Documentación del Código

This document is a guide to understanding the code below, which is used in an ETL process for a job interview.

LIBRARIES USED

The code makes use of the following Python libraries:

- **`pandas`**`: for working with data tables`
- **`numpy`**`: for mathematical operations`
- **`psycopg2`**`: for connecting to a PostgreSQL database`
- **`sqlalchemy`**`: for working with relational databases`

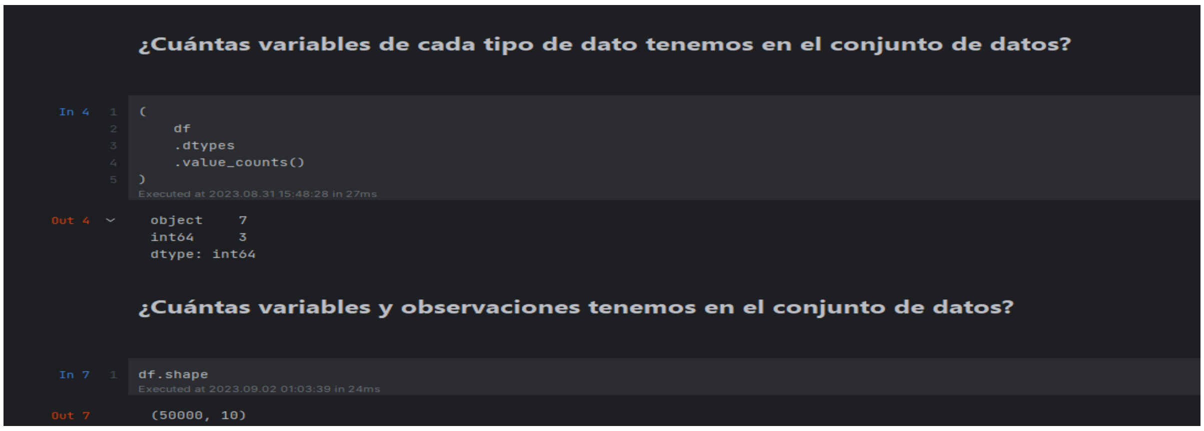
Read data.

A csv file with data of candidates for a job position is read. The read_csv function of the pandas library is used to read the file and the field separator (sep) is specified as ";". The resulting table is printed using the head function.

Exploratory Data Analysis

An exploratory analysis of the data is performed to find out the data type of the variables, how many variables of each type there are, how many observations there are in total, if there are null values in the table and descriptive statistics of the numerical and categorical variables are displayed.

The data set has 50,000 records and 10 columns of which 7 are objects and 3 are numerical.



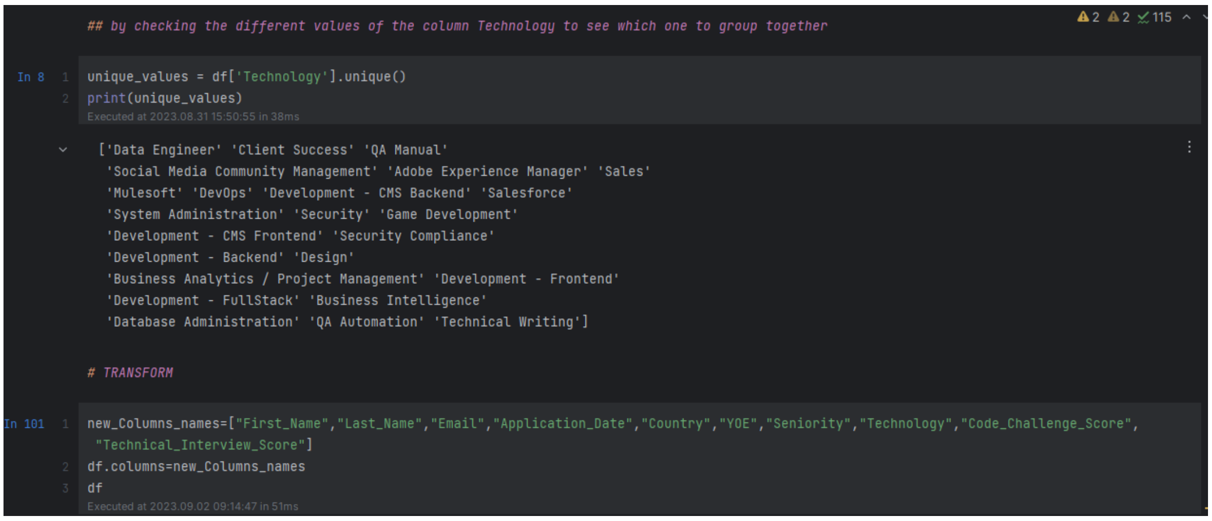
¿Cuántas variables de cada tipo de dato tenemos en el conjunto de datos?

```
In 4
(
    df
    .dtypes
    .value_counts()
)
Executed at 2023.08.31 15:48:28 in 27ms

Out 4
 object    7
 int64     3
 dtype: int64
```

¿Cuántas variables y observaciones tenemos en el conjunto de datos?

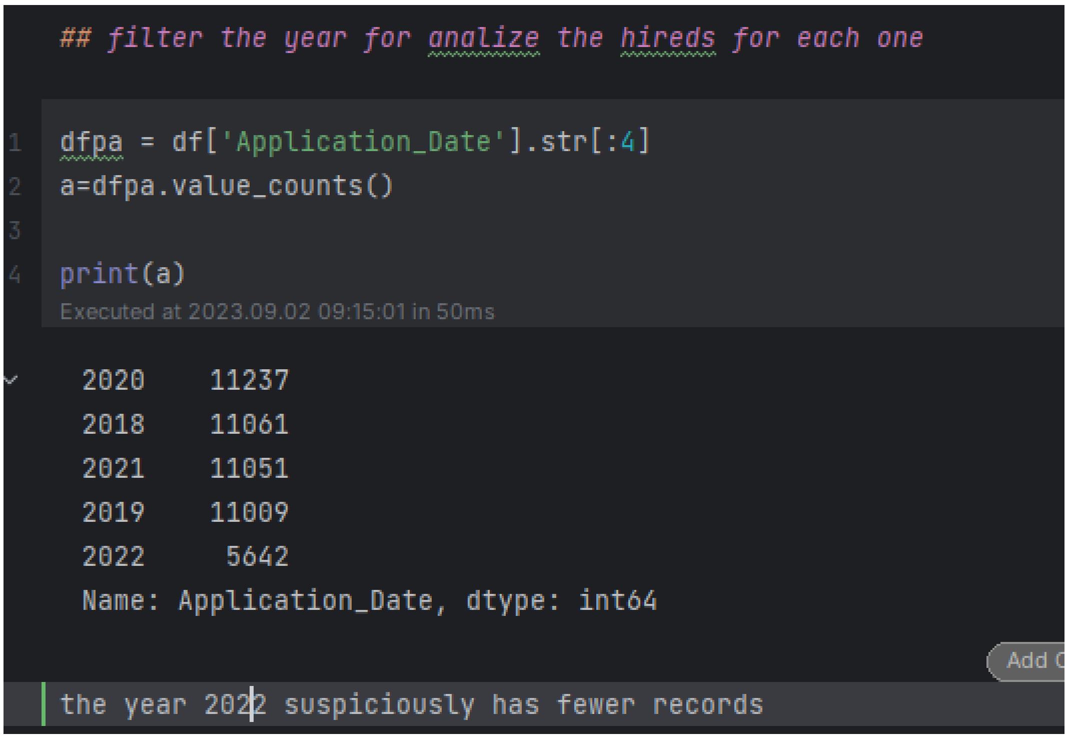
```
In 7
df.shape
Executed at 2023.09.02 01:03:39 in 24ms

Out 7
 (50000, 10)
```

we checking the different values of the column Technology to see which one to group together in the future, because the pie chart is very difficult to visualize because of the number of different variables.



