# **Decision Science**

1. Motivations
2. The Brazilian e-commerce platform Olist
3. Organizing a Project in Data Science and Analytics
4. Notebook like a pro
5. How to debug your code
6. Dealing with new datasets
7. Joining tables

## **The Big Picture**

🔥 5 sessions working as a **Data Consultant** on a real case study!

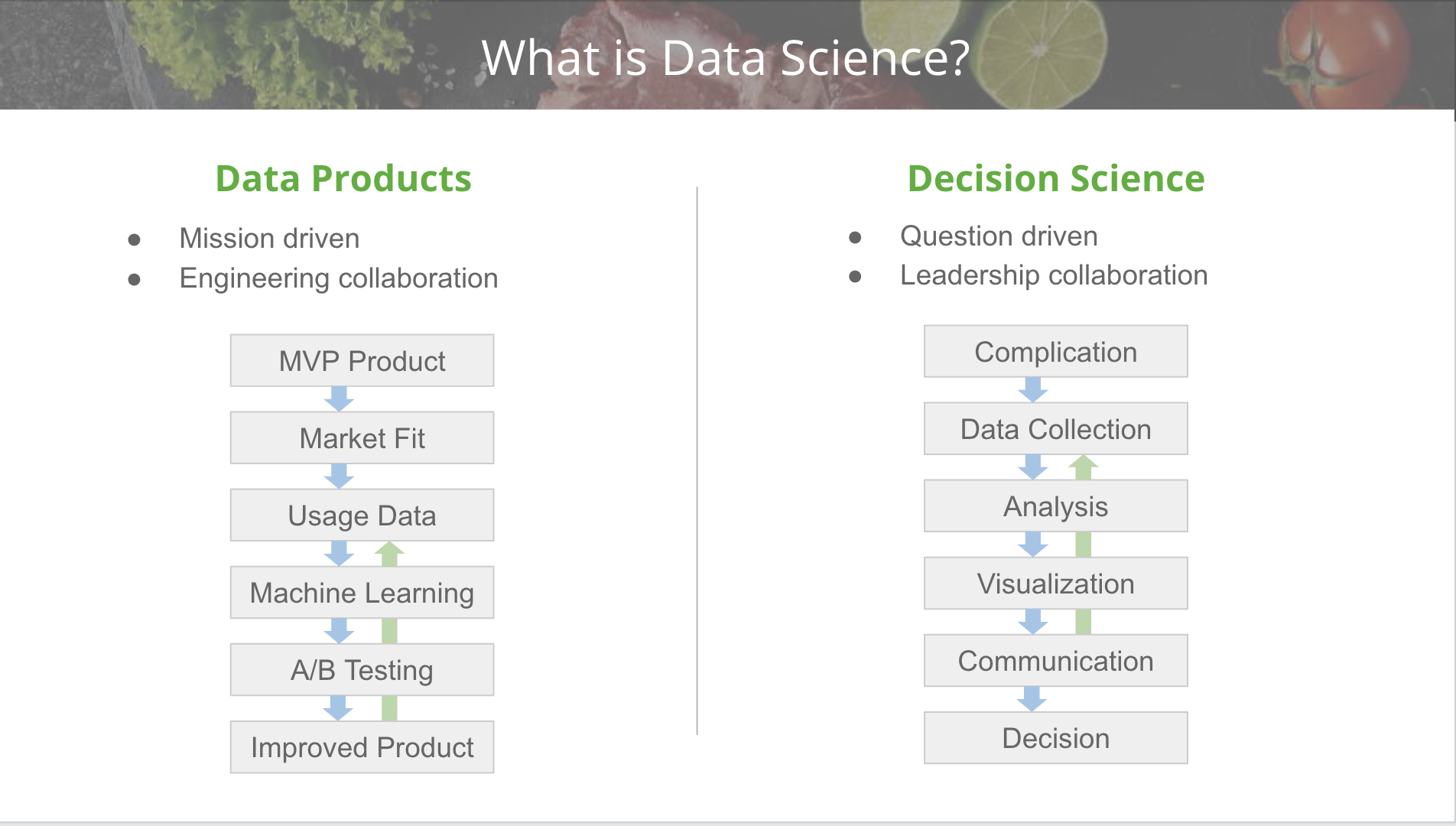
💸 Objective: help the business owners in their **Decision-Making Process**

📈 In-depth analysis of historical business data

🕵🏻 Discover your first (linear) **Statistical Models**: the Linear Regression and the Logistic Regression

* ❗️ No fancy predictions of the future
* 👀 Find correlations between variables, i.e. make your data speak
* 🎲 Use **Statistical Inference** to prove the robustness of your findings

