Remedial Producto Académico 1

20/06/2025

Torres Romero Jorge **UTP0156779**

Extracción de Conocimientos de Bases de Datos
Noveno Cuatrimestre
División: Tecnologías de la Información
Carrera: Desarrollo y Gestión de Software

Mtro. José Francisco Espinosa Garita

Grupo C

Índice de Contenido

Índice de imágenes

Introducción

El objetivo de este proyecto es aplicar los conocimientos adquiridos en la unidad 1 de la asignatura de Extracción de Conocimientos de Bases de Datos. Para ello se trabajara con una base de datos perteneciente al INEGI del Censo 2020.

A lo largo del proyecto, pondremos a prueba nuestras habilidades en la gestión de bases de datos mediante diversas tareas como listar los campos y su tipo de dato, verificar que no haya datos faltantes, contar el número total de observaciones, y hacer cálculos con los datos recibidos etc. Todo esto se llevará a cabo utilizando Python, Pandas y la version mejorada de Pandas, llamada Polars y otras librerias más.

Justificación del Proyecto

Objetivo

La razón por la que escogí la base de datos del INEGI fue debido a que quería un reto y los datos del INEGI satisfacían ese reto por su gran numero de variables y porque contenían datos muy interesantes. Seleccione el estado de Veracruz debido a su relevancia social y su turismo lo que permite identificar patrones diferenciados de desarrollo, desigualdad y servicios públicos, lo que enriquece el análisis

Alcance

Decidí trabajar con las librerías Pandas y Polars, debido a que algunas instrucciones en Pandas son más fáciles de implementar, pero Polars es mucho más rápido y cuenta con funciones adicionales.
Los datos que escogí para calcular y graficar del CSV fueron los siguientes:

- **P5_HLI_HE:** Población de 5 años y más que habla alguna lengua indígena y habla español.
- **PCON_LIMI:** Población con limitación.
- **PSIND_LIM:** Población sin discapacidad, limitación, problema o condición mental.
- **P15YM_AN:** Población de 15 años y más analfabeta.
- **P18YM_PB:** Población de 18 años y más con educación pos-básica.

Limitaciones

El análisis está limitado a los datos disponibles en la base de datos publicos del INEGI para el estado de Veracruz y a las variables seleccionadas. La calidad y la precisión de los resultados dependen directamente de la veracidad y actualización de esos datos, por lo que no estarán actualizados hasta este año

Justificación de los Datos

Escogí estos datos porque me parecieron muy interesantes para mostrar el contraste en la población. Por un lado, podemos observar la cantidad de personas con alguna limitación y las personas sin ninguna limitación, y por otro, la población analfabeta y la que tiene educación pos-básica, lo cual permite un análisis comparativo que puede revelar información significativa sobre las condiciones sociales y educativas en Veracruz.

Creación del Entorno Virtual

Se instaló Anaconda y en la terminal de Windows se ejecutaron los siguientes comandos para iniciar el entorno virtual:

conda create -n ECBD python=3.13

```
  python_abi          pkgs/main/win-64::python_abi-3.13-0_cp313
  setuptools          pkgs/main/win-64::setuptools-78.1.1-py313haa95532_0
  sqlite              pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  tk                  pkgs/main/win-64::tk-8.6.14-h0416ee5_0
  tzdata              pkgs/main/noarch::tzdata-2025b-h04d1e81_0
  vc                  pkgs/main/win-64::vc-14.42-haa95532_5
  vs2015_runtime      pkgs/main/win-64::vs2015_runtime-14.42.34433-hbfb602d_5
  wheel               pkgs/main/win-64::wheel-0.45.1-py313haa95532_0
  xz                  pkgs/main/win-64::xz-5.6.4-h4754444_1
  zlib                pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1


Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ECBD
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

1. Comando para crear el entorno virtual.

conda activate ECBD

```
C:\Users\George TR>source activate ECBD
"source" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\George TR>conda activate ECBD

(ECBD) C:\Users\George TR>
```

1.1. Comando para activar el entorno virtual.

Después, se instalaron las librerías de pandas y polars para poder trabajar con el CSV.

```
C:\Users\George TR>conda activate ECBD

(ECBD) C:\Users\George TR>pip install pandas
Requirement already satisfied: pandas in c:\users\george tr\appdata\roaming\python\python313\site-packages (2.2.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\george tr\appdata\roaming\python\python313\site-packages (from
 pandas) (2.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\george tr\appdata\roaming\python\python313\site-packa
ges (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\george tr\appdata\roaming\python\python313\site-packages (from
pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\george tr\appdata\roaming\python\python313\site-packages (fro
m pandas) (2025.1)
Requirement already satisfied: six>=1.5 in c:\users\george tr\appdata\roaming\python\python313\site-packages (from pyth
on-dateutil>=2.8.2->pandas) (1.17.0)

(ECBD) C:\Users\George TR>pip install polars
Requirement already satisfied: polars in c:\users\george tr\appdata\roaming\python\python313\site-packages (1.30.0)
```

1.2. Instalación de las librerias pandas y polars.

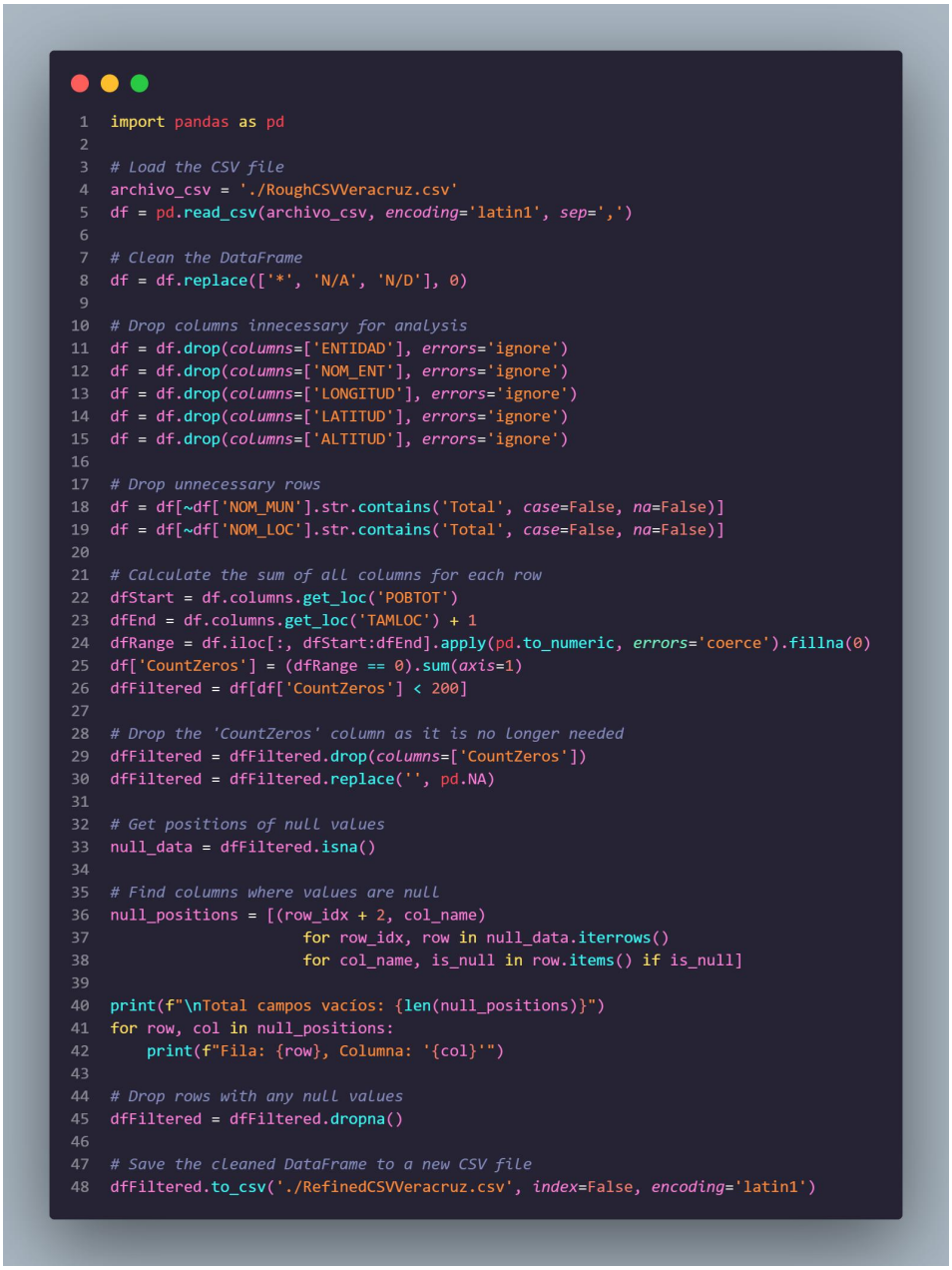
También se instalaron las librerias de matplotlib y numpy para poder graficar y hacer cálculos con los datos del CSV.

