

# PREDICTIVE ANALYTICS And DATA MINING

PADM Spring 22 MASTERS EPITA

- Axel ULLERN

# INTRODUCTION

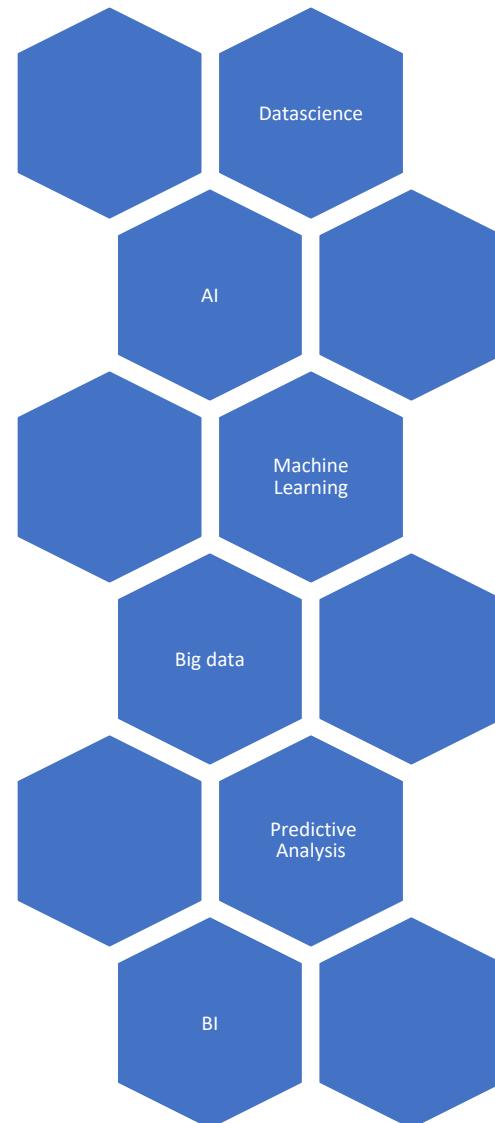


FUNDAMENTALS

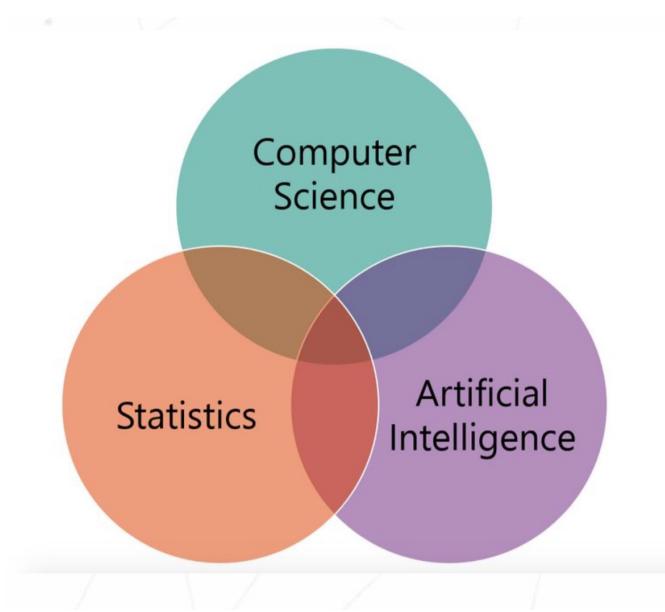


VOCABULARY  
USE CASES OF DATASCIENCE

# PIECES OF VOCABULARY



# And some of the definitions



**Datascience** = Create Business Value through Analysis of data Combining those 3 Technologies

**Machine Learning** = Machine Learning is an Artificial intelligence technology allowing computers to learn without having been explicitly programmed for this purpose. ...

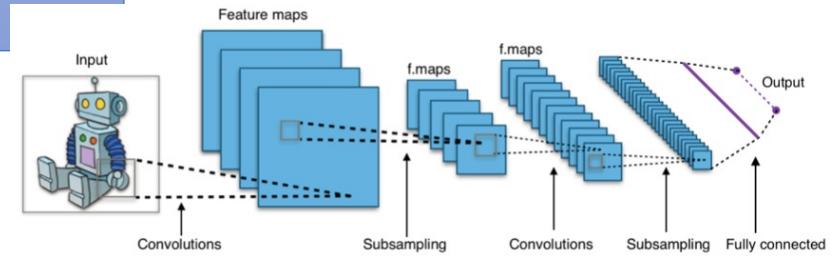
In fact, **Big Data** is the essence of Machine Learning and it is the technology that fully exploits the potential of Big Data.

**Deep Learning** = Class of machine Learning using sophisticated algorithms To solve complex problems

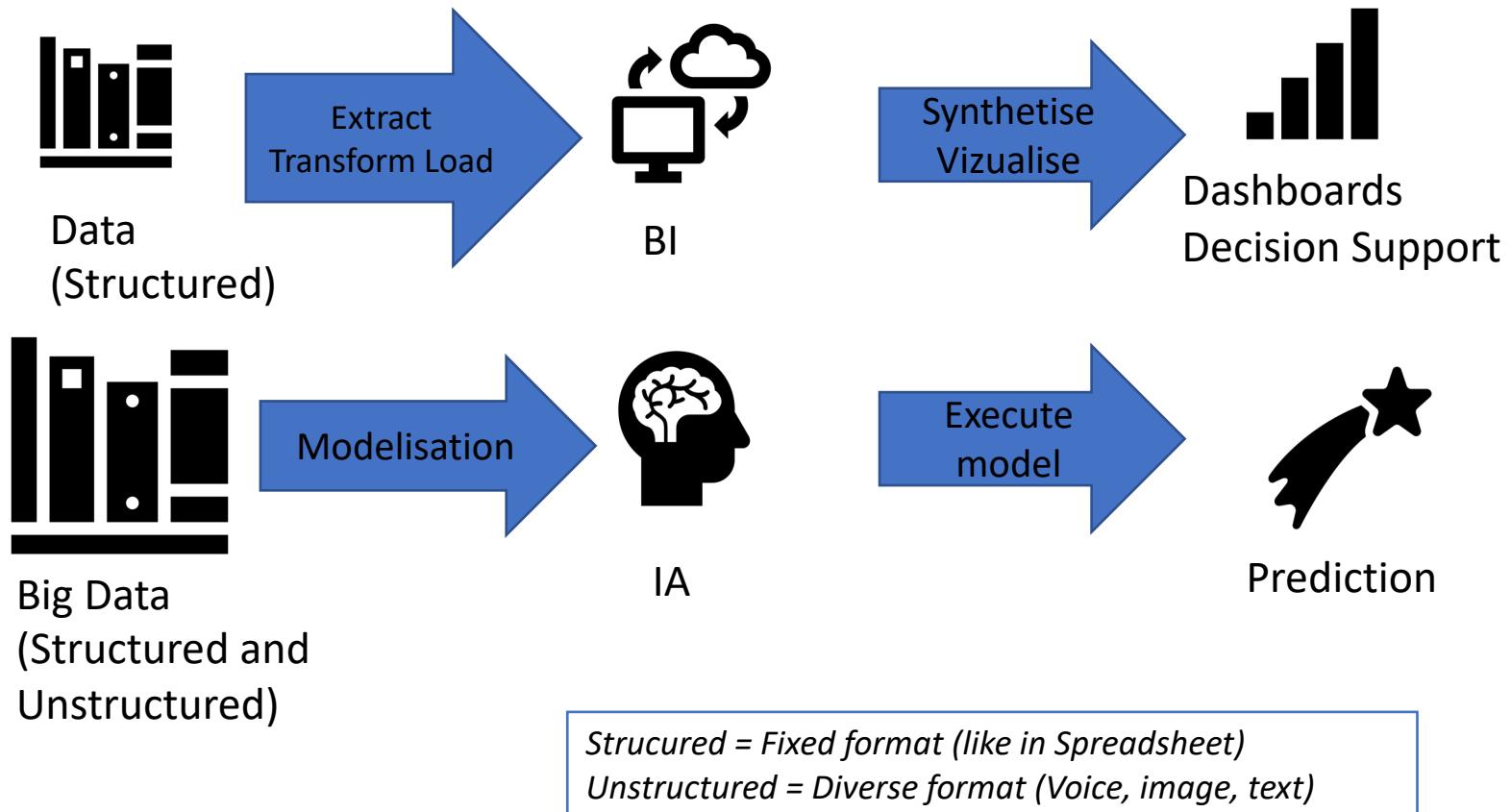
# Deep Learning

Deep learning uses hidden layers of artificial neural networks, and series of complex propositional calculations. Deep learning algorithms oppose shallow learning algorithms because of the number of transformations performed on the data between the input layer and the output layer.

**Many applications :**  
Speech recognition  
Natural Language Processing  
Image Analysis  
Illnesses Diagnosis,  
.....

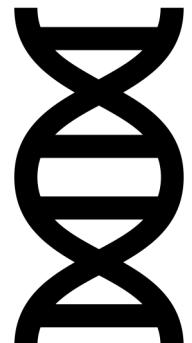
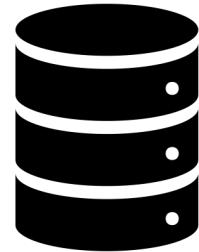
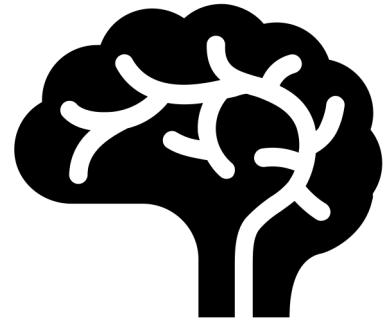


# BUSINESS INTELLIGENCE VS AI



# WHY SO MANY DATAS ?

- Walmart processes more than one million transactions per hour in its stores
- In medicine, epidemiology and genetics we operate billions of data
- Web logs easily get billions of data
- Facebook stores pictures by billion
- The IOT generate huge amounts of data
  
- Finally .. To make reliable predictions requires a lot of data without datas
- **IA or Machine Learning would not exist**



# CUSTOMER EXPERIENCE: DATA SOURCE

Awareness: advertising

Website (Pull) → contacts /Leads

marketing

marketing

Renewal

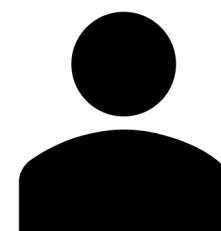
Sale

marketing

Loyalty

marketing

Relationship



Customer

marketing

Exploration

marketing

Evaluation

marketing

Purchase

Sale

# WHICH PREDICTIONS?



Health : illnesses,  
Tumors,...



Customers : Churn,  
recomandation  
engines,advertising..



Industry 4.0 : Anticipate  
failures, Predictive  
maintenance



Banks : Risk  
management



Security : Fraud  
detection , intrusions

# MAIN USES CASES for DATASCIENCE



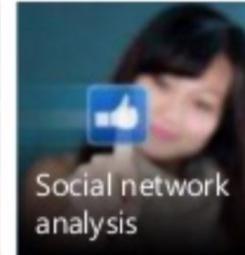
Intelligence  
Gathering



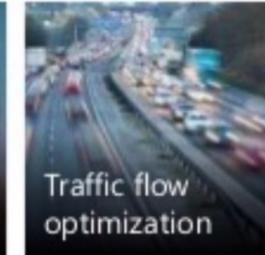
IT infrastructure  
& Web App  
optimization



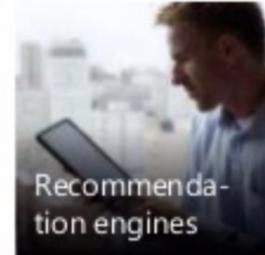
Legal  
discovery and  
document  
archiving



Social network  
analysis



Traffic flow  
optimization



Recommendation  
engines



Churn  
analysis



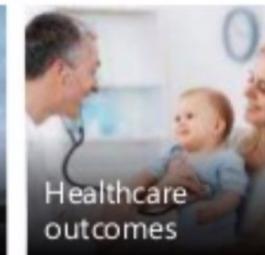
Location-based  
tracking &  
services



Oil & Gas  
exploration



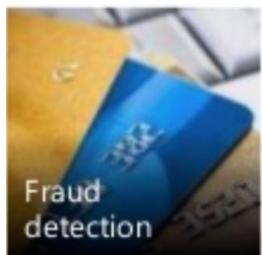
Weather  
forecasting for  
business  
planning



Healthcare  
outcomes



Personalized  
Insurance



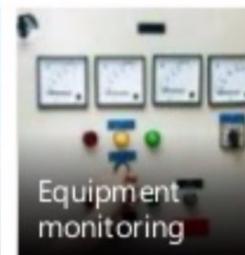
Fraud  
detection



Life sciences  
research



Advertising  
analysis



Equipment  
monitoring

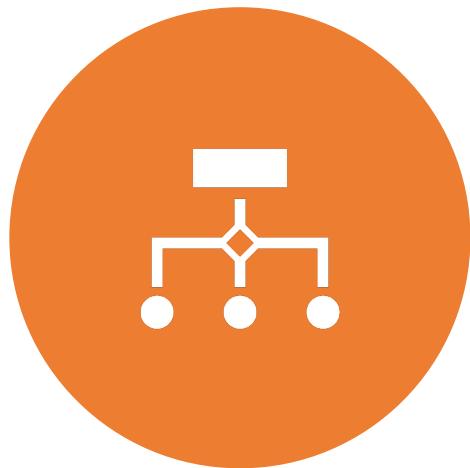


Pricing Analysis



Smart meter  
monitoring

# DATA IN REAL LIFE



HOW COMPANIES PROCESS  
DATA ?



WHAT IS A DATA ANALYSIS  
PROJECT?

# FIRST COMPANIES HAVE TO HAVE A **BUSINESS** PROBLEM TO SOLVE OR **BUSINESSES** ANSWERS TO FIND

- Why customers quit their contract ?
- Should we grant a loan to this customer ?
- Will our customers buy this new product ?
- When this machine will fail ?
- Why employees move to competitors ?



*Big Data and AI will probably provide answers*

*But **will never take decision** : companies have to take decisions*

*Based on those answers*

# Then they have to collect DATA FROM VARIOUS SOURCES



# THE PREP PHASE TAKES LONG TIME !



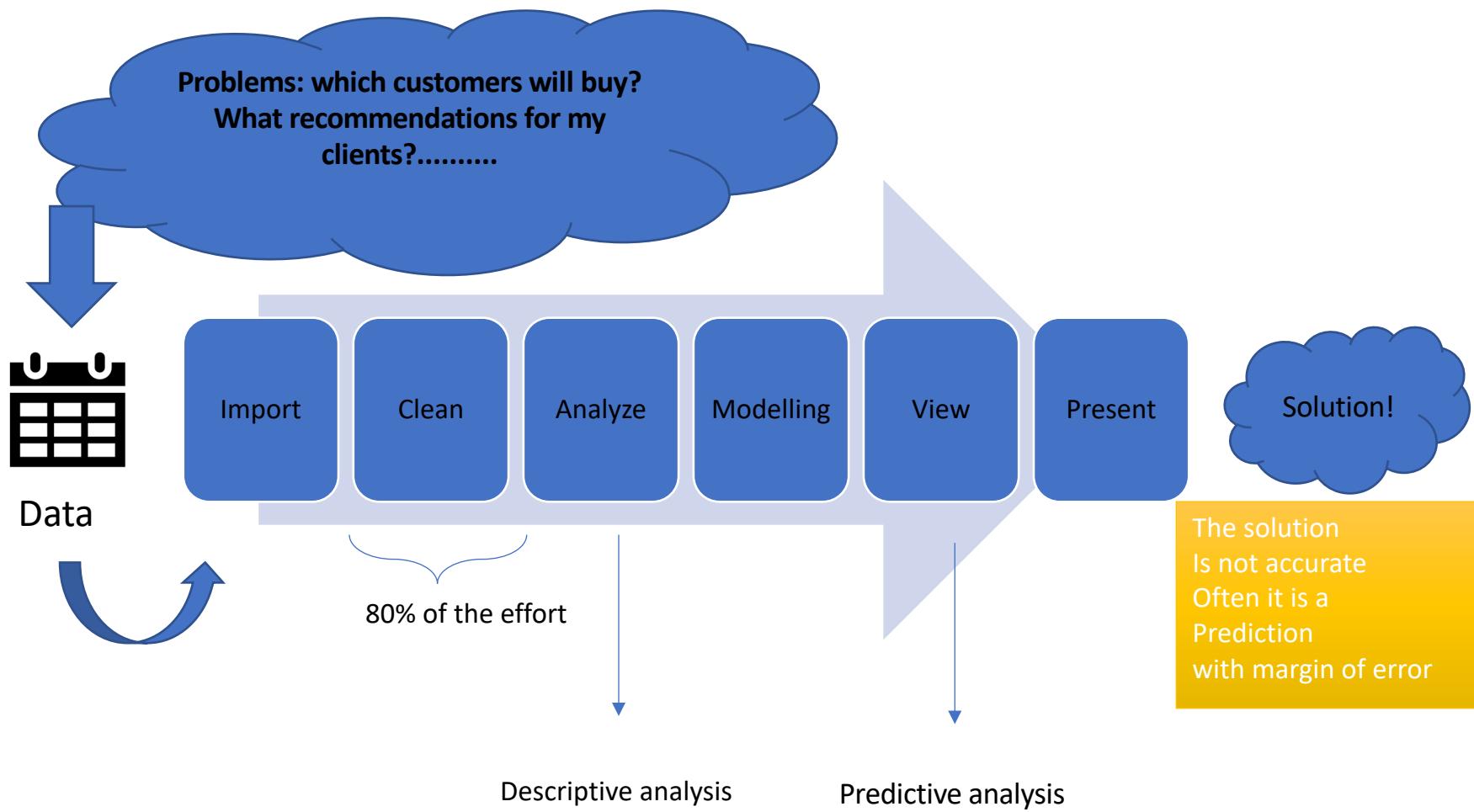
80% of the work to be  
done: data set  
preparation



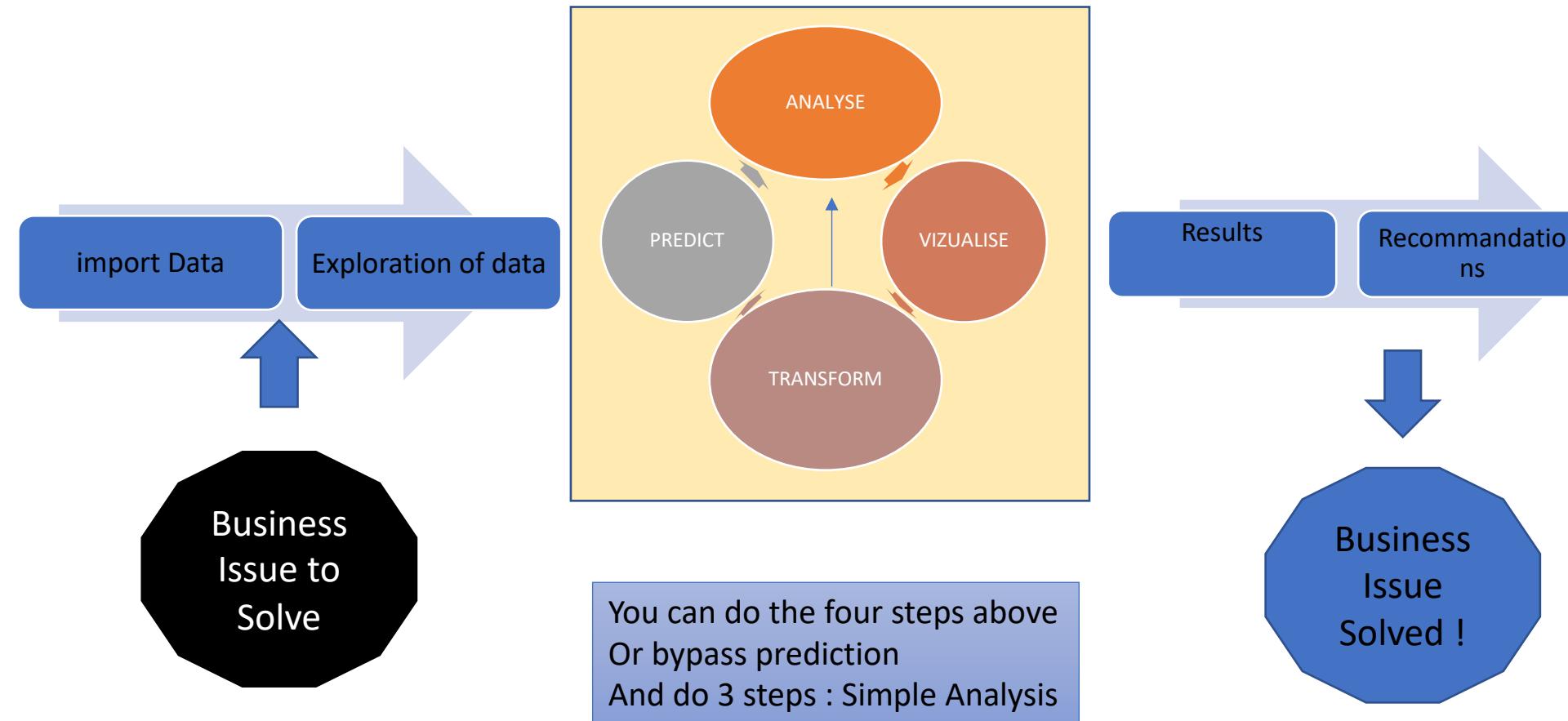
20% analysis



# From the problem to the solution

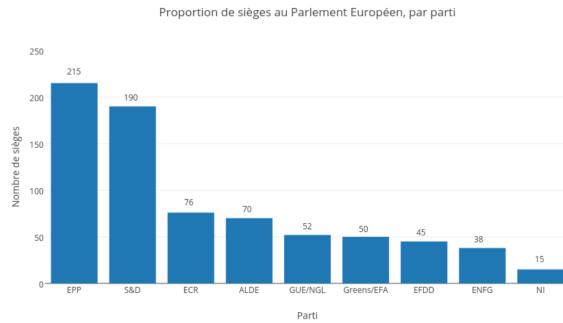


# THE ANALYTICAL PROCESS OF DATA

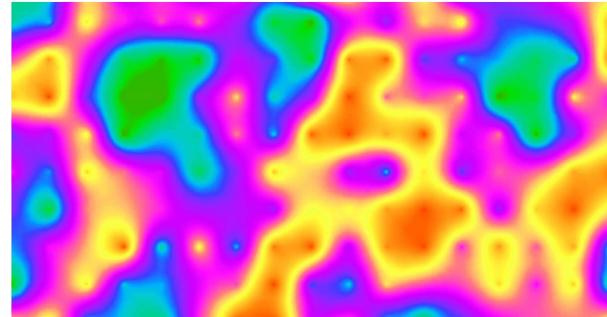


# FORGET EXCEL ! THE VOLUME CHANGE THE SCALE

When a small volume  
Data (100-1000) to be visualised : you can  
have this like in Excel spreadsheet

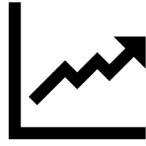


But when you have a **huge**  
Volume ( Millions or Billions)  
Data you get often this



# IMPORTANCE OF CHARTS AND PLOTS in BIG DATA

*" IT IS QUITE IMPOSSIBLE TO VIZUALISE LARGE AMOUNT OF DATA (BIG DATA) SO WE NEED synthesized the data USING STATISTICS AND GRAPH "*



Sometimes the graph is a cloud of points unreadable point cloud on which It is impossible to identify a trend so you have to transform and reduce the volume

# 2 DATA TYPES

1 **Continuous data** They can take any value (integer or decimal).

Eg temperature, pressure, humidity, sales, etc ..