🗣 **Communicate** your results



## **1. Motivations**

❗ Overlooked qualities of Data Analysts and Data Scientists

🤝 **Domain Knowledge** > Technical Expertise

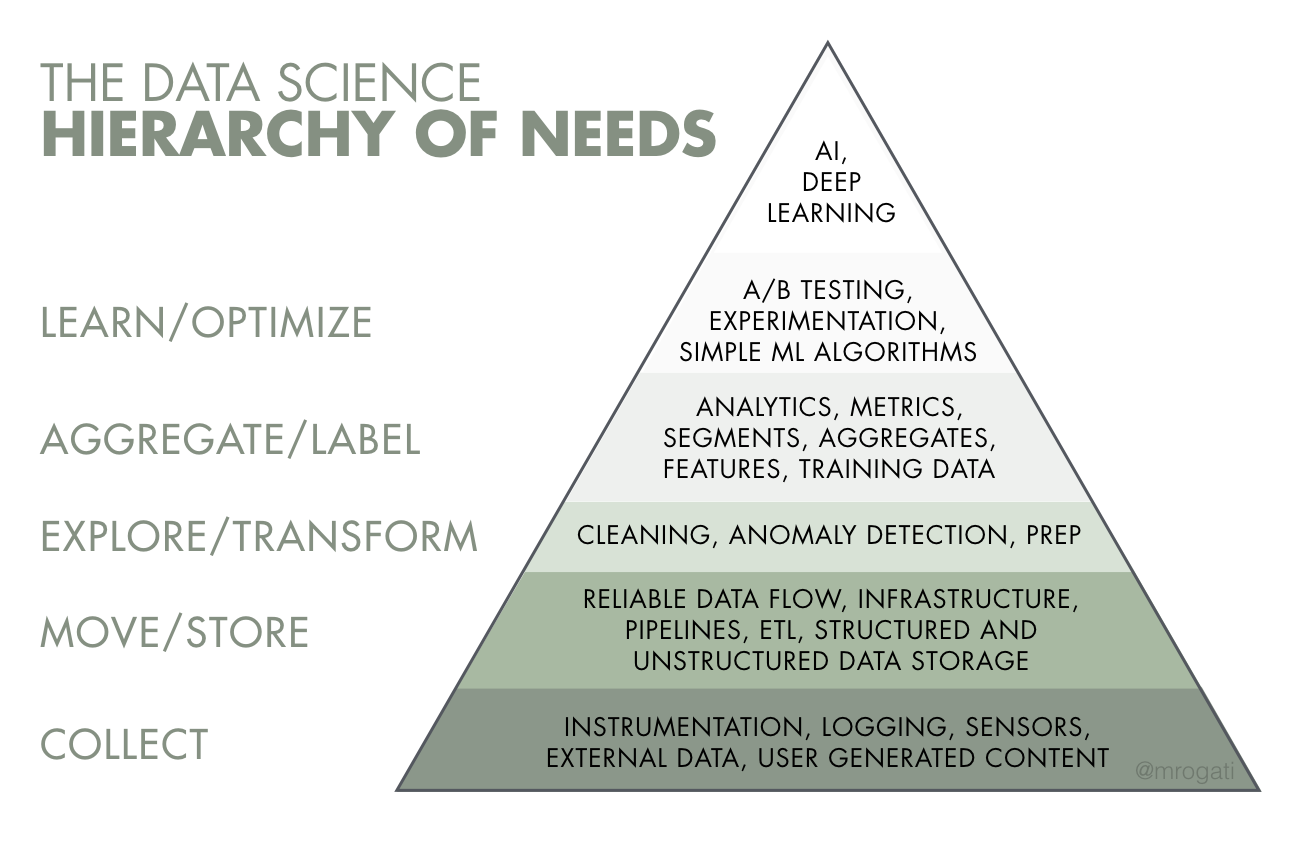
🚀 **Solve real issues**

🤨 Handle **vague problems**

🔥 **Build with scale** in mind

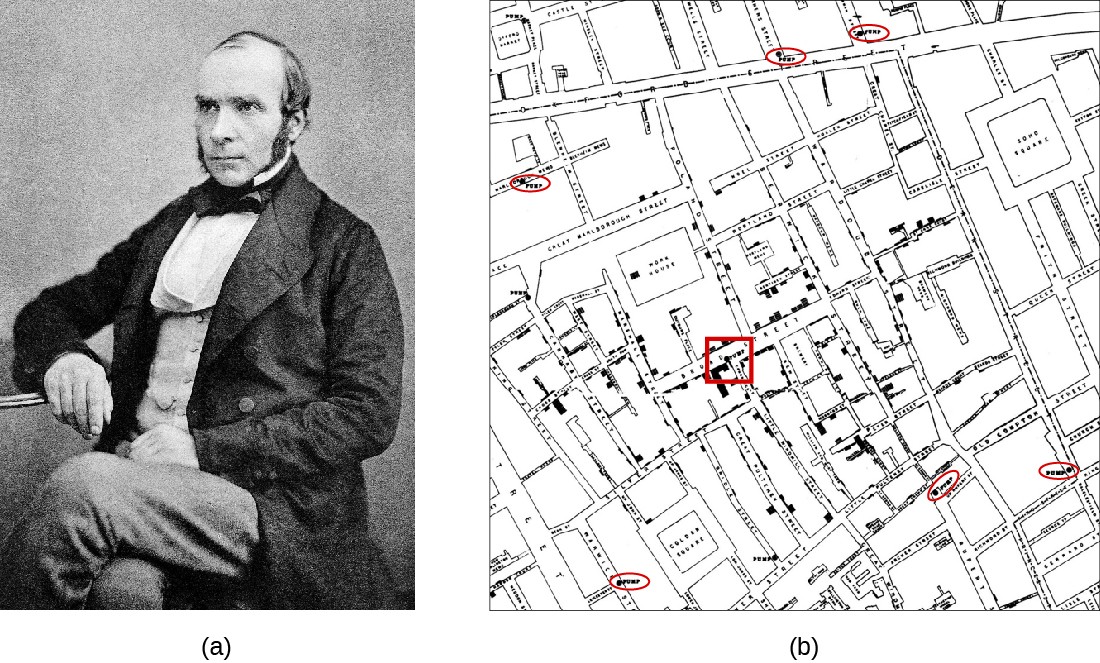
🗣 **Communicate** well

**Pyramid of Needs** (#data preparation is key)



[The AI Hierarchy of Needs](https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007), Monica Rogati, ex VP Data @ LinkedIn