```
C:\WINDOWS\system32\cmd.
atplotlib) (2.2.2)
Requirement already satisfied: packaging>=20.0 in c:\users\george tr\.conda\envs\ecbd\lib\site-packages (from matplotli
b) (24.2)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-11.2.1-cp313-cp313-win_amd64.whl.metadata (9.1 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.3-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\george tr\appdata\roaming\python\python313\site-package
s (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\george tr\appdata\roaming\python\python313\site-packages (from pyth
on-dateutil>=2.7->matplotlib) (1.17.0)
Downloading matplotlib-3.10.3-cp313-cp313-win_amd64.whl (8.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.1/8.1 MB 7.3 MB/s eta 0:00:00
Downloading contourpy-1.3.2-cp313-cp313-win_amd64.whl (223 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.58.1-cp313-cp313-win_amd64.whl (2.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 8.8 MB/s eta 0:00:00
Downloading kiwisolver-1.4.8-cp313-cp313-win_amd64.whl (71 kB)
Downloading pillow-11.2.1-cp313-cp313-win_amd64.whl (2.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.7/2.7 MB 7.2 MB/s eta 0:00:00
Downloading pyparsing-3.2.3-py3-none-any.whl (111 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, contourpy, matplotlib
Successfully installed contourpy-1.3.2 cycler-0.12.1 fonttools-4.58.1 kiwisolver-1.4.8 matplotlib-3.10.3 pillow-11.2.1
pyparsing-3.2.3

(ECBD) C:\Users\George TR>pip install numpy
Requirement already satisfied: numpy in c:\users\george tr\appdata\roaming\python\python313\site-packages (2.2.2)

(ECBD) C:\Users\George TR>
```

1.3. Instalación de las librerias matplotlib y numpy.

Limpieza del CSV

Para limpiar el CSV opté por usar Python y la librería pandas. Lo que hice fue cargar el CSV, convertir los “*”, “N/A”, “N/D” por ‘0', eliminar las variables innecesarias y las observaciones que tengan un “Total…” en las variables “NOM_MUN” y “NOM_LOC”, después cree una variable temporal en donde conté los 0 que tenían las observaciones y solo conservé los que fueran menores a 200, luego conté y encontré los campos nulos, al revisarlos manualmente me di cuenta que no podía rellenar esos campos con ningún valor sin que afecte los cálculos que se harán después, así que opte por eliminar esas filas, ya por último, guardé el CSV ya limpio.



```python
import pandas as pd

# Load the CSV file
archivo_csv = './RoughCSVVeracruz.csv'
df = pd.read_csv(archivo_csv, encoding='latin1', sep=',')

# Clean the DataFrame
df = df.replace(['*', 'N/A', 'N/D'], 0)

# Drop columns innecessary for analysis
df = df.drop(columns=['ENTIDAD'], errors='ignore')
df = df.drop(columns=['NOM_ENT'], errors='ignore')
df = df.drop(columns=['LONGITUD'], errors='ignore')
df = df.drop(columns=['LATITUD'], errors='ignore')
df = df.drop(columns=['ALTITUD'], errors='ignore')

# Drop unnecessary rows
df = df[~df['NOM_MUN'].str.contains('Total', case=False, na=False)]
df = df[~df['NOM_LOC'].str.contains('Total', case=False, na=False)]

# Calculate the sum of all columns for each row
dfStart = df.columns.get_loc('POBTOT')
dfEnd = df.columns.get_loc('TAMLOC') + 1
dfRange = df.iloc[:, dfStart:dfEnd].apply(pd.to_numeric, errors='coerce').fillna(0)
df['CountZeros'] = (dfRange == 0).sum(axis=1)
dfFiltered = df[df['CountZeros'] < 200]

# Drop the 'CountZeros' column as it is no longer needed
dfFiltered = dfFiltered.drop(columns=['CountZeros'])
dfFiltered = dfFiltered.replace('', pd.NA)

# Get positions of null values
null_data = dfFiltered.isna()

# Find columns where values are null
null_positions = [(row_idx + 2, col_name)
                  for row_idx, row in null_data.iterrows()
                  for col_name, is_null in row.items() if is_null]

print(f"\nTotal campos vacios: {len(null_positions)}")
for row, col in null_positions:
    print(f"Fila: {row}, Columna: '{col}'")

# Drop rows with any null values
dfFiltered = dfFiltered.dropna()

# Save the cleaned DataFrame to a new CSV file
dfFiltered.to_csv('./RefinedCSVVeracruz.csv', index=False, encoding='latin1')
```

2. Código de Python para limpiar el CSV.

Estadística Descriptiva

Carga de Datos con CSV

Cargue el CSV a Jupyter Notebook con Pandas y Polars para poder trabajar con los dos de manera simultanea. Los cargue con ‘latin1’ debido a que utf-8 me daba problemas y quite los simbolos extraños de la primera vaiable.

Load the Chiapas CSV

Importing the necessary libraries

[3]:
```
import pandas as pd
import polars as pl
import matplotlib.pyplot as plt
```

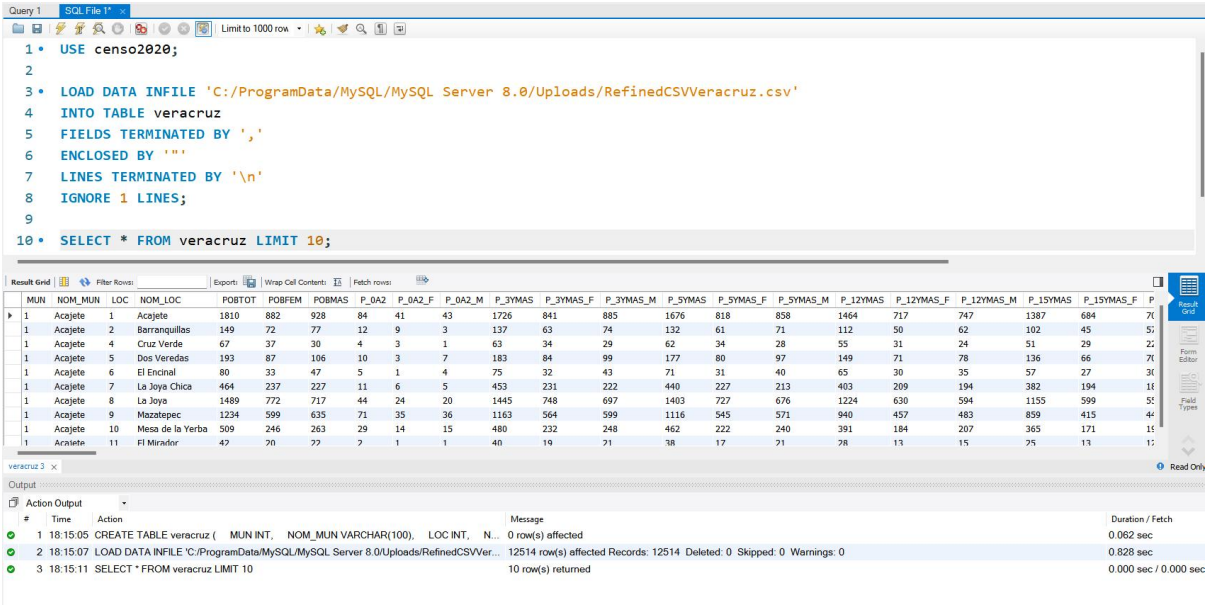
Load the CSV

[4]:
```
dfPd = pd.read_csv('./RefinedCSVChiapas.csv', sep=',', encoding='latin1')
dfPd.columns = dfPd.columns.str.replace('ï»¿', '').str.strip()