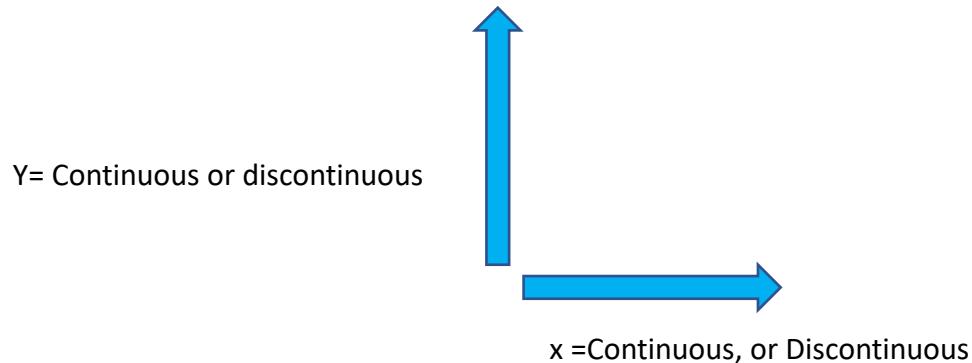
Or: logs onto a website

2 **Discontinuous or categorical data** They have predetermined values or fall into categories :

(BAC + 1 ,MASTERS, ...) or the questionnaire responses (yes, no, do not know)

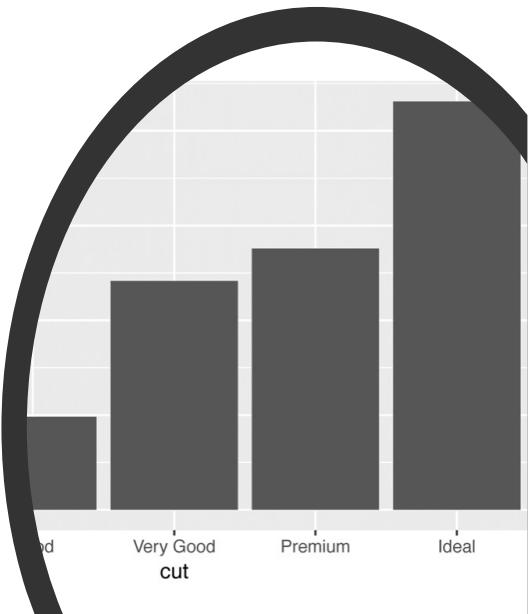
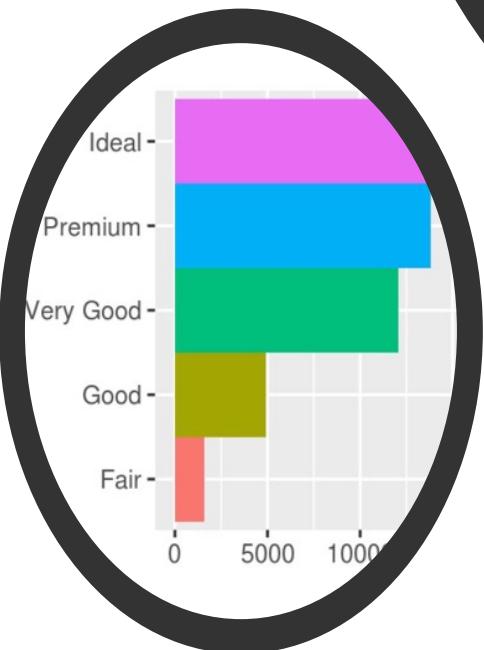
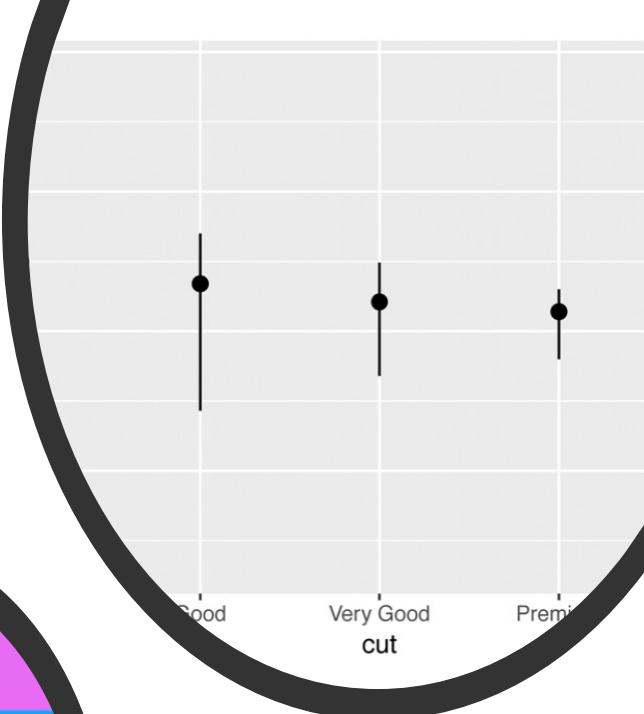
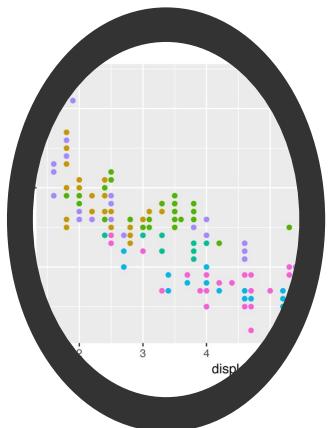
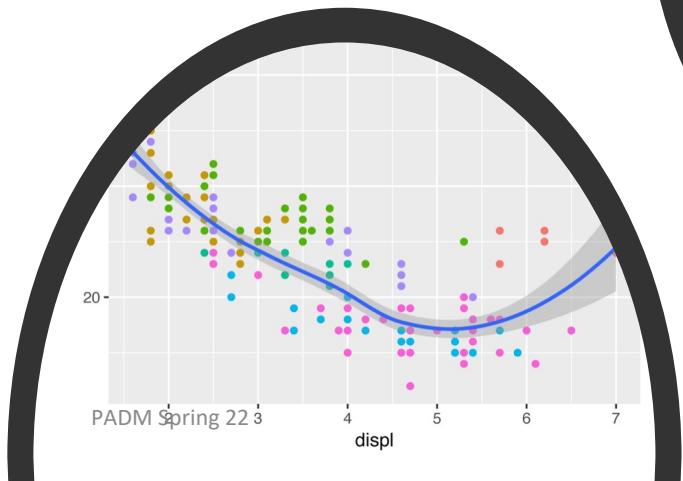
Or socio-professional categories ....

***Consequence*** : We get different kinds of plots to represent series of data



# EXAMPLES

- In all these plots (about diamonds) which kind of
- Data are represented ?
- How much series of data can be plotted
- On the same graph ?

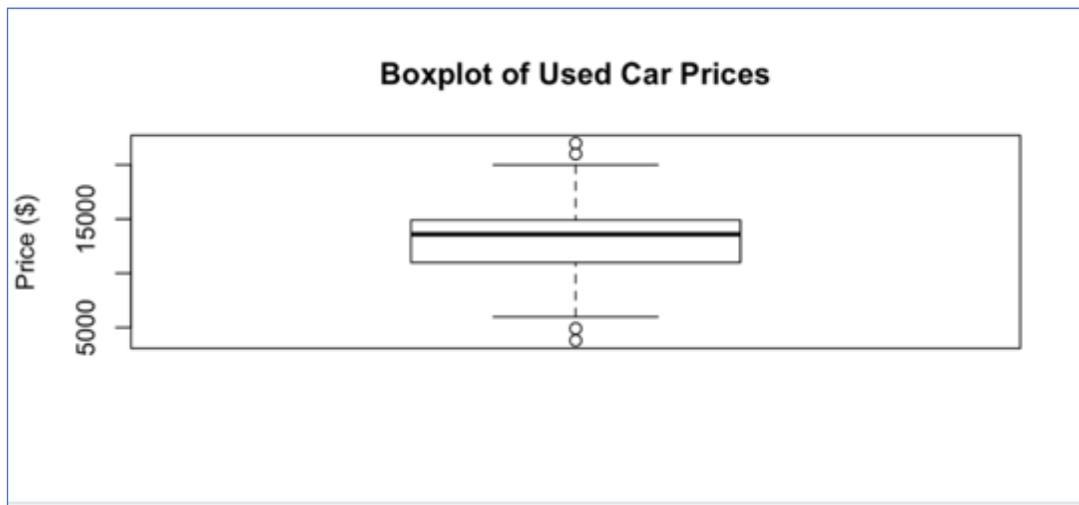


# GENTLE REMINDERS OF STATISTICS

MEDIAN vs MEAN  
BOXPLOT  
CORRELATIONS

## STATISTICS TOOLS

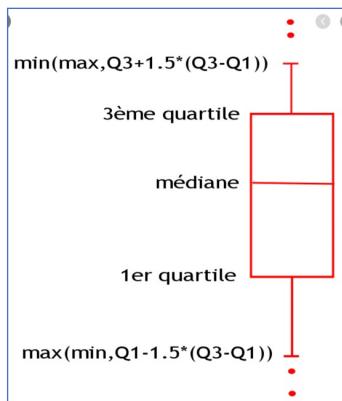
- Understanding the data
- Statistical tool:
- Means,
- Medians,
- Variance and Covariance , etc.
- Highlight correlations between variables



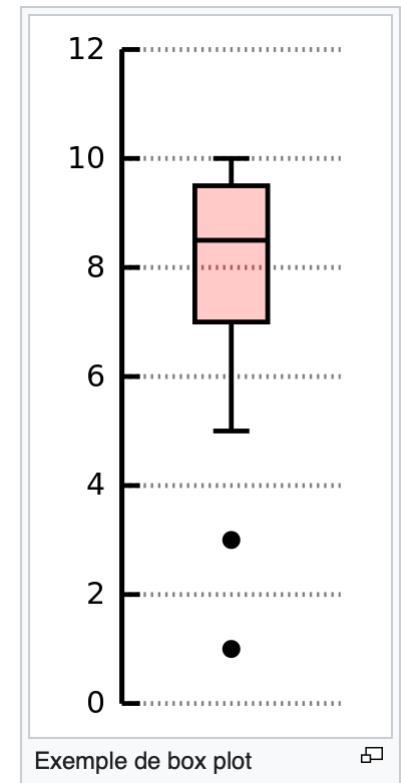
# Median and Boxplot

The series of numbers below must first be sorted in ascending order  
Then the boxplot is used to represent the statistical distribution of the series

$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$x_i$ (non triés)	9	6	7	7	3	9	10	1	8	7	9	9	8	10	5	10	10	9	10	8
$x_{(i)}$ (triés)	1	3	5	6	7	7	7	8	8	8	9	9	9	9	9	10	10	10	10	10



Median = middle of the series here average between  
The 10th and 11th so 8.5



# WHAT THE BOXPLOT SHOWS ?



When the box plot is **short** it implies that much of your data points are similar, since there are many values in a small range



When the box plot is **tall** it implies that much of your data points are quite different, since the values are spread over a wide range



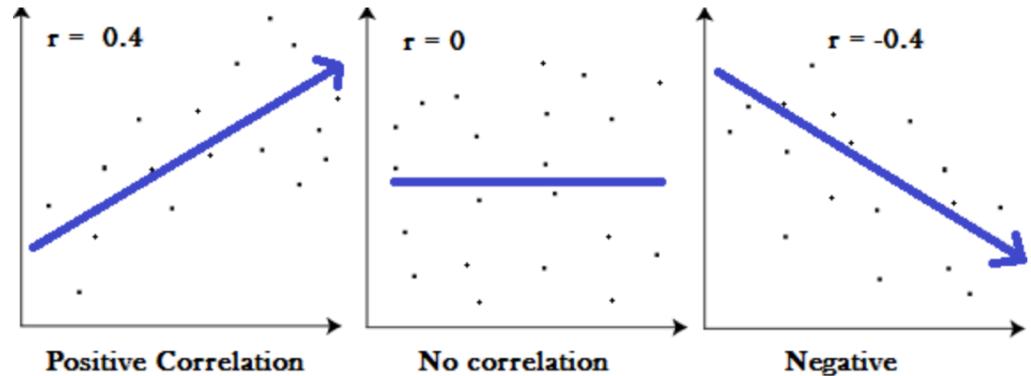
If the median value is closer to the **bottom** then we know that most of the data has lower values. If the median value is closer to the **top** then we know that most of the data has higher values. Basically, if the median line is not in the middle of the box then it is an indication of **skewed data**.



Are the whiskers **very long**? That means your data has a high **standard deviation** and **variance** i.e the values are spread out and highly varying. If you have long whiskers on one side of the box but not the other, then your data may be highly varying only in one direction.

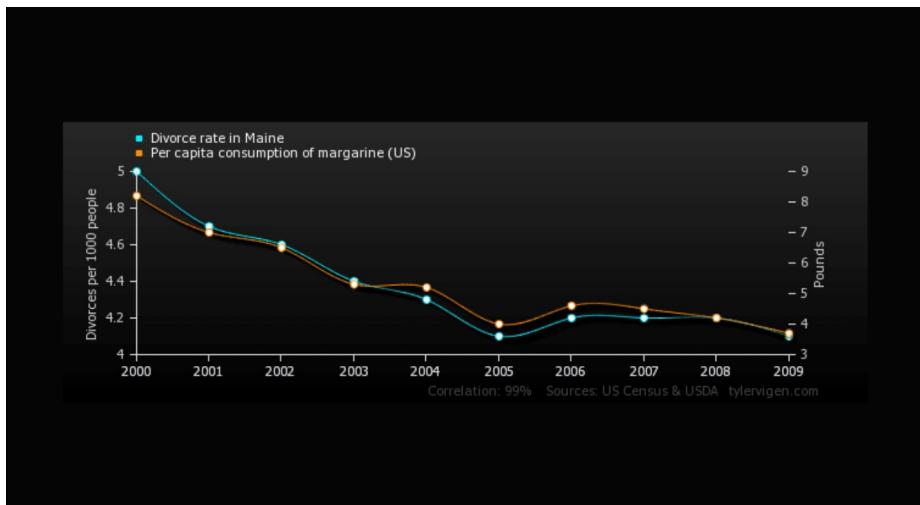
# CORRELATION

- What is correlation? Correlation is a **statistical measure that expresses the extent to which two variables are linearly related** (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect.



# CORRELATION AND CAUSATION

- Correlation or causation?
- A common reasoning error is to say, "X and Y are correlated, so X causes Y". We then confuse correlation and causation
- It could also be that:
  - Y causes X
  - X and Y have a common cause Z
  - X and Y are accidentally linked but have no link of causality.



Based on this graph  
One could pretend that  
Margarine consumption  
Cause divorces !

# If you have 2 continuous variables : check the correlation

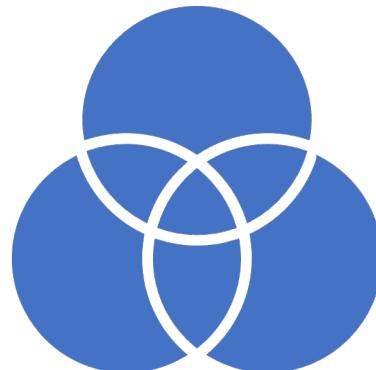
If we find a strong correlation  $>0.6$  or  $<-0.6$  then we ask ourselves the question: causality or not?  
It is then necessary to dig to check if there is a cause or if it is chance.

Some correlations are obvious and have no interest e.g. ice cream sales  
And outside temperature  
Others may have a real interest:  
Walmart found a correlation between sales of baby diapers and sales of  
Beer packs!



# Is there a correlation between a continuous and categorical variable?

- NO, IT MAKES NO SENSE! BOTH VARIABLES MUST BE CONTINUOUS TO CALCULATE THE CORRELATION
- NEVERTHELESS, IT IS INTERESTING TO MEASURE THE RECIPROCAL "INFLUENCE" OF A CONTINUOUS VARIABLE AND A CATEGORICAL VARIABLE.
- BUSINESS (HR) QUESTION: IS LEAVING OR STAYING IN THE COMPANY RELATED TO AGE?
- LEAVE THE COMPANY = CATEGORICAL VARIABLE (YES OR NO)
- AGE = CONTINUOUS VARIABLE
- HOW TO DO IT?
- ANSWER: DRAW TWO BOXPLOTS ONE CORRESPONDING TO THE EMPLOYEES WHO HAVE LEFT THE COMPANY THE OTHER TO EMPLOYEES WHO LEFT



# Attrition and Age: what relationship?

*Attrition = Leaving the company Attrition = Yes employee left , No= employee stayed*



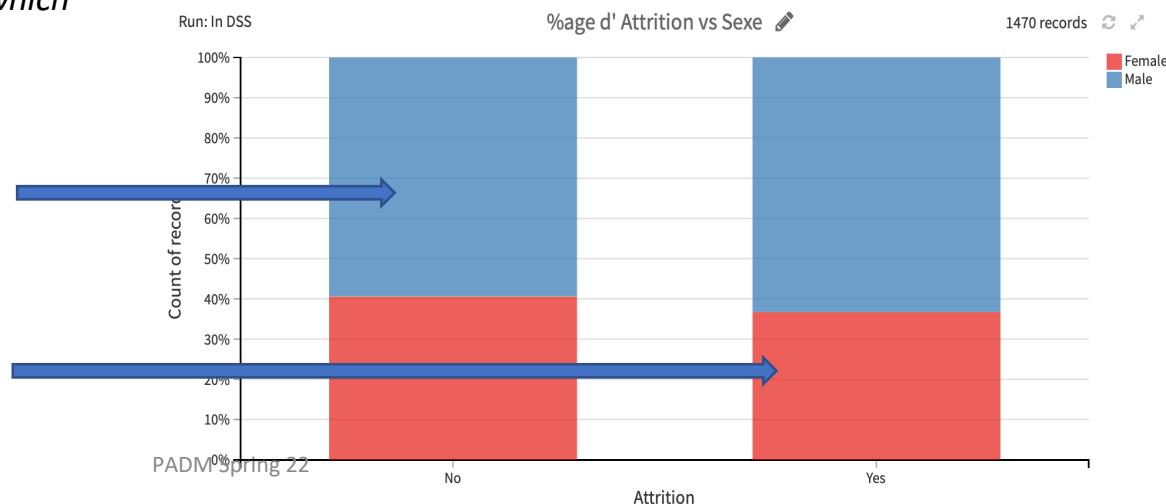
The positioning of the two boxes shows that employees who have left the company is overall younger than those who stayed. It is then necessary to look for the causes of this situation

# Links between two categorical variables

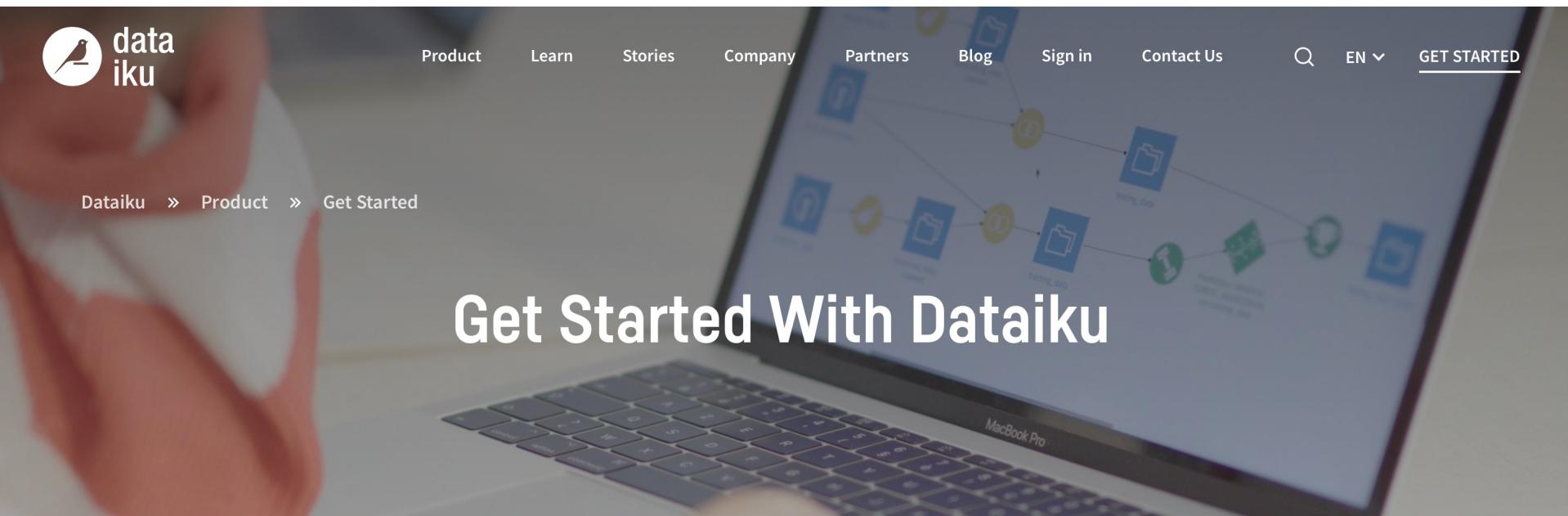
- Business Question: Are women or men most likely to leave
- The company?
- Male or Female = Categorical variable
- Exit or stay = Categorical variable
- How to do it? : a graph with two bars one for employees who left the other for those Who remained, inside each bar two colors for H or F, height of the bars = number of employees We compare the colored areas between the two bars

*By comparing the 2 blue or red zones*

*From each bar we see that there is a little less  
Of Women than Men in proportion to which  
Leave the company*



# And Now.....



# Key Company Milestones

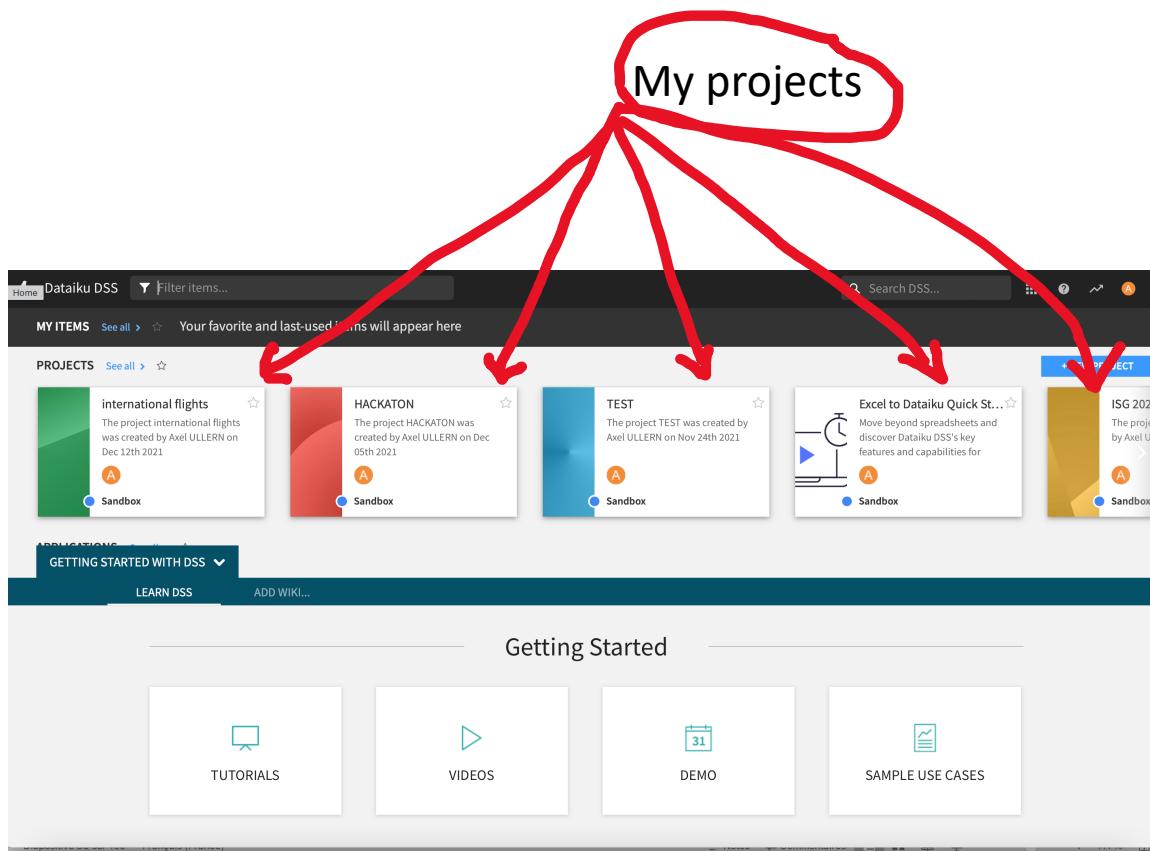
## A Brief History of Dataiku

	2013	2014	2015	2016	2017	2018	2019	2020
COMPANY	Dataiku Created 	20 <sup>th</sup> employee 	Office in New York 	Office in London 	100 <sup>th</sup> employee 	200 <sup>th</sup> employee 7 new offices in EMEA, APAC, Americas	400 <sup>th</sup> employee 	
CUSTOMERS	First customer	First 10+ users customers	50th customer	100th customer	1st CAB Customer Advisory Board 	300th customer	1000+ EGG Visitors	
SOFTWARE	Beta 	version 1 1st tool integrating data prep and ML 	version 2 Real-time collaboration Spark integration 	version 3 Scenarios, automation and version control 	version 4 Advanced scalability, cloud support 	version 5 Deep learning, Containerization, Project wikis 	version 6 Enterprise Scale Elastic AI with Kubernetes 	version 7 Explainable AI and Advanced Visual Statistics 

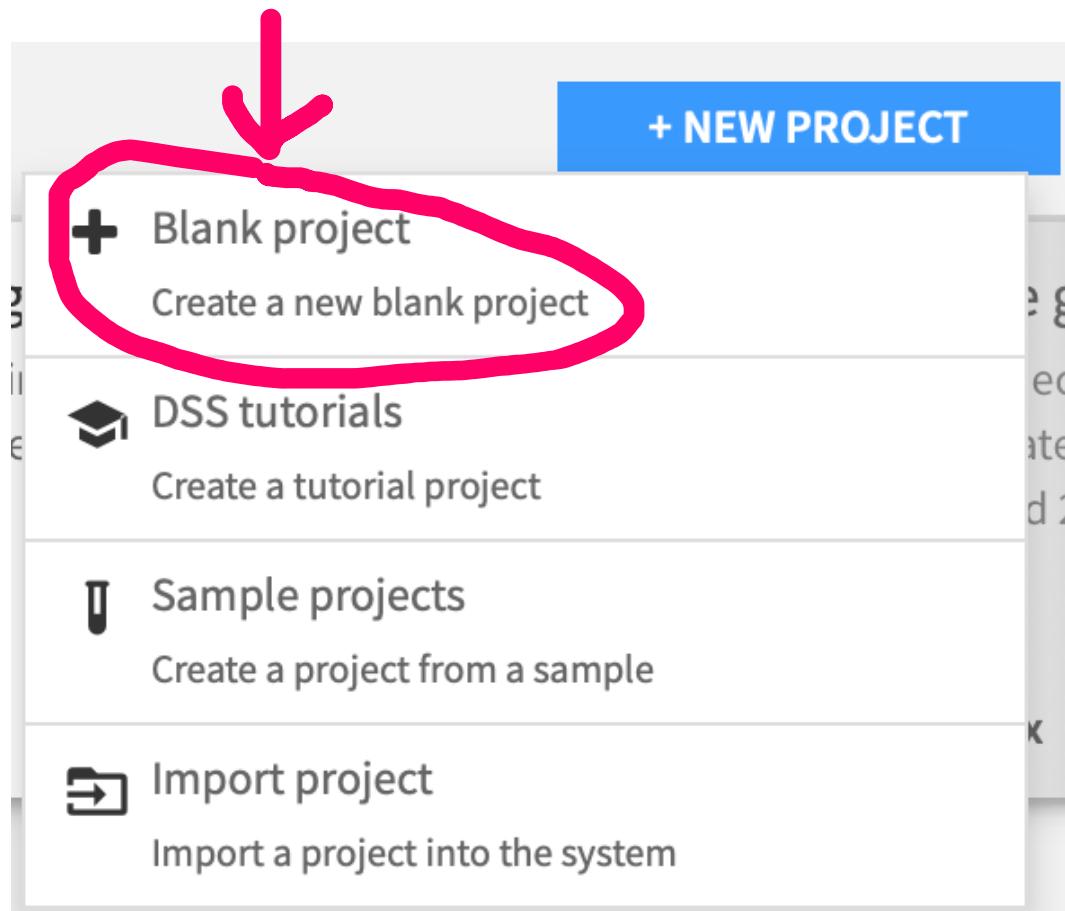
- GETTING STARTED WITH THE PLATFORM
- CREATING A PROJECT
- DATA IMPORT



# Home Page



# CREATING A PROJECT



# NEW PROJECT

+ New project - x

Name

Project Key   
The project key is used to reference datasets between projects. It cannot be changed once the project is created.