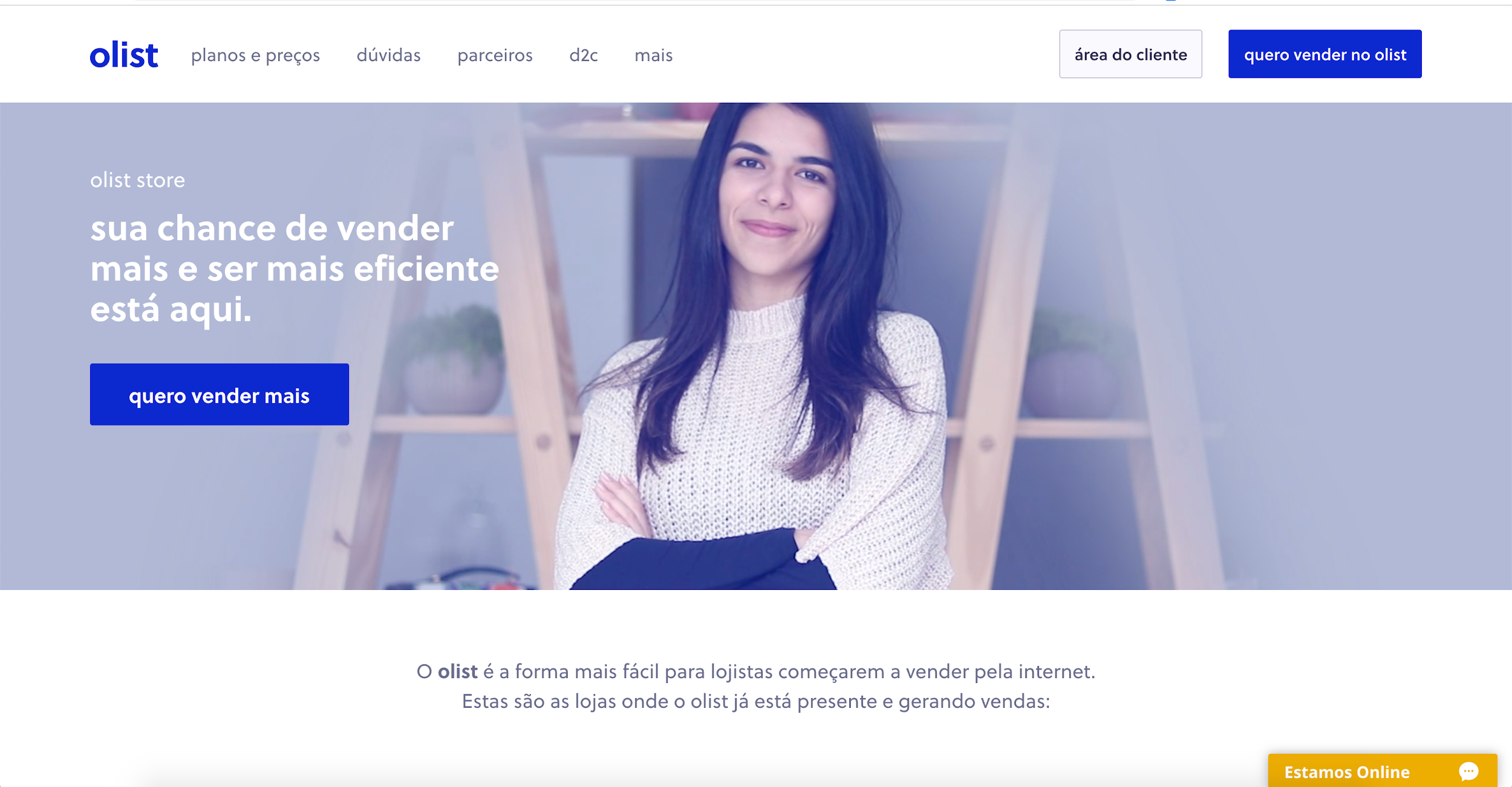
**John Snow & Cholera outbreak** (#communicate)



Source: [Wikipedia, 1854 - Broad Street Cholera Outbreak](https://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak)

## **2. The Brazilian e-commerce platform Olist**

### **Introduction**

****

[**Olist**](https://olist.com/)

🗓 Founded in 2015

🇧🇷 Operates in Brazil

💻 100% digital company   
*(Pure Digital Player, i.e. business operations fully managed online)*

💸 Total Funding Amount

≈

136M USD (accumulated from 2015 to 2021)   
*Source:* [*CrunchBase/Olist/CompanyFinancials*](https://www.crunchbase.com/organization/olist/company_financials)

🛒 Online e-commerce service for sellers

🤝 Connects (small) merchants to the most important marketplaces in Brazil (Amazon, Bahia, Walmart, ...)