dfPl = pl.read_csv('./RefinedCSVChiapas.csv', separator=',', has_header=True, schema_overrides={'AGEB': pl.Utf8}, encoding='latin1')
dfPl = dfPl.rename({col: col.strip().replace("ï»¿", "") for col in dfPl.columns})
```

3. Carga de Datos con CSV

Carga de Datos con Sistema Gestor de Bases de Datos

Se cargo el CSV a MySQL Workbench. En esta parte se hizo una tabla con todas las variables del CSV, además de que tuve algunos problemas utilizando LOAD DATA, esto debido a que MySQL tiene una carpeta especifica en donde se puede utilizar LOAD DATA sin ningun problema.



4. Carga de Datos con MySQL Workbench

Conexión desde Jupyter Notebook

Para conectar Jupyter Notebook con MySQL utilice la libreria SQLAlchemy y Pandas, una vez conectado a SQL hice un SELECT a la tabla veracruz con un limite de 10, esto solo para probar. Afortunadamente esta vez si logre conectarme sin nungun problema.

Connect with MySQL

```
from sqlalchemy import create_engine

ConnectionDB = create_engine("mysql+pymysql://root:GTR07@localhost:3306/censo2020")

dfPdSQL = pd.read_sql("SELECT * FROM veracruz LIMIT 10", ConnectionDB)

dfPdSQL
```

	MUN	NOM_MUN	LOC	NOM_LOC	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	P_0A2_M	...	VPH_CEL	VPH_INTER	VPH_STVP	VPH
0	1	Acajete	1	Acajete	1810	882	928	84	41	43	...	431	212	211	
1	1	Acajete	2	Barranquillas	149	72	77	12	9	3	...	10	0	0	
2	1	Acajete	4	Cruz Verde	67	37	30	4	3	1	...	15	4	4	
3	1	Acajete	5	Dos Veredas	193	87	106	10	3	7	...	31	0	0	
4	1	Acajete	6	El Encinal	80	33	47	5	1	4	...	5	0	0	
5	1	Acajete	7	La Joya Chica	464	237	227	11	6	5	...	97	31	33	
6	1	Acajete	8	La Joya	1489	772	717	44	24	20	...	367	192	174	
7	1	Acajete	9	Mazatepec	1234	599	635	71	35	36	...	224	7	33	
8	1	Acajete	10	Mesa de la Yerba	509	246	263	29	14	15	...	88	1	10	
9	1	Acajete	11	El Mirador	42	20	22	2	1	1	...	5	0	0	

10 rows × 281 columns

5. Conexión y visualización de la carga de datos de MySQL en Jupyter Notebook

Estadística Descriptiva

Listar los campos y los tipos de datos del dataframe.

Necesitaba conocer las variables y los tipos de datos con los que estaba trabajando, por lo que con ayuda de Polars, logré visualizar todos los datos completos con un estilo de “tabla” y con un for para recorrer las variables.

knowing and repairing the CSV

Show the fields and his data type

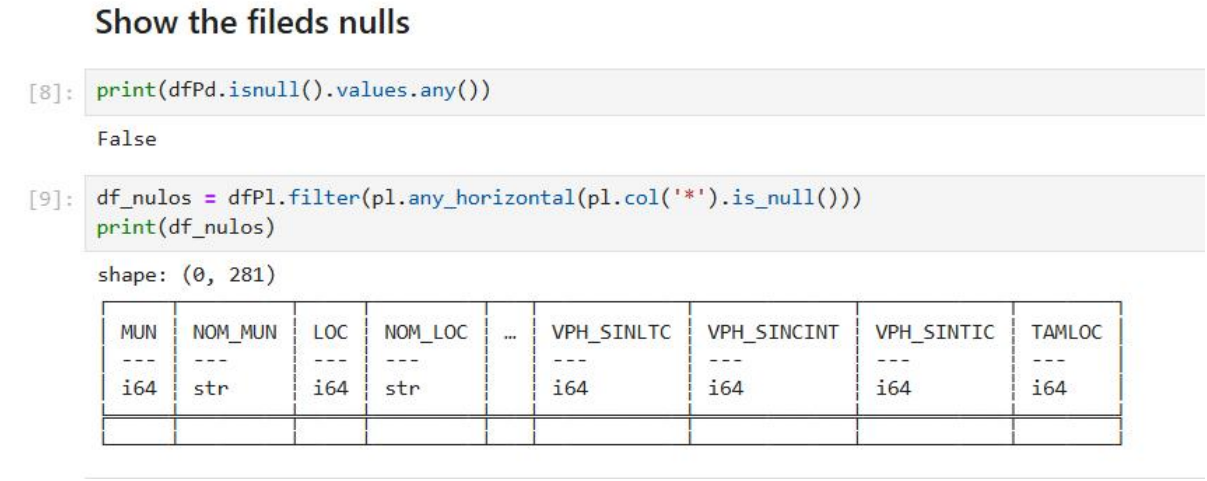
```
[3]: print("| Column Name | Data Type |")
     print("|-------------|-----------|")


     for col, dtype in zip(dfPl.columns, dfPl.dtypes):
         print(f"| {col:<11} | {str(dtype):<9} |")
```

```
| Column Name | Data Type |
|-------------|-----------|
| MUN         | Int64     |
| NOM_MUN     | String    |
| LOC         | Int64     |
| NOM_LOC     | String    |
| POBTOT      | Int64     |
| POBFEM      | Int64     |
| POBMAS      | Int64     |
| P_0A2       | Int64     |
| P_0A2_F     | Int64     |
| P_0A2_M     | Int64     |
| P_3YMAS     | Int64     |
| P_3YMAS_F   | Int64     |
| P_3YMAS_M   | Int64     |
| P_5YMAS     | Int64     |
| P_5YMAS_F   | Int64     |
| P_5YMAS_M   | Int64     |
```

6. Listado de las variables y sus tipos de datos

Verificar si hay datos faltantes.

Una vez que supe con que variables estaba trabajando, necesitaba conocer si en estas variables había quedado algún dato nulo, y el resultado fue falso, por lo que no había ningún dato vacío.



Show the fileds nulls

[8]:
```
print(dfPd.isnull().values.any())
```

