Data extraction, consolidation and cleansing in *etl.ipynb* file

This part was made using the **pandas** package in **Python**.

Working on `orders_` tables

First, I decided to work on `orders_` datasets. After extracting the data I noticed an issue with dates formatting in both `OrderDate` and `RequestedDeliveryDate` columns. I wrote a function `parse_dates()`, which transforms dates in those columns to DD/MM/YYYY format. Besides pd.DataFrame, this function takes the name of the month in which the orders were placed as an argument. This works as a hint for the function to determine, which part of the date is a day and which part is a month. It works well for the `OrderDate` column, because we know in which month the order was placed in. There is a slight issue with using it on the RequestedDeliveryDate column. I will explain it with an example. Imagine the order was placed 1st of January `(OrderDate = 01/01/2025)`, but the `RequestedDeliveryDate` was `02-01-2025` which can both mean 2nd of January or 1st of February. In both of those cases the condition that `RequestedDeliveryDate` should be greater or equal to the `OrderDate` is fulfilled. I decided to set the `RequestedDeliveryDate` as a minimum of the possible delivery dates, so in the example above it would be `02/01/2025`. This problem leaves a place for discussion. Finally, function `orders_table_format()` corrects all the issues with `orders_` tables. I decided to drop columns `CustomerName`, `CustomerID`, `Comments`, because they are not relevant for analysis. I concatenated `orders_` tables to one big `orders` table.

Working on `deliveries` table

Then I moved to working on the `deliveries` table and I stumbled upon various problems. First of all, dates in `ActualDeliveryDate` are inconsistently encoded. To solve this problem I wanted to use the same function as previously, but to make sure it worked correctly, I would need information on e.g. `OrderDate` to determine which number in the date is probably a month. There I found a bigger problem. I wasn't able to join `orders` with `deliveries`, because `OrderID`s in those tables do not match. `OrderID` should be an unique value assigned to exactly one order. An example showing this problem is included in the ***etl.ipynb*** file. It shows that the order with the ID `'O100000'` in `orders` and `deliveries` is clearly not the same order. I decided to remove `ActualDeliveryDate`, because we can't use the information it holds (If we were able to join `orders` and `deliveries` we could count e.g. how fast was the delivery). Another problem is that there are 12000 more records in the `deliveries` table than in `orders`. I didn't manage to discover the reason for that. I noticed a problem in `OrderID` and `DeliveryID`. It would be better if `OrderID` and `DeliveryID` had the same numbers but different prefixes: e.g. if `OrderID == O100004`, `DeliveryID` should be `D100004`. We would avoid potential delivery errors. I also wondered: what happens to lost and damaged packages? Are they sent again with different `OrderIDs`? If that is the case, I think that it should be specified in the `deliveries` table for example by changing the prefix in `DeliveryID` to e.g. 'D**A**-' for sent **A**gain packages. I thought that this could explain why there are more deliveries than orders but there are only 5156 damaged/lost packages that could have potentially been sent again.

Working on `product_master` table

The `product_master` table stores data about all products, which can be identified by its IDs. A column with IDs was called `ProductCode`, whereas in other tables it is called `ProductID`, so to stay consistent, I had to change it to `ProductID`. I also split the column `FamilyHierarchy (Category > SubCategory)` into two separate columns: `ProductCategory` and `ProductSubcategory`.

Working on `complaints` table

I noticed that `complaints` and `deliveries` match on `DeliveryID` and `OrderID`, and that all of the damaged packages are listed in the `complaints` table. Moreover, the `complaints` table contains only damaged deliveries. To avoid some problems later, I suggest adding one more 0 in between the 'CPL-' prefix and ID in `ComplaintID`, so there are six digits for the ID instead of five.

After all the transformations I saved tables as sheets to the ***cleaned_data.xlsx*** file.

Data analysis in PowerBI

I loaded the ***cleaned_data.xlsx*** file to PowerBI.

Execute View

First, I made a pie chart to show the Delivery Status Distribution. We can see that most of the packages (84.78%) arrived on time. I assumed that lost packages are considered as not closed orders, because they didn't arrive to the customer. If there is a problem with the delivery, most frequently it is a delay. FedEx caused most of the problems with the delivery. It is also important to check how many packages each carrier shipped in total, in order to calculate the percentages of properly delivered packages versus those that were damaged, lost, delayed, or delivered on time. It turned out that every carrier delivers a very similar number of packages. Steroid Product 355 is most frequently associated with delivery problems.

Operational View

Monthly product sales remain fairly consistent (around 84000). I noticed that at the beginning of November there were more products sold than typically in a day. This could be attributed to e.g. a marketing campaign. If we could use `ActualDeliveryDate` from the deliveries table we could determine whether the delivery performance has changed over time. If `OrderID` columns matched, we could calculate average delay in days by carrier, which could help us improve in this area. We could also see if there is a correlation between order quantity and delayed/damaged/lost delivery - we could expect huge orders to be more difficult to complete.

The final report was saved to the ***analysis.pbix*** file.