

Dataset: HRDataset_v14.csv

Background: HR Analytics

In this notebook, I explore an HR dataset to analyze employee structure and organizational trends.

The focus is on **data reorganization, filtering, and multi-index structures** using Python and Pandas.

I applied techniques for **data cleaning**, transforming and managing **DataFrames and Series**, and building insights about workforce composition and turnover.

Skills demonstrated: Data cleaning, multi-index DataFrames, filtering, organizational analysis, data persistence

Question 1 - Initial loading and exploration

1. The pandas library was imported; the "read_csv" function was called to read the dataset in question; The pandas function "set_option" was used here with the "display.max_columns" and "None" arguments in order to show all the columns in the original dataset.
2. In this context, the "info()" method was used to bring the names, types, and number of columns in the dataset.

After analysis we can identify the "ManagerName", "PerformanceScore" and "Gender ID" columns that should be converted to the "category" type.

3. As a result of the operation performed, an object of type Pandas Series was selected, now representing a "PerformanceScore" column with its associated index.



```
# Questão 1.1
import pandas as pd
df= pd.read_csv("HRDataset_v14.csv")
pd.set_option("display.max_columns", None)

# Questão 1.2
df.info()

# Questão 1.3
performance_series= df["PerformanceScore"]
performance_series.head(10)
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 311 entries, 0 to 310
Data columns (total 36 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Employee_Name          311 non-null   object
1   EmpID                  311 non-null   int64
2   MarriedID              311 non-null   int64
3   MaritalStatusID        311 non-null   int64
4   GenderID               311 non-null   int64
5   EmpStatusID            311 non-null   int64
6   DeptID                 311 non-null   int64
7   PerfScoreID            311 non-null   int64
8   FromDiversityJobFairID 311 non-null   int64
9   Salary                 311 non-null   int64
10  Termd                  311 non-null   int64
11  PositionID             311 non-null   int64
12  Position                311 non-null   object
13  State                  311 non-null   object
14  Zip                    311 non-null   int64
15  DOB                    311 non-null   object
16  Sex                    311 non-null   object
17  MaritalDesc            311 non-null   object
18  CitizenDesc            311 non-null   object
19  HispanicLatino         311 non-null   object
20  RaceDesc               311 non-null   object
21  DateofHire             311 non-null   object
22  DateofTermination       104 non-null   object
23  TermReason             311 non-null   object
24  EmloymentStatus        311 non-null   object
```

	Employee_Name o.	EmpID int64	MarriedID int64	MaritalStatusID i...	GenderID int64	EmpStatusID int64	DeptID int64	P
0	Adinolfi, Wilson K	10026	0	0	1	1	5	
1	Ait Sidi, Karthikey...	10084	1	1	1	5	3	
2	Akinkuolie, Sarah	10196	1	1	0	5	5	
3	Alagbe,Trina	10088	1	1	0	1	5	
4	Anderson, Carol	10069	0	2	0	5	5	

5 rows, 36 cols 10 / page

<< < Page 1 of 1 > >>



Question 2 - Selection of data for performance analysis

1. In this code we applied the filters requested in the question, applied the ".loc[]" function so that it brought only the records that were equivalent to the filters and visualized the result through the ".head()" method.
2. In this code block, only employees who have an absence number greater than 10 and who have a salary greater than 80000 have been filtered.
3. To perform filtering in pure Python (without the Pandas library), we would start by opening the CSV file using the standard library csv module; we would go line by line using a "for"; we would apply the conditions of the question, more than 10 absences and salary greater than 80,000; We would store the result in a list.

```
# Question 2.1 filters= (df["EmploymentStatus"] == "Active") & (df["PerfScoreID"] == 3)
result= df.loc[filters, ["Employee_Name", "Department", "PerformanceScore", "ManagerName"]] # Question 2.2 criteria= (df[
output= df.loc[criteria] # Question 2.1 result.head(10) # Question 2.3 - Markdown
```

	Employee_Name o...	Department object	PerformanceScore	ManagerName o...	
	Alagbe,Trina ... 10% Andreola, C... .. 10% 8 others 80%	Production 80% Software E... .. 10% IT/IS 10%	Fully Meets ... 100%	Elijah Gray 20% Ketsia Liebig ... 20% 6 others 60%	
3	Alagbe,Trina	Production	Fully Meets	Elijah Gray	
6	Andreola, Colby	Software Enginee...	Fully Meets	Alex Sweetwater	
7	Athwal, Sam	Production	Fully Meets	Ketsia Liebig	
8	Bachiochi, Linda	Production	Fully Meets	Brannon Miller	
9	Bacong, Alejandro	IT/IS	Fully Meets	Peter Monroe	
13	Baron, Francesco A	Production	Fully Meets	Kelley Spirea	
17	Beatrice, Courtney	Production	Fully Meets	Elijah Gray	
19	Becker, Scott	Production	Fully Meets	Webster Butler	
20	Bernstein, Sean	Production	Fully Meets	Amy Dunn	
21	Biden, Lowan M	Production	Fully Meets	Ketsia Liebig	

10 rows, 4 cols 10 / page << < Page 1 of 1 > >> [↓](#)