```
False
```

[9]:
```
df_nulos = dfPl.filter(pl.any_horizontal(pl.col('*').is_null()))
print(df_nulos)
```

```
shape: (0, 281)
```

MUN	NOM_MUN	LOC	NOM_LOC	…	VPH_SINLTC	VPH_SINCINT	VPH_SINTIC	TAMLOC
i64	str	i64	str		i64	i64	i64	i64

6.1.1. Verificando los datos vacíos

Contar el número total de observaciones.

Una vez que verifique que no haya datos nulos en mi CSV me dispuse a contar el numero total de observaciones que tenia mi CSV después de limpiarla por completo, con la función shape de Pandas.

Counting

Count the total of Observations

```
[11]: dfPd.shape[0]

[11]: 12514
```

6.2. Total de observaciones

Obtener el summary del dataframe.

Después de verificar cuantas observaciones eran, se hizo un resumen un con información importante de cada variable del CSV.

Show the summary of the CSV

```
[12]: Summary = dfPd.describe(include='all')
      print(Summary)
```

	MUN	NOM_MUN	LOC	\
count	12514.000000	12514	12514.000000	
unique	NaN	212	NaN	
top	NaN	Tierra Blanca	NaN	
freq	NaN	329	NaN	
mean	110.081509	NaN	341.841218	
std	59.508161	NaN	1453.420052	
min	1.000000	NaN	1.000000	
25%	59.000000	NaN	23.000000	
50%	116.000000	NaN	64.000000	
75%	160.000000	NaN	165.750000	
max	212.000000	NaN	9999.000000	

	NOM_LOC	POBTOT	POBFEM	\
count	12514	12514.000000	12514.000000	
unique	8298	NaN	NaN	
top	Localidades de una vivienda	NaN	NaN	
freq	141	NaN	NaN	
mean	NaN	643.866629	334.681397	
std	NaN	6761.727575	3604.088581	
min	NaN	3.000000	1.000000	
25%	NaN	30.000000	15.000000	
50%	NaN	121.000000	60.000000	
75%	NaN	341.750000	172.000000	
max	NaN	443063.000000	237967.000000	

6.3. Summary del dataframe

¿Identificar cuales son los nombres de las variables?

Los nombres de las variables ya las había mostrado antes, cuando mostré los tipos de datos, por lo que en esa consulta puedo ver el nombre de todas las variables.

knowing and repairing the CSV

Show the fields and his data type

```
[3]: print("| Column Name | Data Type |")
     print("|-------------|-----------|")


     for col, dtype in zip(dfPl.columns, dfPl.dtypes):
         print(f"| {col:<11} | {str(dtype):<9} |")
```

```
| Column Name | Data Type |
|-------------|-----------|
| MUN         | Int64     |
| NOM_MUN     | String    |
| LOC         | Int64     |
| NOM_LOC     | String    |
| POBTOT      | Int64     |
| POBFEM      | Int64     |
| POBMAS      | Int64     |
| P_0A2       | Int64     |
| P_0A2_F     | Int64     |
| P_0A2_M     | Int64     |
| P_3YMAS     | Int64     |
| P_3YMAS_F   | Int64     |
| P_3YMAS_M   | Int64     |
| P_5YMAS     | Int64     |
| P_5YMAS_F   | Int64     |
| P_5YMAS_M   | Int64     |
```

6.4. Listado de las variables y sus tipos de datos

¿Cuántos municipios hay?

Contar el total de municipios fue muy fácil, debido a que la variable MUN esta ordenada de 1...212, por lo tanto solo tenia que mostrar el ultimo número para saber el total de municipios que hay.

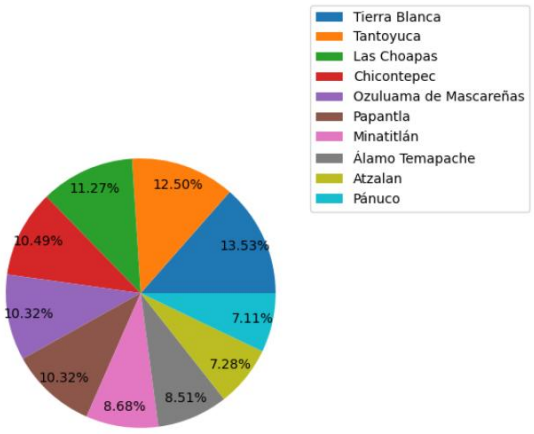
Show the total of municipalities

```
3]: countMun = dfPd[ 'MUN' ].iloc[ -1 ]
    print ( countMun )

212
```

6.5. Total de Municipios

Para hacer el gráfico de los municipios conté únicamente los primeros 10 que más aparecen en el CSV, y extraje su tamaño y nombre del municipio para mostrarlo en el gráfico.

```
[30]: municipalitiesG = dfPl["NOM_MUN"].value_counts().sort("count", descending=True).head(10)

      sizes = municipalitiesG["count"].to_list()
      labels = municipalitiesG["NOM_MUN"].to_list()

      plt.figure(figsize=(4, 4))
      plt.pie(sizes, autopct='%.2f%%', pctdistance=0.85)
      plt.axis('equal')
      plt.legend(labels=labels, bbox_to_anchor=(1.05, 1.5), loc='upper left')
      plt.show()
```



6.5.1. Código y gráfica de los municipios

¿Cuántas localidades hay?

Contar el total de localidades fue más complejo de lo que pensaba, debido a que había localidades que se repetían más de una vez en diferentes municipios y la variable LOC repetía los números por cada municipio. Entonces ocupe las variables MUN y LOC para verificar que solo se cuenten una vez las localidades por cada municipio.

Repeated localities

```
35]: repLoc = dfPd.groupby('NOM_LOC')['MUN'].nunique()
     repLoc = repLoc[repLoc > 1]
     print(repLoc)
```

```
NOM_LOC
Abrevadero          2
Acatitla            2
Actopan             3
Acuapa              2
Adalberto Tejeda    6
                   ..
Zaragoza            2
Zongolica           2
Álamo               3
Álvaro Obregón      3
Úrsulo Galván      14
Name: MUN, Length: 1204, dtype: int64
```

6.6. Localidades repetidas

Show the total of localities

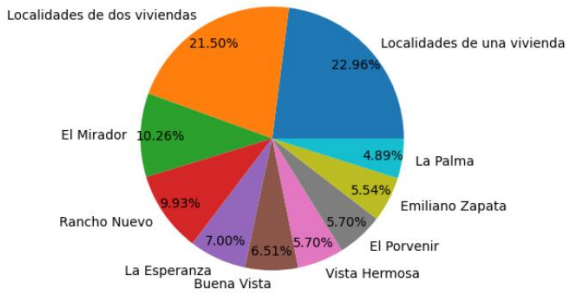
```
34]: countLoc = dfPd.groupby( 'MUN' )[ 'LOC' ].nunique()
     totalLoc = countLoc.sum()
     print(totalLoc)
```

```
12514
```

6.6.1. Total de Localidades

Después hice el mismo proceso de la grafica de municipios pero para la de localidades y lo limite a 10.