✅ Offers *Logistic and Inventory Management Services* to sellers

❗️ Does *NOT* sell anything directly to consumers

The **Seller** workflow ⚙️

The seller:

1. joins Olist
2. uploads their product catalogues
   * (Olist) displays these catalogues to existing marketplaces (Amazon, Bahia, Walmart, ...)
3. gets notified whenever a product is sold
4. hands over the ordered items to third-party logistic carriers

*Note: Multiple sellers can be involved in one customer's order!*

The **Customer** workflow ⚙️

The customer:

1. browses products on marketplaces (Amazon, Bahia, Walmart, ...)
2. purchases products listed via Olist.store
3. gets an expected date for delivery
   * *ETA = Estimated Time of Arrival (of the orders)*
4. receives the order(s)
5. leaves a review

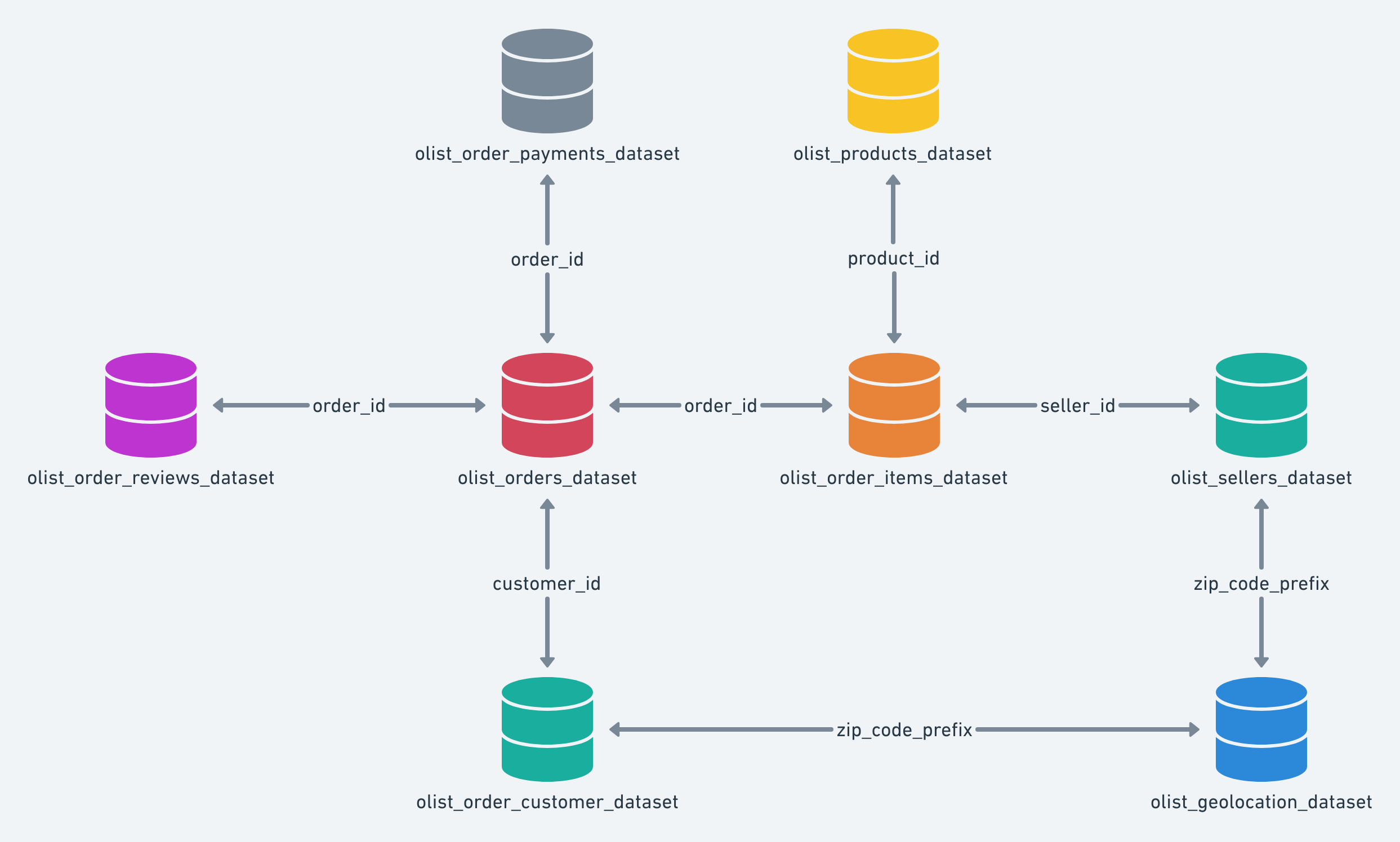
*Note: Between 2016 and mid-2018, a review could be left as soon as the order was sent, meaning that a customer could potentially leave a review for a product they hadn't received yet!*

### **Dataset**

*About the data:* 🔢

* Released on Kaggle in November 2018
  + [kaggle.com/olistbr/brazilian-ecommerce](https://www.kaggle.com/olistbr/brazilian-ecommerce)
* **Information about ~100k orders made between 2016 and 2018**
* **8 csv files, ~120mb**
* Real data!
  + Marketplaces' and merchants' identifiable information have been anonymized
* The full documentation is available in the /data directory of the 1st challenge of the unit

Data Model:



### **Revenues**

💸 📈 **Olist's Revenues**:

Sellers have to pay to use Olist.

How does Olist earn revenues from the sellers?

Variable revenues

Olist charges a **10% cut** on each product sold by a seller, if the order was delivered.

Fixed revenues

Sellers have to pay Olist **80 BRL per month** to use the platform.