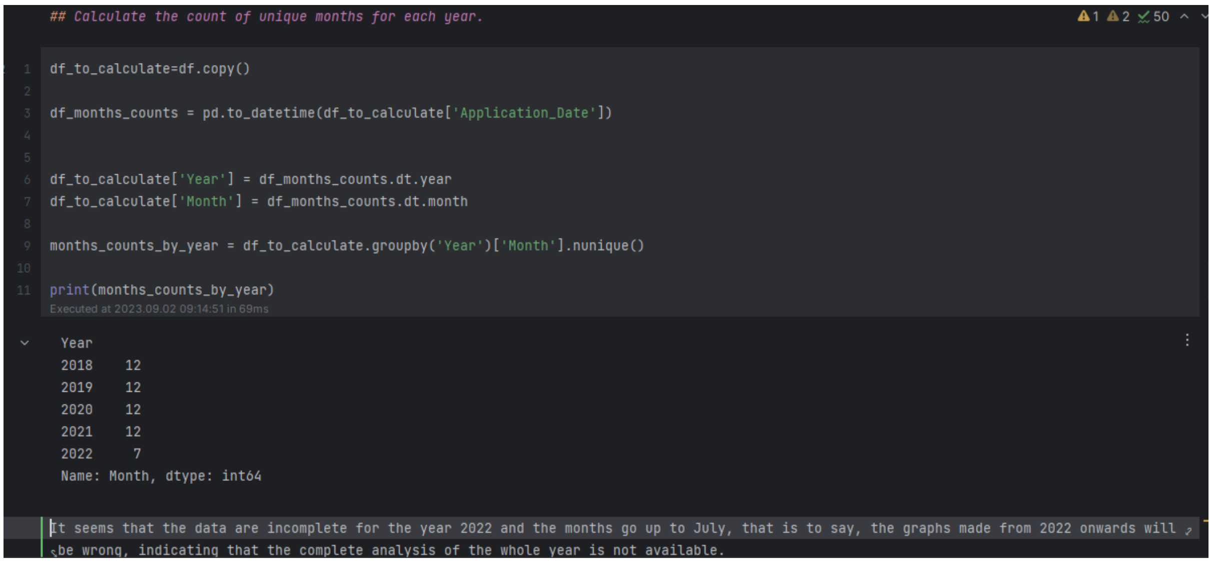
```
## by checking the different values of the column Technology to see which one to group together

In 8
unique_values = df['Technology'].unique()
print(unique_values)
Executed at 2023.08.31 15:50:55 in 38ms

 ['Data Engineer' 'Client Success' 'QA Manual'
  'Social Media Community Management' 'Adobe Experience Manager' 'Sales'
  'Mulesoft' 'DevOps' 'Development - CMS Backend' 'Salesforce'
  'System Administration' 'Security' 'Game Development'
  'Development - CMS Frontend' 'Security Compliance'
  'Development - Backend' 'Design'
  'Business Analytics / Project Management' 'Development - Frontend'
  'Development - FullStack' 'Business Intelligence'
  'Database Administration' 'QA Automation' 'Technical Writing']

# TRANSFORM

In 101
new_Columns_names=["First_Name","Last_Name","Email","Application_Date","Country","YOE","Seniority","Technology","Code_Challenge_Score",
 "Technical_Interview_Score"]
df.columns=new_Columns_names
df
Executed at 2023.09.02 09:14:47 in 51ms
```

we analyze the **Aplication_Date** column, specifically the year, as it is one of the most important columns for the visualizations. Within the analysis we realize that the year 2022 has less records.



```
## filter the year for analize the hireds for each one

dfpa = df['Application_Date'].str[:4]
a=dfpa.value_counts()

print(a)
Executed at 2023.09.02 09:15:01 in 50ms

 2020    11237
 2018    11061
 2021    11051
 2019    11009
 2022     5642
 Name: Application_Date, dtype: int64

the year 2022 suspiciously has fewer records
```

here we realize that only 7 months were taken into account for the year 2022.



```
## Calculate the count of unique months for each year.

df_to_calculate=df.copy()

df_months_counts = pd.to_datetime(df_to_calculate['Application_Date'])


df_to_calculate['Year'] = df_months_counts.dt.year
df_to_calculate['Month'] = df_months_counts.dt.month

months_counts_by_year = df_to_calculate.groupby('Year')['Month'].nunique()

print(months_counts_by_year)
```

Executed at 2023.09.02 09:14:51 in 69ms

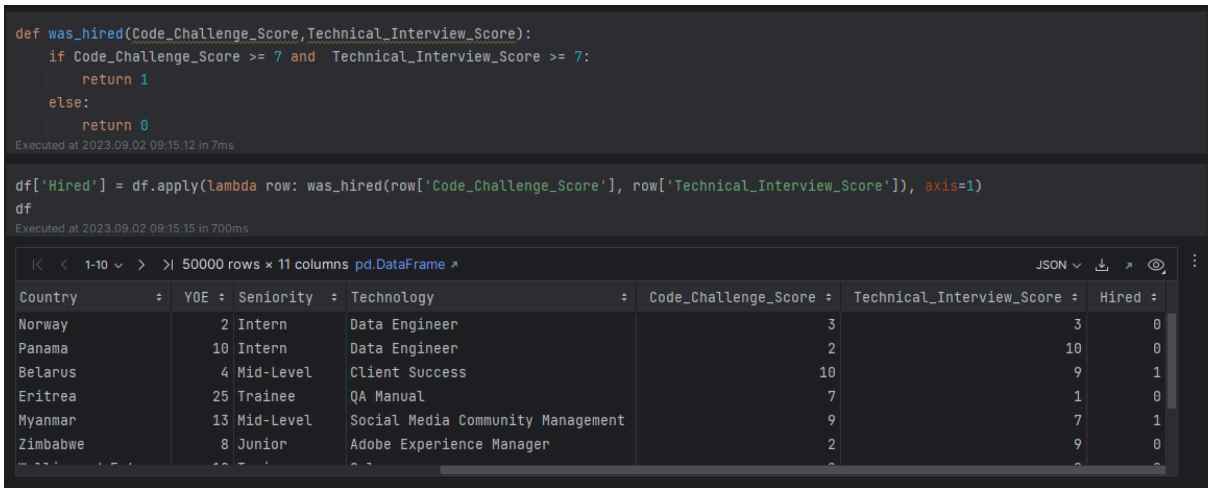
```
Year
2018    12
2019    12
2020    12
2021    12
2022     7
Name: Month, dtype: int64
```

It seems that the data are incomplete for the year 2022 and the months go up to July, that is to say, the graphs made from 2022 onwards will be wrong, indicating that the complete analysis of the whole year is not available.

Transform Of Data

Somes transformations are made to the data, such as changing the column names, adding a **Hired** column indicating whether the candidate was hired or not.

The **was_hired** function takes two arguments: **Code_Challenge_Score** and **Technical_Interview_Score**. This function returns 1 if both scores are greater than or equal to 7, which means that the candidate was hired. If not, it returns 0, which means that the candidate was not hired.



```
def was_hired(Code_Challenge_Score,Technical_Interview_Score):
    if Code_Challenge_Score >= 7 and  Technical_Interview_Score >= 7:
        return 1
    else:
        return 0
```

Executed at 2023.09.02 09:15:12 in 7ms

```
df['Hired'] = df.apply(lambda row: was_hired(row['Code_Challenge_Score'], row['Technical_Interview_Score']), axis=1)
df
```

Executed at 2023.09.02 09:15:15 in 700ms

1-10 50000 rows × 11 columns pd.DataFrame

Country	YOE	Seniority	Technology	Code_Challenge_Score	Technical_Interview_Score	Hired
Norway	2	Intern	Data Engineer	3	3	0
Panama	10	Intern	Data Engineer	2	10	0
Belarus	4	Mid-Level	Client Success	10	9	1
Eritrea	25	Trainee	QA Manual	7	1	0
Myanmar	13	Mid-Level	Social Media Community Management	9	7	1
Zimbabwe	8	Junior	Adobe Experience Manager	2	9	0

Then, this function is applied to each row of the DataFrame df using the apply method. The axis=1 argument indicates that the application of the function will be performed along the rows instead of the columns. The result is stored in a new column named 'Hired' in the DataFrame df.