```
[33]: localitiesG = dfPl["NOM_LOC"].value_counts().sort("count", descending=True).head(10)

      sizes = localitiesG["count"].to_list()
      labels = localitiesG["NOM_LOC"].to_list()

      plt.figure(figsize=(4, 4))
      plt.pie(sizes, labels=labels, autopct='%.2f%%', pctdistance=0.85)
      plt.axis('equal')
      plt.show()
```



6.6.2. Gráfica de las localidades

¿Cuántas observaciones por municipio hay?

Una vez que supe cuantos municipios había me dediqué a contar las observaciones que tenía cada municipio y las mostré a manera de tabla con un for.

Show the observations por municipalities

```
[40]: dfPlMun = (dfPl.group_by(['MUN', 'NOM_MUN']).len().sort('MUN'))

print("| MUN  | NOM_MUN                           | OBSERVATIONS |")
print("|------|-----------------------------------|--------------|")

for row in dfPlMun.iter_rows(named=True):
    print(f"| {row['MUN']:>4} | {row['NOM_MUN']:<34} | {row['len']:>12} |")
```

```
| MUN  | NOM_MUN                           | OBSERVATIONS |
|------|-----------------------------------|--------------|
|    1 | Acajete                           |           35 |
|    2 | Acatlán                           |            4 |
|    3 | Acayucan                          |           82 |
|    4 | Actopan                           |          119 |
|    5 | Acula                             |           23 |
|    6 | Acultzingo                        |           37 |
|    7 | Camarón de Tejeda                 |           28 |
|    8 | Alpatláhuac                       |           37 |
|    9 | Alto Lucero de Gutiérrez Barrios  |          102 |
|   10 | Altotonga                         |           98 |
|   11 | Alvarado                          |          131 |
|   12 | Amatitlán                         |           26 |
|   13 | Naranjos Amatlán                  |           37 |
|   14 | Amatlán de los Reyes              |           65 |
|   15 | Angel R. Cabada                   |          105 |
|   16 | La Antigua                        |           22 |
```

6.7. Total de observaciones por municipios

¿Cuántas observaciones por localidades hay?

E igualmente conté las observaciones para las localidades de la misma forma que hice para los municipios.

Show the observations por localities

```
[45]: dfPlLoc = (dfPl.group_by(['MUN', 'NOM_MUN', 'NOM_LOC']).len().sort(['MUN', 'NOM_LOC']))

print("| ID   | NOM_LOC                                  | OBSERVATIONS |")
print("|------|------------------------------------------|--------------|")

ID = 1
for row in dfPlLoc.iter_rows(named=True):
    print(f"| {ID:>4} | {row['NOM_LOC']:<41} | {row['len']:>12} |")
    ID += 1
```

```
| ID   | NOM_LOC                                  | OBSERVATIONS |
|------|------------------------------------------|--------------|
|    1 | Acajete                                  |            2 |
|    2 | Acocota                                  |            1 |
|    3 | Barranquillas                            |            1 |
|    4 | Cinco de Mayo (Los Chávez)               |            1 |
|    5 | Coletxa                                  |            1 |
|    6 | Cruz Verde                               |            1 |
|    7 | Cuesta del Vaquero                       |            1 |
|    8 | Dos Veredas                              |            1 |
|    9 | El Capulín                               |            1 |
|   10 | El Encinal                               |            1 |
|   11 | El Encinal Dos                           |            1 |
|   12 | El Mirador                               |            1 |
|   13 | El Mirador Dos (Parte Baja)              |            1 |
|   14 | El Muñeco                                |            1 |
|   15 | El Quemado                               |            1 |
|   16 | El Rincón de Sedeño                      |            1 |
```

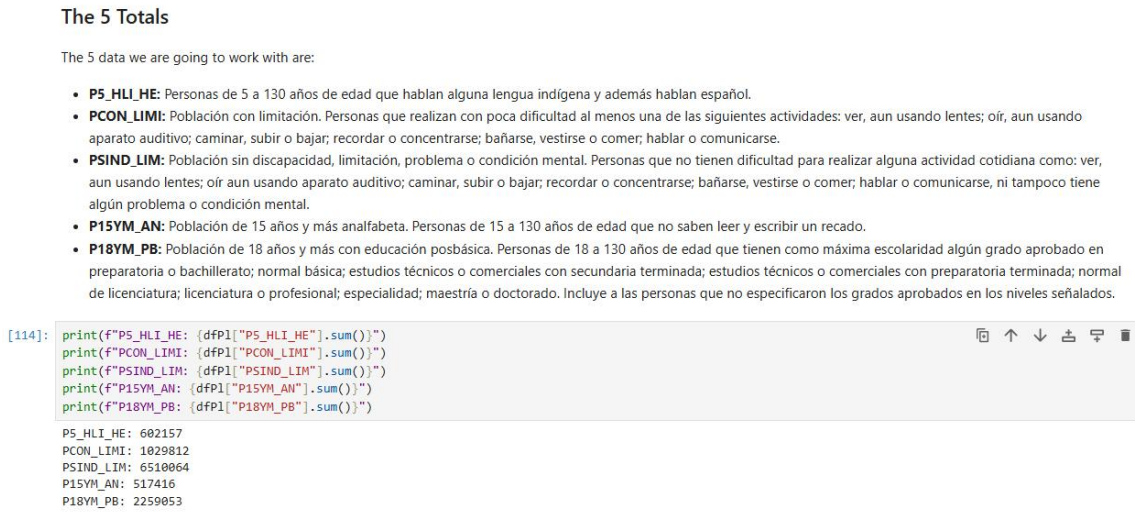
6.8. Total de observaciones por localidades

5 totales de las variables que desee.

Una vez que supe más sobre el CSV, me enfoqué a 5 datos clave que quise investigar más a profundidad.

- **P5_HLI_HE:** Población de 5 años y más que habla alguna lengua indígena y habla español.
- **PCON_LIMI:** Población con limitación.
- **PSIND_LIM:** Población sin discapacidad, limitación, problema o condición mental.
- **P15YM_AN:** Población de 15 años y más analfabeta.
- **P18YM_PB:** Población de 18 años y más con educación pos-básica.

El significado de estos datos lo saque del diccionario de datos que venia con el CSV del INEGI. Además que estos datos, me servirán después para poder hacer algunos cálculos y graficar los resultados obtenidos.



The 5 Totals

The 5 data we are going to work with are:

- **P5_HLI_HE:** Personas de 5 a 130 años de edad que hablan alguna lengua indígena y además hablan español.
- **PCON_LIMI:** Población con limitación. Personas que realizan con poca dificultad al menos una de las siguientes actividades: ver, aun usando lentes; oír, aun usando aparato auditivo; caminar, subir o bajar; recordar o concentrarse; bañarse, vestirse o comer; hablar o comunicarse.
- **PSIND_LIM:** Población sin discapacidad, limitación, problema o condición mental. Personas que no tienen dificultad para realizar alguna actividad cotidiana como: ver, aun usando lentes; oír aun usando aparato auditivo; caminar, subir o bajar; recordar o concentrarse; bañarse, vestirse o comer; hablar o comunicarse, ni tampoco tiene algún problema o condición mental.
- **P15YM_AN:** Población de 15 años y más analfabeta. Personas de 15 a 130 años de edad que no saben leer y escribir un recado.
- **P18YM_PB:** Población de 18 años y más con educación posbásica. Personas de 18 a 130 años de edad que tienen como máxima escolaridad algún grado aprobado en preparatoria o bachillerato; normal básica; estudios técnicos o comerciales con secundaria terminada; estudios técnicos o comerciales con preparatoria terminada; normal de licenciatura; licenciatura o profesional; especialidad; maestría o doctorado. Incluye a las personas que no especificaron los grados aprobados en los niveles señalados.