[CANCEL](#) [CREATE](#)

CREATE



**DIAMONDS**

The project *DIAMONDS* was created by Axel ULLERN on May 09th 2021

A master  
Click to add tags  
Sandbox

**Flow**

- 0 DATASETS
- 0 RECIPES
- 0 MODELS

**Lab**

- 0 NOTEBOOKS
- 0 ANALYSES

**Dashboards**

- 1 DASHBOARD

**Wiki**

- 0 ARTICLES

**Tasks**

- 3 TASKS

+ IMPORT YOUR FIRST DATASET



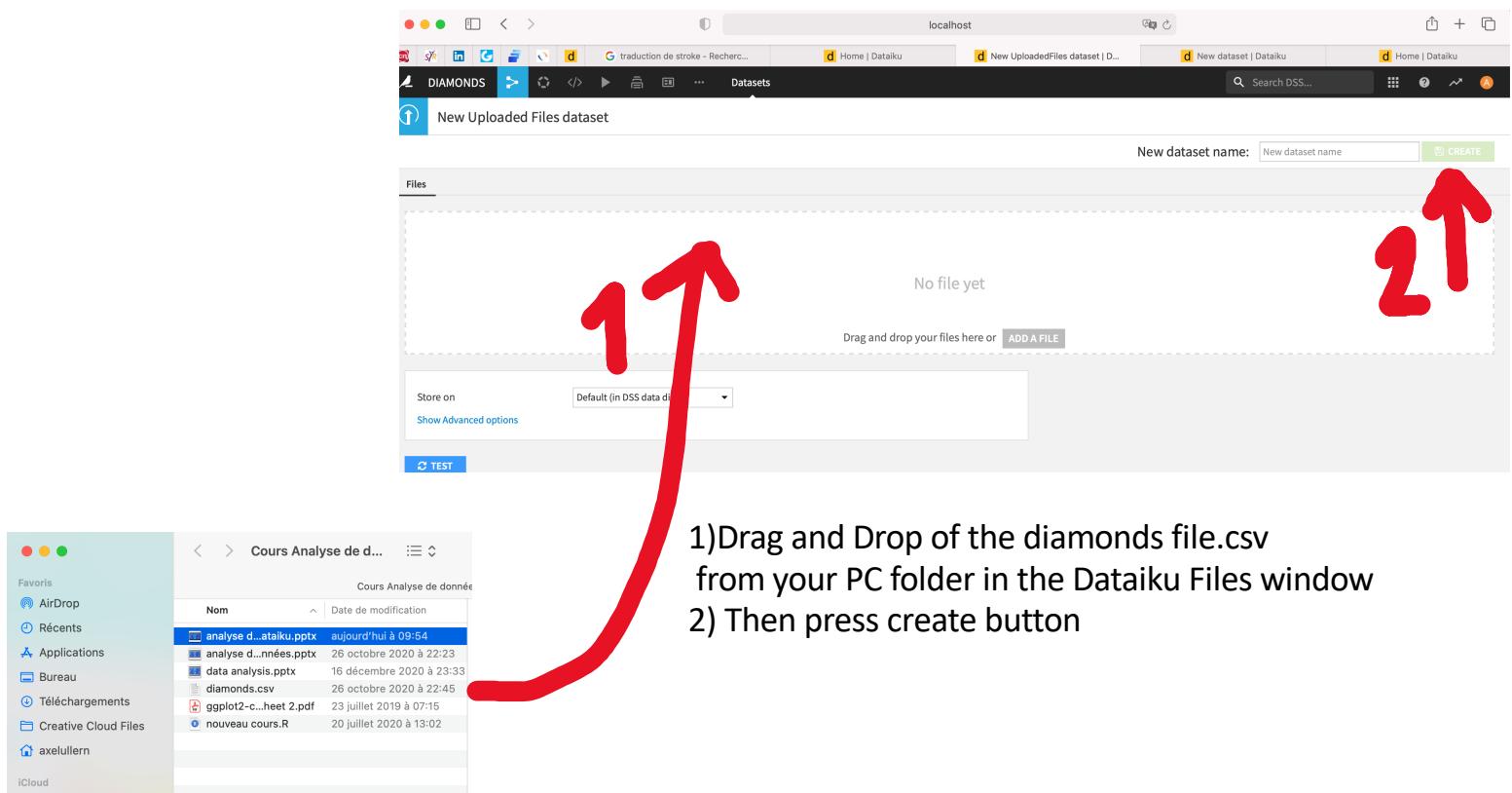
Import your data

# UPLOAD YOUR FILE

The screenshot shows the DataStax Studio (DSS) interface. At the top, there's a navigation bar with icons for Home, DIAMONDS, and various project-related buttons. A search bar on the right says "Search DSS...". Below the navigation is a title "New dataset". The main area contains several cards:

- Files**: Contains links to "Upload your files", "Server's filesystem", and "Files in folder".
- Hadoop**: States "Hadoop connection is not enabled on your DSS instance. Please contact your administrator".
- SQL**: Lists supported databases: MySQL, PostgreSQL, Vertica, Amazon Redshift, Snowflake, IBM Netezza, Other SQL, Greenplum, Teradata, Oracle, MS SQL Server, SAP Hana, Google BigQuery, and Athena.
- Cloud Storages**: Lists cloud storage providers: Amazon S3, Azure Blob Storage, Google Cloud Storage, FTP, SFTP, SCP, HTTP, and HTTP (with cache).
- NoSQL**: Lists MongoDB, Cassandra, and ElasticSearch.
- DSS**: Lists dataset management options: Managed dataset, Folder, Dataset from another project, Metrics, Internal stats, and Editable.
- Import existing**: Allows choosing a connection to import from or importing from a catalog.
- More dataset options?**: Provides a link to browse available plugins.

A large red arrow points from the bottom left towards the "Upload your files" link in the Files section.



# DATA PREPARATION

DIAMONDS DATASET

DATA DISCOVERY

1ST SIMPLE ANALYSES

# Loaded dataset

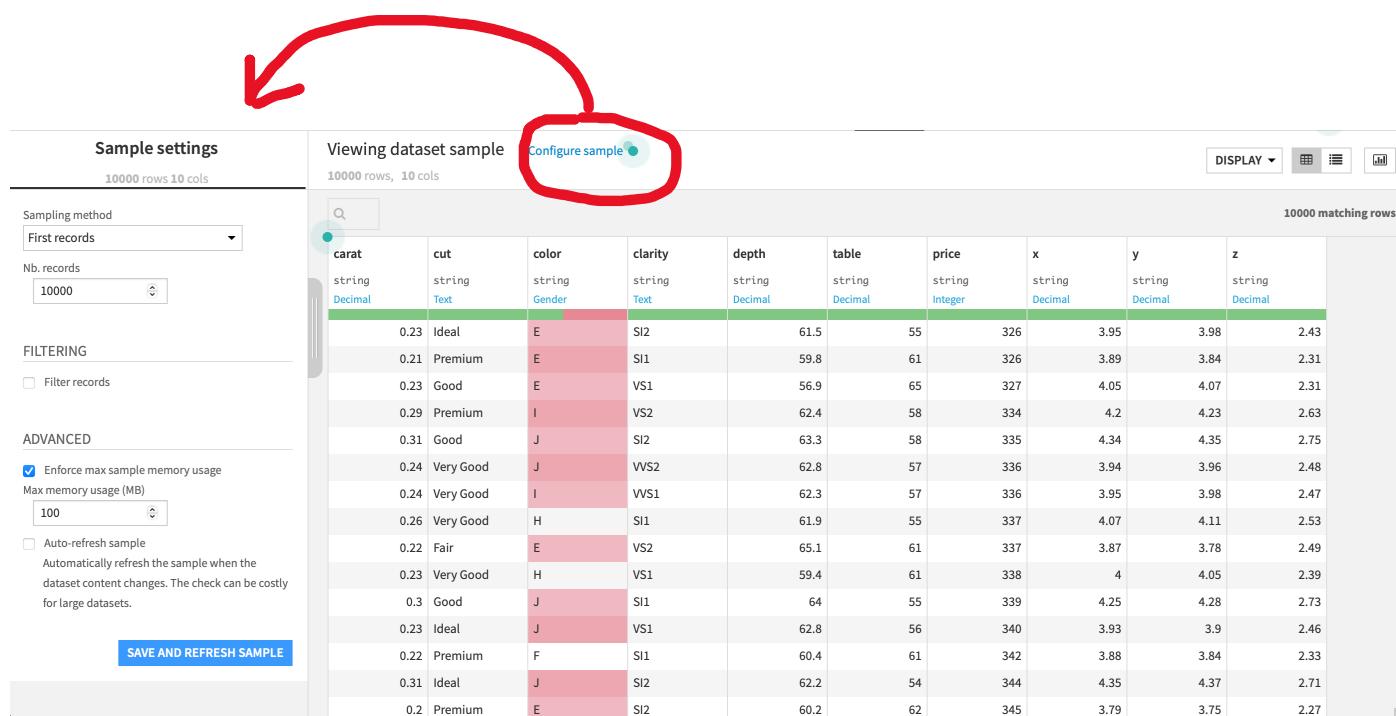
Up diamonds 

Viewing dataset sample [Configure sample](#) 

10000 rows, 10 cols

carat	cut	color	clarity	depth	table	price	x	y	z	---
string	string	string	string	string	string	string	string	string	string	---
Decimal	Text	Gender	Text	Decimal	Decimal	Integer	Decimal	Decimal	Decimal	---
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43	
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31	
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31	
0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63	
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75	
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48	
0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47	
0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53	
0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49	
0.23	Very Good	H	VS1	59.4	61	338	4	4.05	2.39	
0.3	Good	J	SI1	64	55	339	4.25	4.28	2.73	
0.23	Ideal	J	VS1	62.8	56	340	3.93	3.9	2.46	
0.22	Premium	F	SI1	60.4	61	342	3.88	3.84	2.33	
0.21	Good	I	SI2	66.0	54	344	4.25	4.27	2.71	

# Configure sampling with configure sample



The screenshot shows a data visualization interface with two main sections: 'Sample settings' on the left and 'Viewing dataset sample' on the right.

**Sample settings:**

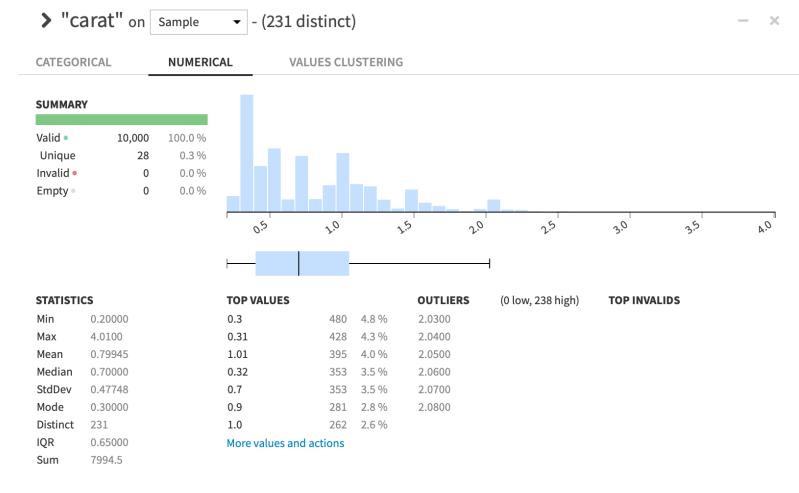
- Sampling method:** First records
- Nb. records:** 10000
- FILTERING:** Filter records (unchecked)
- ADVANCED:**
  - Enforce max sample memory usage
  - Max memory usage (MB):** 100
  - Auto-refresh sample
  - Automatically refresh the sample when the dataset content changes. The check can be costly for large datasets.
- SAVE AND REFRESH SAMPLE** button

**Viewing dataset sample:**

- Configure sample** button (highlighted with a red oval and a red arrow pointing from the title).
- DISPLAY** dropdown menu with icons for list, grid, and chart.
- 10000 matching rows** message.
- Table Headers:** carat, cut, color, clarity, depth, table, price, x, y, z.
- Table Data:** A grid of 15 rows and 10 columns. The first few rows show data for carats ranging from 0.20 to 0.31, with various cut, color, and clarity grades.

# ANALYZE EACH COLUMN

- To the right of the name of each column A small triangle
- Click on this triangle and select Analyze



# 1st Analyses

- What is the largest carat?
- What is the average price?
- What percentage of diamonds of the ideal cut?

# Transformations and visualisation

---

TRANSFORMING DATA WITH RECIPES  
VISUALIZE TRANSFORMATIONS WITH FLOW

# RECIPES

Recipes are used to transform the dataset, these are simple operations such as:

Cleaning

Renaming

Creating a new column,

Changing date formats

Etc...

The screenshot shows the Alteryx interface with two main panels. On the left is the 'Viewing dataset sample' panel, which displays a sample of the 'diamonds' dataset with 53940 rows and 10 columns. The columns are: carat, cut, color, clarity, depth, table, price, x, y, and z. The data types are listed above the first row: carat (double), cut (string), color (string), clarity (string), depth (string), table (string), price (double), x (string), y (string), and z (string). The first few rows of data are shown. On the right is the 'LAB' panel, which contains the 'Visual recipes' palette. This palette includes various icons for different operations: Sync, Prepare, Sample / Filter, Group, Distinct, Window, Join with..., Split, Top N, Sort, Pivot, and Stack. A red oval highlights the 'Visual recipes' section. A blue arrow points from the text 'When the dataset is open click actions' down to the 'LAB' panel. Another blue arrow points up from the text 'Visual Recipes are very useful To prepare to clean up a dataset' towards the 'Visual recipes' section.

When the dataset is open click actions

Visual Recipes are very useful  
To prepare to clean up a dataset



## Dataiku creates a new dataset after running recipe.

---

- This ensures that the initial data is not overwritten and allow returned to initial dataset at any time.
- DSS proposes a new name: it is often useful to put a name that corresponds to the transformation performed

# Example: Creating a New Column

In the game diamonds we want to create a volume column whose content is the multiplication of the 3 dimensions x, y, and z (last 3 columns of the dataset)

Step 1: Recipe Prepare Selection



Step 2: Give the new dataset a name  
Then create the recipe

The screenshot shows the 'New data preparation recipe' dialog box. It has two tabs: 'Input dataset' (selected) and 'Output dataset'. Under 'Input dataset', the 'diamonds' dataset is selected. Under 'Output dataset', the 'Name' field is filled with 'diamonds\_xyz'. The 'Store into' dropdown is set to 'filesystem\_managed' and the 'Format' dropdown is set to 'CSV'. At the bottom right, there are 'CANCEL' and 'CREATE RECIPE' buttons, with 'CREATE RECIPE' being highlighted by a red arrow.

# Recipe making

Step 3: Build the recipe script and select the processors

Script

Visualisation des transformations

The screenshot shows the DIAMONDS software interface. At the top, there's a navigation bar with 'DIAMONDS', 'compute\_diamonds\_xyz', and various icons. Below it, a 'Recipes' tab is selected. The main area has two tabs: 'Script' (which is active) and 'Visualisation des transformations'. The 'Script' tab shows a table of diamond data with columns like carat, cut, color, clarity, depth, table, price, x, y, and z. Below the table, there are several status messages and a red box highlighting the 'ADD A NEW STEP' button. A blue arrow points upwards from the bottom of the screen towards this button.

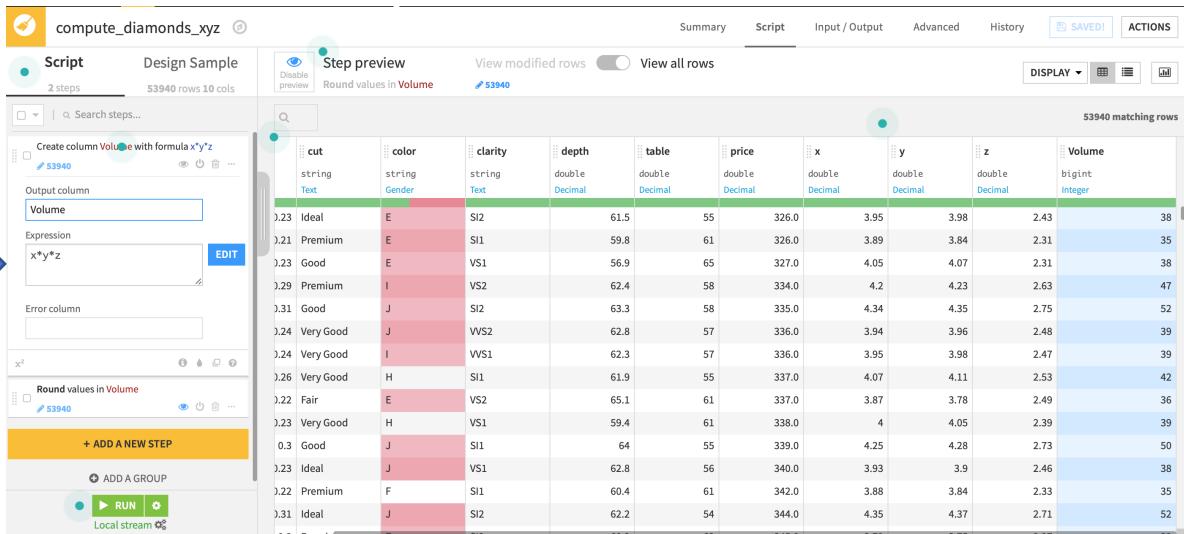
Allows you to add a new  
Step in the script and open the window for choosing processors

Processors library	
<input type="checkbox"/> Filter data	13
<input type="checkbox"/> Data cleansing	11
<input type="checkbox"/> Strings	11
<input checked="" type="checkbox"/> Math / Numbers	15
<input type="checkbox"/> Split / Extract	7
<input type="checkbox"/> Web logs	6
<input type="checkbox"/> Dates	8
<input type="checkbox"/> Geography	9
<input type="checkbox"/> Enrich	11
<input type="checkbox"/> Reshaping	16
<input type="checkbox"/> Natural Language	5
<input type="checkbox"/> Joins	3
<input type="checkbox"/> Complex objects	7
<input type="checkbox"/> Code	4
<input type="checkbox"/> Misc	10
<input type="checkbox"/> Formula	
<input type="checkbox"/> Extract numbers	
<input type="checkbox"/> Split currencies in column	
<input checked="" type="radio"/> Compute quantile	
<input type="checkbox"/> Filter rows/cells on numerical range	
<input type="checkbox"/> Flag rows on numerical range	
<input type="checkbox"/> Filter rows/cells with formula	
<input type="checkbox"/> Flag rows with formula	
<input type="checkbox"/> x <sup>2</sup> Convert number formats	
<input type="checkbox"/> Convert a UNIX timestamp to a date	
<input type="checkbox"/> Discretize (bin) numerical values	

Here the chosen processor is  
Maths/Numbers because we want to  
Create a new column  
Calculated from the others  
existing columns

# Creating the new column in the recipe

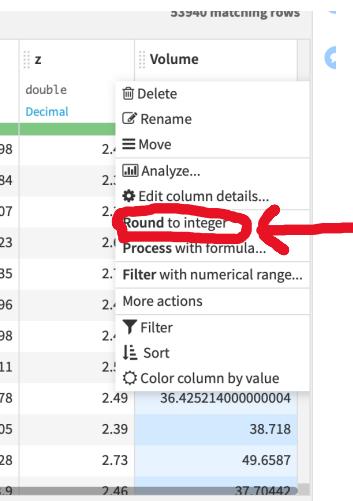
Step 4: we name the new volume column and indicate the calculation formula:  
 $\text{Volume} = \text{x} * \text{y} * \text{z}$  ( $\text{x}$ ,  $\text{y}$  and  $\text{z}$  are the names of the last 3 columns of the dataset)



The screenshot shows the Dataiku DSS interface for the 'compute\_diamonds\_xyz' recipe. On the left, there's a sidebar with 'Script' and 'Design Sample' tabs. Under 'Script', there are two steps: 'Create column Volume with formula x\*y\*z' and 'Round values in Volume'. The 'Output column' dropdown is set to 'Volume'. The 'Expression' field contains the formula 'x\*y\*z'. The main area shows a preview of 53940 rows with columns: cut, color, clarity, depth, table, price, x, y, z, and Volume. The 'Volume' column is of type bigint. At the bottom, there are buttons for '+ ADD A NEW STEP', '+ ADD A GROUP', and a run button.



The new column is created



A context menu is open over the 'Volume' column. The options listed are: Delete, Rename, Move, Analyze..., Edit column details..., Round to integer (which is circled in red), Process with formula..., Filter with numerical range..., More actions, Filter, Sort, Color column by value, and a specific row value (36.425214000000004). The 'Volume' column is currently set to 'Decimal'.

