SHAMPOO     19123
14            BAJA CALIFORNIA          JABON DE TOCADOR     18348
15            BAJA CALIFORNIA             CHILES EN LATA     16676
16            BAJA CALIFORNIA                  GALLETAS     15873
17            BAJA CALIFORNIA                 PANTALLAS     15703
18            BAJA CALIFORNIA                  CEREALES     15398
19            BAJA CALIFORNIA               DESODORANTE     14748
20        BAJA CALIFORNIA SUR                  REFRESCO     27770
21        BAJA CALIFORNIA SUR                       FUD     17776
```

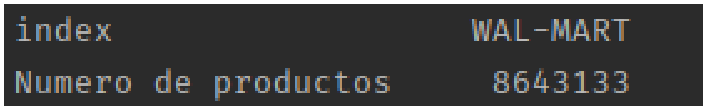
Question 3: Which is the commercial chain with the highest number of monitored products?



```
# question 3 answer
def top_chain(self):
    """
    Prints the chain with the most products.

    Reads the 'cadenaComercial' column to obtain the chain with the most
     products including the number of products for the chain.

    Returns:
        None

    Typical usage example:
    analysis = ExploratoryAnalysis(file_name)
    analysis.top_chain()
    """
    self._read_columns('cadenaComercial')
    self.df = self.df.value_counts().reset_index().iloc[0, :]
    self.df.rename({0: 'Numero de productos'}, inplace=True)
    print(self.df)
    del self.df
    gc.collect()
```

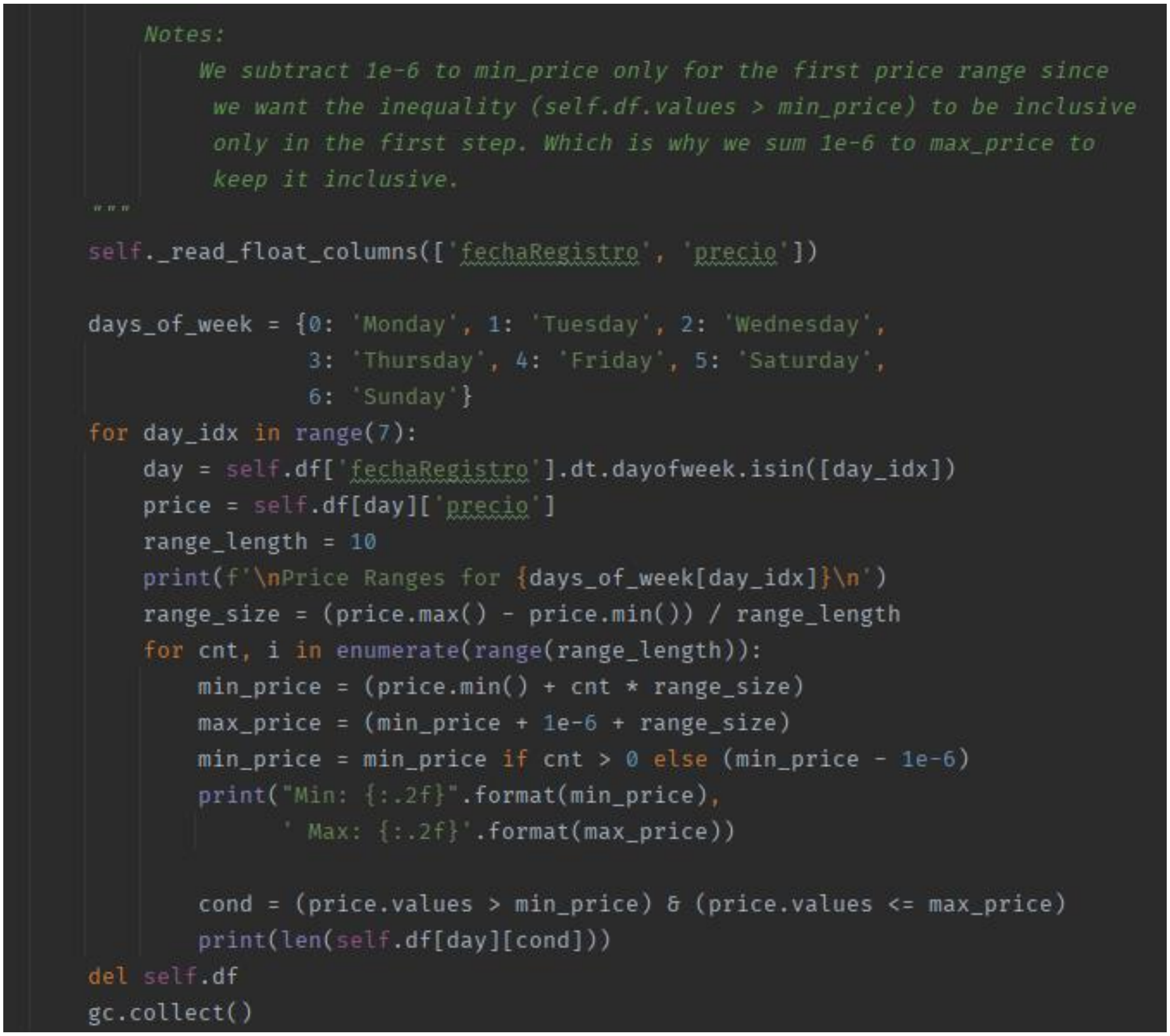
For this answer I used a value_counts which returns all the occurrences per each unique row of the given column. I used iloc just to get the first element since the question is only asking for the chain with the highest number of products. Fortunately for us pandas' value_counts method already returns each chain with their occurrences sorted.

Answer 3:



```
index                     WAL-MART
Numero de productos        8643133
```

Question 4: Use the data to find an interesting fact.

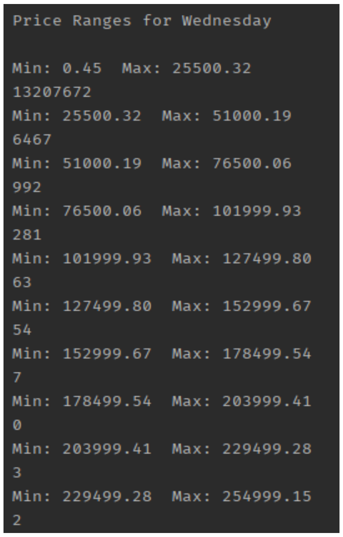


```
            Notes:
                We subtract 1e-6 to min_price only for the first price range since
                 we want the inequality (self.df.values > min_price) to be inclusive
                 only in the first step. Which is why we sum 1e-6 to max_price to
                 keep it inclusive.
        """
        self._read_float_columns(['fechaRegistro', 'precio'])

        days_of_week = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday',
                        3: 'Thursday', 4: 'Friday', 5: 'Saturday',
                        6: 'Sunday'}
        for day_idx in range(7):
            day = self.df['fechaRegistro'].dt.dayofweek.isin([day_idx])
            price = self.df[day]['precio']
            range_length = 10
            print(f'\nPrice Ranges for {days_of_week[day_idx]}\n')
            range_size = (price.max() - price.min()) / range_length
            for cnt, i in enumerate(range(range_length)):
                min_price = (price.min() + cnt * range_size)
                max_price = (min_price + 1e-6 + range_size)
                min_price = min_price if cnt > 0 else (min_price - 1e-6)
                print("Min: {:.2f}".format(min_price),
                      ' Max: {:.2f}'.format(max_price))

                cond = (price.values > min_price) & (price.values <= max_price)
                print(len(self.df[day][cond]))
        del self.df
        gc.collect()
```

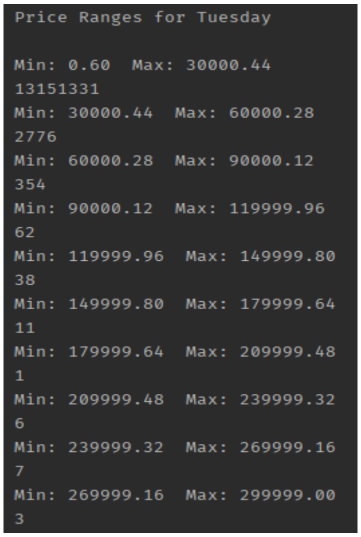
I decided to check the price ranges per day of the week in the data. For each day there will be 10 price ranges printed out along with the number of prices per price range. Weekends are included as well.

Geovanni Velázquez Medina

Partial answer for question 4:



```
Price Ranges for Wednesday

Min: 0.45  Max: 25500.32
13207672
Min: 25500.32  Max: 51000.19
6467
Min: 51000.19  Max: 76500.06
992
Min: 76500.06  Max: 101999.93
281
Min: 101999.93  Max: 127499.80
63
Min: 127499.80  Max: 152999.67
54
Min: 152999.67  Max: 178499.54
7
Min: 178499.54  Max: 203999.41
0
Min: 203999.41  Max: 229499.28
3
Min: 229499.28  Max: 254999.15
2
```



```
Price Ranges for Tuesday

Min: 0.60  Max: 30000.44
13151331
Min: 30000.44  Max: 60000.28
2776
Min: 60000.28  Max: 90000.12
354
Min: 90000.12  Max: 119999.96
62
Min: 119999.96  Max: 149999.80
38
Min: 149999.80  Max: 179999.64
11
Min: 179999.64  Max: 209999.48
1
Min: 209999.48  Max: 239999.32
6
Min: 239999.32  Max: 269999.16
7
Min: 269999.16  Max: 299999.00
3
```

The days with the most products reported were Wednesday and Tuesday. What I found interesting is that each day's price ranges follow a Pareto distribution. The quantity of products changes but the shape of the distribution does not.

Question 5: What are the lessons learned from this exercise?

I learned that what I thought of Profeco's data was wrong. Since the number of products per price has something resembling a Pareto distribution instead of a Gaussian which was what I expected at first. Other than that, I learned that many simple products such as school notebooks appear in this database.

Question 6: Can you identify other ways to approach this problem? Explain.

To benefit Profeco one option I can think of is to try to predict the number of reports per day. This due to the fact that in the answer for question 4 I found that more reports were received on Wednesday and Tuesday. By predicting the number of products per day we can have three options to benefit Profeco:

1. Lower employees needed on days when the number of reports is predicted to be low. This way Profeco can offer a more flexible schedule for its employees.

   Pros:

   - Possible micro savings by not having as many employees available when not needed (assuming they get paid by the hour).

   Cons:

   - Prediction errors may cause a shortage of employees when more are needed.

2. Use automated recordings or email answers for reports which are common such as sodas.

   Pros:

- Possible savings by automating part of the process. This could help ameliorate employee stress by having some people's reports solved by automated systems.
- People's satisfaction by possibly simplifying the process and potentially reducing wait times when demand is high for their services.

Cons:

- A system that works inefficiently would make the problem worse. People trying to make a report would probably just try to make the report in person.

3. Try to fix the root of the problem researching the causes for the reports on the days with the most appearances.

Pros:

- Possibly inexpensive alternative that could reduce the number of reports drastically.

Cons:

- Unknown variables which affect the number of occurrences.
- Root of the problem could be something Profeco is unable to change.