```
[114]: print(f"P5_HLI_HE: {dfPl["P5_HLI_HE"].sum()}")
       print(f"PCON_LIMI: {dfPl["PCON_LIMI"].sum()}")
       print(f"PSIND_LIM: {dfPl["PSIND_LIM"].sum()}")
       print(f"P15YM_AN: {dfPl["P15YM_AN"].sum()}")
       print(f"P18YM_PB: {dfPl["P18YM_PB"].sum()}")
```

```
P5_HLI_HE: 602157
PCON_LIMI: 1029812
PSIND_LIM: 6510064
P15YM_AN: 517416
P18YM_PB: 2259053
```

7. Los totales de las variables elegidas

Calcular el mínimo, el máximo y el rango

Una vez elegidos los datos, iba a calcular algunas cosas con ellos. Lo primero que hice fue calcular el mínimo, el máximo y el rango de la población de 5 años y más que habla alguna lengua indígena y habla español, y de la población con algún tipo de limitación, y representarlos por medio de un diagrama de caja.

Calculate the minimum, maximum and range

P5_HLI_HE

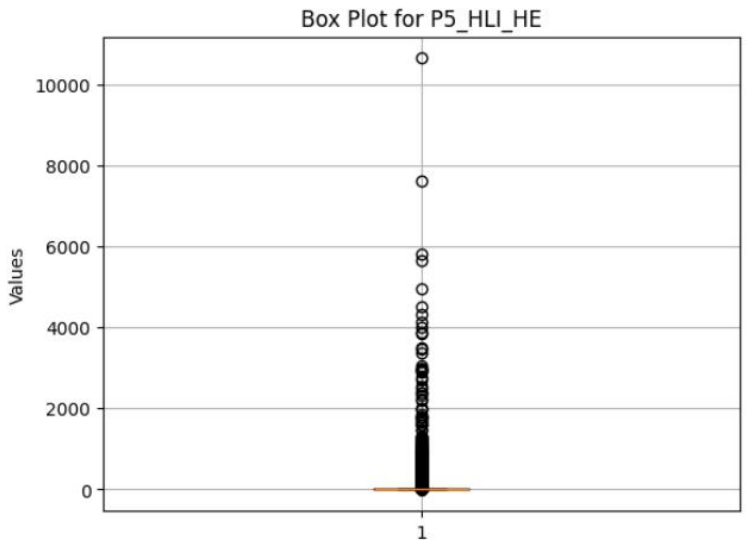
```
[115]: colPLI1 = "P5_HLI_HE"

min_val1 = dfPl[colPLI1].min()
max_val1 = dfPl[colPLI1].max()
range1 = max_val1 - min_val1

print(f"Column: {colPLI1}")
print(f"Minumum: {min_val1}")
print(f"Maximum: {max_val1}")
print(f"Range: {range1}")
```

```
Column: P5_HLI_HE
Minumum: 0
Maximum: 10637
Range: 10637
```

7.1. Código para calcular el mínimo, el máximo y el rango de la variable P5_HLI_HE

Graphic

```
]: dataForBoxPlot1 = dfPl[colPLI1].to_list()
plt.boxplot(dataForBoxPlot1)
plt.title(f"Box Plot for {colPLI1}")
plt.ylabel("Values")
plt.grid(True)
plt.show()
```



7.1.1. Código y gráfica del mínimo, el máximo y el rango de la variable P5_HLI_HE

PCON_LIMI

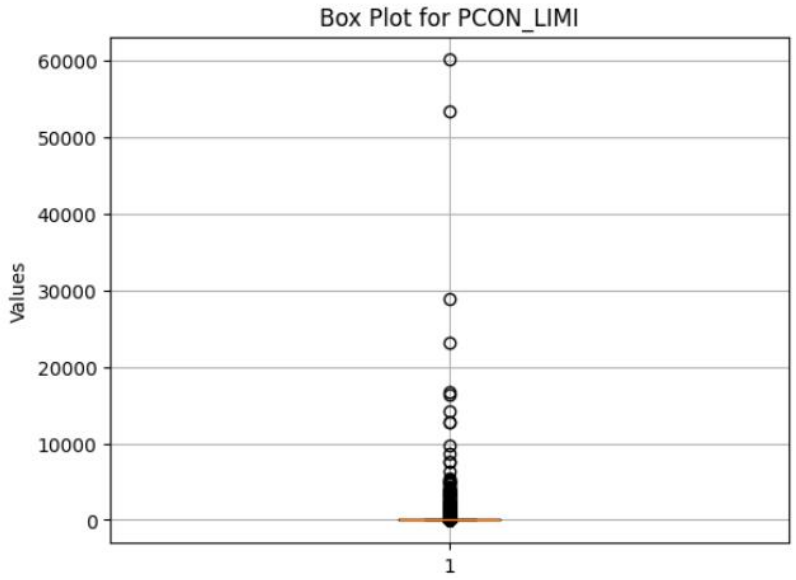
```
colPCL1 = "PCON_LIMI"

min_val2 = dfPl[colPCL1].min()
max_val2 = dfPl[colPCL1].max()
range2 = max_val2 - min_val2

print(f"Column: {colPCL1}")
print(f"Minumum: {min_val2}")
print(f"Maximum: {max_val2}")
print(f"Range: {range2}")
```

```
Column: PCON_LIMI
Minumum: 0
Maximum: 60043
Range: 60043
```

7.1.2. Código para calcular el mínimo, el máximo y el rango de la variable PCON_LIMI

Graphic

```
dataForBoxPlot2 = dfPl[colPCL1].to_list()
plt.boxplot(dataForBoxPlot2)
plt.title(f"Box Plot for {colPCL1}")
plt.ylabel("Values")
plt.grid(True)
plt.show()
```



7.1.3. Código y gráfica del mínimo, el máximo y el rango de la variable PCON_LIMI

Calcular la media, la mediana y la moda

Para calcular la media, la mediana y la moda, decidí ocupar las variables de la población con alguna limitación y las personas sin ninguna limitación, y representarlas con una gráfica de barras utilizando algunas funciones de Polars.

PCON_LIMI

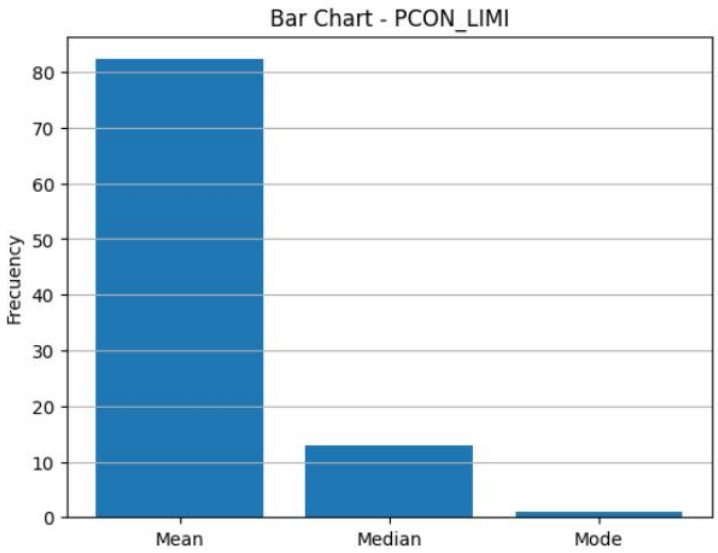
```
[92]: colPCL2 = "PCON_LIMI"

mean1 = dfPl[colPCL2].mean()
median1 = dfPl[colPCL2].median()
modeDf1 = (dfPl.group_by(colPCL2).len().sort("len", descending=True).limit(1))

mode1 = modeDf1[0, colPCL2]

print(f"Column: {colPCL2}")
print(f"Mean: {mean1}")
print(f"Median: {median1}")
print(f"Mode: {mode1}")
```

```
Column: PCON_LIMI
Mean: 82.29279207287837
Median: 13.0
Mode: 1
```

7.2. Código para calcular la media, la mediana y la moda de la variable PCON_LIMI

Graphic