Allows rounding  
Decimal digits

# Last step: Execute the recipe

- Click on Run and wait for the "Job Succeeded" message
- **WARNING: AS LONG AS THIS OPERATION HAS NOT BEEN DONE THE NEW DATASET IS NOT CREATED!**

The screenshot shows a data processing interface with the following details:

- Left Panel (Steps):**
  - Create column **Volume** with formula  $x*y*z$  (53940 rows)
  - Round values in **Volume** (53940 rows)
  - Column **Volume**: single | multiple | pattern | all
  - Rounding mode: Round
  - Significant digits: 0
  - Decimal places: 0
- Table View:** A grid showing 53940 matching rows. The columns are: cut, color, clarity, depth, table, price, x, y, z, Volume. The Volume column is of type bigint and Integer.
- Bottom Panel:**
  - + ADD A NEW STEP
  - RUN button (highlighted with a blue arrow pointing up from the bottom left)
  - Local stream icon
- Status Bar:** Job succeeded. View details. Explore dataset diamonds\_xyz.

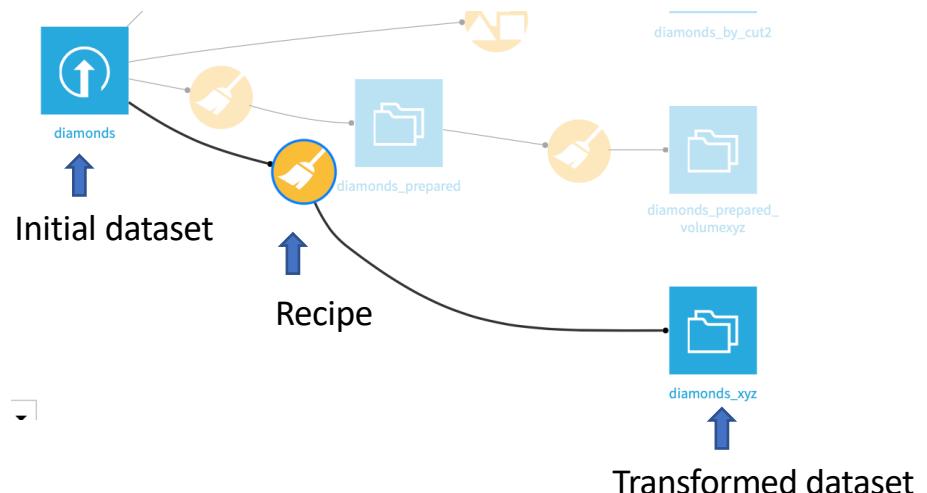
# FLOW



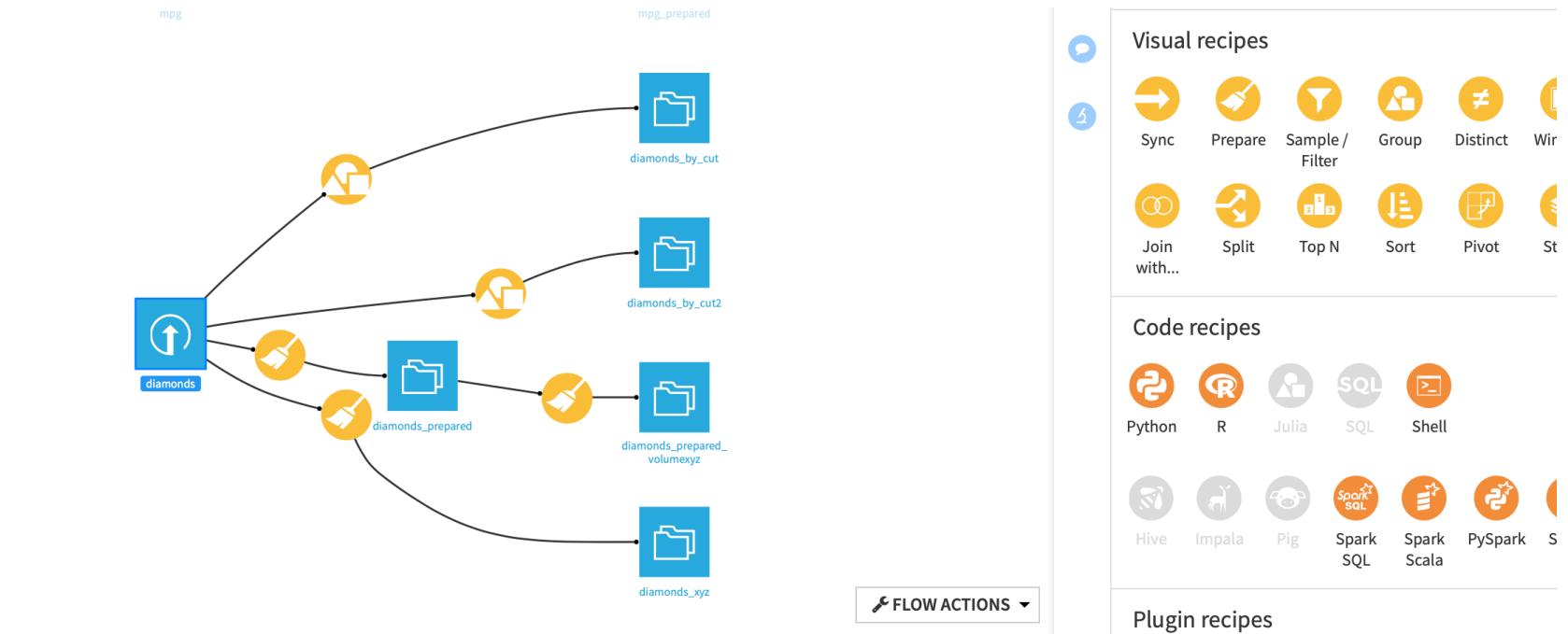
It is accessible at any time with this icon ↑

It shows datasets and recipes

It allows you to see all the stages of a project



# Flow example



From the dataset diamonds we created 5 new datasets and 5 recipes (2 groups and 3 preparations)

# ANOTHER RECIPE: FILTERING

Objective: This is to filter the rows of the table on the basis of criteria (e.g. prices < \$ 1000, carats >3)

Filtering with recipe:

The screenshot shows the Dataiku interface with the 'diamonds' dataset selected. On the left, there's a sidebar with 'ACTIONS' and various icons like 'View in Row', 'Tag', 'Export', and 'Publish'. Below it is the 'LAB' button. Under 'Visual recipes', several icons are listed: Sync, Prepare, Sample / Filter, Top N, Sort, Pivot, and Stack. A red arrow points from the 'Sync' icon towards the main workspace.

**No step yet in your script.**

You can add steps either by:

- Clicking on column headers.
- Clicking on cells.
- Highlighting some text.
- Paste if you've copied steps.

OR

+ ADD A NEW STEP

**Processors library**

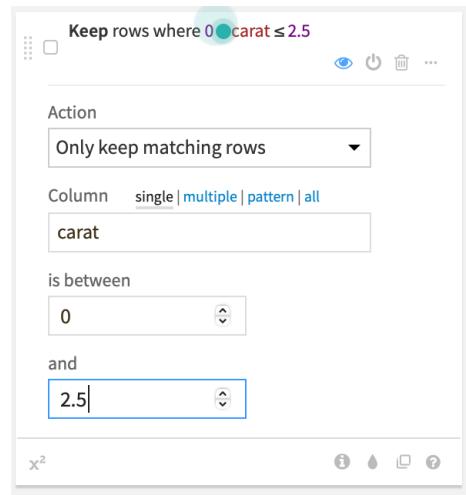
Processor	Count
Filter data	13
Data cleansing	11
Strings	11
Math / Numbers	15
Split / Extract	7
Web logs	6
Dates	8
Geography	9
Enrich	11
Reshaping	16
Natural Language	5
Joins	3
Complex objects	7
Code	4
Misc	10

A red arrow points from the 'ADD A NEW STEP' button towards the 'Processors library' panel.

# Practice

Create a dataset from the diamond set containing the carats $\leq 2.5$

Here's what to  
Put in recipe  
Filtering

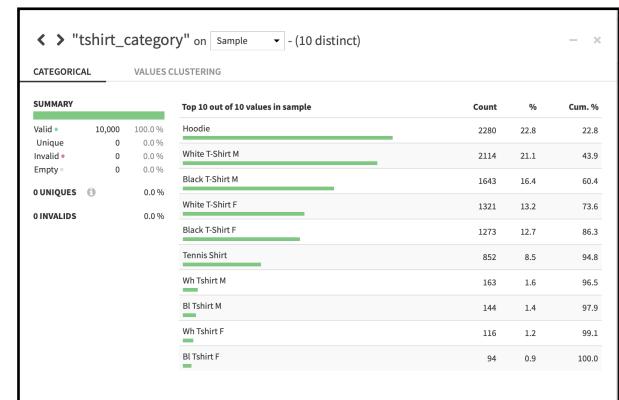


*You will examine the transformed dataset to check in the carat column that all carat are now  $>2.5$  then run the recipe and observe the Flow*

# DATA PREPARATION

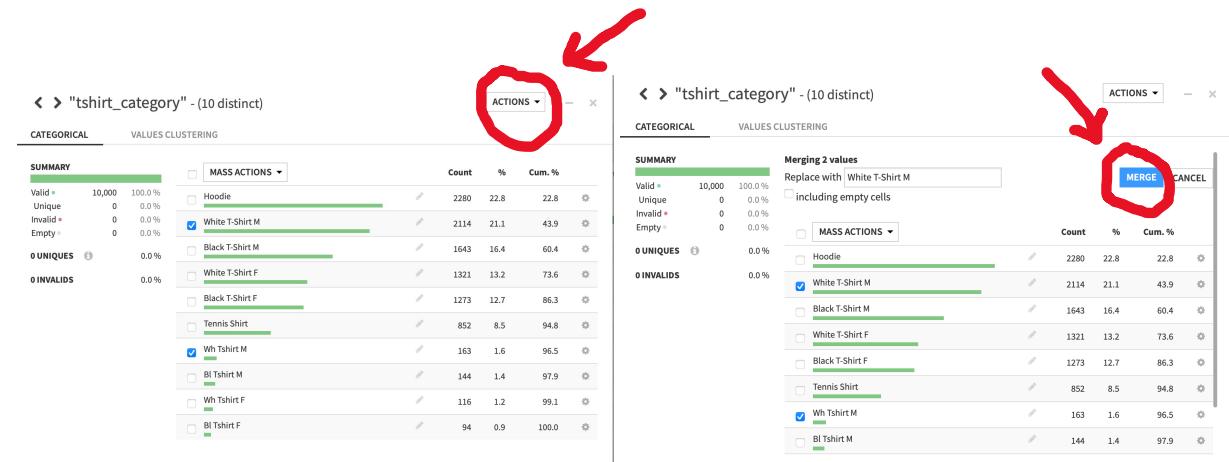
*By clicking on the triangle  
Next to the column  
tshirt\_category then Analyze  
The Marketing Product Manager  
Realizes that many  
Codification errors have been made:  
White T shirt M and Wh Tshirt M are the  
same  
Products and must be grouped in  
The same category.*

- Many datasets have errors in the coding (labels) of the data
- that may distort the analysis, Dataiku will very easily allow us to change these codifications
- Ex: load into Dataiku the dataset "orders" which includes sales of T-Shirt.



# Merge product categories into 1 single

- Create a prepare recipe and then directly in the recipe click on the t shirt category column
- And select the two categories to merge then click on ACTIONS then MASS ACTIONS, accept or change the name of the merged category then click MERGE





## Exercise: Finish the recipe

---

- Merge the other categories of erroneous Tshirt with the right categories in the same way
- Then do RUN when the recipe is ready
- Examine the flow
- Open the modified dataset
- Verify that all categories are correct

# ANALYSIS AND INSIGHTS

Choosing data to visualise

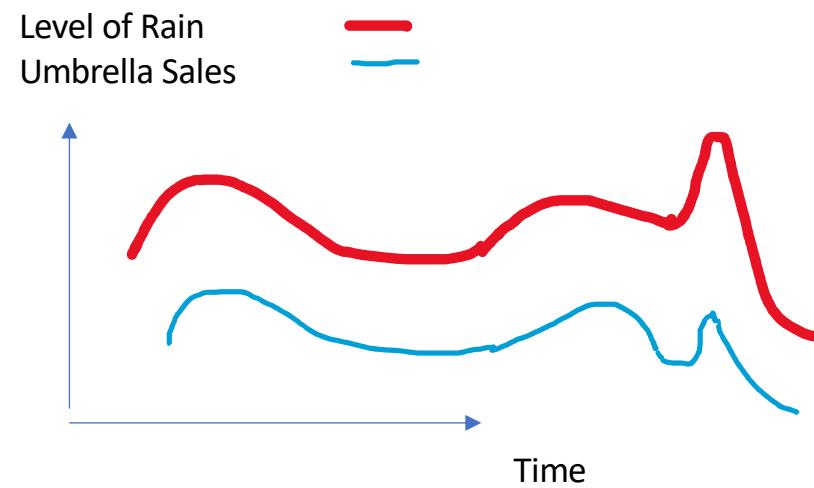
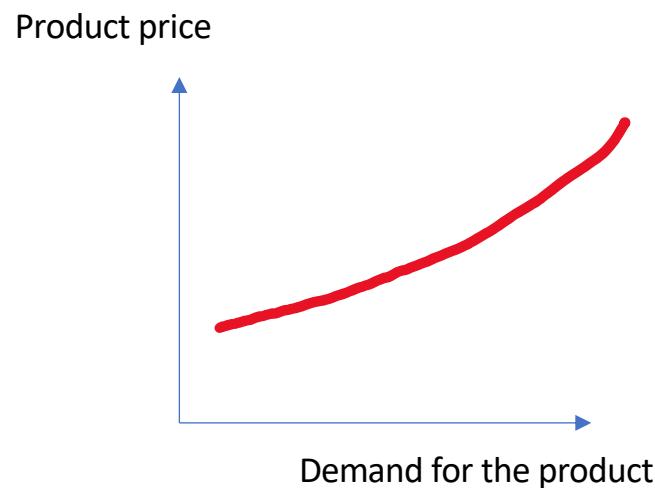
Creating graphs

Get insights from graphs

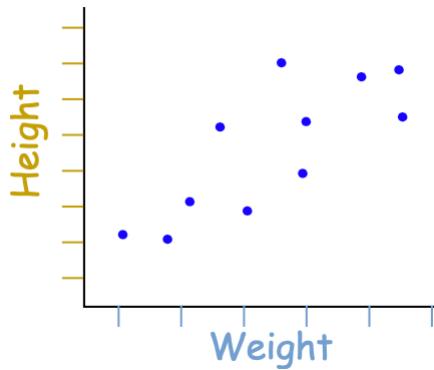
# What do we see on a graph ?

This makes possible to see the **link (or influence)** between two variables: the one that is on the abscissa and the one that is in ordinates.

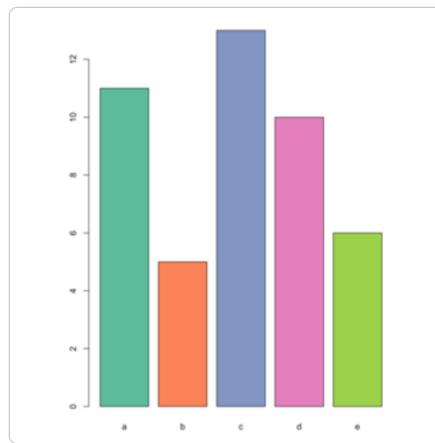
The correlation between several variables can also be studied by comparing their respective evolution



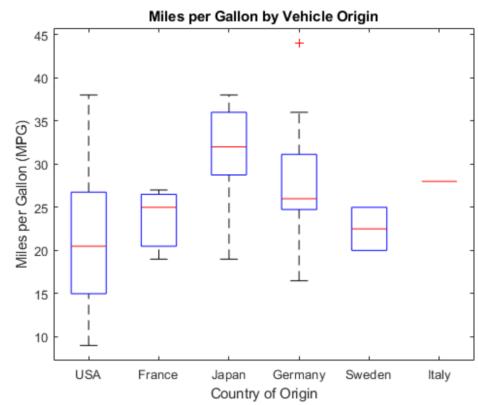
# Main graphs type



Scatterplot or dot diagram  
**The two variables in X and Y are Continues**  
*Allows you to see the possible link between The two variables*



Bar plot or bar chart  
**The variable in X is categorical**  
**No Y variable:** the height of the bar  
Corresponds to the quantity of the variable X  
In each category  
*Allows you to compare the importance of categories*



Boxplot or Whisker Box  
**The variable in X is categorical**  
**The variable in Y is continuous**  
*Allows you to compare different categories In X between them according to the variable In Y*

# Graphs with Dataiku

The screenshot shows the Dataiku interface. At the top, there is a navigation bar with tabs: Summary, Explore, Charts (which is highlighted with a red circle), Statistics, Status, History, and Settings. Below the navigation bar is a 'DISPLAY' dropdown menu. The main area displays a table with the following data:

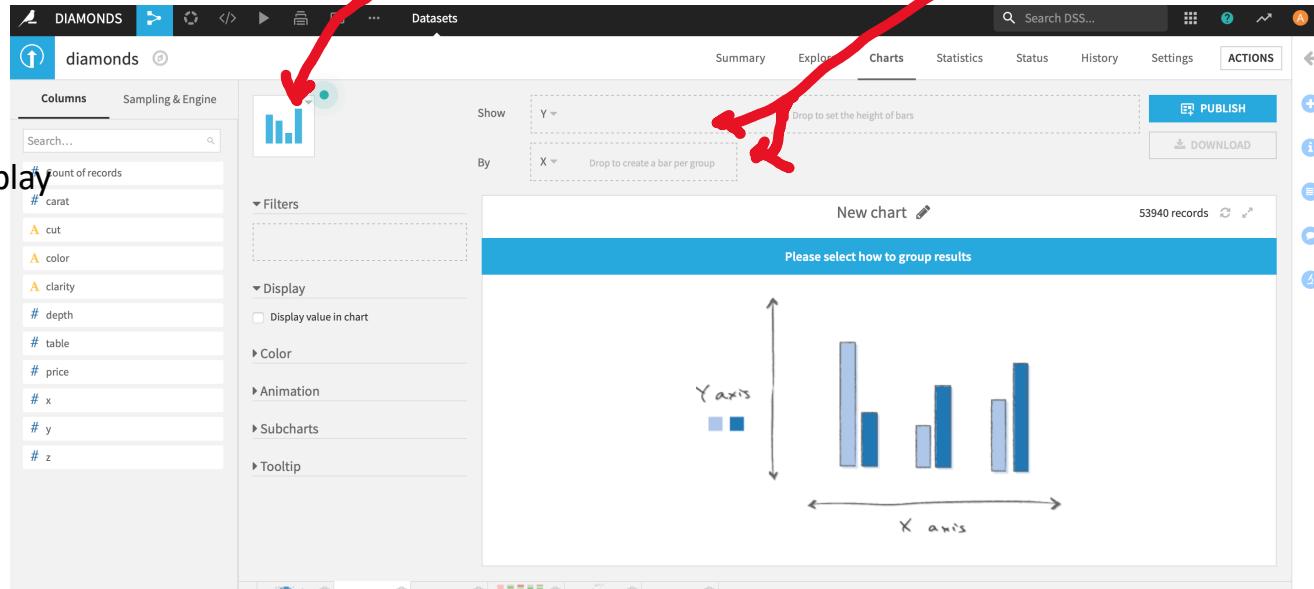
depth	table	price	x	y	z
string	string	string	string	string	string
Decimal	Decimal	Integer	Decimal	Decimal	Decimal
61.5	55	326	3.95	3.98	2.43
50.8	61	326	3.80	3.84	2.31

To the right of the table is a large red arrow pointing from the 'Charts' tab towards a detailed view of the chart selection menu. This menu is titled 'Basics' and includes categories for Bar charts (Histogram, Stacked, Stacked 100%, Stacked 100%), Lines & Curves (Lines, Stacked Area, Mix, Stacked Area 100%), and Pie & Donuts (Pie, Donut). There is also a 'OTHER' section.

Click on "Charts" then choose the type of graph and build the graph from the data

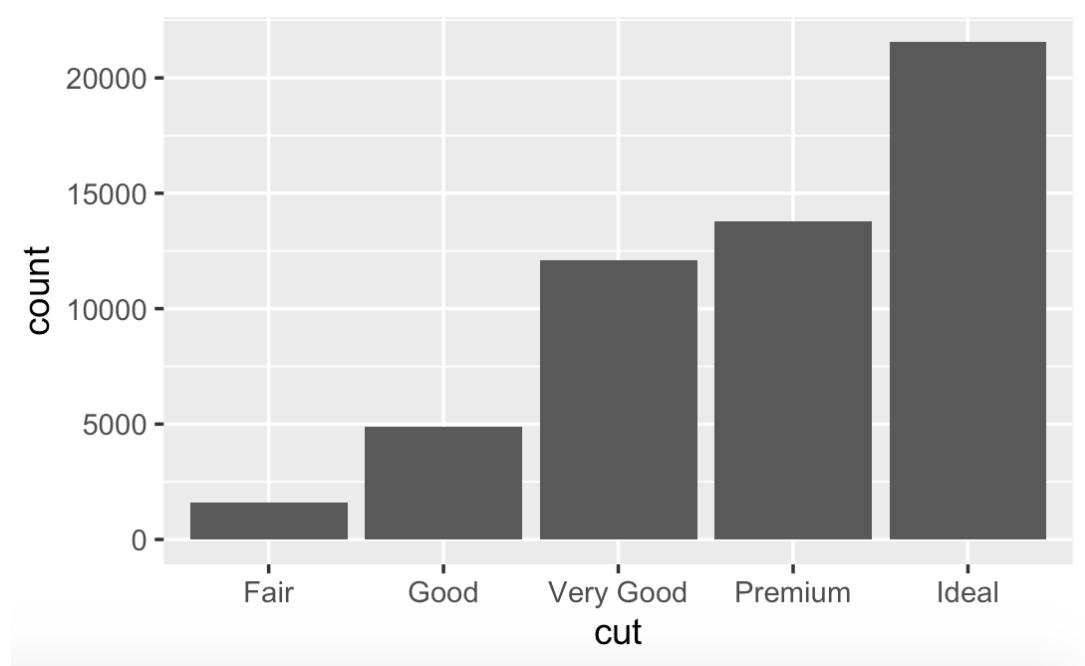
# Building Graph

2 Sélect  
variables to display  
in X  
the Y



## BAR GRAPH

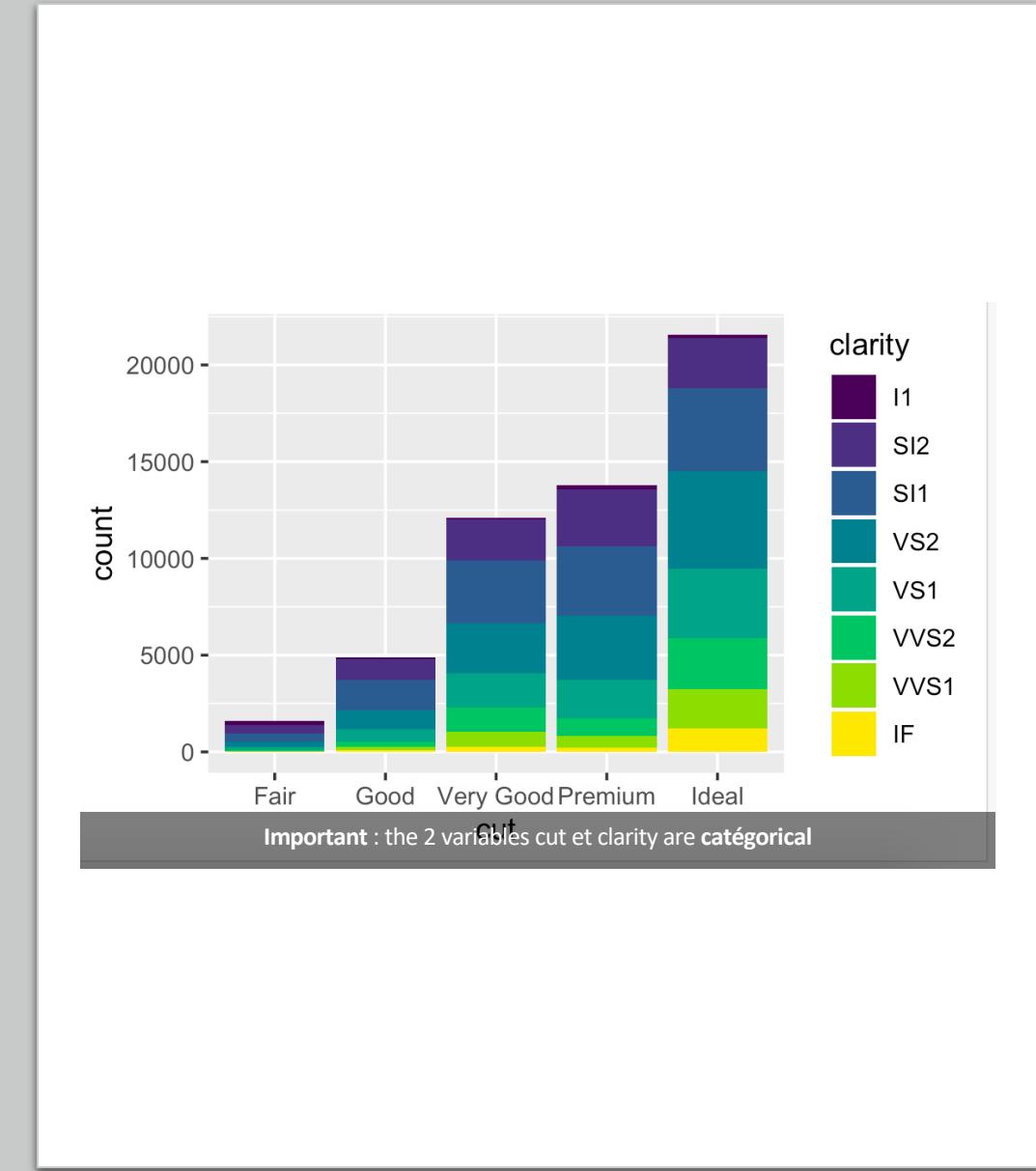
This graph is adapted to categorical (or discontinuous) variables  
In X the categories of the variable , in Y the number of occurrences of the variable in the dataset