Connection To The DataBase

A connection to a PostgreSQL database is established using the psycopg2 library. A table named Applicants_interview_good is created and the to_sql function of the pandas library is used to load the data from the df table into the database table.

The detailed description of the code used to interact with a PostgreSQL database in a Docker container. The code is divided into several sections, each with its own functionality and specific goals and can be created as follows

sudo docker run -d --name=postgres -p 5432:5432 -v postgres-volume:/var/lib/postgresql/data -e POSTGRES_PASSWORD=mysecretpass postgres

.

Connection Parameter Section

This section defines the parameters required to establish the connection to the PostgreSQL database in the Docker container. These parameters include:

host: The IP address or hostname of the Docker container.

port: The port mapped in the Docker container for the PostgreSQL database.

database: The name of the database you want to connect to.

user: The user name for the connection to the database.

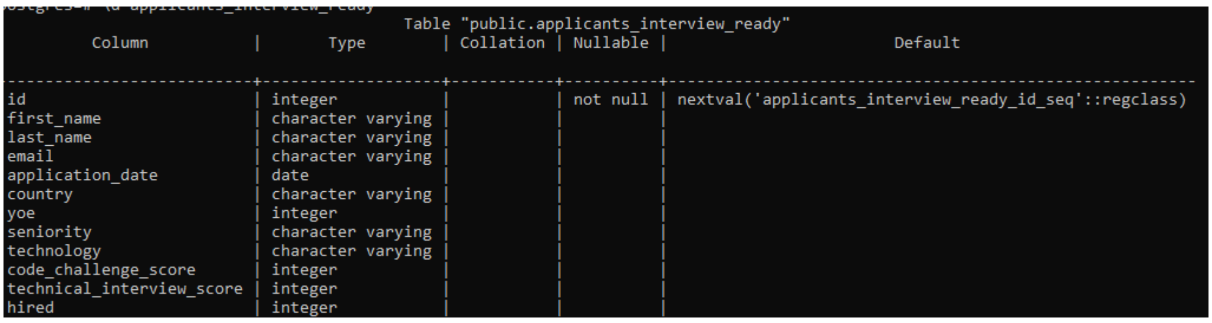
password: The user's password for the connection.

Connection Attempt

Within a "try-except" block, this code attempts to establish a connection to the database using the previously defined parameters. In case of a successful connection, it displays the PostgreSQL version as confirmation.

Tables Query

This code performs a query on the information_schema catalog to get the list of tables in the database and displays them in the console. This helps to verify which tables exist in the database.



```
                                          Table "public.applicants_interview_ready"
         Column           |       Type        | Collation | Nullable |                        Default

--------------------------+-------------------+-----------+----------+-------------------------------------------------------
 id                       | integer           |           | not null | nextval('applicants_interview_ready_id_seq'::regclass)
 first_name               | character varying |           |          |
 last_name                | character varying |           |          |
 email                    | character varying |           |          |
 application_date         | date              |           |          |
 country                  | character varying |           |          |
 yoe                      | integer           |           |          |
 seniority                | character varying |           |          |
 technology               | character varying |           |          |
 code_challenge_score     | integer           |           |          |
 technical_interview_score | integer          |           |          |
 hired                    | integer           |           |          |
```

Closing the Connection

Once all the necessary operations have been performed, the connection is closed properly using cursor.close() and connection.close().

Creating a Table in the Database

Table Creation Query

This section contains an SQL query that creates a table in the database. You can customize this query to suit your specific table structure needs. The query checks if the table already exists before attempting to create it.

Executing the Query

The table creation query is executed using the database connection cursor, and changes are committed with connection.commit().

Loading Data from a DataFrame to the Database

Creating the Connection String

SQLAlchemy is used to create a connection string to the PostgreSQL database. This string contains information about the user, password, IP address and port of the Docker container, as well as the database name.

Loading Data from the DataFrame

The code loads data from a pandas DataFrame (df) to a specific table in the PostgreSQL database. The if_exists parameter determines the behavior in case the table already exists (in this case, it replaces the existing table).

Closing the Connection

Once the data has been loaded into the database, the connection is closed using engine.dispose().

Querying Data from the Database

Data Query

This code block performs a SQL query to retrieve data from a specific table (applicants_interview_good) in the PostgreSQL database. The query is executed and the results are obtained.

Displaying the Results

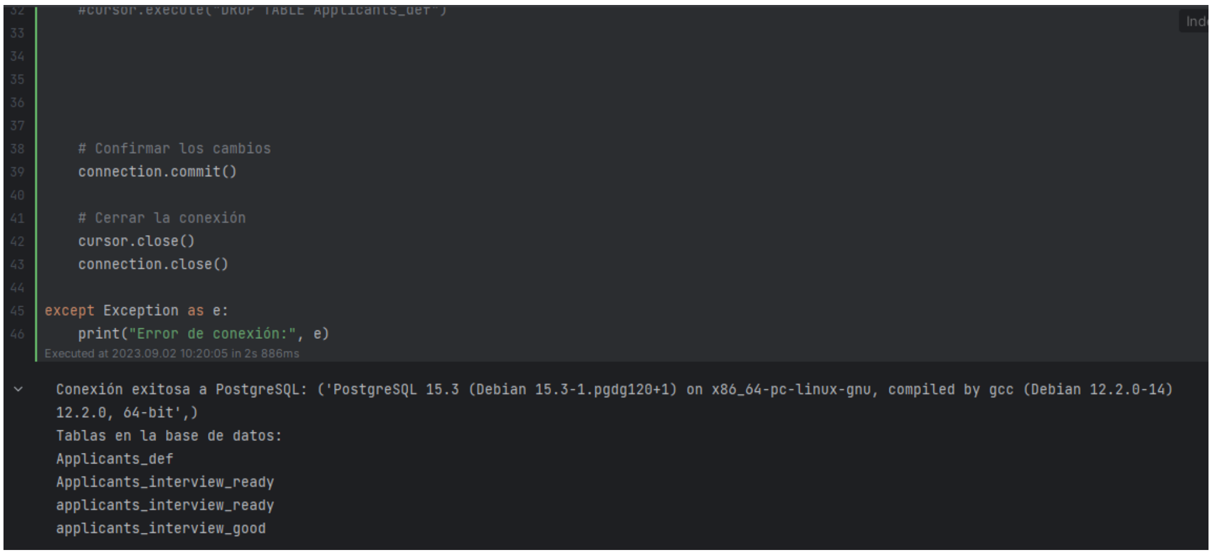
The query results are stored in the results variable and displayed in the console for viewing.

It is important to remember to customize SQL queries, table names and other parameters according to the specific needs of your application and database. Also, make sure you have the proper dependencies, such as the psycopg2 and SQLAlchemy library, to run this code successfully.

