Exploratory Data Analysis (EDA)
Lab 2: advanced pandas

Julien Velcin

2025-2026

Objective: In the previous lab, you learned how to process a single, clean table. But real-world investigations are rarely that simple. Today, you are handed the French Road Traffic Injury dataset (ONISR). The clues are scattered across multiple files, and the data is messy. Your mission is to clean the evidence, connect the different tables, and extract actionable insights.

Prerequisites: Download the 4 CSV files (`caracteristiques-2018.csv`, `lieux-2018.csv`, `vehicules-2018.csv`, `usagers-2018.csv`) and the PDF data dictionary from https://velcin.github.io/eda. Keep the PDF open, you will need it to translate the category codes!

Part 1 : Securing the scene (data loading and cleaning)

1.1 Loading the evidence: Load the 4 CSV files into four separate Pandas DataFrames (`df_acc`, `df_places`, `df_vehicles`, `df_users`). *Hint:* French open data often uses a specific separator (like `;` instead of `,`) and encoding. If you get an encoding error, try `encoding='utf-8'` or `encoding='latin-1'`.

1.2 First glance: For each DataFrame, check the shape and the data types. How many accidents were recorded in 2018?

1.3 The missing pieces: Missing data is a common issue (due to typos, measurement impossibility, etc.).

- Calculate the percentage of missing values (`NaN`) for each column in the `df_places` table. Print the resulting percentages with an appropriate f-string. The output should be:

```
Num_Acc     0.00%
catr        0.00%
voie       37.89%
(...)
```

- The column `vosp` (reserved lane) has a lot of missing values. Read the PDF dictionary. Does a `NaN` here mean "Unknown" or just "Not applicable" (no reserved lane)?
- Impute (fill) the missing values in `vosp` with the appropriate default value (use `.fillna()`).

1.4 Time formatting: In `df_acc`, look at the `hrmn` (time of accident) column. It is likely formatted as an integer (e.g., 1430 for 14:30) or a weird string. Write a function to clean this column and convert it into a standard "Hour" format into a new column (`accident_hour`).

- First, use `zfill` to make sure the variable is of length 4
- Select the first two digits to get the hour only (eg., "14")
- Don't forget to change variable type to `int` by casting.

Part 2 : Interrogating the suspects (univariate analysis and groub by)

Now that the scene is secure, let's interview the victims and look at the vehicles involved.

2.1 User profiling: In the `df_users` table, the column `grav` indicates the severity of the injury (Unscathed, Killed, Hospitalized, Light injury). Plot a bar chart showing the distribution of injury severity. *Warning:* The numbers 1, 2, 3, 4 are categories, not continuous integers! Use the PDF to map these numbers to readable text labels before plotting.

2.2 The vulnerability hypothesis: We suspect that pedestrians and cyclists (`catu` column) suffer more severe injuries than car drivers. Use the `.groupby()` method to calculate the proportion of "Killed" and "Hospitalized" for each category of user. Was our hypothesis correct? *Hint:* For printing percentage values in a nice way, you can use the `to_string` function by specifying the `float_format` similarly to f-string.

2.3 Vehicle types: In `df_vehicles`, find the top 5 most common types of vehicles (`catv`) involved in accidents.

Part 3 : Connecting the clues (relational data and merges)

A suspect's alibi only makes sense when crossed with the crime scene log. We need to join our tables!

3.1 The primary key: Identify the column that serves as the unique identifier linking all these tables together.

3.2 Merging places and accidents: Create a new DataFrame called `df_master` by joining `df_acc` and `df_places`. *Hint:* Use `pd.merge()`. Should you use a left join, inner join, or outer join? Why?

3.3 Adding users: Now, merge `df_users` into your `df_master` table. *Careful:* One accident can involve multiple users. What happens to the number of rows in your `df_master` after this join?

3.4 The Lyon Bicycle Case: Your chief wants a specific report. Calculate the exact number of accidents involving a bicycle (`catv == 1`) that occurred in the city of Lyon (`dep == 690` or `com == ...` depending on the dataset version). *Hint:* You will need to join `df_master` with `df_vehicles` to get this answer.

Part 4 : The big picture (visualization and storytelling)

It's time to present your findings to the judge. The visual cues must be clear and objective.

4.1 City Comparison: Build a bar chart comparing the total number of bicycle accidents in 3 major cities: Paris, Lyon, and Marseille.

4.2 The Population Bias: Wait! Paris has much more inhabitants than Lyon. Comparing raw numbers can lead to a statistical bias.

- Find the approximate population of these 3 cities on the internet.
- Normalize your data: calculate the number of bicycle accidents *per 100,000 inhabitants*.
- Plot the new normalized bar chart. How does the story change?

4.3 Refining the plot: Apply the "Less is More" principle to your final plot.

- Remove unnecessary grid lines.
- Add a clear, explanatory title (e.g., "X is the most dangerous city for cyclists per capita", not just "Accidents by City").
- Add axis labels when it's necessary.
- Highlight the highest bar with a specific color (e.g., red), and keep the others gray.

4.4 The Demographics of danger (Box plots): A witness claims that young people are involved in the most severe accidents. Let's verify the distribution of ages across the different levels of injury severity.

- In the `df_users` table, create a new column `age` by subtracting the birth year (`an_nais`) from the year of the dataset (2018).
- Use the `seaborn` library to create a box plot (`sns.boxplot()`) displaying the `age` on the Y-axis and the severity (`grav_label`) on the X-axis.
- Look at the medians and the outliers. Does the visual evidence support the witness's claim?

Part 5 : The cold case (bonus / going Further)

For the fast detectives who finished early. No hints provided here!

5.1 Temporal trends: Download the ONISR data for the year 2017 on the data.gouv website. Concatenate the 2017 and 2018 datasets. Plot a line chart showing the evolution of the number of accidents month by month across the two years.

5.2 The weather factor: Is there a correlation between the weather conditions (`atm`) and the severity of the accidents? Use a heatmap or a stacked bar chart to prove your point.

5.3 Age vs. vehicle: Create a scatter plot (or boxplots) analyzing the age of the driver versus the category of the vehicle (e.g., do younger people drive older cars? Are they more involved in motorcycle accidents?).