It is easy to see how much diamonds of each category in the dataset

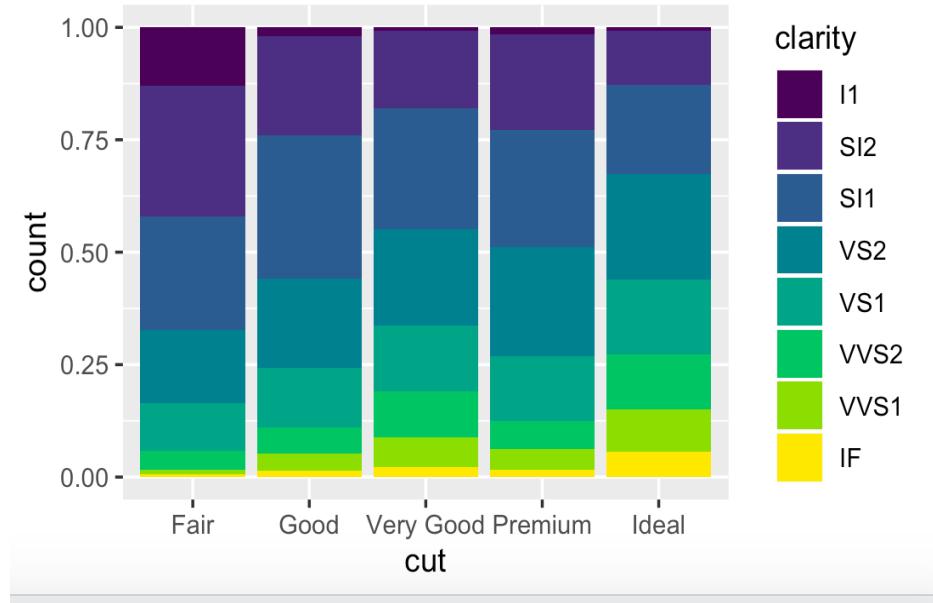
# Bars stacked

- The bars are stacked with a second variable (clarity )
- We can see how much clarities in the different cut (1st variable)

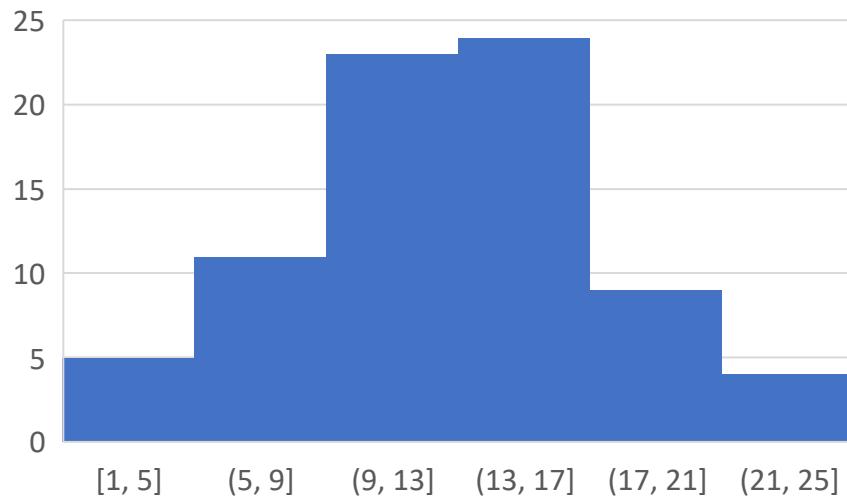


# Bars Stacked 100%

It is easier to compare the inside of the bars if they have the same height



# HISTOGRAM



The histogram allows you to cut  
A continuous variable in "slices"  
Called "bin"  
Each slice behaves like  
one of the categories  
of a categorical variable

For the dataset diamonds this type of  
Graph adapts well to the  
Carat variable on the abscissa  
And counting the number of diamonds  
per carat slice

**OBJECTIVE:** to determine which factor(s) most influence the price of diamonds

Cut ? Carat? Color ? .....



# Exercise : Display link between 2 then 3 variables

## Try to determine what is influencing price

1. You want to see if there is an influence of cut on price :
  1. Which is the nature of those 2 variables ?
  2. Which graph will you choose ?
  3. Build the graph in dataiku
  4. Then try to interpret it ? Does the result confirm Expectations ?
2. You want to compare the colors inside the different cut
  1. Which graph ?
  2. Build the graph
  3. What do you observe ?
3. Same for clarity inside cut
4. Influence of carat on price ? (same questions )

**Price:** price in US dollars ( $\$326--\$18,823$ )

**Carat:** weight of the diamond (0.2--5.01)

**Cut:** quality of the cut (Fair, Good, Very Good, Premium, Ideal)

**Color:** diamond colour, from D (best) to J (worst)

**Clarity:** a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

# Exercice : Filter and analyse deeper

1. On former exercise you made a graph with carats on the abscissa and prices on ordinates  
What is strange with this graph ?
1. On the same graph Color the dots according to cut Are you satisfied with the result ?
2. Filter the diamonds game by selecting the large carats  $> 2.4$
3. Then make a graph carat / price with colored points based on the cut

The company Telco is very concerned about the lack of loyalty of its customers and would like to understand the reasons for the loss of its customers (churn)

Let's talk business!

As an external consultant,  
You will analyze the situation  
of this company  
will make recommendations

- Load Telco Churn datasetHow many churn vs no churn clients? (Churn rate)
- Does the client's gender have an influence on churn? Base your answer on a graph
- Is the InternetService, TechSupport, ... have an influence on the Churn? Base your answer on a graph
- Do TotalCharges have an influence on churn? Create the appropriate chart that will support your answer
- What is tenure? How is it related to Churn?

In conclusion: What recommendations do you make to the Telecom operator to better retain its customers ?

- Churn =Yes the customer has broken his contract or no longer buys (lost customer) Churn= No the customer is kept (loyalty)
- )



# DATA PREPARATION AND TRANSFORMATION

# Aggregation (group by)

**Theory:** Aggregation consists of concentrating the dataset according to the categories of a variable  
We then have one line per category

**Example:** here are the grades of 3 high school students, there are several lines for the same high school student because each has several grades

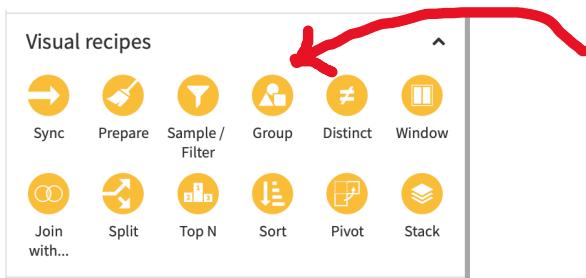
Aggregate consists in calculating the average (or the sum) of the grades of a student  
we then would have a new table

With one row per student and its average grade: the new table is more synthetic

STUDENT	SUBJECT	GRADE
ARTHUR	MATHS	12
INES	PHYSICS	15
SEBASTIEN	FRENCH	8
ARTHUR	FRENCH	11
INES	MATHS	11
SEBASTIEN	PHYSICS	16

*Here the aggregation key  
Is the student :  
We would like to group  
by student the table  
And calculate average  
grade per student*

# Aggregation with Dataiku



1) Choose recipe Group

The screenshot shows the 'New group recipe' dialog box. It has tabs for 'Input dataset' and 'Output dataset'. Under 'Input dataset', 'diamonds' is selected. Under 'Group By', there is a dropdown menu with 'Nothing selected' and a list of columns: carat, cut, color, clarity, depth, table, price, x, y, z. At the bottom are 'CANCEL' and 'CREATE RECIPE' buttons.

2) select aggregation Key

3) Specify which calculations have to be made in the final table

The screenshot shows the configuration screen for the 'compute\_diamonds\_by\_cut' recipe. It includes sections for 'Group Keys' (with a 'cut' key selected), 'Per field aggregations' (with a table listing fields like carat, color, clarity, depth, table, price, x, y, z with various aggregation options like Distinct, Min, Max, Avg, Sum, Concat, Std-dev, Count, First, Last), and a 'RUN' button at the bottom.

# Application :Group by

Group  
diamonds per  
cut

Calculate  
average price  
per cut

# JOINING TABLES

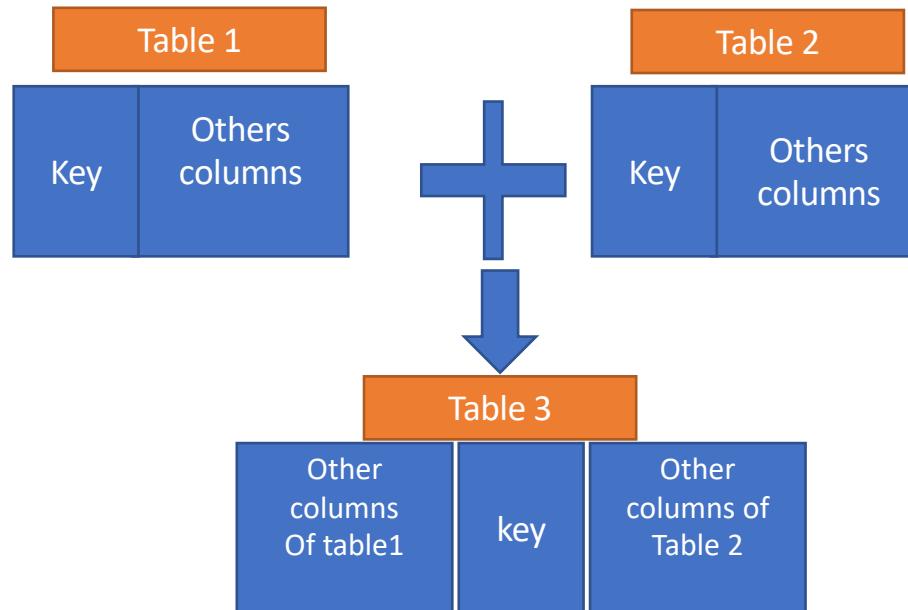
IT IS VERY COMMON  
AND USEFUL TO JOIN  
TABLES



IN ORDER TO GET A NEW  
TABLE CONTAINING LOTS  
OF INFORMATIONS

## ENRICH DATA WITH JOIN BETWEEN 2 TABLES

Two tables can only be "joined" if they have  
A common column (variable), this variable is called "key"

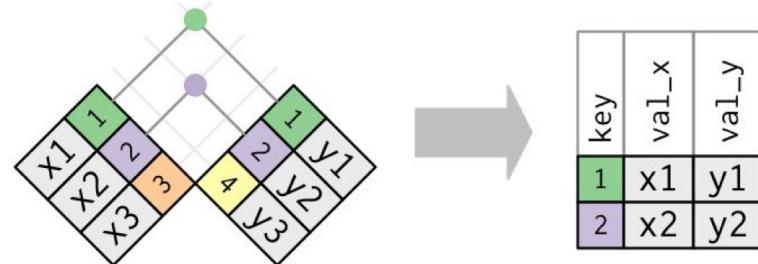


We get in the new table 3 all informations we had in both table 1 and 2  
(the key is not repeated twice !)

*The column key in table 1 and the column key in table 2  
may have different names provided  
The information in the 2 columns are the same*

# Several kind of joins

**Inner join** =Only rows that have the same value in the Key column are retained in the new table



Inner Join

Rows 3 and 4 are dropped in the join  
The result of the join is a table with  
A minimum number of rows

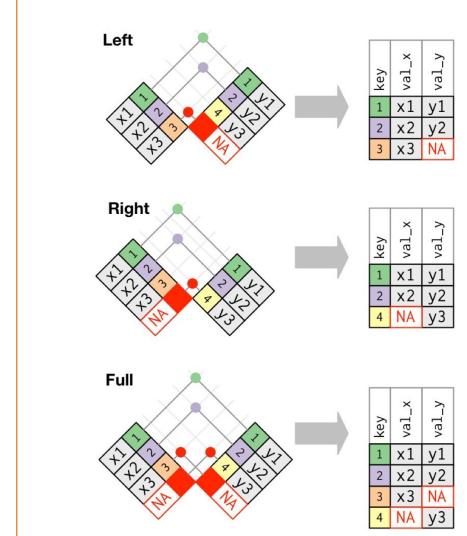
# OUTER\_JOIN

## Outer Joins

An inner join keeps observations that appear in both tables. An *outer join* keeps observations that appear in at least one of the tables. There are three types of outer joins:

- A *left join* keeps all observations in x.
- A *right join* keeps all observations in y.
- A *full join* keeps all observations in x and y.

These joins work by adding an additional “virtual” observation to each table. This observation has a key that always matches (if no other key matches), and a value filled with NA.



# JOIN WITH DATAIKU

**Datasets: orders.csv and customers.csv**

- 1) Pre-join data preparation: Aggregate the orders.csv table so that there is only one row per customer
- 2) Select the two datasets to join from the flow (right click on each)
- 3) Select the join recipe
- 4) Give the name to the new merged table (or keep the proposed name)
- 5) Check Dataiku has found the key
- 6) Change the key (if needed) by pressing the equal sign)
- 7) Click on Create



Join

# Building the join recipe

Datasets names to be joined →

← Name of new Dataset

Click here

New join recipe

Input datasets	Output dataset
<p>customers DATASET - View</p> <p>orders_by_customer DATASET - View</p> <p>Additional inputs can be added after the recipe creation.</p>	<p>Name customers_joined</p> <p>Store into filesystem_managed</p> <p>Format CSV</p> <p>NEW DATASET   USE EXISTING DATASET</p>

CANCEL CREATE RECIPE

PADM Spring 22

# The join recipe

Dataiku propose a left join  
(you can change it)

+ ADD INPUT

It is possible to make  
The join with a third table

Dataiku recognised  
The correct key even if the 2  
Names are slightly different  
This is why it is recommended to click on selected column  
To choose one of the 2 columns key in the final table

Click on Run if choices proposed are ok

**Attention :**  
*If the join key  
Has a different wording in  
both datasets  
Then you have to click  
on the sign =  
To specify the key  
and select  
Manually  
The 2 columns*

# The final result

Check all columns from original datasets are present

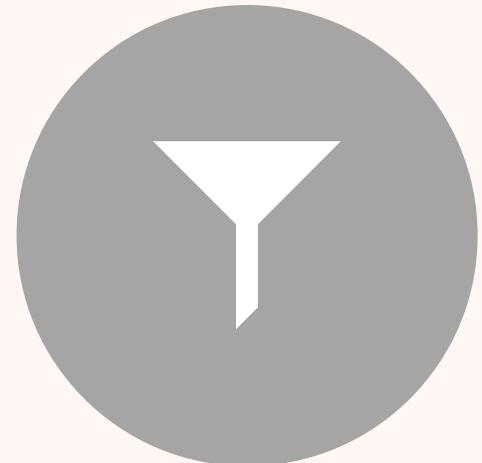
key

customerID	gender	birthdate	user_agent	ip_address	campaign	customer_id	first_order_date	pages_visited_avg	total_sum	co
886900	M	1953/10/05	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	69.8.136.110	False	886900	2016-01-26T00:00:00.000Z	10.0	80.0	1
038040	F	1946/12/21	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	67.74.254.174	False	038040	2014-06-04T00:00:00.000Z	11.0	38.0	1
698696	F	1987/12/01	Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	95.135.70.206	False	698696	2014-08-17T00:00:00.000Z	9.5	53.0	1
sn9keh	M	1995/09/23	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	129.50.205.26	False	sn9keh	2016-09-16T00:00:00.000Z	9.0	20.0	1
496907	M	1872/06/08	Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	81.98.234.29	False	496907	2014-04-28T00:00:00.000Z	11.0	85.0	1
486658	F	1948/07/29	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	76.177.15.150	False					1
345231	F	1885/02/18	Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0	174.83.130.76	False					1
vjavbl	F	1938/03/05	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	125.128.130.195	False					1
801797	M	1957/07/19	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	137.155.219.158	False	801797	2014-11-14T00:00:00.000Z	11.5	60.0	1
037662	F	1938/08/11	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	126.217.132.156	False	037662	2015-09-01T00:00:00.000Z	12.5	41.0	1
180904	F	1988/02/02	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	42.226.241.79	True					1
851551	M	1992/10/14	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	108.245.154.218	False	851551	2015-09-21T00:00:00.000Z	8.0	19.0	1
884231	F	1951/11/29	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	69.127.96.134	False					1
2p9myw	F	1972/01/11	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4285.147 Safari/537.36	133.52.95.35	False					1

# A REAL CASE : Employee Attrition



CREATE A NEW PROJECT IN  
DATAIKU



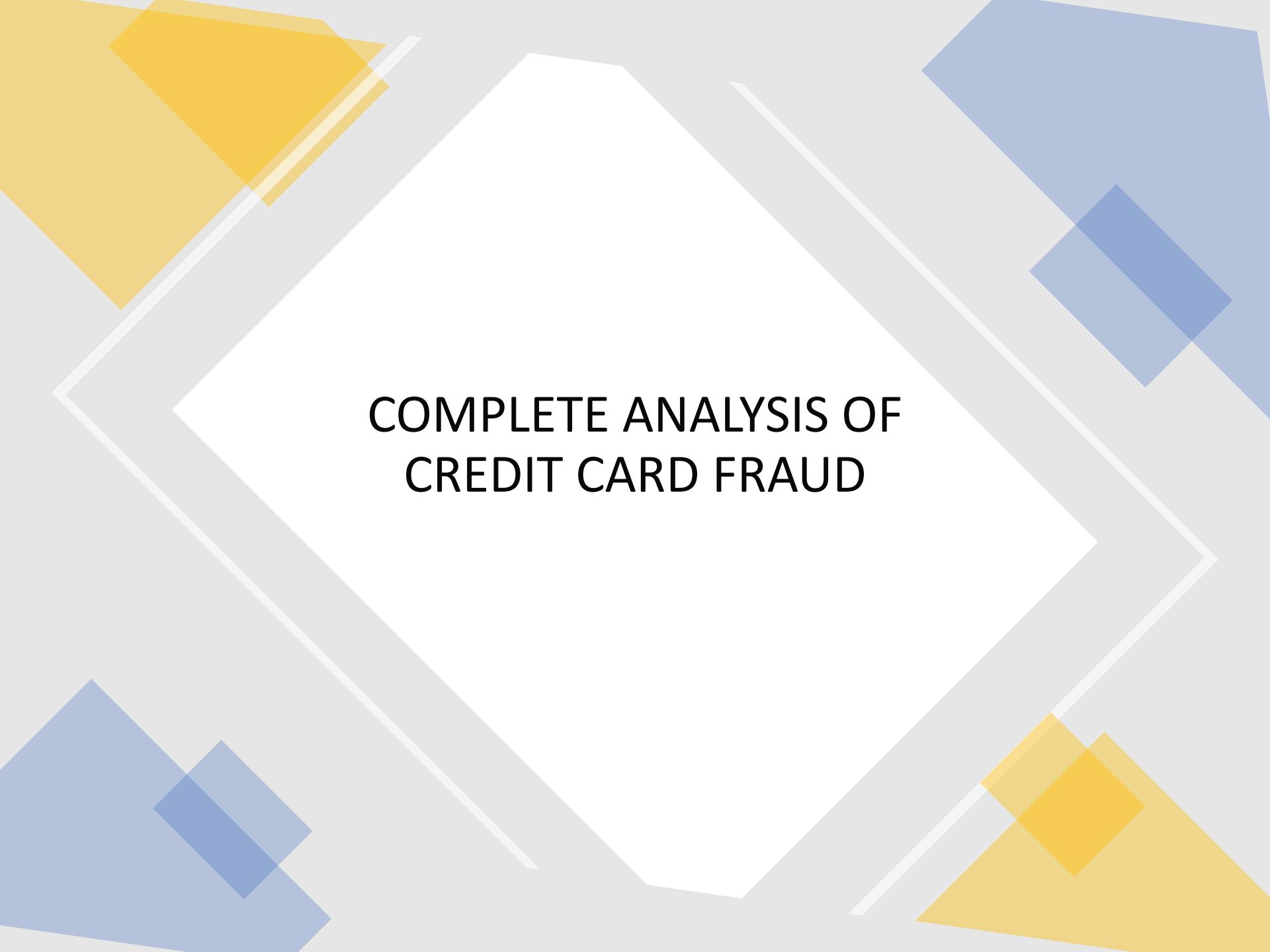
IMPORT DATASET « HR-  
EMPLOYEE-ATTRITION »

# The Turnover seems too high: can you find why ?

You are commissioned by the HR Director as a specialized consulting firm to help this company.

You will analyze the data and extract as much information as possible to understand the situation and determine potential causes of turnover

Then you will formulate your recommendations that will be presented to the Executive Committee



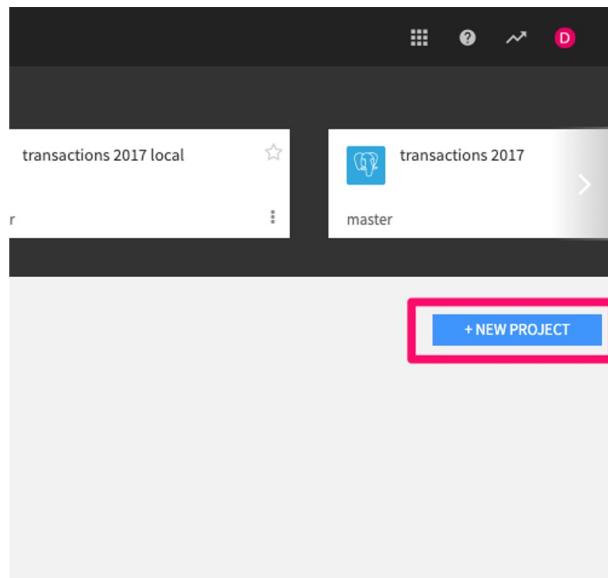
# COMPLETE ANALYSIS OF CREDIT CARD FRAUD

# Agenda

Discovery  
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
- 3. Creating Projects and Connecting to Data**
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

# Create a new project



Name your project “Credit Card Fraud” + your name

+ New project

Name  Credit Card Fraud Pat

Project Key  The project key is used to reference datasets between projects. It cannot be changed once the project is created.