Verification of Data in the Database

We verify that the data has been loaded correctly in the Applicants_interview_ready table of the database.

we can verify that we are connected

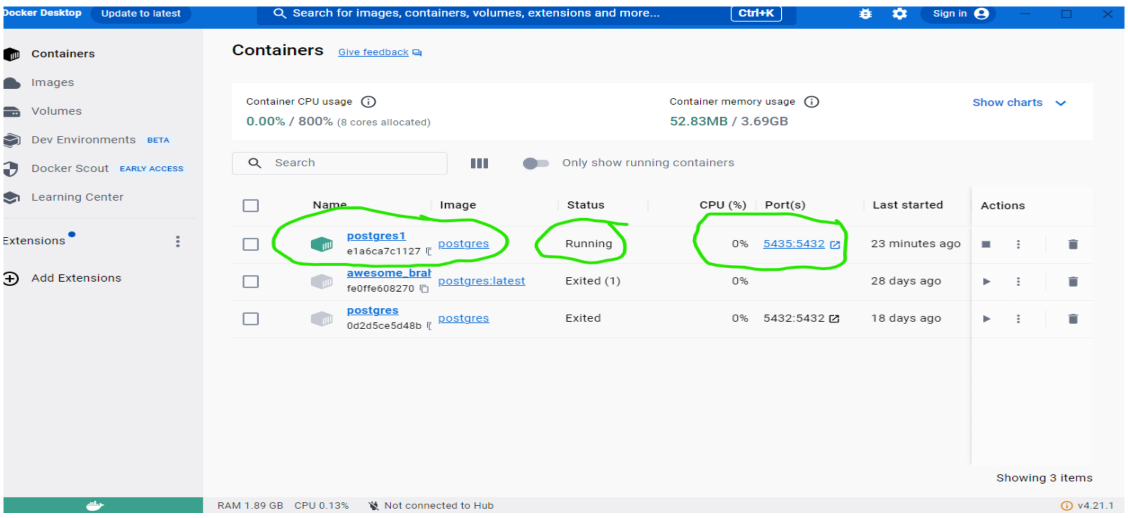


```
    #cursor.execute("DROP TABLE Applicants_def")




    # Confirmar los cambios
    connection.commit()

    # Cerrar la conexión
    cursor.close()
    connection.close()

except Exception as e:
    print("Error de conexión:", e)
Executed at 2023.09.02 10:20:05 in 2s 886ms
```

```
Conexión exitosa a PostgreSQL: ('PostgreSQL 15.3 (Debian 15.3-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14)
12.2.0, 64-bit',)
Tablas en la base de datos:
Applicants_def
Applicants_interview_ready
applicants_interview_ready
applicants_interview_good
```

Conncection With PowerBI

Usamos Docker para crear la base de datos:
Usamos el puerto 5435 que mapea al puerto 5432 dentro del container,que es el puerto predeterminado que usa postgres. Nosotros usamos el puerto 5435 de nuestra computadora, porque tenemos ocupada el predeterminado (5432) con otra base de datos



Containers

Container CPU usage 0.00% / 800% (8 cores allocated)

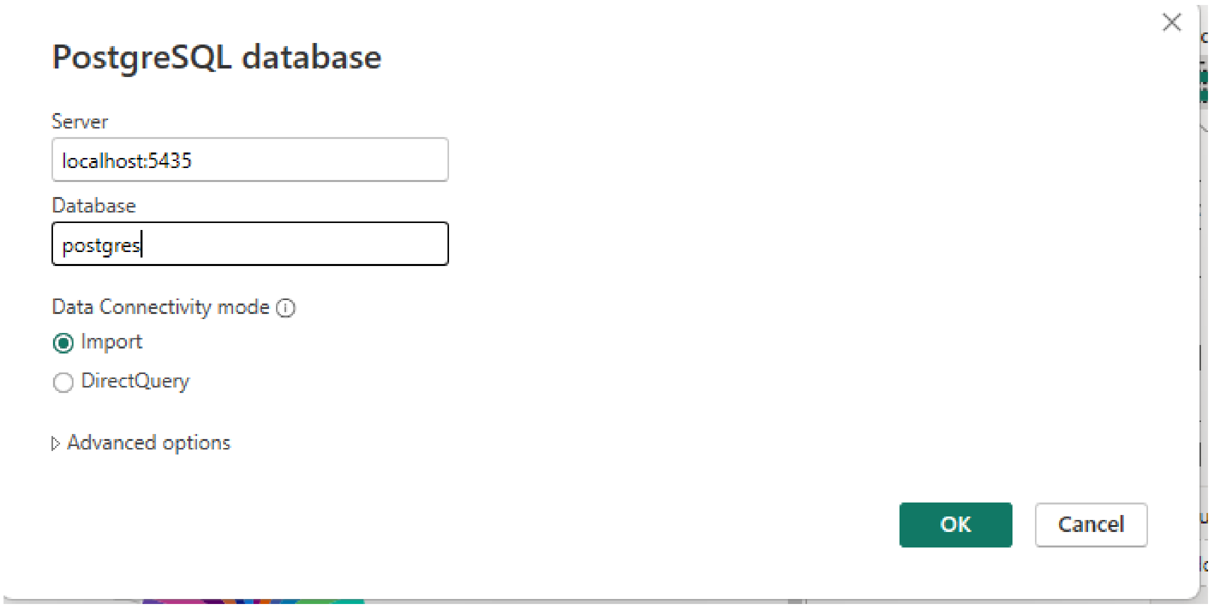
Container memory usage 52.83MB / 3.69GB

Name	Image	Status	CPU (%)	Port(s)	Last started
postgres1 e1a6ca7c1127	postgres	Running	0%	5435:5432	23 minutes ago
awesome_brah fe0ffe608270	postgres:latest	Exited (1)	0%		28 days ago
postgres 0d2d5ce5d48b	postgres	Exited	0%	5432:5432	18 days ago

Showing 3 items

RAM 1.89 GB CPU 0.13% Not connected to Hub v4.21.1

We use Docker to create the database:
We use port 5435 which maps to port 5432 inside the container,which is the default port that postgres uses. We use the port 5435 of our computer, because we have occupied the default port (5432) with another database.