### **Costs**

💸 📉 **Olist's Costs**

IT Costs

* We will make the (*simplified*) assumption that the IT Costs scale with the square root of the total number of sellers, as well as the square root of the total number of items sold.
* In other words, we assume that IT costs grow relatively slower and slower for each additional seller and additional item sold (economies of scale)

Reputation Costs

*Bad reviews affect Olist's reputation and long term profits*.

Let's quantify these impacts as follows:

| **review\_score** | **cost (BRL)** |
| --- | --- |
| 1 | 100 |
| 2 | 50 |
| 3 | 40 |
| 4 | 0 |
| 5 | 0 |

❗️ Warning ❗️

* Keep in mind that we will be using these simplified assumptions for our Data Science Consulting project because we have five days to answer the question.
* In a Data Analyst interview, you will be given some datasets (.csv) and you will be asked to answer questions in a limited amount of days (3-5 days).
  + So you will also need to make some assumptions
  + Also, because in the recruiting process, you do not have access to the real costs of the company

### **Your mission**

You have been tasked by Olist's CEO [Tiago Dalvi](https://www.linkedin.com/in/tiagodalvi/?locale=pt_BR) to make sense of the dataset to give recommendations about the following question:

*How to increase customer satisfaction (so as to increase profit margin) while maintaining a healthy order volume?*

## **3. Project Organization**

### **3.1 Repository**

.

├── 01-Project-Setup

│ │ *# Your whole code logic and data, this is your "package"*

│ ├── data-context-and-setup

│ │ ├── data *# your data source (git ignored)*

│ │ │ ├── csv

│ │ │ │ ├── olist\_customers\_dataset.csv

│ │ │ │ ├── olist\_orders\_dataset.csv

│ │ │ │ └── ...

│ │ │ └── README.md *# database documentation*

│ │ │

│ │ └── olist *# your data-processing logic*

│ │ ├── data.py

│ │ ├── product.py

│ │ ├── seller.py

│ │ ├── utils.py

│ │ └── \_\_init\_\_.py *# turns the olist folder into a "package"*

│ │

│ │ *# Your notebooks & analyses, day-by-day, challenge-by-challenge*

│ ├── data-data-preparation

│ └── data-exploratory-analysis

├── 02-Statistical-Inference

│ └── ...

├── 03-Linear-Regression

│ └── ...

└── 04-Logistic-Regression

└── ...

🧑🏻‍🏫 Our methodology:

1. Manipulate data in Notebooks .ipynb
2. Encode confirmed processing steps in Python .py scripts within **classes**
3. Import your scripts and instantiate your **classes** when needed for new analysis

👉 We will create the following classes:

* Olist
* Order
* Seller
* Product

😉 Feel free to revisit this Olist data science consulting mission after the bootcamp to create additional classes for further analysis!

### **3.2 Classes**

A **class** is just a template designed to build **instances** which share common **attributes**

💡 Think about a "Mold" *vs.* a "Cake"



💻 Let's create our first class in a new project

mkdir lewagon-project

cd lewagon-project

mkdir csv

mkdir notebooks

mkdir lewagon

touch lewagon/student.py

touch lewagon/\_\_init\_\_.py

code .

**class** **Student**:

*# class attribute*

school = 'lewagon'

*# initializer of instance attributes*

**def** \_\_init\_\_(self, name, age): *# Note the `self` parameter*

self.name = name.capitalize()

self.age = age

*# instance method*

**def** says(self, something): *# x.f(y) equivalent to Class.f(x,y)*

print(f'**{**self.name**}** says **{**something**}**')

boris = Student('boris', 31) *# Instantiating a student named `boris`*

print(boris.school) *# who is 31 yrs old*

boris.says('hi') *# eq. to Student.says(boris, 'hi')*

lewagon

Boris says hi

**Class method**

💻 What if we want to instantiate a student from their birth date instead ?

**from** **datetime** **import** date

**class** **Student**:

*# [...]*

*# Class method*

@classmethod

**def** from\_birth\_year(cls, name, birth\_year): *# Note the `cls` parameter*

**return** cls(name, date.today().year - birth\_year)

zuza = Student.from\_birth\_year('zuza', 1963) *# Instantiating a student name `zuza`*

print(zuza.age) *# who was born in 1963*

zuza.says("Heyyyyy, I'm younger !!!!")

58

Zuza says Heyyyyy, I'm younger !!!!

**Class inheritance**

**class** **DataStudent**(Student):

cursus = 'datascience'

**def** \_\_init\_\_(self, name, age, batch):

super().\_\_init\_\_(name, age)

self.batch = batch

boris = DataStudent('boris', 30, 359)

boris.\_\_dict\_\_

{'name': 'Boris', 'age': 30, 'batch': 359}

## **Code refactoring**

Now that we know how the class inheritance works, let's talk about best practices:

❌ Multiple classes in one .py file

✅ One class per .py file

touch lewagon/data\_student.py

*# lewagon/data\_student.py*

**class** **DataStudent**(Student):

cursus = 'datascience'

**def** \_\_init\_\_(self, name, age, batch):

super().\_\_init\_\_(name, age)

self.batch = batch

❗️ When creating a sub-class that inherits from a parent class, do *NOT* forget to import the parent class ❗️

*# lewagon/data\_student.py*

*# the import will only work if executed from lewagon-project directory*

*# unless you have set up the PYTHONPATH*

**from** **lewagon.student** **import** Student

**class** **DataStudent**(Student):

cursus = 'datascience'

**def** \_\_init\_\_(self, name, age, batch):

super().\_\_init\_\_(name, age)

self.batch = batch

boris = DataStudent('boris', 30, 359)

boris.batch

359

💡 Call the IPython [**autoreload**](https://ipython.readthedocs.io/en/stable/config/extensions/autoreload.html) extension to avoid restarting the kernel everytime you modify the .py within your package.