CANCEL CREATE

# Data Sources Overview: 4 Datasets

## ***transactions\_2017*** and ***transactions\_2018***

- consumer credit card transactions

## ***cardholder\_info***

- credit score for each cardholder

## ***merchant\_info***

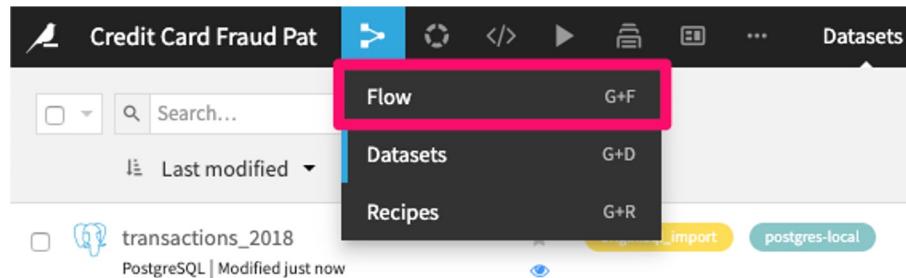
- Lookup table for merchants

# Import data

The screenshot shows a user interface for importing datasets. At the top, there is a blue button labeled '+ IMPORT YOUR FIRST DATASET'. Below it, the main interface displays several categories of datasets:

- Files**: Includes options to "Upload your files", "Server Filesystem", and "Files in folder". A red arrow points to the "Upload your files" link.
- Hadoop**: Includes "HDFS" and "Hive".
- SQL**: Includes "MySQL", "PostgreSQL", "Vertica", "Amazon Redshift", "Snowflake", "IBM Netezza", and "Other SQL". It also lists "Greenplum", "Teradata", "Oracle", "MS SQL Server", "SAP HANA", and "Google BigQuery".
- Cloud Storages**: Includes "Amazon S3", "Azure Blob Storage", "Google Cloud Storage", "FTP", "SFTP", "SCP", "HTTP", and "HTTP (with cache)".
- NoSQL**: Includes "MongoDB", "Cassandra", and "ElasticSearch".
- Social**: Includes "Twitter".
- DSS**: Includes "Managed dataset", "Folder", "Dataset from another project", "Metrics", "Internal stats", and "Editable".
- Import existing**: Contains a link "Choose connection to import from" and "Import from catalog", which is highlighted with a red box.

# You should now have 4 datasets in your flow



transactions\_2018



cardholder\_info



transactions\_2017



merchant\_info

Double click on a dataset  
to view it

# Explore the transaction data

The screenshot shows a data exploration interface with a top navigation bar featuring icons for file operations, a search bar, and tabs for 'Summary', 'Explore' (which is selected), 'Charts', 'Statistics', 'Status', 'History', 'Settings', and 'ACTIONS'. Below the navigation is a sub-header with a profile icon, the dataset name 'transactions\_2017', and a 'Configure sample' link. It also indicates there are '10000 rows, 9 cols'. A 'DISPLAY' dropdown and three visualization icons are on the right. The main area displays a table with 10000 matching rows, showing columns for transaction\_id, authorized\_flag, purchase\_date, card\_id, merchant\_id, merchant\_category\_id, item\_category, purchase\_amount, and signature\_provided. The first few rows of data are visible.

transaction_id	authorized_flag	purchase_date	card_id	merchant_id	merchant_category_id	item_category	purchase_amount	signature_provided
59760	1	2017-05-08 14:19:45	C_ID_e0fd181776	M_ID_1eb5b5e073	417	B	34.4	0
59761	1	2017-05-08 14:22:04	C_ID_e7676db3b8	M_ID_7ca715839e	771	D	99.3	1
59762	1	2017-05-08 14:24:45	C_ID_a15f532627	M_ID_4885bb4884	518	D	359.25	0
59763	1	2017-05-08 14:25:12	C_ID_3092725553	M_ID_b8a0b209fe	206	B	60.08	1
59764	1	2017-05-08 14:25:47	C_ID_5c32762ada	M_ID_e1b2d6bf5c	373	B	241.27	0
59765	1	2017-05-08 14:26:27	C_ID_45e7b34f81	M_ID_d5c94df60e	434	B	418.73	0
59766	1	2017-05-08 14:27:47	C_ID_17226c9d33	M_ID_6f939b9049	278	C	51.73	0
59767	1	2017-05-08 14:28:47	C_ID_9dd08a4854	M_ID_b86f065dde	783	C	90.41	0

Each row is a transaction. The '**authorized\_flag**' shows whether it was approved or not.

# Column attributes

transaction_id	authorized_flag	purchase_date	card_id
bigint Integer	bigint Integer	string Date (unparsed)	string Text
59760	1	2017-05-08 14:19:45	C_ID_e01e01e01e01e01e
59761	1	2017-05-08 14:22:04	C_ID_e7676db3b8
59762	1	2017-05-08 14:24:45	C_ID_a15f532627
59763	1	2017-05-08 14:25:12	C_ID_3092725553
59764	1	2017-05-08 14:25:47	C_ID_5c32762ada
59765	1	2017-05-08 14:26:27	C_ID_45e7b34f81
59766	1	2017-05-08 14:27:47	C_ID_17226c9d33
59767	1	2017-05-08 14:28:47	C_ID_9dd08a4854

Column name

Schema - the storage type

ex. string, int, double, array

Meaning - the “human readable” meaning

ex. text, integer, IP address, gender

We can see information about each column and its contents

# Agenda

Discovery  
Hands-on Exercise

Purple Track

1. Background
  2. DSS Login and Set up Account
  3. Creating Projects and Connecting to Data
  4. **Stacking Datasets**
  5. Joins
  6. Computation Engines
  7. Data Cleaning [1]
  8. Charts [1]
  9. Splitting Data
  10. Machine Learning
- Green Track
11. Filtering Data
  12. Data Cleaning [2]
  13. Group by Aggregations
  14. Charts [2]

# Stack



- Merge two or more datasets into one
- Equivalent to a “union all” SQL statement
- Good for datasets with substantially the same columns/schema

# Stack the transactions datasets together

Credit Card Fraud Pat > Flow

Search DSS... DESELECT

4 datasets

transactions\_2018

cardholder\_info

Shift + Click to select multiple datasets

transactions\_2017

merchant\_info

2 datasets

+ RECIPE + DATASET

Build Tag Share Unwatch Star

Delete

Visual recipes

Join Stack

Code recipes

Python R SQL Shell

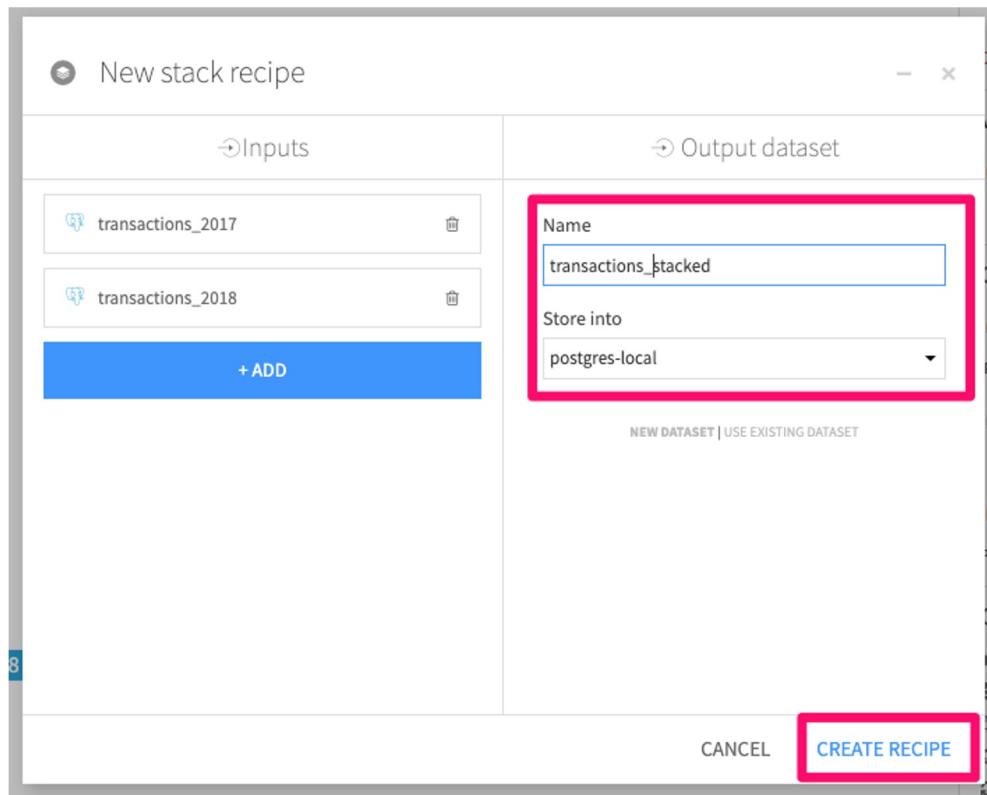
Hive Impala Pig Spark SQL Spark Scala

PySpark Spark R

Other actions

Add to a scenario Change connection

# Rename the output dataset “transactions\_stacked”



# Stack

The screenshot shows a data stacking interface with the following components:

- Top Bar:** Credit Card Fraud Pat, various icons, Recipes.
- Panel Headers:** compute\_transactions\_stacked, Summary, Settings, Input / Output, Advanced, History.
- Left Panel (Flowchart):** A vertical flowchart showing the data pipeline:
  - Pre-filters
  - Selected columns (highlighted with a green checkmark)
  - Origin column
  - Post-filter
  - Output (highlighted with a green checkmark)
- Input Datasets:** transactions\_2017 (blue dot) and transactions\_2018 (green dot).
- Columns Selection:** Union of input schemas.
- Column List:** A list of columns from both datasets:

authorized_flag	blue dot	green dot
card_id	blue dot	green dot
city_id	blue dot	green dot
category_1	blue dot	green dot
installments	blue dot	green dot
category_3	blue dot	green dot
merchant_category_id	blue dot	green dot
month_lag	blue dot	green dot
purchase_amount	blue dot	green dot
purchase_date	blue dot	green dot
category_2	blue dot	green dot
merchant_id	blue dot	green dot
- Text Overlay:** You should see all columns in both datasets.
- Bottom Buttons:** RUN (highlighted with a red box), In-database (SQL).
- Text Overlay:** Click run to create the output dataset.

# Your flow should look like this



cardholder\_info



merchant\_info



transactions\_2017



transactions\_2018



transactions\_stacked

→

→

→

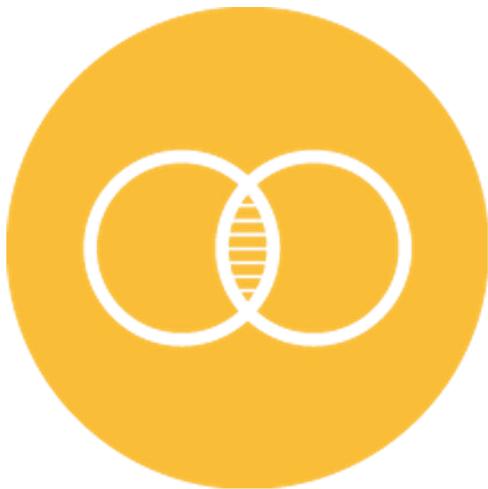
# Agenda

Discovery  
Hands-on Exercise

Purple Track

1. Background
  2. DSS Login and Set up Account
  3. Creating Projects and Connecting to Data
  4. Stacking Datasets
  5. Joins
  6. Computation Engines
  7. Data Cleaning [1]
  8. Charts [1]
  9. Splitting Data
  10. Machine Learning
- Green Track
11. Filtering Data
  12. Data Cleaning [2]
  13. Group by Aggregations
  14. Charts [2]

# Join



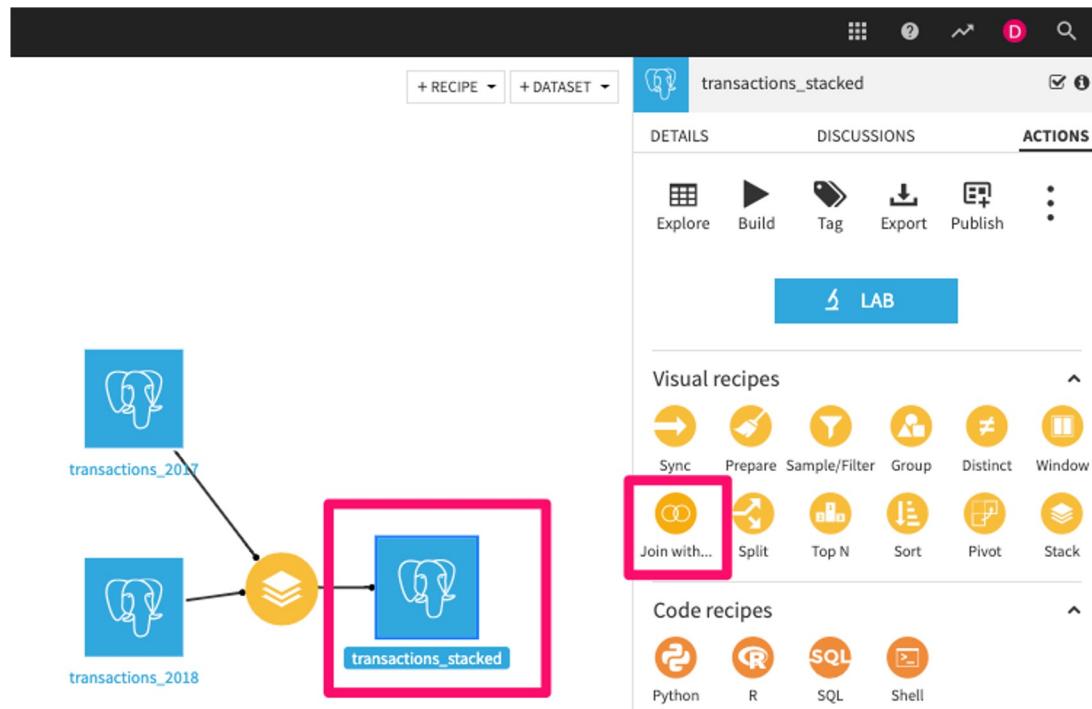
- Join two or more datasets
- Perform inner joins, left outer joins, full outer joins, right joins, etc.
- Define join conditions using the UI or SQL

## Join the stacked transactions dataset with the other two datasets

1. LEFT JOIN `transactions_stacked` dataset with `cardholder_info` dataset ON `'card_id'` = `'internal_card_mapping'`
2. LEFT JOIN `transactions` dataset with `merchant_info` dataset ON `'merchant_id'` and `'merchant_category_id'`
3. ADD PREFIXES “`cardholder`” and “`merchant`” to columns from the two datasets respectively

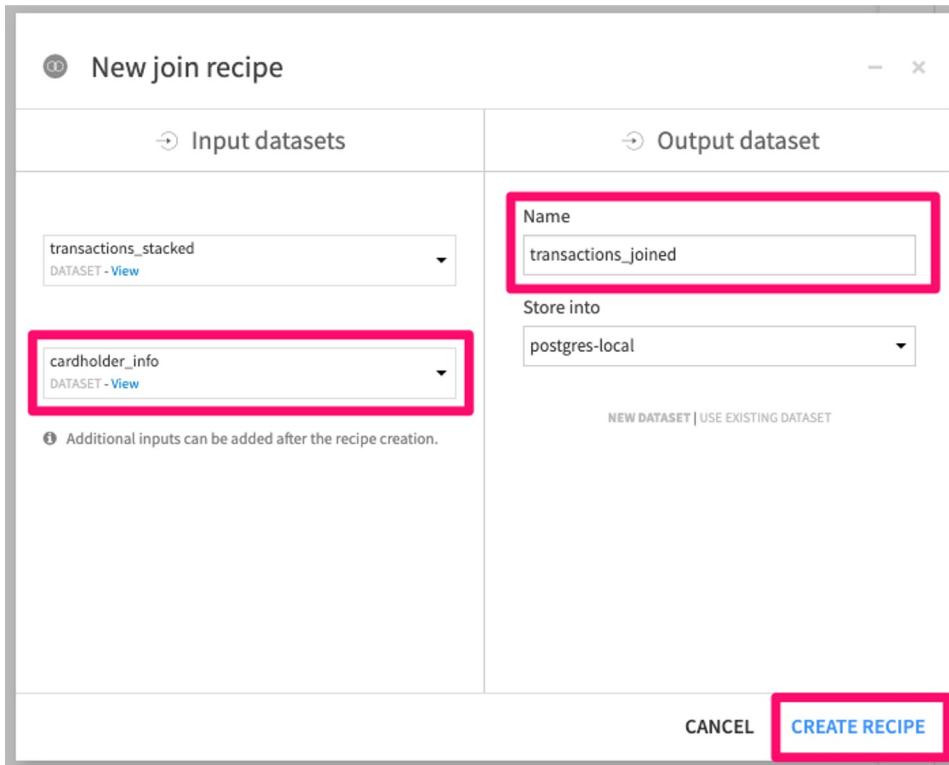
Do this all in ONE **Join** recipe

# Join the stacked transactions dataset with the other two datasets



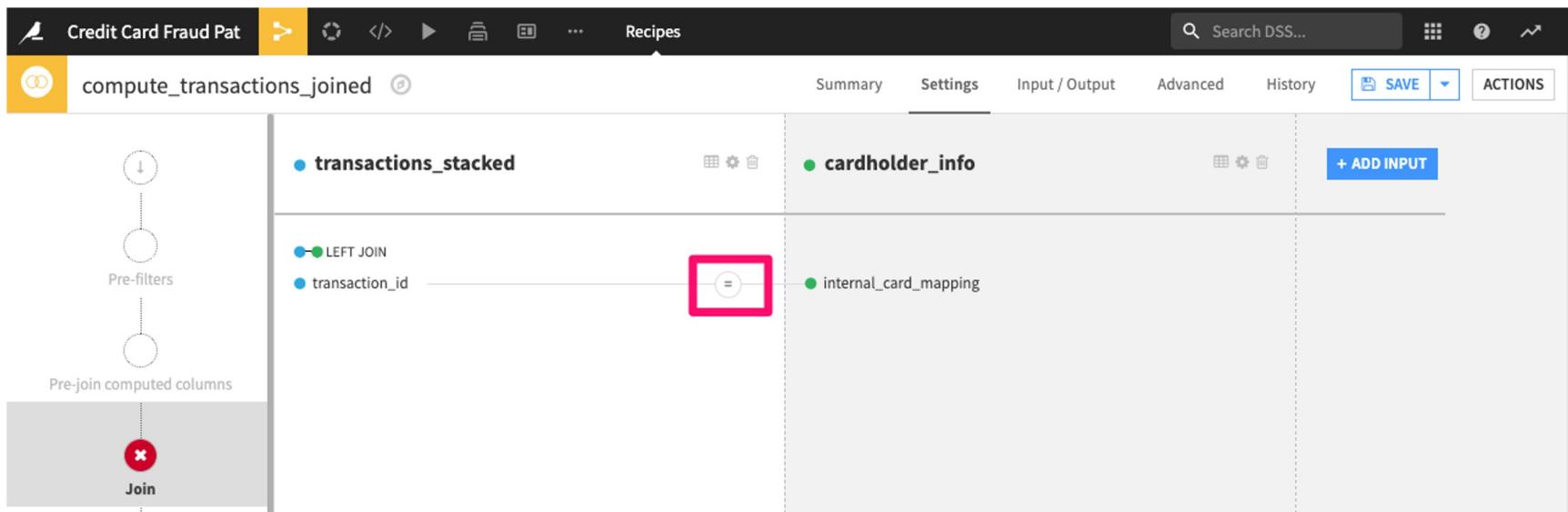
- Click on the **transactions\_stacked** dataset
- Choose the **Join** recipe

# Join the stacked transactions dataset with the other two datasets



- Select the `cardholder_info` dataset from the dropdown
- Name the output dataset `transactions_joined`

## LEFT JOIN transactions\_stacked with cardholder\_info



Sometimes DSS guesses the wrong desired join columns

Click the **equals** sign to change the join

## LEFT JOIN transactions\_stacked with cardholder\_info

Join

JOIN TYPE CONDITIONS

Match when all the following conditions are satisfied

LEFT JOIN transaction\_id = internal\_card\_mapping

Click to change join columns



Change the columns to join on

'card\_id' = 'internal\_card\_mapping'

Join

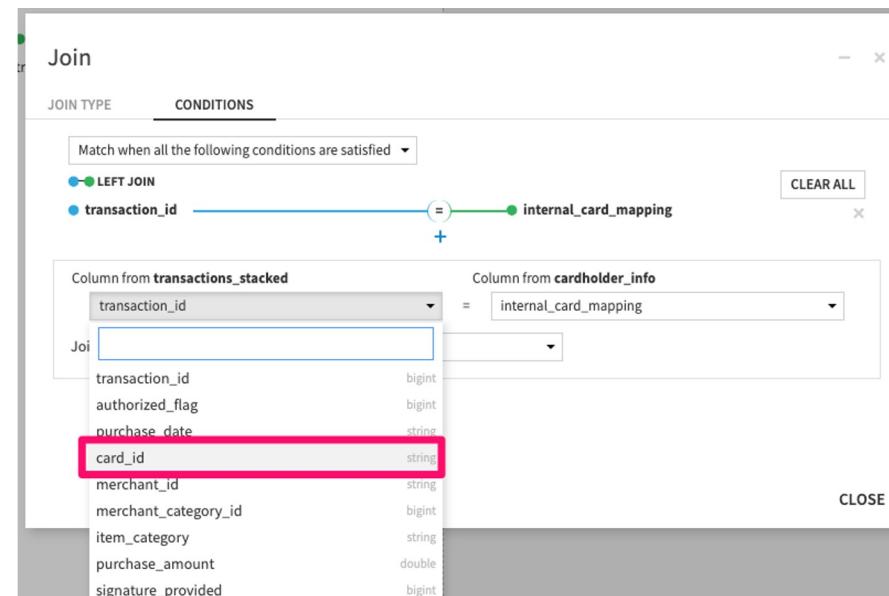
JOIN TYPE CONDITIONS

Match when all the following conditions are satisfied

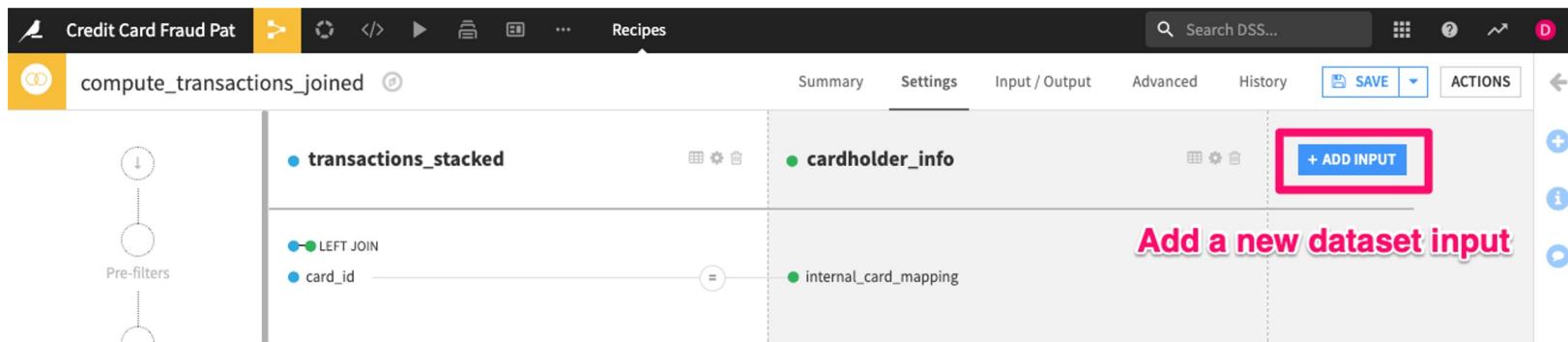
LEFT JOIN transaction\_id = internal\_card\_mapping

Column from transactions\_stacked  
transaction\_id  
authorized\_flag  
purchase\_date  
**card\_id**  
merchant\_id  
merchant\_category\_id  
item\_category  
purchase\_amount  
signature\_provided

Column from cardholder\_info  
internal\_card\_mapping



# LEFT JOIN transactions\_stacked with merchant\_info



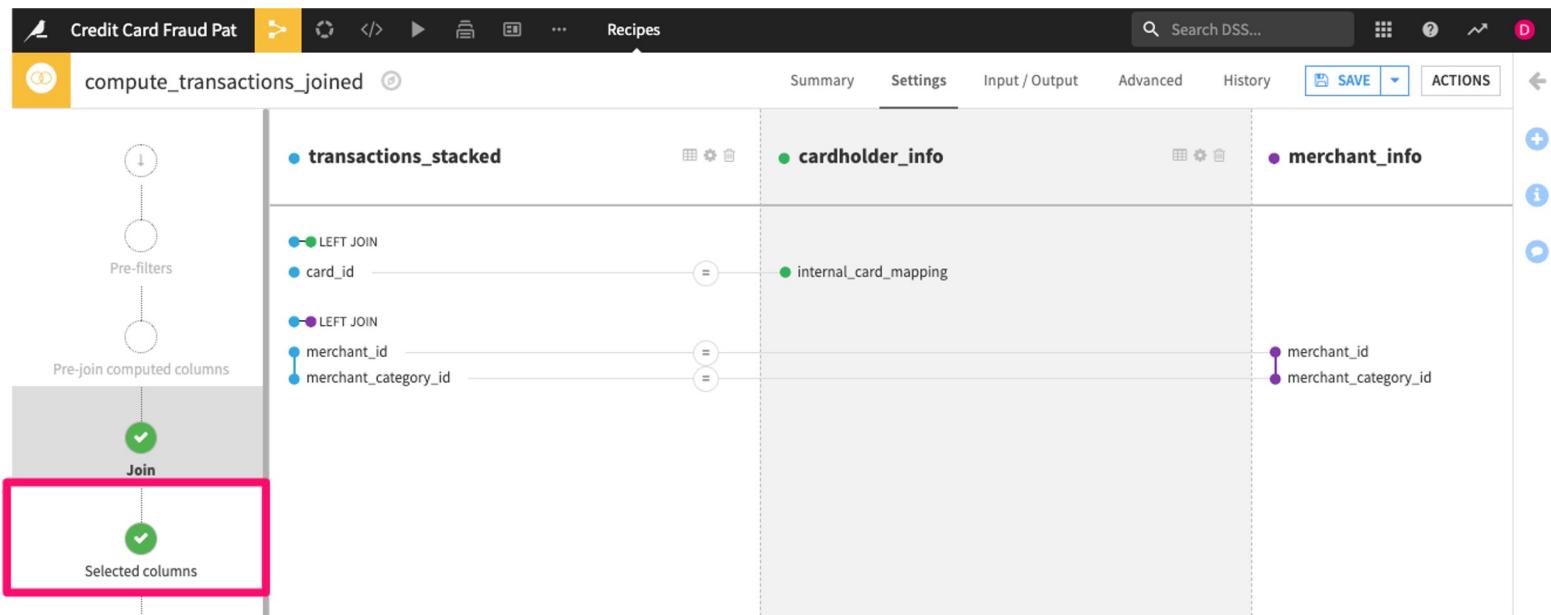
Add a new dataset input

Select the merchant\_info dataset from the dropdown

The modal dialog has the following fields:

- Existing input dataset:** transactions\_stacked
- New input dataset:** merchant\_info (highlighted with a red box)
- Buttons:** CANCEL, ADD DATASET

# LEFT JOIN transactions\_stacked with merchant\_info



This looks good!