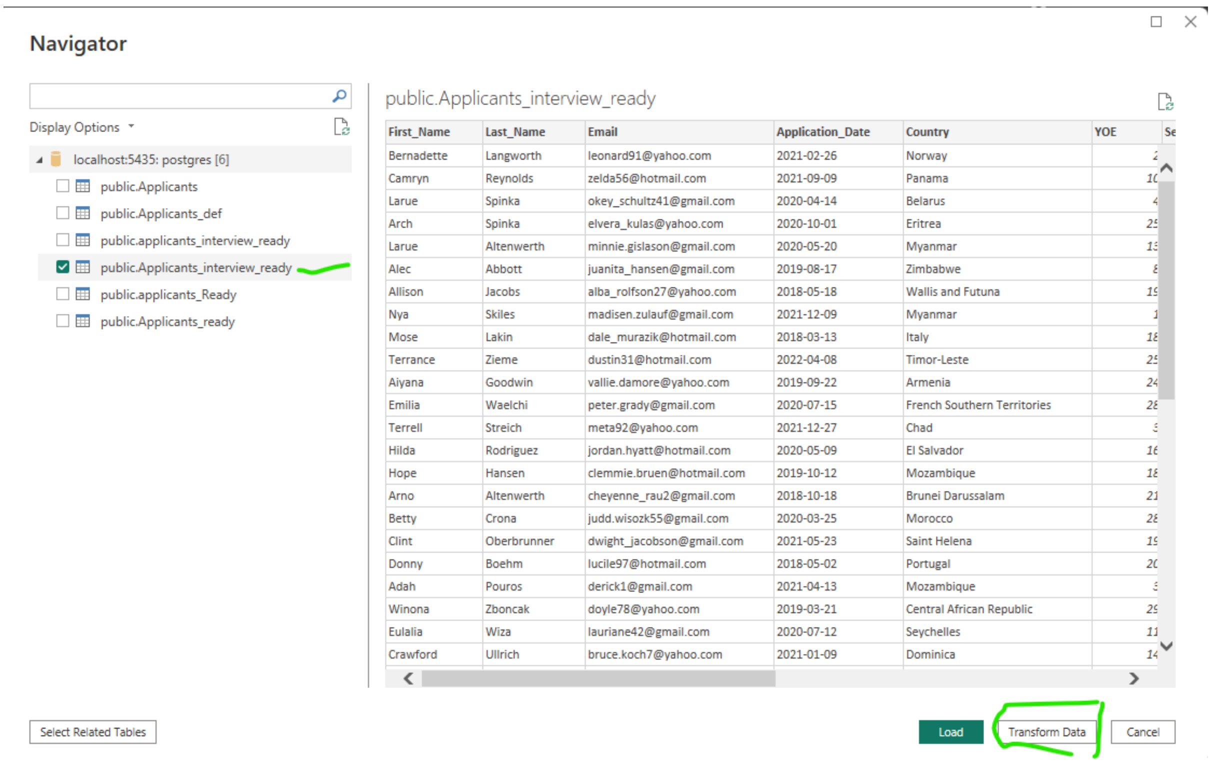
PostgreSQL database

Server: localhost:5435

Database: postgres

Data Connectivity mode
[✓ Import] DirectQuery

let's go to the transformation section.



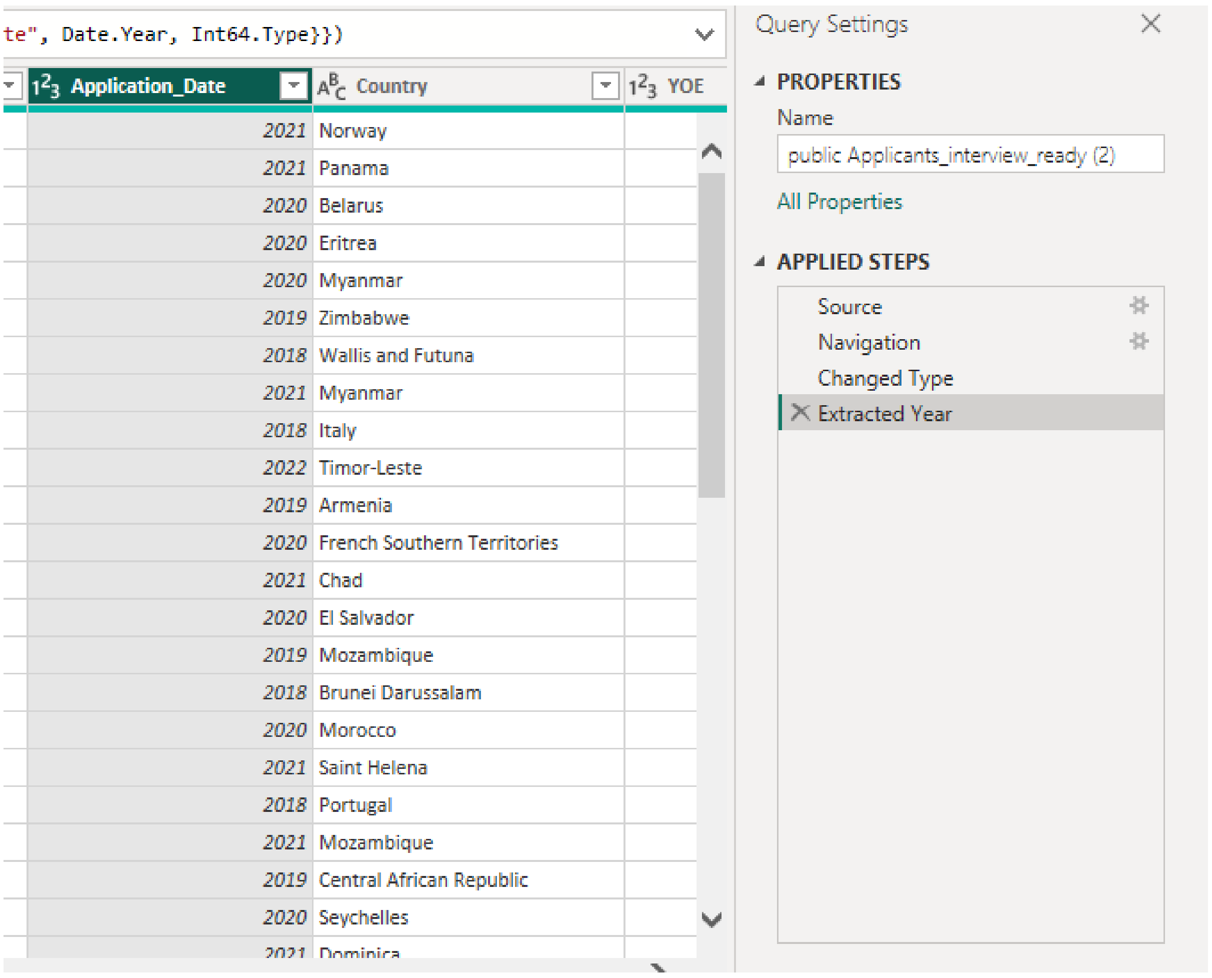
Navigator

- localhost:5435: postgres [6]
  - ☐ public.Applicants
  - ☐ public.Applicants_def
  - ☐ public.applicants_interview_ready
  - ☑ public.Applicants_interview_ready
  - ☐ public.applicants_Ready
  - ☐ public.Applicants_ready

public.Applicants_interview_ready

First_Name	Last_Name	Email	Application_Date	Country	YOE	Se
Bernadette	Langworth	leonard91@yahoo.com	2021-02-26	Norway	2	
Camryn	Reynolds	zelda56@hotmail.com	2021-09-09	Panama	10	
Larue	Spinka	okey_schultz41@gmail.com	2020-04-14	Belarus	4	
Arch	Spinka	elvera_kulas@yahoo.com	2020-10-01	Eritrea	25	
Larue	Altenwerth	minnie.gislason@gmail.com	2020-05-20	Myanmar	13	
Alec	Abbott	juanita_hansen@gmail.com	2019-08-17	Zimbabwe	8	
Allison	Jacobs	alba_rolfson27@yahoo.com	2018-05-18	Wallis and Futuna	19	
Nya	Skiles	madisen.zulauf@gmail.com	2021-12-09	Myanmar	1	
Mose	Lakin	dale_murazik@hotmail.com	2018-03-13	Italy	18	
Terrance	Zieme	dustin31@hotmail.com	2022-04-08	Timor-Leste	25	
Aiyana	Goodwin	vallie.damore@yahoo.com	2019-09-22	Armenia	24	
Emilia	Waelchi	peter.grady@gmail.com	2020-07-15	French Southern Territories	28	
Terrell	Streich	meta92@yahoo.com	2021-12-27	Chad	3	
Hilda	Rodriguez	jordan.hyatt@hotmail.com	2020-05-09	El Salvador	16	
Hope	Hansen	clemmie.bruen@hotmail.com	2019-10-12	Mozambique	18	
Arno	Altenwerth	cheyenne_rau2@gmail.com	2018-10-18	Brunei Darussalam	21	
Betty	Crona	judd.wisozk55@gmail.com	2020-03-25	Morocco	28	
Clint	Oberbrunner	dwight_jacobson@gmail.com	2021-05-23	Saint Helena	19	
Donny	Boehm	lucile97@hotmail.com	2018-05-02	Portugal	20	
Adah	Pouros	derick1@gmail.com	2021-04-13	Mozambique	3	
Winona	Zboncak	doyle78@yahoo.com	2019-03-21	Central African Republic	29	
Eulalia	Wiza	lauriane42@gmail.com	2020-07-12	Seychelles	11	
Crawford	Ullrich	bruce.koch7@yahoo.com	2021-01-09	Dominica	14	