In [1]: %load\_ext autoreload

In [2]: %autoreload 2

**Naming conventions**

* Packages and modules have **short, all-lowercase** names: pandas, lewagon.
* Class names use **UpperCamelCase**: DataFrame, Student.
* Variables and functions use **lower\_snake\_case**: name, first\_name, from\_birth\_year().

👉<https://www.python.org/dev/peps/pep-0008/#naming-conventions>

### **3.3 Package and Module Imports in Python**

❓ How can we call custom Python .py scripts in a Jupyter Notebook or an iPython session ❓

**from** **olist.data** **import** \*

🤔 **Problems**:

* The way you import your modules depends on where you run your notebook or your iPython session
* It becomes quickly tedious to import modules living in a completely different place!

👉 **A first solution**:

Add the following code in each iPython session or Notebook:

root\_path = "/Users/..../lewagon-project"

**import** **sys**; sys.path

sys.path.append(root\_path)

💪 **A more robust and easier solution: the PYTHONPATH**:

Add your repository path to a PYTHONPATH global environment variable

*# open your zshrc file*

code ~/.zshrc

*# add this at the bottom of your zshrc file*

export PYTHONPATH='/Users/.../lewagon-project'

*# restart your zsh shell*

exec zsh

Restart your Jupyter Notebook server or iPython

*# you can check your pythonpath specifically*

**import** **os**; os.environ['PYTHONPATH']

*# double check your root\_dir is your path list for package imports*

**import** **sys**; sys.path

### **3.4 Where am I ?**

We need to access our data files from inside our Python code.

Our data will be in a folder in our file system:

/

├── Users *# on macOS - `home` on Linux / WSL*

│ ├── your-username

│ │ ├── code *# all coding projects you work on*

│ │ │ ├── your-github-username *# the coding projects you own*

│ │ │ │ ├── project-name *# your project folder*

│ │ │ │ │ ├── data *# your data folder*

│ │ │ │ │ │ ├── orders.csv

│ │ │ │ │ │ └── ...

│ │ │ │ │ ├── package-name

│ │ │ │ │ │ ├── data.py

│ │ │ │ │ │ ├── order.py

│ │ │ │ │ │ └── ...

│ │ │ │ ├── other-project

│ │ │ │ └── ...

│ │ ├── Documents

│ │ └── ...

So the *absolute path* to your data file is for example:

/Users/your-username/code/your-github-username/project-name/data/orders.csv

But we can't use that in our code! Why not?

**Current Working Directory**

*From where* are you executing your Notebook or iPython session or .py file?

*# iPython session or Jupyter Notebook or .py file*

**import** **os**; os.getcwd()

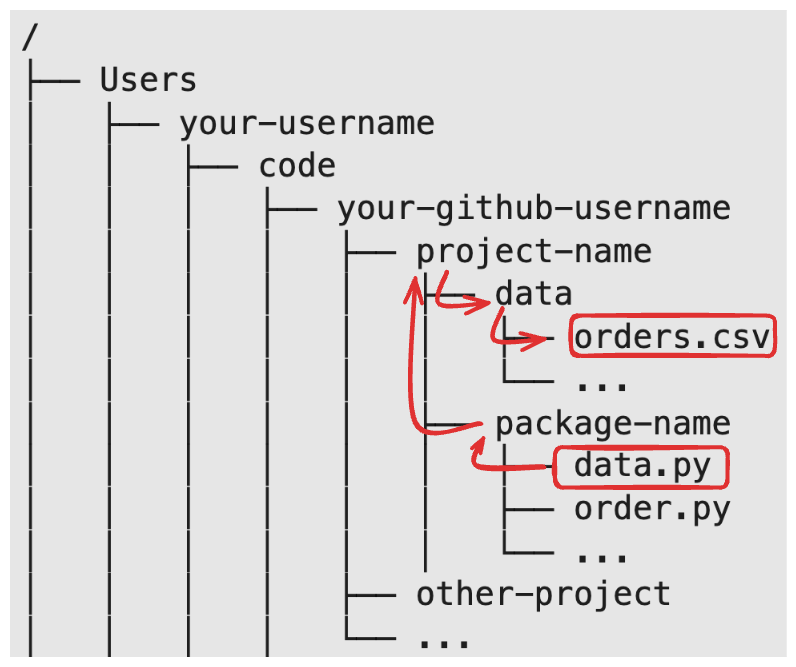
☝️ A **relative path** always starts from the **"current working directory"**

So we can't use relative paths in our code either. Why not?

It will break if we execute the code from inside another folder.

**How to solve this?**

If you know the location of your python file, you can find the location of your data!



**Location of a python file**

Where is this module.py located on the disk?

Use the following piece of code within your python file:

\_\_file\_\_ *# Contains the absolute path of module.py, including the filename*

How to get the directory of a python file?

os.path.dirname(\_\_file\_\_) *# Returns the directory where your module.py is stored*

How to get the directory above the directory of a python file?

os.path.dirname(os.path.dirname(\_\_file\_\_))

⚠️ \_\_file\_\_ is only available from python files, not from a notebook

**Assembling paths**

How to create paths from components?