Now let's look at the **Selected Columns** tab

# Add prefixes to columns from joined datasets

The screenshot shows a data pipeline interface with three joined datasets:

- transactions\_stacked**:
  - Manually select columns (checked)
  - Prefix: none
  - Selected columns (highlighted with a red box): transaction\_id, authorized\_flag, purchase\_date, card\_id, merchant\_id, merchant\_category\_id, item\_category, purchase\_amount.
- cardholder\_info**:
  - Manually select columns (checked)
  - Prefix: card (highlighted with a red box)
  - Selected columns (highlighted with a red box): card\_internal\_card\_mapping, card\_first\_active\_month, card\_reward\_program, card\_latitude, card\_longitude, card\_fico\_score, card\_age.
- merchant\_info**:
  - Manually select columns (checked)
  - Prefix: merchant (highlighted with a red box)
  - Selected columns (highlighted with a red box): merchant\_merchant\_id, merchant\_merchant\_category\_id, merchant\_subsector\_description, merchant\_latitude, merchant\_longitude.

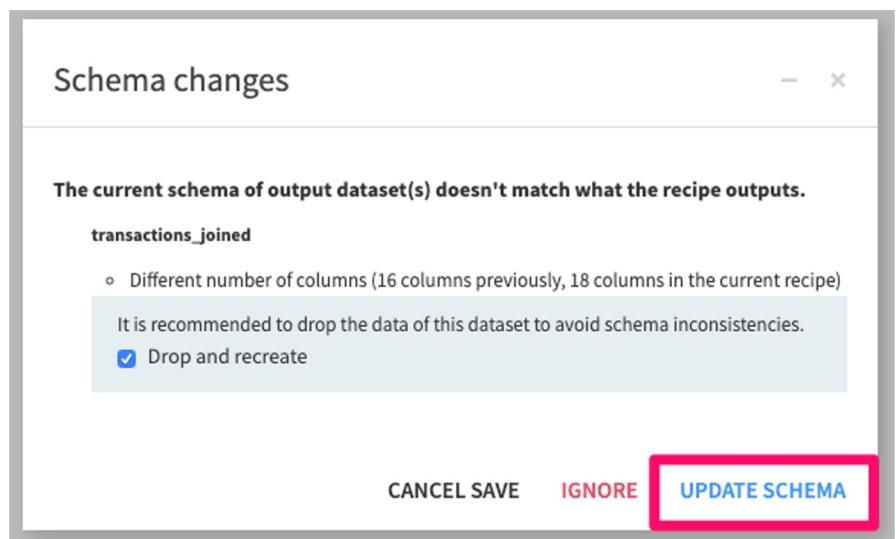
A pink arrow points from the text "Exclude this" to the "card\_internal\_card\_mapping" checkbox in the cardholder\_info dataset.

- Deselect 'card\_internal\_card\_mapping'
- Select 'merchant\_latitude' and 'merchant\_longitude'

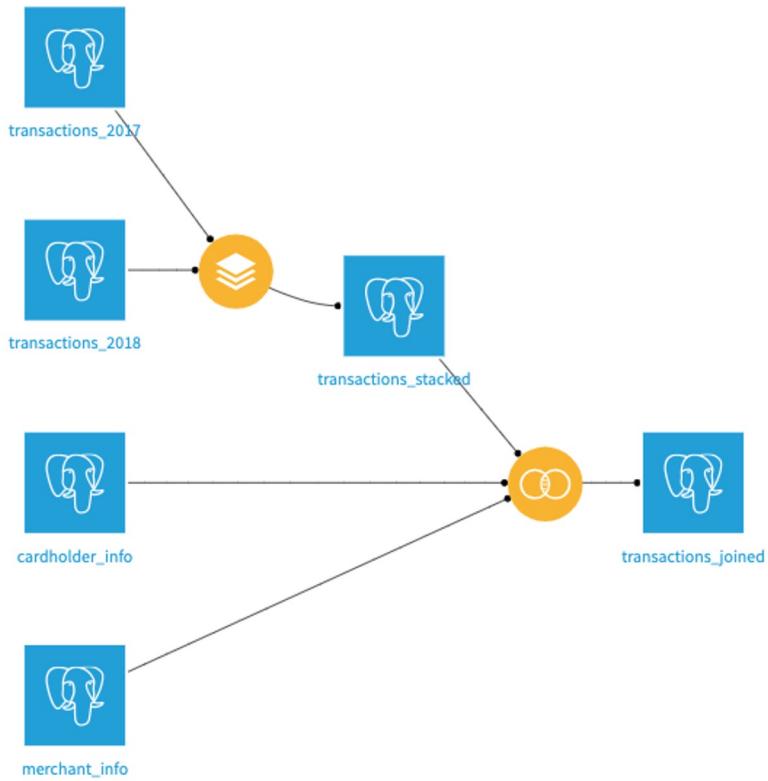
Any ideas why merchant latitude/longitude cols weren't included initially?

# Run your join

Don't be afraid of schema updates! (Click the blue button)



# Flow check



# Agenda

Discovery  
Hands-on Exercise

Purple Track

1. Background
  2. DSS Login and Set up Account
  3. Creating Projects and Connecting to Data
  4. Stacking Datasets
  5. Joins
  6. Computation Engines
  7. Data Cleaning [1]
  8. Charts [1]
  9. Splitting Data
  10. Machine Learning
- Green Track
11. Filtering Data
  12. Data Cleaning [2]
  13. Group by Aggregations
  14. Charts [2]

# Prepare



~90 different processors, including

- Find and replace
- Date manipulations
- Mathematical operations on columns
- Splitting columns
- Extracting parts of columns
- Text parsing and NLP
- Geographic transformations

# Create a **Prepare** recipe from the **transactions\_joined** dataset

The screenshot illustrates the workflow for creating a data preparation recipe. On the left, a visual representation shows a dataset named "ns\_stacked" connected to a central node, which then connects to a dataset named "transactions\_joined". The "transactions\_joined" dataset is highlighted with a pink border. In the center, a sidebar menu lists various recipe types: "Explore", "Build", "Visual recipes" (Sync, Prepare, Join with..., Split), and "Code recipes" (Python, R). The "Prepare" icon is highlighted with a red box. On the right, a modal window titled "New data preparation recipe" is open. It shows the "Input dataset" set to "transactions\_joined" and the "Output dataset" set to "Name: transactions\_joined\_prepared" and "Store into: postgres-local". At the bottom right of the modal, the "CREATE RECIPE" button is also highlighted with a red box.

New data preparation recipe

Input dataset	Output dataset
transactions_joined	Name: transactions_joined_prepared Store into: postgres-local

CANCEL CREATE RECIPE

## Round 1: Create time-based features

1. Parse dates: `card_first_active_month` and `purchase_date`
2. Extract date components from `purchase_date_parsed`
  - year, month, day, dow, hour
3. Create a `purchase_weekend` (sat/sun) column using a DSS formula
4. Create a `days_active` column: the time from the cardholder's `first_active_month` until `purchase_date`

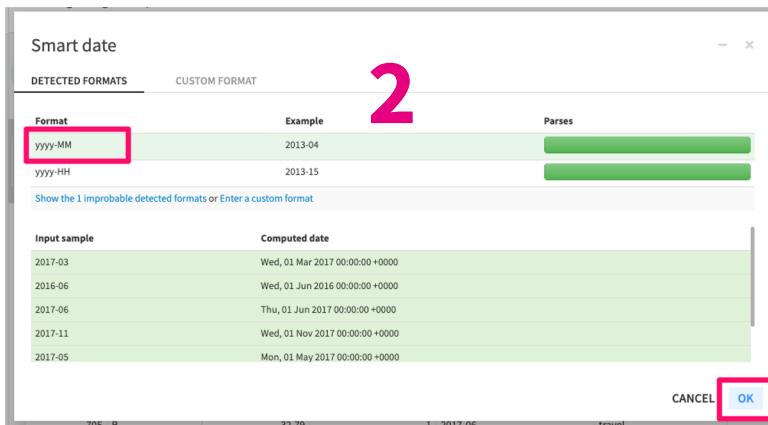
# Parse card\_first\_active\_month

Viewing design sample

10000 rows, 18 cols

1

category_id	item_category	purchase_amt	signature_provided	card_first_active_month	card_reward_pro
130	D	244.58			
884	D	384.75			
661	D	55.1			
367	A	45.46			
298	C	337.3			
68	B	241.42			
705	D	77.26			



3

compute\_transactions\_joined

Script

Design Sample

1 step

10000 rows 18 cols

Parse date in card\_first\_active\_month

10000

Column single | multiple | pattern | all

card\_first\_active\_month

Output column (empty for in-place)

card\_first\_active\_month\_parsed

Date format(s)

yyyy-MM

ADD FORMAT

Locale

Translate automatically

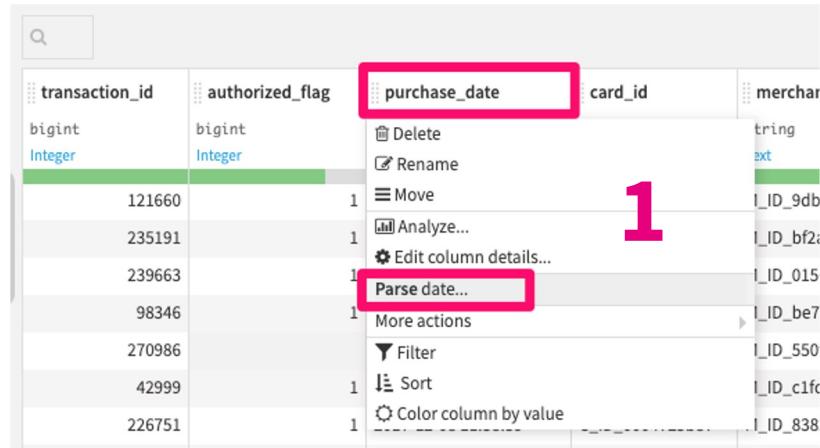
Timezone

UTC

+ ADD A NEW STEP

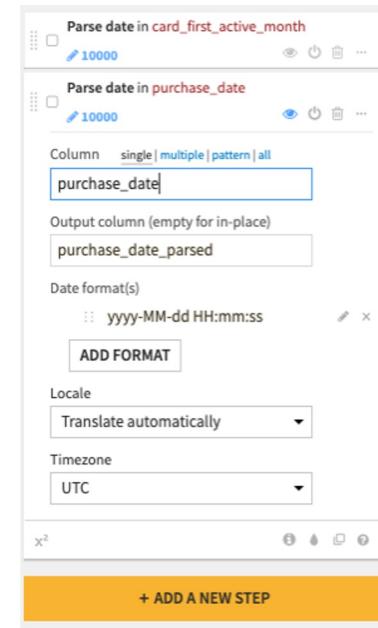
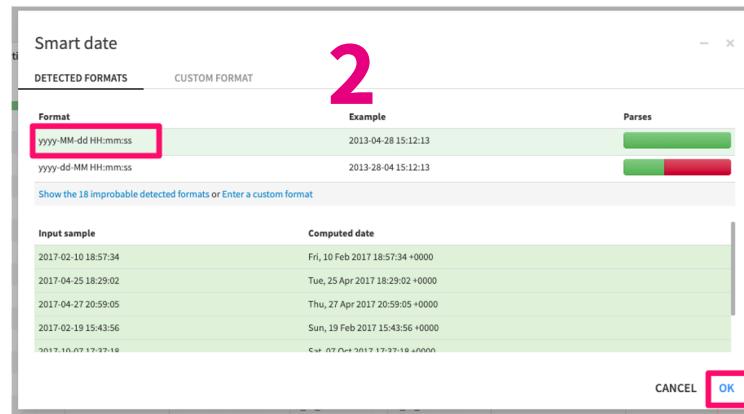
We want yyyy-MM format

# Parse purchase\_date



A screenshot of a database table interface. The table has columns: transaction\_id, authorized\_flag, purchase\_date, card\_id, and merchant. The purchase\_date column is highlighted with a red box. A context menu is open over the purchase\_date column, with the 'Parse date...' option highlighted by a pink box and a large pink number '1'.

transaction_id	authorized_flag	purchase_date	card_id	merchant
121660	1		I_ID_9db	
235191	1		I_ID_bf2i	
239663	1		I_ID_015	
98346	1		I_ID_be7	
270986			I_ID_550	
42999	1		I_ID_c1fc	
226751	1		I_ID_838	



A screenshot of the 'Parse date in purchase\_date' dialog. It shows the 'purchase\_date' column selected. The output column is set to 'purchase\_date\_parsed'. The date format is specified as 'yyyy-MM-dd HH:mm:ss'. The locale is set to 'Translate automatically' and the timezone is 'UTC'. A large pink number '3' is displayed next to the dialog.

**Note:** if you wanted to parse in-place, you could delete the output column name to do that

# Extract date components from `purchase_date_parsed`

authorized_flag	purchase_date	purchase_date_parsed	card_id	merchant_id
bigint Integer	string Date (unparsed)	1 2017-08-14 11:10:00		M_ID_9dbbaabb0a
		1 2017-12-15 13:39:53		M_ID_bf2ac203cf
		1 2017-12-19 08:26:41		M_ID_0156e34767
		1 2017-07-13 13:34:21		M_ID_be730907ce
		2018-01-17 08:09:41		M_ID_550f11f3df
		1 2017-04-05 15:24:48	87	M_ID_c1fc572ee0
		1 2017-12-08 11:38:39	87	M_ID_83823fb6e7
		1 2017-10-05 22:17:10	87	M_ID_0074dd7c6e
		1 2017-10-11 10:43:30	C_ID_0004725b87	M_ID_8bebe220344
		2017-10-11T10:43:30.000Z	C_ID_0004725b87	M_ID_8d577d8382
		1 2017-01-07 17:05:48	C_ID_0004725b87	M_ID_8d577d8382
		2017-01-07T17:05:48.000Z		

1

2

Extract date components from `purchase_date_parsed`

Date column: `purchase_date_parsed`

Timezone: UTC

'Year' column: `purchase_year`

'Month' column: `purchase_month`

'Day' column: `purchase_day`

'Day of week' column: `purchase_dow`

'Hour' column: `purchase_hour`

'Minutes' column:

Clean up column names:  
for example...  
`purchase_date_parsed_year` →  
`purchase_year`

Add:  
`purchase_dow`  
`purchase_hour`

# Create binary purchase\_weekend column

Use a DSS Formula!

We are defining weekends  
as Friday/Saturday

+ ADD A NEW STEP

Let's add this one using the  
+ ADD A NEW STEP button

The screenshot shows the DSS Processor library interface. On the left, there is a sidebar with various processor categories: Filter data, Data cleansing, Strings, Math / Numbers, Split / Extract, Web logs, Dates, Geography, Enrich, Reshaping, Natural Language, Joins, Complex objects, Code, and Misc. To the right of the sidebar is a list of processors numbered 13 down to 10. Processor number 15, labeled 'Formula', is highlighted with a red box. The right side of the screen contains detailed information about the 'Formula' processor, including its description, formula language capabilities, usage examples, and help links.

**Processors library**

90 processors

Processor	Description
Filter data	
Data cleansing	
Strings	
Math / Numbers	
Split / Extract	
Web logs	
Dates	
Geography	
Enrich	
Reshaping	
Natural Language	
Joins	
Complex objects	
Code	
Misc	
Find and replace	
Split column	
Transform string	
<b>Formula</b>	<b>This processor computes new columns</b>
Extract with regular expression	
Concatenate columns	
Simplify text	
Tokenize text	
Extract ngrams	
Extract numbers	
Negate boolean value	

**Formula**

This processor computes new columns

The formula language provides:

- Math functions
- String manipulation functions
- Date handling functions
- Boolean and conditional expressions

**Usage examples**

- Compute a numerical expression:
- Manipulate strings: `toLowerCase`
- Create a rule-based column: `if`

**Getting help**

# Create binary purchase\_weekend column

The screenshot shows a step in a data processing workflow. The title is "Create column purchase\_weekend with formula". The formula is `if(purchase_dow>5,1,0)`. The output column is set to "purchase\_weekend". The expression is the same as the formula. There is an "EDIT" button highlighted with a red box. Below the expression is an "Error column" field and an "x<sup>2</sup>" field with a comment icon. At the bottom is a yellow button "+ ADD A NEW STEP".

You can add  
comments to  
document potentially  
complex formulas!

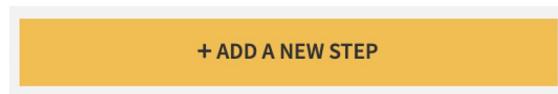
Click “EDIT” to pull up examples and reference

The screenshot shows a formula editor window. The formula is `if(purchase_dow>5,1,0)`. The "SAVE" button is highlighted with a red box. To the right, there is a "Results" section and an "Examples" section. The examples list includes:

- `nb_events + 3`  
Evaluates to the value of the `nb_events` cell + 3
- `max(x1, x2)`  
Returns the highest value of two cells
- `if (x1 > x2, "big", "small")`  
Returns "big" if `x1 > x2`, "small" otherwise
- `numval("column with space") * 2`  
For column names with special characters, use `strval("name")` or.

At the bottom, a green bar says "Formula is valid" with a checkmark.

# Compute the time from first active month to purchase date



Let's add this one using the **+ ADD A NEW STEP** button

Processors library

<input type="checkbox"/> Filter data	13
<input type="checkbox"/> Data cleansing	11
<input type="checkbox"/> Strings	11
<input type="checkbox"/> Math / Numbers	15
<input type="checkbox"/> Split / Extract	7
<input type="checkbox"/> Web logs	7
<input checked="" type="checkbox"/> Dates	8
<input type="checkbox"/> Geography	9
<input type="checkbox"/> Enrich	12
<input type="checkbox"/> Reshaping	16
<input type="checkbox"/> Natural Language	5
	-

34069      1      8 processors

+ ADD A NEW STEP

Compute time difference between  
card\_first\_active\_month\_parsed and  
 purchase\_date\_parsed

10000

Time since column

until

Other column

Output time unit

Output column

Reverse output

+ ADD A NEW STEP

Name the output column:  
**days\_active**

# You can **Group** your date cleaning steps

The screenshot shows the Alteryx interface with a context menu open over a selected step. The menu includes options like 'Group' (which is highlighted with a red box), 'Copy 5 steps', 'Paste after selection', 'Toggle enable/disable', 'Delete', 'Color', 'Extract date components from purchase\_date\_parsed', and 'Create column purchase\_weekend with formula if(purchase\_dow>5,1,0)'. To the right, a 'Date Formatting' group is shown, containing five steps: 'Parse date in cardholder\_first\_active\_month', 'Parse date in purchase\_date', 'Compute time difference between cardholder\_first\_active\_month\_parsed and purchase\_date\_parsed', 'Extract date components from purchase\_date\_parsed', and 'Create column purchase\_weekend with formula if(purchase\_dow>5,1,0)'. A descriptive comment at the bottom states: 'Parse dates, compute days active, extract date components from purchase date, and create purchase weekend'. A checkbox for 'Always show comment' is checked. A small red box highlights the 'comment' icon at the bottom right of the group.

Call your group “**Date Formatting**”

This is much easier to read!

You can also modify colors or add descriptions !

## Round 2: Create geolocation features

1. Create geopoint columns
  - `merchant_location` using the `merchant latitude & longitude`
  - `cardholder_location` using the `cardholder latitude & longitude`
2. Reverse geocode
  - `merchant_location` to get the `merchant_state`
  - `cardholder_location` to get the `cardholder_state`

# Create merchant geopoint column

Processors library

Geography

create geo

1 processors

Create GeoPoint from lat/lon

Use the processor search

Create GeoPoint from **merchant\_latitude** & **merchant\_longitude**

**10000**

Input 'latitude' column  
**merchant\_latitude**

Input 'longitude' column  
**merchant\_longitude**

Output 'GeoPoint' column  
**merchant\_location**

x<sup>2</sup> ⓘ ⚡ ⌂ ?

# Create cardholder geopoint column

The screenshot shows a data processing interface with a search bar at the top containing the text "create geo". Below the search bar, there is a list of processors under the heading "Processors library". One processor is selected, highlighted with a pink border: "Create GeoPoint from lat/lon". The processor has the following configuration:

- Processor ID: 1
- Type: Geography
- Description: Create GeoPoint from lat/lon
- Processor details:
  - This processor creates a GeoPoint column by combining latitude and longitude columns.
  - Geolocation column: card\_location
  - For each row, the latitude and longitude values are combined into a single GeoPoint column.

To the right of the processor list, a detailed view of the selected processor is shown:

**Create GeoPoint from card\_latitude & card\_longitude**

**10000**

**Input 'latitude' column**  
card\_latitude

**Input 'longitude' column**  
card\_longitude

**Output 'GeoPoint' column**  
card\_location

**x<sup>2</sup>**

Do the same thing for the cardholder locations!

**For step 2, we will be using the reverse geocoding plugin...**

# Plugins

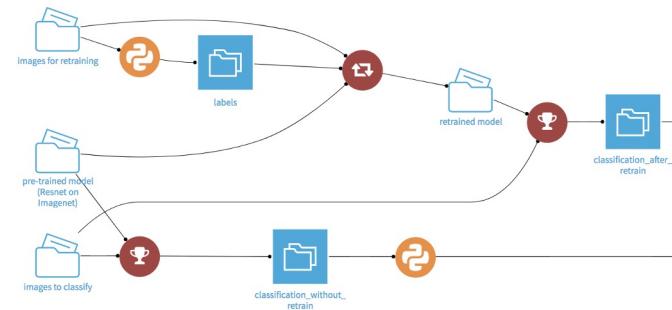
## Many Types

In DSS plugins, the user can develop :

- **Custom Recipes**
  - A basic user interface allowing the users to interact with variables of the underlying code: R, Python, SQL in Python
  - Example : Query a rest API with some settings; apply a statistical function to a field of dataset
- **Custom Dataset**
  - Example : query a salesforce table.
- **Much More**
  - New Preparation Step Processor
  - Custom Scenario Step
  - Dataset Exporter
  - Web App/R Markdown Template

Plugins are then available as a DSS extension

The screenshot shows a configuration page for a dataset named 'senate\_lobbyingdisclosure\_2015'. It includes tabs for 'Connector', 'Schema', and 'Advanced'. Below the tabs, there's a 'Learn more' link. The main area contains fields for 'API Key' (with placeholder text '\*\*\*\*\*') and 'Dataset' (set to 'us.gov.senate.lobbyingdisclosure'). There's also a 'Number of pages' input field set to '0'. At the bottom is a green 'TEST & GET SCHEMA' button.