Question 3 - Structure with multiple indexes and segmentation by manager

1. The mentioned columns have been selected and stored in a new dataframe called "novo_df".
2. The Pandas "set_index" function was called to configure the "Department" and "ManagerName" columns as the indexes of the new dataframe.
3. The use of multi-indexes in this context helps to organize the data according to hierarchical levels, such as "Department" - "Manager" - "Employee"; it facilitates the visualization of the dataset and ends up facilitating the interpretation and understanding of the data; It facilitates data aggregation and in addition, there is data optimization, without the need to create additional columns or multiple filters.

```
# Question 3.1 novo_df= df[["Department", "ManagerName", "PerformanceScore", "Salary"]] # Question 3.2 multi_index_df= novo_df.set_index(["Department", "ManagerName"])
multi_index_df.head(5) # Issue 3.3 - Markdown
```

	PerformanceScore	Salary int64	
('Pr...	Exceeds	62506	
('IT...	Fully Meets	104437	
('Pr...	Fully Meets	64955	
('Pr...	Fully Meets	64991	
('Pr...	Fully Meets	50825	

5 rows, 2 cols 10 / page << < Page 1 of 1 > >> [↓](#)

Question 4 - Transformations and adjustments

1. In this question, we store in the variable map_scores a dictionary corresponding the value to its numerical correspondent and then apply the ".map" function in the "PerformanceScore" column using the variable defined above as a parameter.
2. In the code below, the expression "Salary * (1 + Absences / 100)" is used with the "Salary" and "Absences" columns as a parameter, returning a value that indicates the total cost of the employee (salary + absences) to the company.
3. Adjusted and calculated metrics such as the "Cost_Index" column help managers make better decisions; make data more reliable with regard to the reality of the company and its employees; It allows you to observe the company's actual spending and adds a weight related to behavior (absences).

```
# Questão 4.1 map_scores= { "Exceeds": 3, "Fully Meets": 2, "Needs Improvement": 1, "PIP": 0 }
df["PerformanceScore"] = df["PerformanceScore"].map(map_scores)
df[["Employee_Name", "PerformanceScore"]].head(10) # Questão 4.2
df["Cost_Index"] = df["Salary"] * (1 + df["Absences"] / 100)
df[["Employee_Name", "Salary", "Absences", "Cost_Index"]].head(10) # Questão 4.3 - Markdown
```

	Employee_Name o...	Salary int64 47837 - 104437	Absences int64 1 - 19	Cost_Index float64 49750.48 - 12219...	
	Adinolfi, Wil... ~ 10% Ait Sidi, Kar... ~ 10% 8 others ~ 80%				
0	Adinolfi, Wilson K	62506	1	63131.06	
1	Ait Sidi, Karthikey...	104437	17	122191.29	
2	Akinkuolie, Sarah	64955	3	66903.65	
3	Alagbe, Trina	64991	15	74739.65	
4	Anderson, Carol	50825	2	51841.5	
5	Anderson, Linda	57568	15	66203.2	
6	Andreola, Colby	95660	19	113835.4	
7	Athwal, Sam	59365	19	70644.35	
8	Bachiochi, Linda	47837	4	49750.48	
9	Bacong, Alejandro	50178	16	58206.48	

10 rows, 4 cols 10 / page << < Page 1 of 1 > >> [Download](#)

Question 5 - Persistence of data

- The dataframe created in question 3 and modified in question 4, with the column "Cost_Index" was brought into this code and transformed into a standard CSV (with a comma) and another with a semicolon (;).
- The pandas "read_csv" method was used here to read the two CSVs and the "display" method to show the first 5 rows of each df returned.
- If the tab used in the save is not the same as the one informed in the reading, Pandas (or other software) can read all columns as a single string or even generate incorrect extra columns; errors also happen when the encoding in which the file was saved is not the same as the one in which it is being read, as an example: file saved in "UTF-8", but read in "Latin-1" can lead to errors such as "André" (with the acute accent) appearing as another character.

```
# Questão 5.1
final_df = df[["Department", "ManagerName", "PerformanceScore", "Salary", "Cost_Index"]]
final_df.to_csv("Dataset_virgula.csv", index=False)
final_df.to_csv("Dataset_pontoevirgula.csv", sep=";", index=False)
final_df.head()
```

```
# Questão 5.2
df_virgula = pd.read_csv("Dataset_virgula.csv")
df_pontoevirgula = pd.read_csv("Dataset_pontoevirgula.csv", sep=";")

display(df_virgula.head())
display(df_pontoevirgula.head())
```

Questão 5.3 - Markdown

	Department object	ManagerName o...	PerformanceScore	Salary int64	Cost_Index float64	
0	Production	Michael Albert	3	62506	63131.06	
1	IT/IS	Simon Roup	2	104437	122191.29	
2	Production	Kissy Sullivan	2	64955	66903.65	
3	Production	Elijah Gray	2	64991	74739.65	
4	Production	Webster Butler	2	50825	51841.5	

5 rows, 5 cols 10 / page << < Page 1 of 1 > >> [Download](#)

	Department object	ManagerName o...	PerformanceScore	Salary int64	Cost_Index float64	
0	Production	Michael Albert	3	62506	63131.06	
1	IT/IS	Simon Roup	2	104437	122191.29	
2	Production	Kissy Sullivan	2	64955	66903.65	
3	Production	Elijah Gray	2	64991	74739.65	
4	Production	Webster Butler	2	50825	51841.5	

5 rows, 5 cols 10 / page << < Page 1 of 1 > >> [Download](#)