```
[101]: labelsForBarChart1 = ["Mean", "Median", "Mode"]
valuesForBarChart1 = [mean1, median1, mode1]

plt.bar(labelsForBarChart1, valuesForBarChart1)
plt.ylabel("Frecuency")
plt.title(f"Bar Chart - {colPCL2}")
plt.grid(axis='y')
plt.show()
```



7.2.1. Código y gráfica de la media, la mediana y la moda de la variable PCON_LIMI

PSIND_LIM

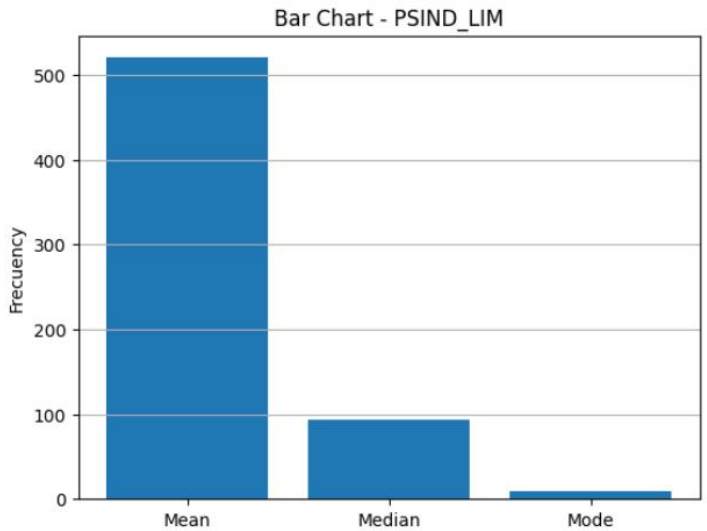
```
colPSL1 = "PSIND_LIM"

mean2 = dfPl[colPSL1].mean()
median2 = dfPl[colPSL1].median()
modeDf2 = (dfPl.group_by(colPSL1).len().sort("len", descending=True).limit(1))

mode2 = modeDf2[0, colPSL1]

print(f"Column: {colPSL1}")
print(f"Mean: {mean2}")
print(f"Median: {median2}")
print(f"Mode: {mode2}")
```

```
Column: PSIND_LIM
Mean: 520.2224708326675
Median: 94.0
Mode: 9
```

7.2.2. Código para calcular la media, la mediana y la moda de la variable PSIND_LIM

Graphic

```
labelsForBarChart2 = ["Mean", "Median", "Mode"]
valuesForBarChart2 = [mean2, median2, mode2]

plt.bar(labelsForBarChart2, valuesForBarChart2)
plt.ylabel("Frecuency")
plt.title(f"Bar Chart - {colPSL1}")
plt.grid(axis='y')
plt.show()
```



7.2.3. Código y gráfica de la media, la mediana y la moda de la variable PSIND_LIM

Calcular la desviación estándar y la varianza

Para calcular la desviación estándar y la varianza ocupe la variable de la población de 15 años y más analfabeta y la variable de la población de 18 años y más con educación pos-básica, y las represente con una gráfica de barras.

Calculate standard deviation and variance

P15YM_AN

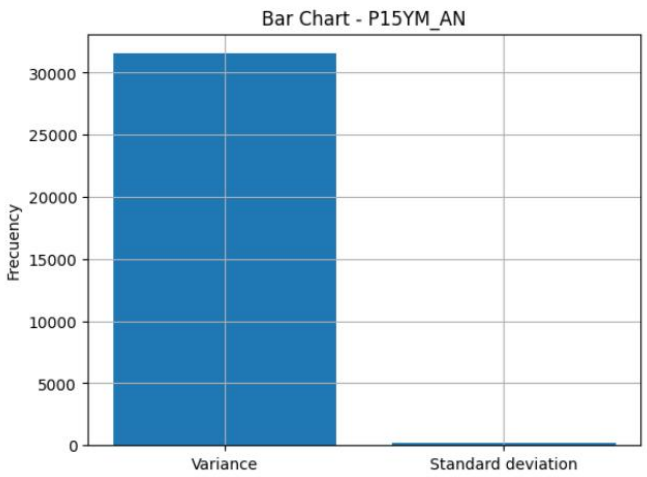
```
colP15AN1 = "P15YM_AN"

variance1 = dfPl[colP15AN1].var()
stdDeviaton1 = dfPl[colP15AN1].std()

print(f"Column: {colP15AN1}")
print(f"Variance: {variance1}")
print(f"Standard deviation: {stdDeviaton1}")
```

```
Column: P15YM_AN
Variance: 31528.73918726251
Standard deviation: 177.5633385225185
```

7.3. Código para calcular la desviación estándar y la varianza de la variable P15YM_AN

Graphic

```
labelsForBarChart3 = ["Variance", "Standard deviation"]
valuesForBarChart3 = [variance1, stdDeviaton1]

plt.bar(labelsForBarChart3, valuesForBarChart3)
plt.ylabel("Frecuency")
plt.title(f"Bar Chart - {colP15AN1}")
plt.grid(True)
plt.show()
```



7.3.1. Código y gráfica de la desviación estándar y la varianza de la variable P15YM_AN

P18YM_PB

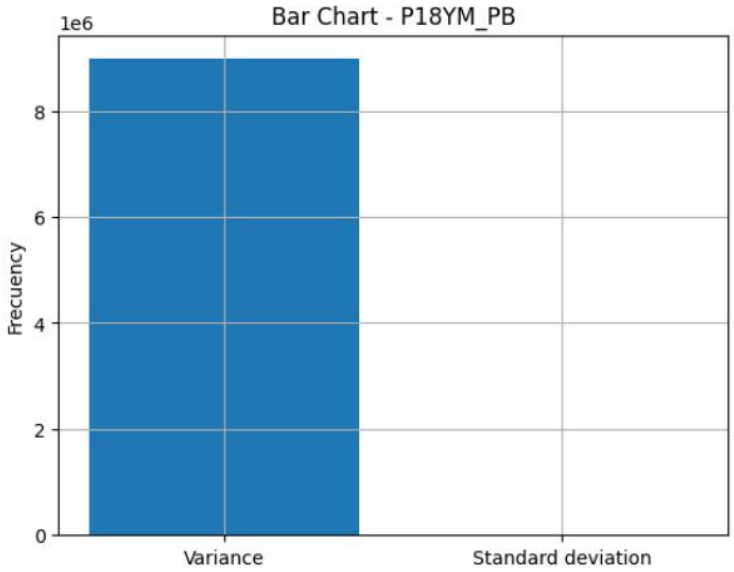
```
colP18EP1 = "P18YM_PB"

variance2 = dfPl[colP18EP1].var()
stdDeviaton2 = dfPl[colP18EP1].std()

print(f"Column: {colP18EP1}")
print(f"Variance: {variance2}")
print(f"Standard deviation: {stdDeviaton2}")
```

```
Column: P18YM_PB
Variance: 8975545.77163052
Standard deviation: 2995.9215229425686
```

7.3.2. Código para calcular la desviación estándar y la varianza de la variable P18YM_PB

Graphic