❌ "folder\_path/subfolder/data.csv"

✅ os.path.join("folder\_path", "subfolder", "data.csv")

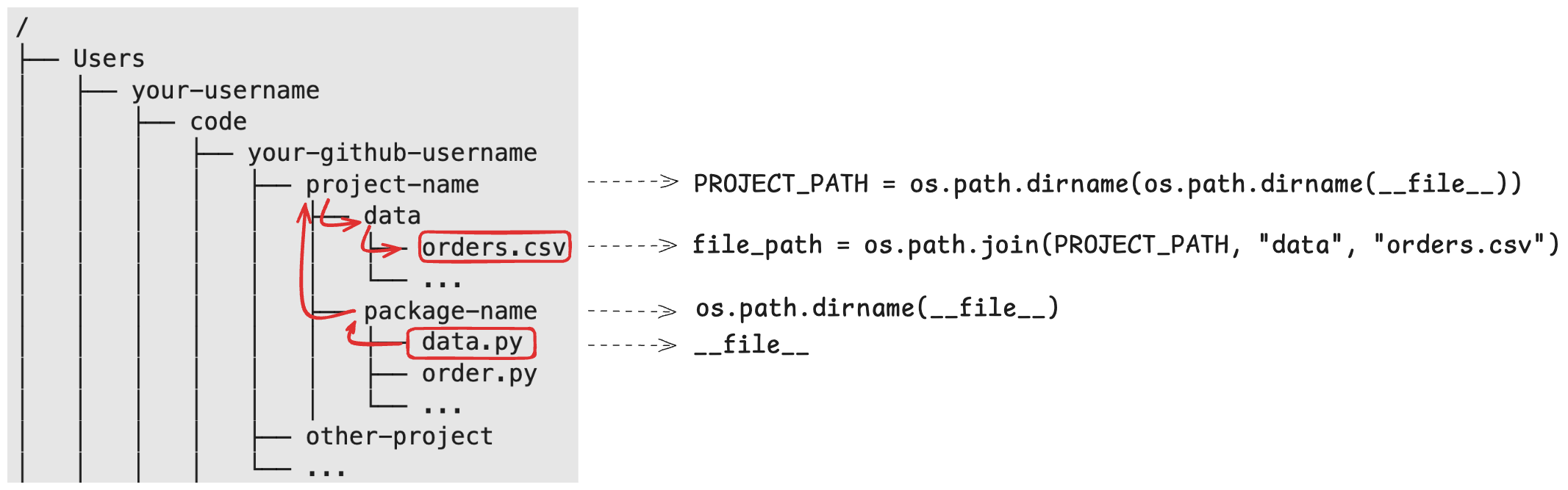
Slashes (/) only work on POSIX systems like Unix, Linux, FreeBSD, macOS.

Windows (outside of WSL) requires backslashes (\)

**Putting it all together**

How to create a path to your data files from your *module.py* file?

1. Obtain the path to your *module.py*
2. Use os.path.dirname() to find the root path of your project
3. Use os.path.join() to construct a path to your data files



PROJECT\_FOLDER = os.path.dirname(os.path.dirname(\_\_file\_\_))

filename = os.path.join(PROJECT\_FOLDER, "data", "orders.csv")

## **4 Notebook like a pro**

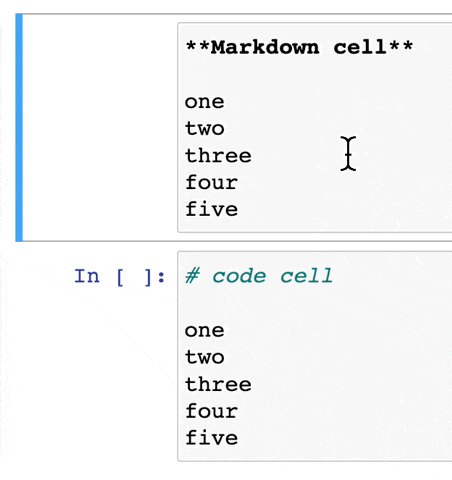
**General Principles of a Notebook**

* Your Notebook should be runnable from top to bottom (Run --> Run All Cells)
* Name your variables carefully
* Clear your draft code
* Merge cells when relevant (Shift M)
* Un-merge when debugging (Ctrl Shift -)

**Most frequently used Notebook shortcuts**

* Esc - A 👉 insert a cell above
* Esc - B 👉insert a cell below
* Esc - D - D 👉 delete a cell
* Esc - arrows 👉 move between cells
* Shift + Enter 👉 execute the current cell and move the focus to the next one
* Cmd + Enter (Mac Users) 👉 run the current cell and keep the focus on the same cell  
  Ctrl + Enter (Windows or Linux Users)
* Shift + Tab 👉 to get the docs! Repeat this combo many times to open the docs *"permanently"*

**Multi-cursor select 🔥**

* Hit Alt / Option on your keyboard and drag your cursor:  
  
* Similar: hit Ctrl / Cmd on your keyboard and click on multiple places
* Select next occurence of the selection with Cmd-D or Ctrl-D like in VS Code?  
  + The default setting in Jupyter is Cmd-Shift-D or Ctrl-Shift-D.
  + Open jupyter lab, go to Settings > Settings Editor > Keyboard Shortcuts
  + Find Select Next Occurence and change it to Cmd-D or Ctrl-D

## **5. Debug**

*# iPython debugger*

breakpoint()

>>> s (step into)

>>> n (next = step over)

>>> c (continue to next error)

>>> u (up stack trace), d(down)

>>> return (continue until current function return)

>>> l (provide more context)

>>> q (quit) or exit

☝️ Works in iPython sessions and in Jupyter Notebooks as well!