# Reverse geocode the merchant location

Use the processor search



Processors library

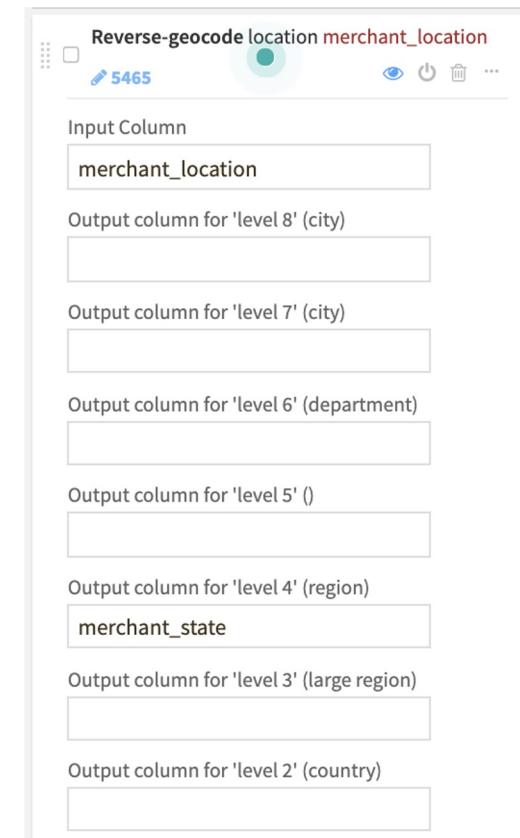
reverse geocoding

1 processors

Geography    1    Reverse geocoding

Enrich    1

**Reverse**  
This processor...  
Output is pro...  
**Administr**  
The process...  
each level de...



Reverse-geocode location merchant\_location

5465

Input Column  
merchant\_location

Output column for 'level 8' (city)

Output column for 'level 7' (city)

Output column for 'level 6' (department)

Output column for 'level 5' ()

Output column for 'level 4' (region)  
merchant\_state

Output column for 'level 3' (large region)

Output column for 'level 2' (country)

## We can choose what to extract.

In “region”, enter  
“merchant\_state”

# Reverse geocode the cardholder location

Reverse-geocode location **merchant\_location**

5465

Input Column: **merchant\_location**

Output column for 'level 8' (city)

1 2017-12-29T

⋮

- Copy this step
- Paste after this step
- Comment
- Color
- Duplicate step

Feeling lazy? Duplicate and modify!

Reverse-geocode location **card\_location**

8990

Input Column: **card\_location**

Output column for 'level 8' (city)

Output column for 'level 7' (city)

Output column for 'level 6' (department)

Output column for 'level 5' ()

Output column for 'level 4' (region)

**card\_state**

Output column for 'level 3' (large region)

# Compute the distance between cardholders and merchants

Use the processor search

The screenshot shows a data processing interface with two main sections. On the left, a 'Processors library' sidebar lists categories like 'Geography'. A search bar at the top contains the text 'compute distance', which is highlighted with a red rectangle. Below the search bar, a list of processors is shown, with one item selected: 'Compute distance between geopoints'. This item has a red rectangle around its name. To the right of the search bar, it says '1 processors'. The right side of the interface shows the configuration details for the selected processor. It includes fields for 'Distance between column' (set to 'card\_location'), 'and' (set to 'Another geopoint column'), 'Other column' (set to 'merchant\_location'), 'Output distance unit' (set to 'Miles'), and 'Output column' (set to 'merchant\_cardholder\_distance'). At the bottom right is a yellow button labeled '+ ADD A NEW STEP'.

Processors library

Geography

compute distance

1 processors

Compute distance between geopoints

This processor computes the distance between a geopoint and another geopoint.

You can compare a column with:

- A specific geopoint
- Another geopoint column

The computation outputs a number of distance units (kilo).

For more info, [please see the processor's reference](#)

Compute distance between **card\_location** and **merchant\_location**

9813

Distance between column  
**card\_location**

and  
Another geopoint column ▾  
**merchant\_location**

Other column  
**merchant\_location**

Output distance unit  
**Miles**

Output column  
**merchant\_cardholder\_distance**

+ ADD A NEW STEP

## Note the “invalid” data

DSS has two “data types” listed:

- 1) **Storage Type:** indicates how dataset backend should store data
- 2) **Meaning:** rich semantic type that can be more specific and is automatically detected

merchant_state	merchant_state_enName
string	string
US State	US State
Indiana	Indiana
New York	New York
Ontario	Ontario

The **red** refers to invalid meanings. These will not necessarily cause errors, they are more to help guide you.

Including non-US data is fine for now, so we can leave the “invalid” data in.

# Double check that you've added all steps

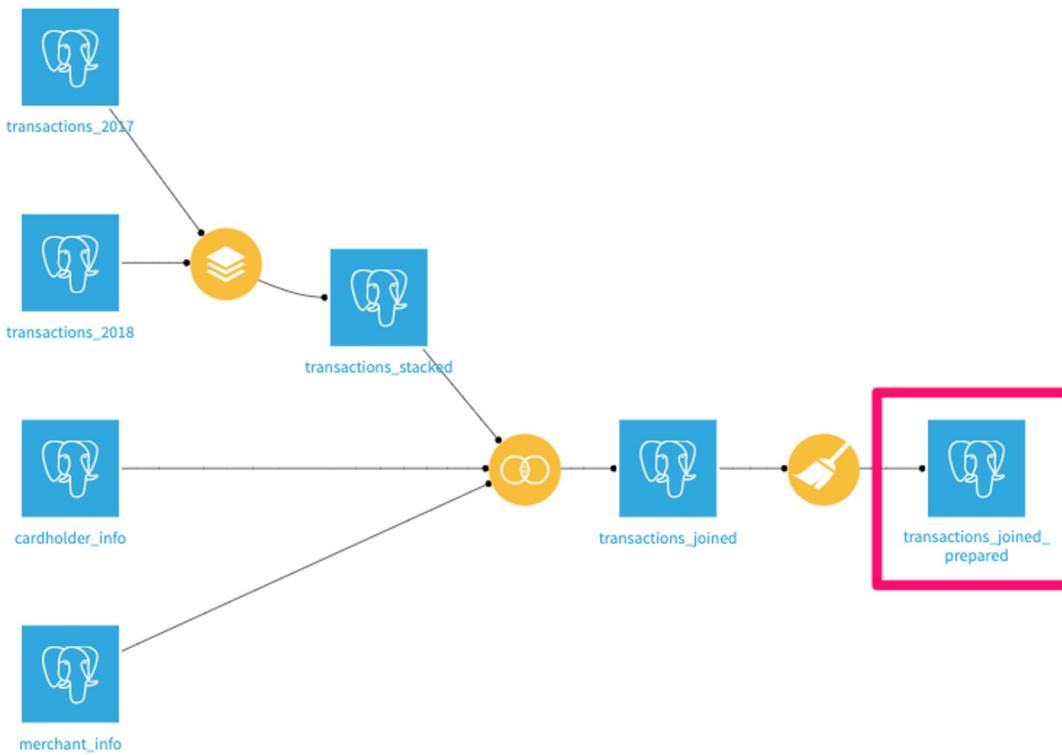
The screenshot shows a data transformation interface with a sidebar on the left and a main panel on the right. The sidebar contains a tree view of data steps:

- Date formatting**:
  - 10000 Parse date in `card_first_active_month`
  - 10000 Parse date in `purchase_date`
  - 10000 Extract date components from `purchase_date_parsed`
  - 10000 Create column `purchase_weekend` with formula `if(purchase_dow>5,1,0)`
  - 10000 Compute time difference between `card_first_active_month_parsed` and `purchase_date_parsed`
- Parse dates, compute days active, extract date components from purchase date, and create purchase weekend**:
  - Always show comment
- Create GeoPoint from `merchant_latitude` & `merchant_longitude`**:
  - 9813
- Create GeoPoint from `card_latitude` & `card_longitude`**:
  - 10000
- Reverse-geocode location `merchant_location`**:
  - 8615
- Reverse-geocode location `card_location`**:
  - 8762
- Compute distance between `card_location` and `merchant_location`**:
  - 9813



...and then click “RUN”

# Analyze the transactions\_joined\_prepared dataset



# Click on a column name and ‘Analyze’ to see more

Summary   Explore   Charts   Status   History   Setting

card\_internal\_card\_num\_map   card\_fico\_score   card\_reward\_program   merge

string   Text

724 C\_ID\_d80632d9df

557 C\_ID\_d80632d9df

279 C\_ID\_ba6b5f72ec

Analyze...   Edit column schema...   Filter   Sort   Color column by value

- Distributions
- Summary stats
- Value counts

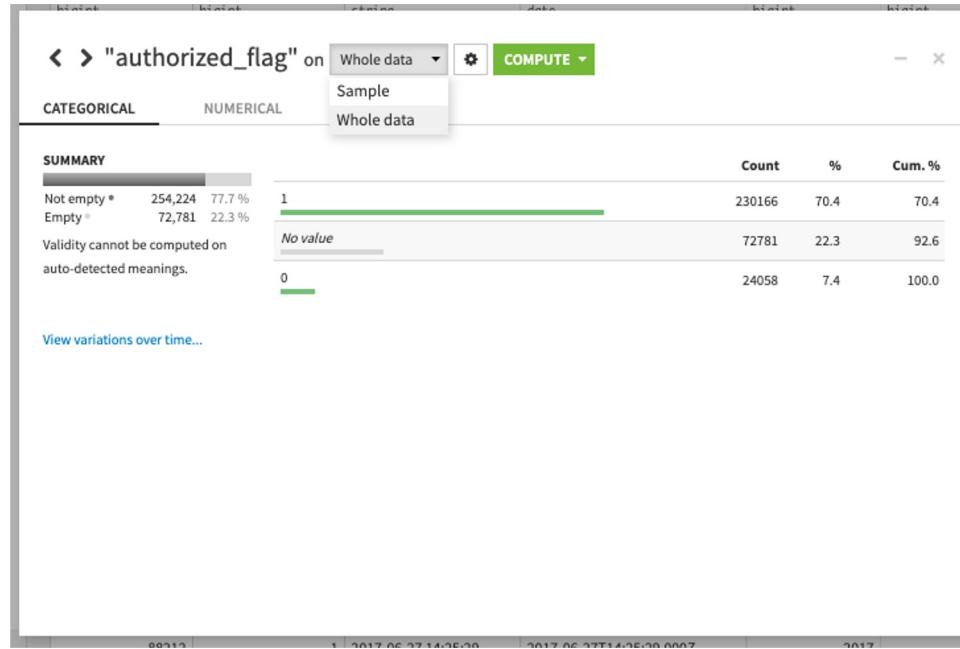


Stats can be computed on the sample or whole dataset

# Questions

1. What is the highest individual purchase amount?
1. What percentage of transactions are unauthorized (fraudulent)?

# When analyzing a column, you can run stats on the entire dataset



# Questions

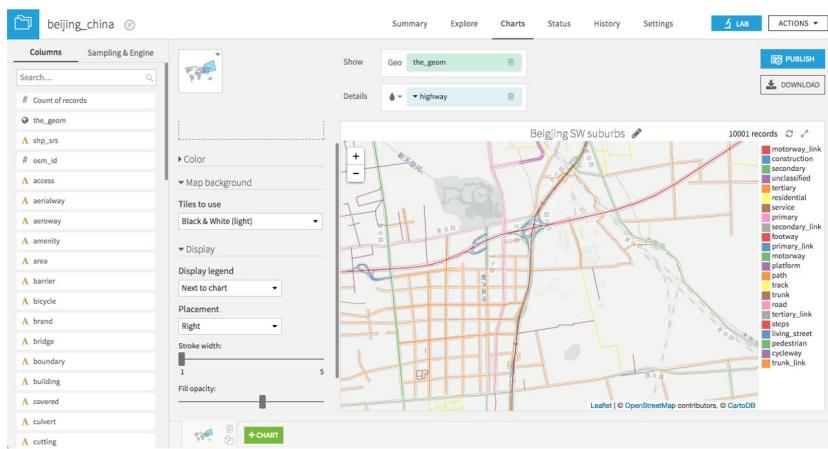
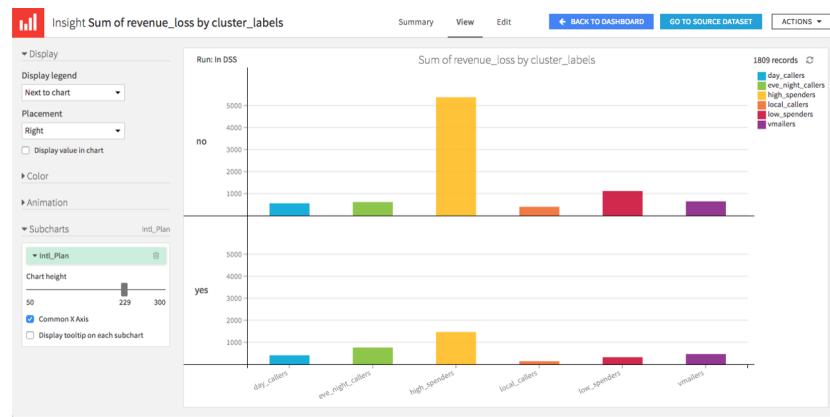
1. What is the highest individual purchase amount?  
\$48,147
1. What percentage of transactions are unauthorized?  
7.4%

# Agenda

Discovery  
Hands-on Exercise

Purple Track

1. Background
  2. DSS Login and Set up Account
  3. Creating Projects and Connecting to Data
  4. Stacking Datasets
  5. Joins
  6. Computation Engines
  7. Data Cleaning [1]
  8. Charts [1]
  9. Splitting Data
  10. Machine Learning
- Green Track
11. Filtering Data
  12. Data Cleaning [2]
  13. Group by Aggregations
  14. Charts [2]



# Build a chart on the transactions\_stacked\_prepared dataset

Task:

- Map the fraud rate by US state
- Which states have the highest rates?

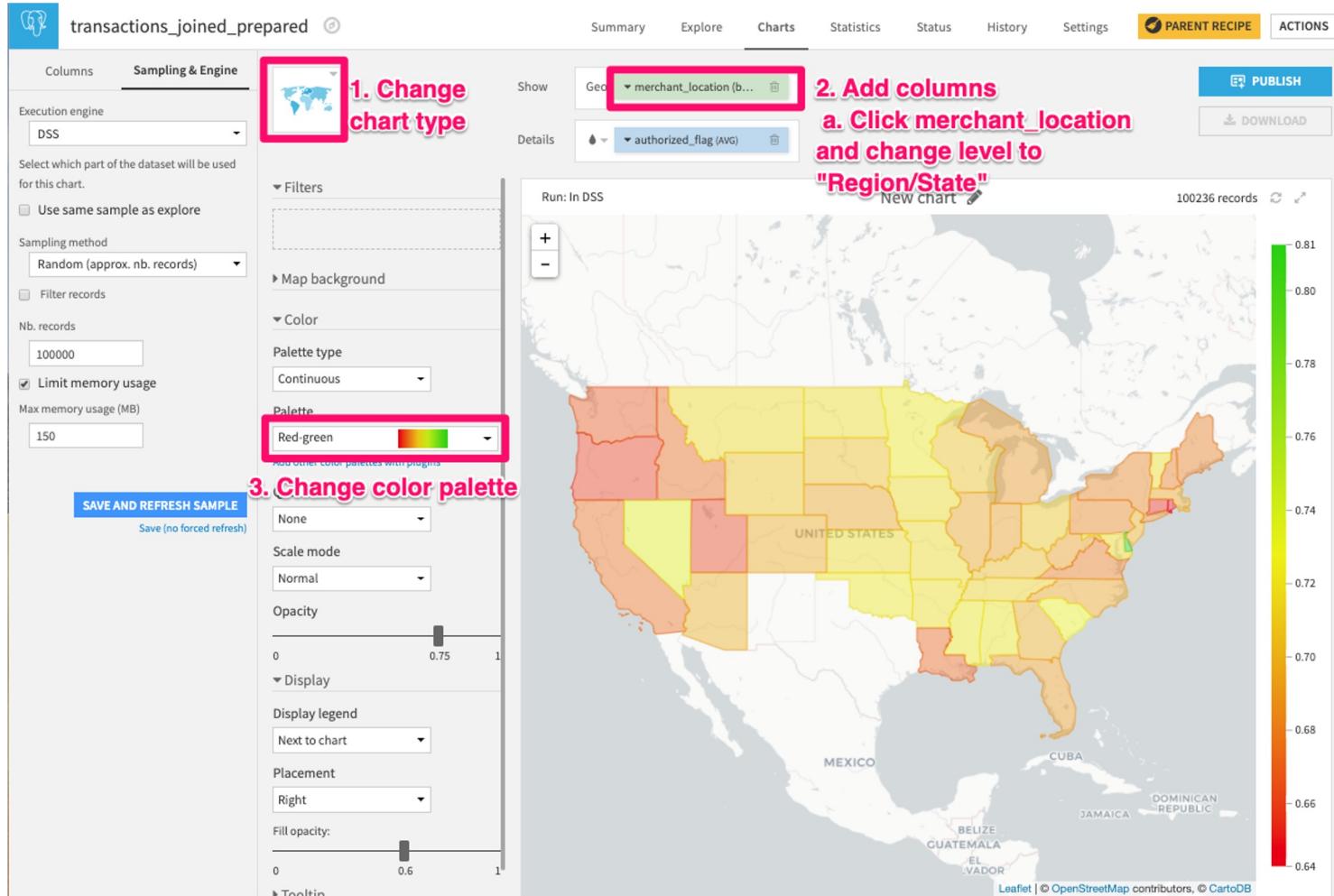
Details:

- Filled administrative map
- Geo - merchant\_location
- Fill - authorized\_flag (AVG)

Note:

- Change the sample to 100,000 random (left hand menu)

# Maps



# Scatter plots

Task:

- Plot the purchase\_amount by FICO score - show which transactions are fraudulent
- Does it look like people with higher FICO scores have more fraudulent transactions?
- What about high vs. low transaction amounts?

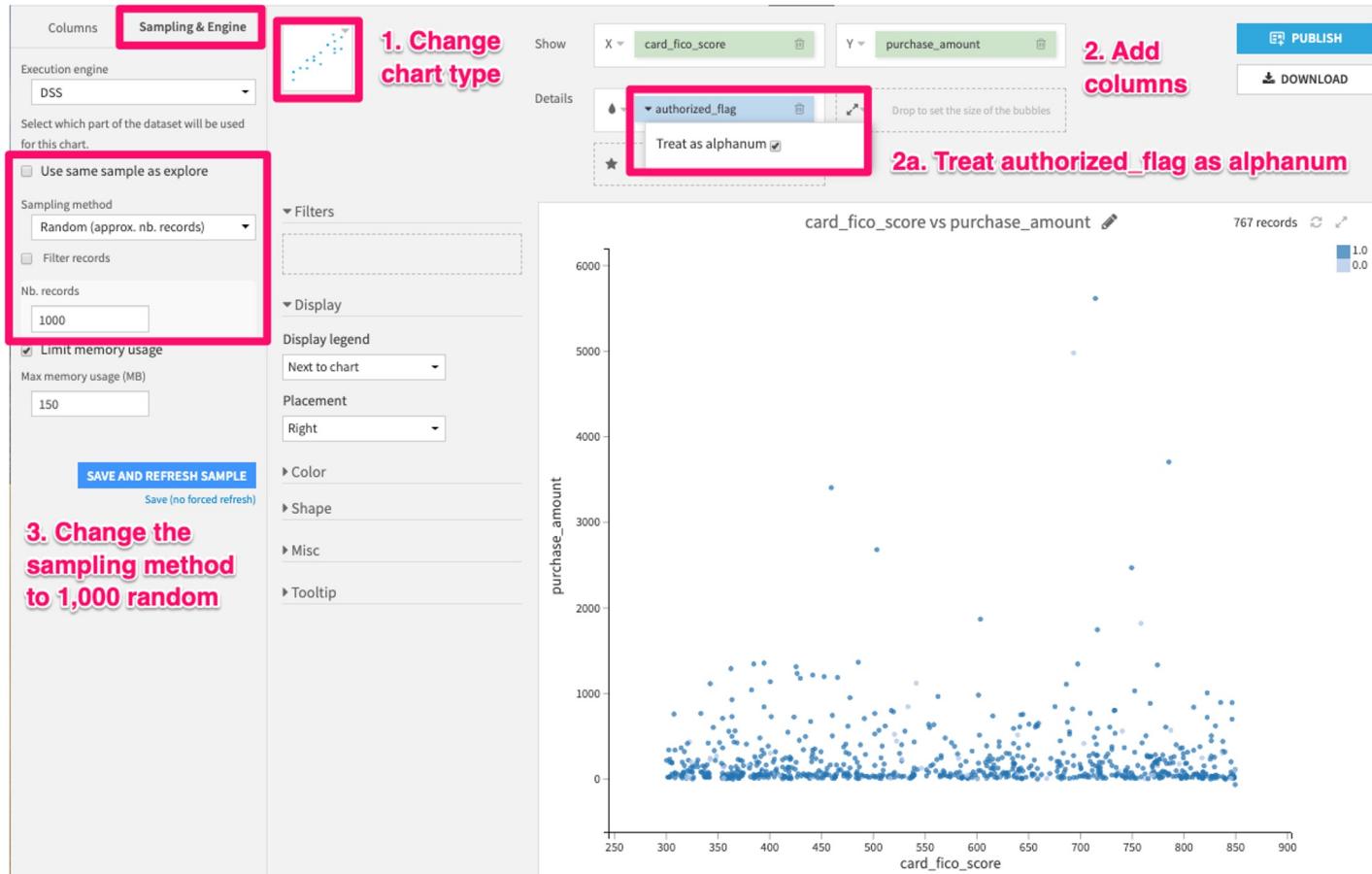
Details:

- Scatter plot
- X-axis - card\_fico\_score
- Y-axis - purchase\_amount
- Color - authorized\_flag

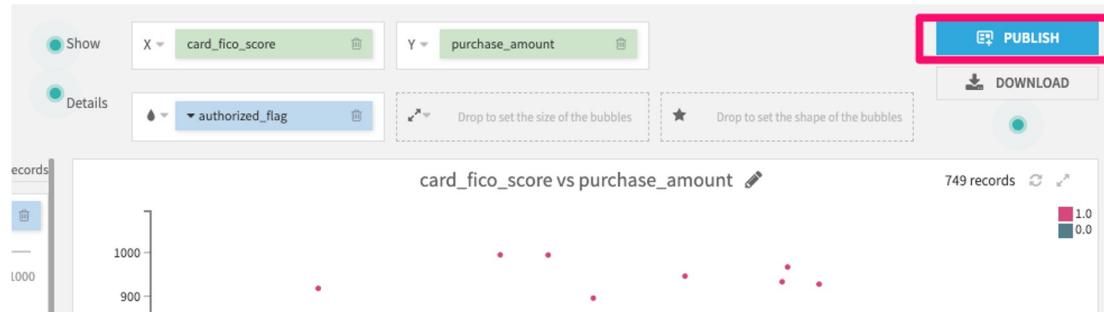
Note:

- Change the sample to 1,000 random (left hand menu)
- Filter for only purchases less than \$1000

# Scatter plots

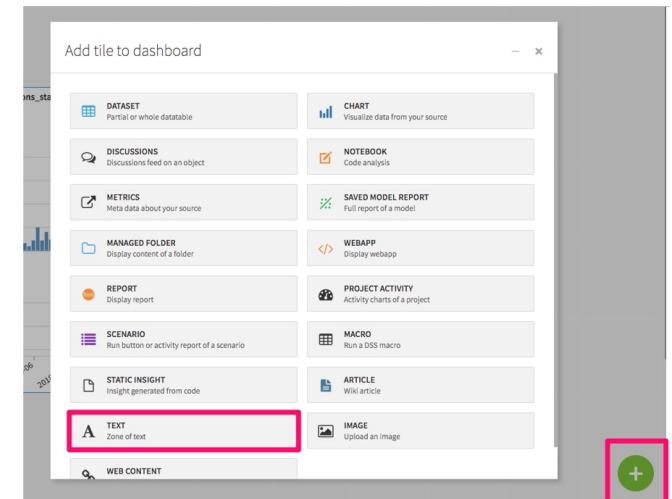


# Publish both of your charts to a dashboard



Add more content to your dashboard by clicking the green plus button

- Add a title (e.g. Credit Card Fraud Patterns)



# Dashboards