We transform the column type to Date and leave only the year.



```
te", Date.Year, Int64.Type}})
```

Application_Date	Country	YOE
2021	Norway	
2021	Panama	
2020	Belarus	
2020	Eritrea	
2020	Myanmar	
2019	Zimbabwe	
2018	Wallis and Futuna	
2021	Myanmar	
2018	Italy	
2022	Timor-Leste	
2019	Armenia	
2020	French Southern Territories	
2021	Chad	
2020	El Salvador	
2019	Mozambique	
2018	Brunei Darussalam	
2020	Morocco	
2021	Saint Helena	
2018	Portugal	
2021	Mozambique	
2019	Central African Republic	
2020	Seychelles	
2021	Dominica	

Query Settings

PROPERTIES

Name: public Applicants_interview_ready (2)

All Properties

APPLIED STEPS

- Source
- Navigation
- Changed Type
- Extracted Year

We group and reduce the number of different variables in the Technology column to be able to plot it and make it look better:

Software Development:
Development - CMS Backend
Development - CMS Frontend
Development - Backend
Development - Frontend
Development - FullStack
Game Development

Project Management and Analysis:
Business Analytics / Project Management
Business Intelligence

Sales and Customer Relations:
Client Success
Sales
Salesforce

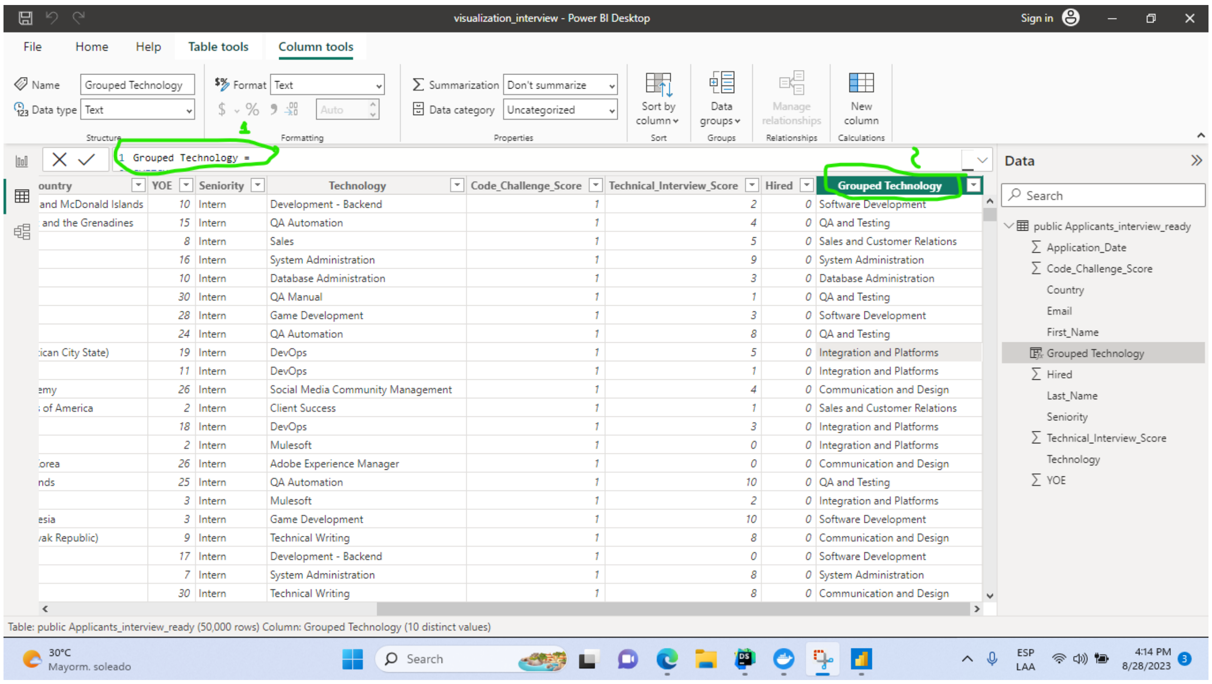
Security and Compliance:
Security
Security Compliance

QA and Testing:
QA Manual
QA Automation

Communication and Design:
Social Media Community Management
Adobe Experience Manager
Design
Technical Writing

Integration and Platforms:
Mulesoft
DevOps

Those that are not found here, it is because they have kept their initial name.



Name: Grouped Technology
Data type: Text
Format: Text
Summarization: Don't summarize
Data category: Uncategorized

```
1 Grouped Technology =
```

ountry	YOE	Seniority	Technology	Code_Challenge_Score	Technical_Interview_Score	Hired	Grouped Technology
and McDonald Islands	10	Intern	Development - Backend	1	2	0	Software Development
and the Grenadines	15	Intern	QA Automation	1	4	0	QA and Testing
	8	Intern	Sales	1	5	0	Sales and Customer Relations
	16	Intern	System Administration	1	9	0	System Administration
	10	Intern	Database Administration	1	3	0	Database Administration
	30	Intern	QA Manual	1	1	0	QA and Testing
	28	Intern	Game Development	1	3	0	Software Development
	24	Intern	QA Automation	1	8	0	QA and Testing
ican City State)	19	Intern	DevOps	1	5	0	Integration and Platforms
	11	Intern	DevOps	1	1	0	Integration and Platforms
emy	26	Intern	Social Media Community Management	1	4	0	Communication and Design
of America	2	Intern	Client Success	1	1	0	Sales and Customer Relations
	18	Intern	DevOps	1	3	0	Integration and Platforms
	2	Intern	Mulesoft	1	0	0	Integration and Platforms
orea	26	Intern	Adobe Experience Manager	1	0	0	Communication and Design
nds	25	Intern	QA Automation	1	10	0	QA and Testing
	3	Intern	Mulesoft	1	2	0	Integration and Platforms
esia	3	Intern	Game Development	1	10	0	Software Development
vak Republic)	9	Intern	Technical Writing	1	8	0	Communication and Design
	17	Intern	Development - Backend	1	0	0	Software Development
	7	Intern	System Administration	1	8	0	System Administration
	30	Intern	Technical Writing	1	8	0	Communication and Design

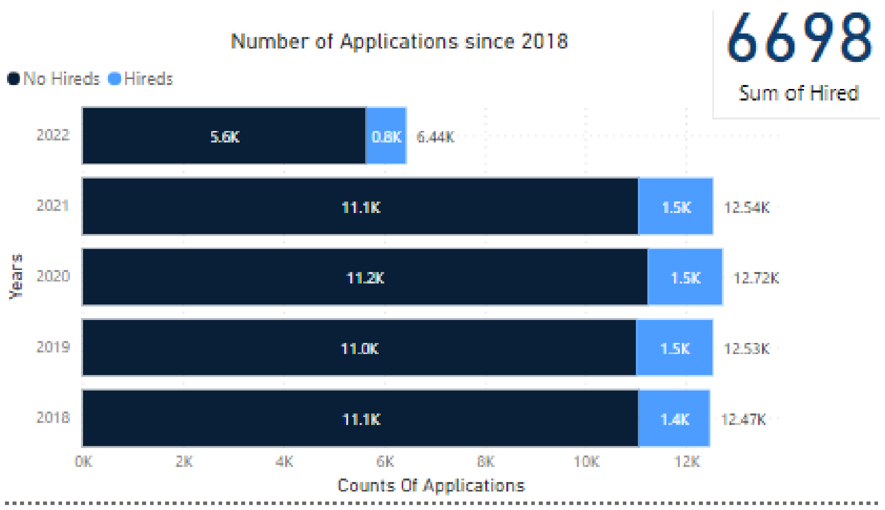
Data
- public Applicants_interview_ready
  - Application_Date
  - Code_Challenge_Score
  - Country
  - Email
  - First_Name
  - Grouped Technology
  - Hired
  - Last_Name
  - Seniority
  - Technical_Interview_Score
  - Technology
  - YOE

Table: public Applicants_interview_ready (50,000 rows) Column: Grouped Technology (10 distinct values)

right click on the Technology column and click on create new column(2). And we put the following query(1).