```
labelsForBarChart4 = ["Variance", "Standard deviation"]
valuesForBarChart4 = [variance2, stdDeviaton2]

plt.bar(labelsForBarChart4, valuesForBarChart4)
plt.ylabel("Frecuency")
plt.title(f"Bar Chart - {colP18EP1}")
plt.grid(True)
plt.show()
```



7.3.3. Código y gráfica de la desviación estándar y la varianza de la variable P18YM_PB

Calcular los quartiles (25%, 50,% 75%)

Por ultimo para calcular los quartiles ocupe las variables de la población con alguna limitación y las personas sin ninguna limitación, y las representé con un diagrama de caja, para esto intente agregar lineas de diferente color y estilo para que se pudieran distinguir los quartiles, sin embargo por los valores muy altos y bajos apenas si se pueden distinguir las lineas. .

Calculate percentiles

PCON_LIMI

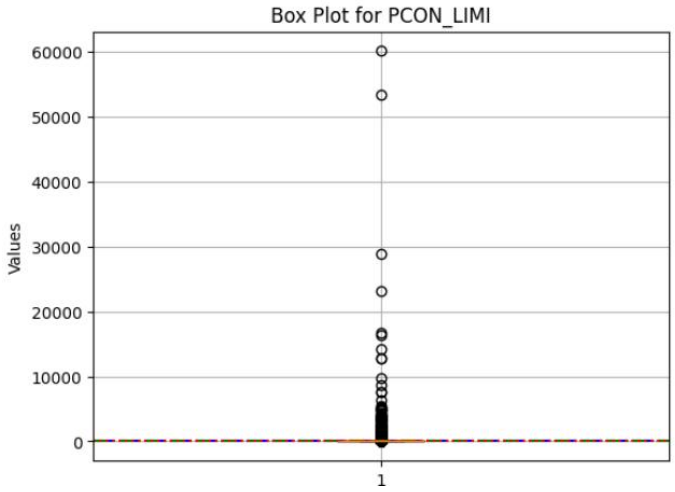
```
colPCL3 = "PCON_LIMI"

p25_1 = dfPl[colPCL3].quantile(0.25, interpolation="nearest")
p50_1 = dfPl[colPCL3].quantile(0.50, interpolation="nearest")
p75_1 = dfPl[colPCL3].quantile(0.75, interpolation="nearest")

print(f"Column: {colPCL3}")
print(f"Percentile 25: {p25_1}")
print(f"Percentile 50: {p50_1}")
print(f"Percentile 75: {p75_1}")
```

```
Column: PCON_LIMI
Percentile 25: 3.0
Percentile 50: 13.0
Percentile 75: 43.0
```

7.4. Código para calcular los quartiles de la variable PCON_LIMI

Graphic

```
dataForBoxPlot3 = dfPl[colPCL3].to_list()

plt.boxplot(dataForBoxPlot3)
plt.axhline(p25_1, color='blue', linestyle='--', label='25%')
plt.axhline(p50_1, color='green', linestyle='-.', label='50%')
plt.axhline(p75_1, color='red', linestyle=':', label='75%')
plt.title(f"Box Plot for {colPCL3}")
plt.ylabel("Values")
plt.grid(True)
plt.show()
```



7.4.1. Código y gráfica de los quartiles de la variable PCON_LIMI

PSIND_LIM

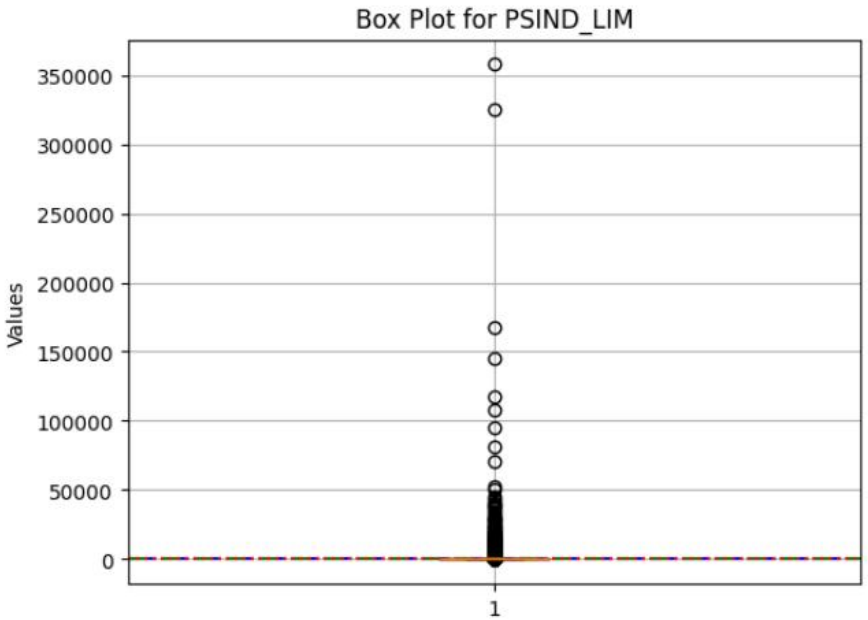
```
colPSL2 = "PSIND_LIM"

p25_2 = dfPl[colPSL2].quantile(0.25, interpolation="nearest")
p50_2 = dfPl[colPSL2].quantile(0.50, interpolation="nearest")
p75_2 = dfPl[colPSL2].quantile(0.75, interpolation="nearest")

print(f"Column: {colPSL2}")
print(f"Percentile 25: {p25_2}")
print(f"Percentile 50: {p50_2}")
print(f"Percentile 75: {p75_2}")
```

```
Column: PSIND_LIM
Percentile 25: 24.0
Percentile 50: 94.0
Percentile 75: 272.0
```

7.4.2. Código de los quartiles de la variable PSIND_LIM

Graphic

```
dataForBoxPlot4 = dfPl[colPSL2].to_list()

plt.boxplot(dataForBoxPlot4)
plt.axhline(p25_2, color='blue', linestyle='--', label='25%')
plt.axhline(p50_2, color='green', linestyle='-.', label='50%')
plt.axhline(p75_2, color='red', linestyle=':', label='75%')
plt.title(f"Box Plot for {colPSL2}")
plt.ylabel("Values")
plt.grid(True)
plt.show()
```



7.4.3. Código y gráfica de los quartiles de la variable PSIND_LIM

Conclusión

En conclusión, este proyecto me permitió fortalecer mis habilidades de análisis de datos y familiarizarme con herramientas que antes desconocía, pero que resultaron ser muy útiles. Aprendí a trabajar con librerías como Polars, Pandas y Matplotlib, lo cual enriqueció mi comprensión del procesamiento y visualización de datos.

Aunque enfrenté varios obstáculos: como una instalación incorrecta de Anaconda Navigator, problemas para visualizar los datos completos en Jupyter Notebook, trabajar inicialmente con un archivo CSV equivocado y perder la contraseña de MySQL. Cada uno de estos desafíos representó una oportunidad para aprender y mejorar. A pesar de los altibajos, logré superar todas las dificultades.

Bibliografia

Desviación estándar. (s/f). Jmp.com. Recuperado el 21 de junio de 2025, de https://www.jmp.com/es/statistics-knowledge-portal/measures-of-central-tendency-and-variability/standard-deviation

Manage workspace members. (s/f). Index - Polars user guide. Pola.Rs. Recuperado el 21 de junio de 2025, de https://docs.pola.rs/

ORM quick start — SQLAlchemy 2.0 documentation. (s/f). Sqlalchemy.org. Recuperado el 21 de junio de 2025, de https://docs.sqlalchemy.org/en/20/orm/quickstart.html

Ortega, C. (2022, diciembre 21). Desviación estándar: Qué es, usos y cómo obtenerla. QuestionPro. https://www.questionpro.com/blog/es/desviacion-estandar/

Quick start guide — Matplotlib 3.10.3 documentation. (s/f). Matplotlib.org. Recuperado el 21 de junio de 2025, de https://matplotlib.org/stable/users/explain/quick_start.html

User Guide — pandas 2.3.0 documentation. (s/f). Pydata.org. Recuperado el 21 de junio de 2025, de https://pandas.pydata.org/docs/user_guide/index.html