Luan

Question 1a)
Python result method -> exercise_one_a()

For this question, I decided to generate two graphs. The first helps us understand at what time of day invoice generation is more and less concentrated, while the second helps us understand not only the generation of invoices, but also any changes that may have occurred with the invoice, that is, any payment event, reversal, etc.

Both graphs had very similar behaviors and their distribution of activities throughout the hours of the day are very similar. Therefore

Using the generated graphs as a basis, we can understand that the period of least use of services and that can cause less operational loss would be between 2 and 5 in the morning. In addition, I believe that this time interval of approximately 3 hours can help us with rollback plans, if necessary.

Question 2a)
Python result method -> exercise_two_a()

For this question, I thought it was important to have two comparisons. So, in one scenario, all the different invoiceIds were used as the sample space, and in the other, all the different invoices with the created type. With this, we were able to find a percentage of 78.892% of invoices paid for the first scenario and 78.890% for the second scenario. The difference is almost negligible and it refers to three invoices that do not have the created event. I highlighted the study to find this information in the execise_two_a() method.

Regarding invoices paid late, using the sample space of both the first and second scenarios, the result was the same percentage using approximation to the third decimal place. The value was 6.556% of the total invoices paid in overdue.

Question 2b)
Python result method -> exercise_two_b()

To find the relationship between invoices paid in overdue and the months of the year, I generated a graph grouping the month of creation of the invoices that were paid in overdue. As shown in what was plotted in execise_two_b() we can see that we had delays in the months of April, May, June and July. The third month mentioned was the one that presented more than 50% of the invoices paid late and the third month presented a negligible value (less than 0.1%)

Question 2c)
Python result method -> exercise_two_c()

To find the reasons why an invoice is paid in time, I created three different groups.

1 - group all invoices paid in time,
2 - group all invoices paid in overdue,
3 - group all expired invoices.

After that, I calculated for each record the time needed for the invoice to be considered late (invoiceDue - invoiceCreated). With that, I was able to generate intervals where it was possible to analyze for each grouping how much the time until an invoice is considered late impacts its payment, late payment or expiration.

With that, it was possible to note some interesting points.

point A - The vast majority of invoices that are paid late (more than 90%) are cases in which the time to delay is less than 1 minute, which suggests that there may not be enough time to process the payment or there may be a delay in the response.

point B - Most invoices that are not even paid (expired) are cases in which the delay time is greater than 3 hours, which really suggests cases in which payment will probably not be made.

Therefore, we can see that the delay time directly impacts the scenario in which it will be (paid in time, paid in overdue or expired).

Question 3a)
Python result method -> exercise_three_a()

About execise_three_a() we basically find all invoices that have undergone some reversion and compare them with the number of different invoiceIds (sample space). With this, I obtained a percentage of 0.18% (as shown in the graph). For this question, it is important to pay attention to scenarios in which the same invoice has more than one reversal (we should consider only 1 case, even if it has N reversals).

Question 3b)
Python result method -> exercise_three_b()

Regarding execise_three_b(), it was important to group the invoices by id and also by their amount, in addition to filtering only the records with the type reversed or reversing. With this, I generated some time intervals representing the difference between the time that a reversing status (start) is logged and the reversed status (end). This helps to understand the time that is normally necessary to complete a reversion.
As the graph shows, most reversals take 5 to 10 seconds or 10 to 20 seconds.
In addition, the general average to complete these reversals was also calculated, the result was: 16,902 seconds.

Question 3c)
Python result method -> exercise_three_c()

Regarding execise_three_c(), this was a bit more complex. The idea was to analyze and compare whether there was any relationship between the time since the invoice was created (its age) and whether it was a total or partial reversion.

After grouping and setting the time intervals, it was possible to understand (from graphs 1 and 2 of the method) that:

1 - Normally, total reversals are made on invoices that were created less than a day ago, that is, more recent invoices.

2 - It is noticeable that partial reversions are much more related to invoices that were created a long time ago. As the graph shows, most partial reversals are made on invoices that were created more than 2 weeks ago.

Therefore, it seems logical that the “age” of the invoice (that is, since it was created) directly implies that it has a greater possibility of being a partial reversion (older age) or total reversion (younger age).

Final personal considerations:

To be honest, I didn't have much knowledge of Python, just from what I used in some courses at college, but I loved the challenge and managed to learn a lot. Basically speaking, to handle the file data I used pandas and to plot the graphs I used matplotlib. With these tools it was possible to create visualizations and better understand the data that the spreadsheet provides us.

Regarding the business rule, I had a bit of difficulty understanding why we had some invoices without the creation event, even though this only happened in 3 IDs, I thought it could have an impact. So, in the end, in the exercises in which this was impacted, I decided to bring both comparisons, that is, comparing with the total number of IDs and also with the total number of created invoices.

Furthermore, I liked that we need to think a lot about highly complex cases, for example, having to group information by different types, then retrieve the IDs of these lines and compare them with other records. At times, it became clear that disregarding some cases could completely impact the data that was being extracted, which is why it is such an important function and demands a lot of responsibility to ensure that the extracted data is correctly considering the logic we thought.

Once again, I would like to thank you for the opportunity, it was quite challenging and "stretching".