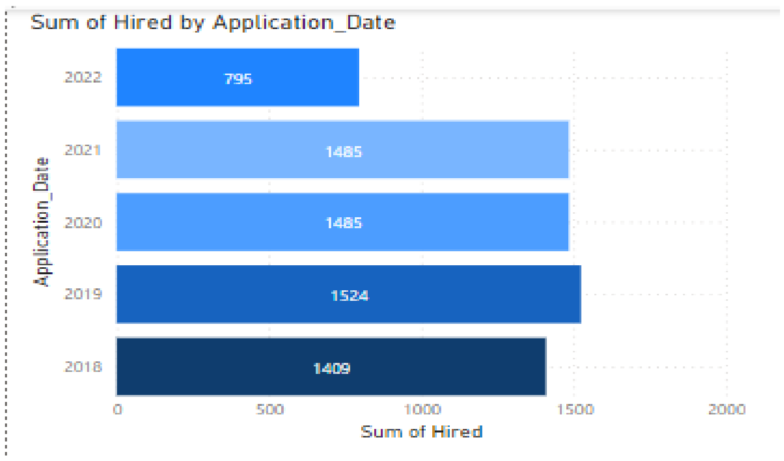
```
Grouped Technology =
SWITCH (
    'public Applicants_interview_ready'[Technology],  // Nombre de la columna que
contiene los valores a agrupar
    "Data Engineer", "Data Engineer",
    "Development - CMS Backend", "Software Development",
    "Development - CMS Frontend", "Software Development",
    "Development - Backend", "Software Development",
    "Development - Frontend", "Software Development",
    "Development - FullStack", "Software Development",
    "Game Development", "Software Development",
    "Business Analytics / Project Management", "Project Management and Analysis",
    "Business Intelligence", "Project Management and Analysis",
    "Client Success", "Sales and Customer Relations",
    "Sales", "Sales and Customer Relations",
    "Salesforce", "Sales and Customer Relations",
    "System Administration", "System Administration",
    "Security", "Security and Compliance",
    "Security Compliance", "Security and Compliance",
    "QA Manual", "QA and Testing",
    "QA Automation", "QA and Testing",
    "Social Media Community Management", "Communication and Design",
    "Adobe Experience Manager", "Communication and Design",
    "Design", "Communication and Design",
    "Technical Writing", "Communication and Design",
    "Mulesoft", "Integration and Platforms",
    "DevOps", "Integration and Platforms",
    "Database Administration", "Database Administration",
    'public Applicants_interview_ready'[Technology]  // Valor por defecto si no
coincide con ninguna categoría
)
```

VISUALIZATIONS

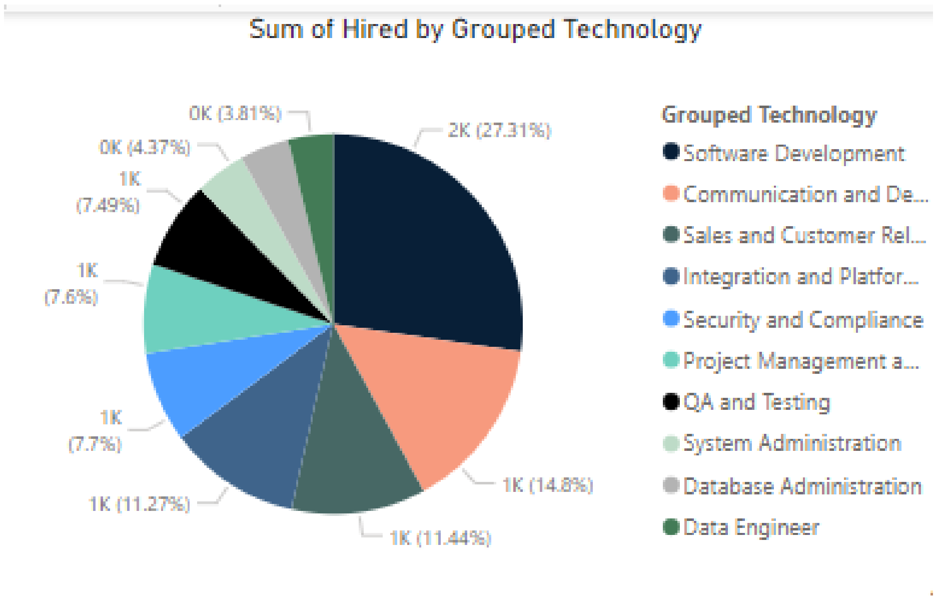


Here we can see the number of people who applied for the different positions, where in each of the years less than 20% of those who apply are hired each year. We can also see that in the last 4 years and 7 months about 6,698 people have been hired.

To analyze it in more detail, we have this chart:



For this horizontal bar chart, we observe a higher number of hires in 2019, with a total of 1524, compared to the other years which remain relatively stable. However, in the year 2022, only 795 hires are recorded, suggesting that some data is likely missing for this year, which could explain the decrease in hires.

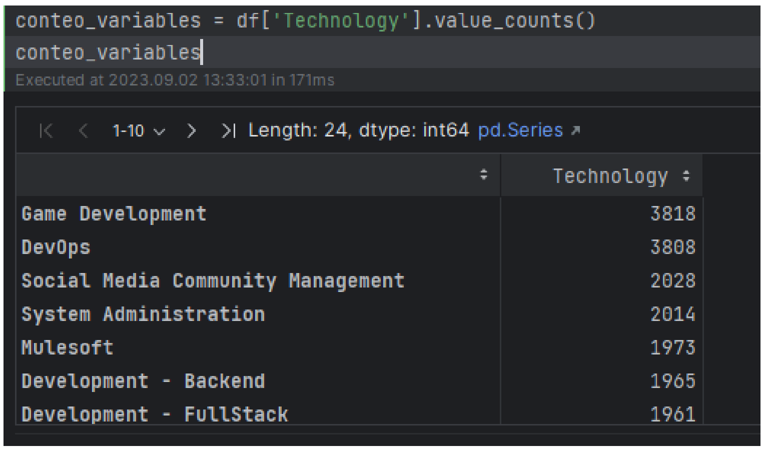


As we can see, the least hired are the Data Engineers with **255 hired**, representing **3.81%,** and the most hired are those grouped in **"Software Development"**.
que serían:

- Development - CMS Backend
- Development - CMS Frontend
- Development - Backend
- Development - Frontend
- Development - FullStack
- Game Development

with more than **2000 hires**, representing **27.31 %**. Although it should be clarified that **Software Development** is the largest grouping that was made with 6 grouped variables. **Communication and Design** represent 14.8% are only grouped with 2 variables and it could seem that the most representative variable is in .**Communication and Design** .