⚠️ Always do a proper "clean exit" out of the debugger in Jupyter, otherwise you will have to restart the kernel.

You can also add a new cell under an error and run:

%debug *# raises `breakpoint()` just before the error!*

## **6. Dealing with new datasets**

* Check DataFrame attributes .shape, .columns, .dtypes
* Use DataFrame methods .head(), .info(), .describe(), nunique(), .isna().sum()
* Plot distributions ...

**Use YData Profiling 🔥🔥🔥** One-click EDA (Exploratory Data Analysis):

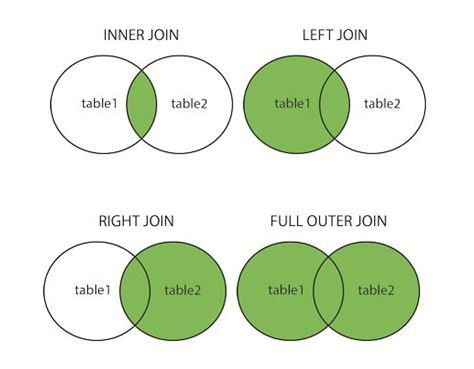
* Essentials
* Quantile statistics (min, Q1, median, Q3, max, ...)
* Most frequent values
* Histogram
* Correlations
* Missing values matrix, count, heatmap and dendrogram of missing values

👉 [github.com/ydataai/ydata-profiling](https://github.com/ydataai/ydata-profiling)

**import** **ydata\_profiling**

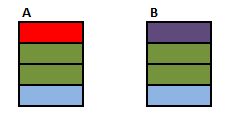
mpg.profile\_report()

## **7. (Refresher) Get your head around inner join *vs.*outer join**

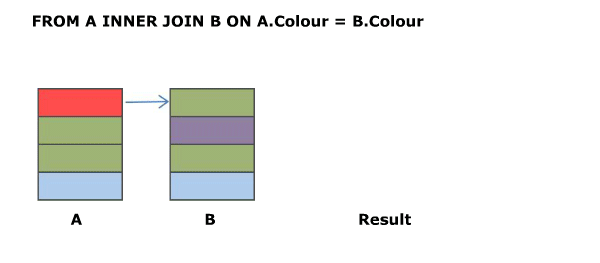
****

❓ What will be the shape of the inner joined table ❓

A.merge(B, on='color', how='inner')



Answer: (5,2)



📚 Read the following thread:

* [stackoverflow.com/questions/different-between-inner-join-and-outer-join](https://stackoverflow.com/questions/38549/what-is-the-difference-between-inner-join-and-outer-join#answer-27458534)

## **🚀 Your turn!**

* **Sessions 1 & 2: Data Preparation (40%)**
  + Understand the problem
  + Organize your repo
  + Make sense of the data
* **Sessions 3 & 4: Data Analysis (40%)**
  + Break down the problem into smaller actionable analyses
  + Use Linear / Logistic Regression
* **Session 5: Communicate (20%)**
  + Realize a quantitative [Cost-Benefit Analysis](https://www.investopedia.com/terms/c/cost-benefitanalysis.asp)
  + Present your results and convince your audience of your recommendations

### **What we'll do in the challenges**

**Challenge 1: setup our project**

Setup our project structure  
│ ├── folder-for-the-first-challenge

│ │ ├── data *# your data source (git ignored)*

│ │ │ ├── csv

│ │ │ │ ├── olist\_customers\_dataset.csv

│ │ │ │ ├── olist\_orders\_dataset.csv

│ │ │ │ └── ...

│ │ │ └── README.md *# database documentation*

│ │ │

│ │ └── olist *# your data-processing logic*

│ │ ├── data.py

│ │ ├── product.py

│ │ ├── seller.py

│ │ ├── utils.py

│ │ └── \_\_init\_\_.py *# turns the olist folder into a "package"*

1. Change PYTHON\_PATH so we can use our package from anywhere on our machine

**Challenge 2: Write reusable code to load our data**

**Write code in a jupyter notebook** to load all the csv files into a dictionary of Pandas DataFrames:  
{

'orders': a DataFrame **with** the data **from** **the** orders csv,

'sellers': a DataFrame **with** the data **from** **the** sellers csv,

...

}

**Refactor the code** from the notebook into olist/data.py:  
We'll create a class Olist with a get\_data() method that will return the dictionary of Pandas DataFrames.  
This way we'll be able to load our data from anywhere using:  
**from** **olist.data** **import** Olist

data = Olist().get\_data()

orders = data['orders']

**Challenge 3: Explore the data**

Explore our data, using the refactored code in our package to load the data.

## **Bibliography**

Good reads 📚

* [12 things I wish I’d known before starting as a Data Scientist](https://medium.com/deliberate-data-science/12-things-i-wish-id-known-before-starting-as-a-data-scientist-45989be6300e), Jason Goodman, Data Scientist @ Airbnb
* [Practical Proactivity in Data Science](https://www.quora.com/q/quoradata/Practical-Proactivity-For-Data-Science), Eric Mayefsky, ex-Head of Data Science @ Quora
* [The AI Hierarchy of Needs](https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007), Monica Rogati, ex VP Data @ LinkedIn