Pero esta tabla nos evita sesgos confirmando que la variable más representativa se encuentra en Software Development:

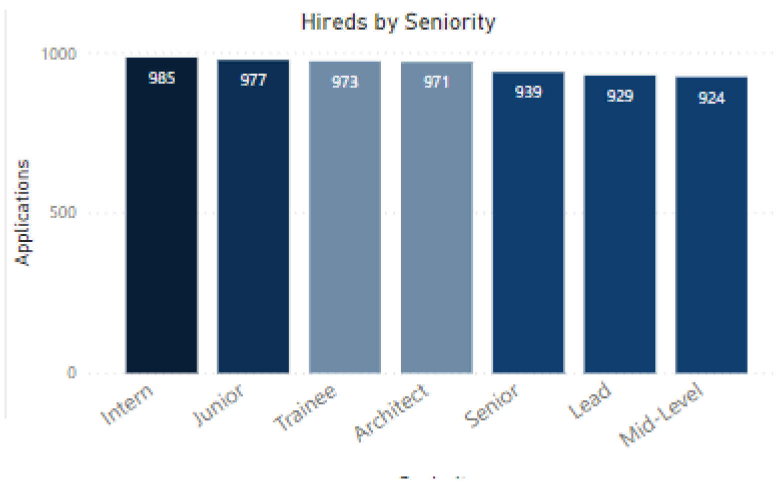


```
conteo_variables = df['Technology'].value_counts()
conteo_variables
```

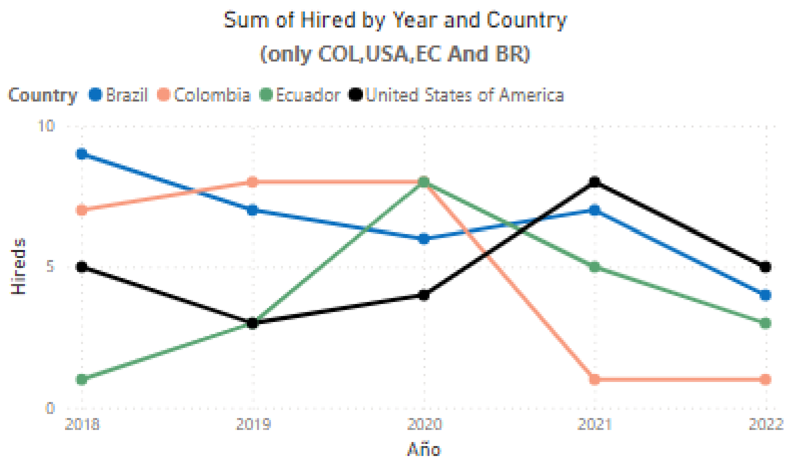
Executed at 2023.09.02 13:33:01 in 171ms

1-10 Length: 24, dtype: int64 pd.Series

	Technology
Game Development	3818
DevOps	3808
Social Media Community Management	2028
System Administration	2014
Mulesoft	1973
Development - Backend	1965
Development - FullStack	1961



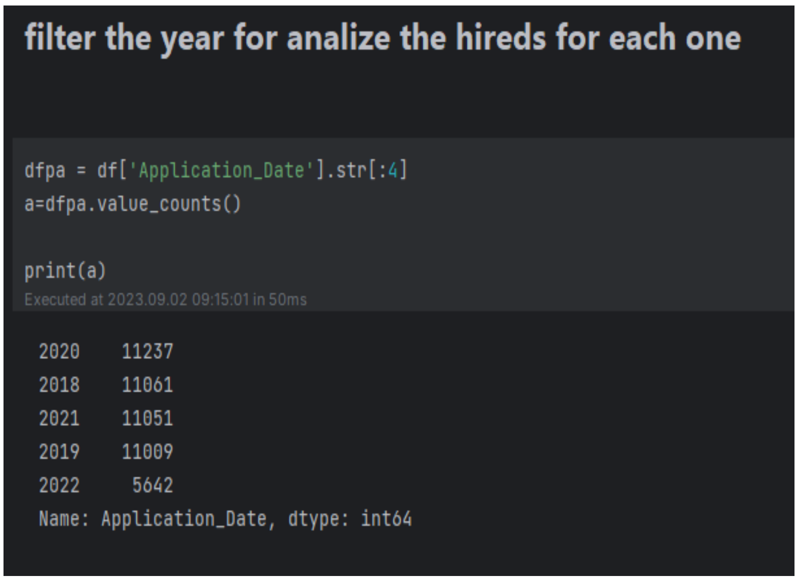
Within this graph, each hire is classified into 7 different levels according to their experience. It is important to note that the level with the highest number of hires corresponds to "Intern" and then “Junior” which may be consistent given that these are individuals who are doing internships or work experience programs and have a higher probability of being hired with the objective of reducing costs and hiring people with a lot of enthusiasm.   However, it is worth mentioning that no clear pattern can be identified that relates the level of experience or expertise ("seniority") to the number of hires. In other words, although "Intern" leads in terms of number of hires, there is no clear correlation between seniority level and number of hires.



In this graph of hires by country, we can see significant variability in hiring levels over the years. Notably, the United States (USA) shows a steady increase in the number of hires as we move forward in time.

However, it is important to note that Colombia, despite starting with a considerable number of hires in the early years, experiences a sharp decline in the year 2022, where only 1 hire is recorded. This drastic drop could suggest a decrease in job opportunities in Colombia.

with these tables we can better understand the chart:



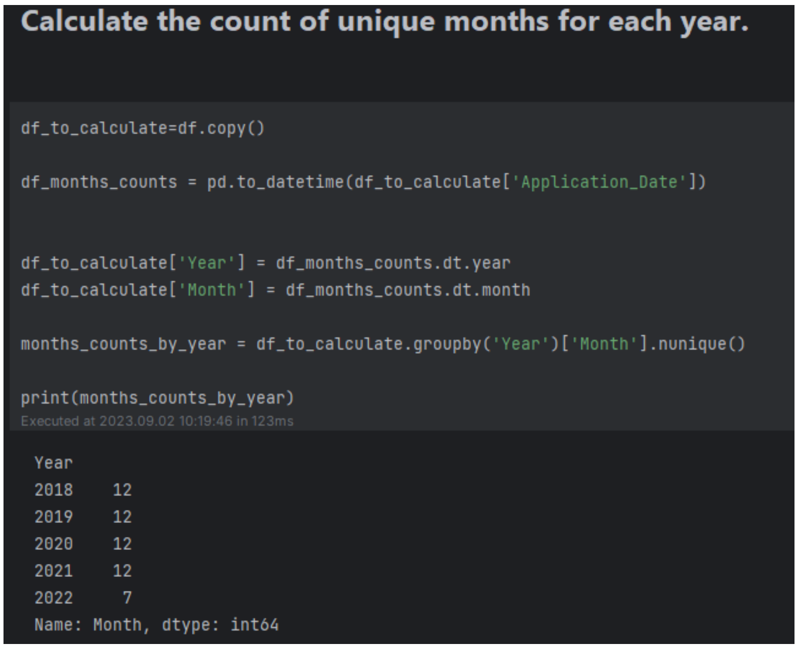
filter the year for analize the hireds for each one

```
dfpa = df['Application_Date'].str[:4]
a=dfpa.value_counts()

print(a)
```

Executed at 2023.09.02 09:15:01 in 50ms

```
2020    11237
2018    11061
2021    11051
2019    11009
2022     5642
Name: Application_Date, dtype: int64
```

\



Calculate the count of unique months for each year.

```
df_to_calculate=df.copy()

df_months_counts = pd.to_datetime(df_to_calculate['Application_Date'])


df_to_calculate['Year'] = df_months_counts.dt.year
df_to_calculate['Month'] = df_months_counts.dt.month

months_counts_by_year = df_to_calculate.groupby('Year')['Month'].nunique()

print(months_counts_by_year)
```

Executed at 2023.09.02 10:19:46 in 123ms

```
Year
2018    12
2019    12
2020    12
2021    12
2022     7
Name: Month, dtype: int64
```

It is essential to keep in mind that this analysis is based on data available up to July 2022, and the lack of data for the second half of the year could be affecting the figures. Therefore, the hiring charts by year and by country may not fully reflect the current labor situation in Colombia due to the lack of data for the full year.