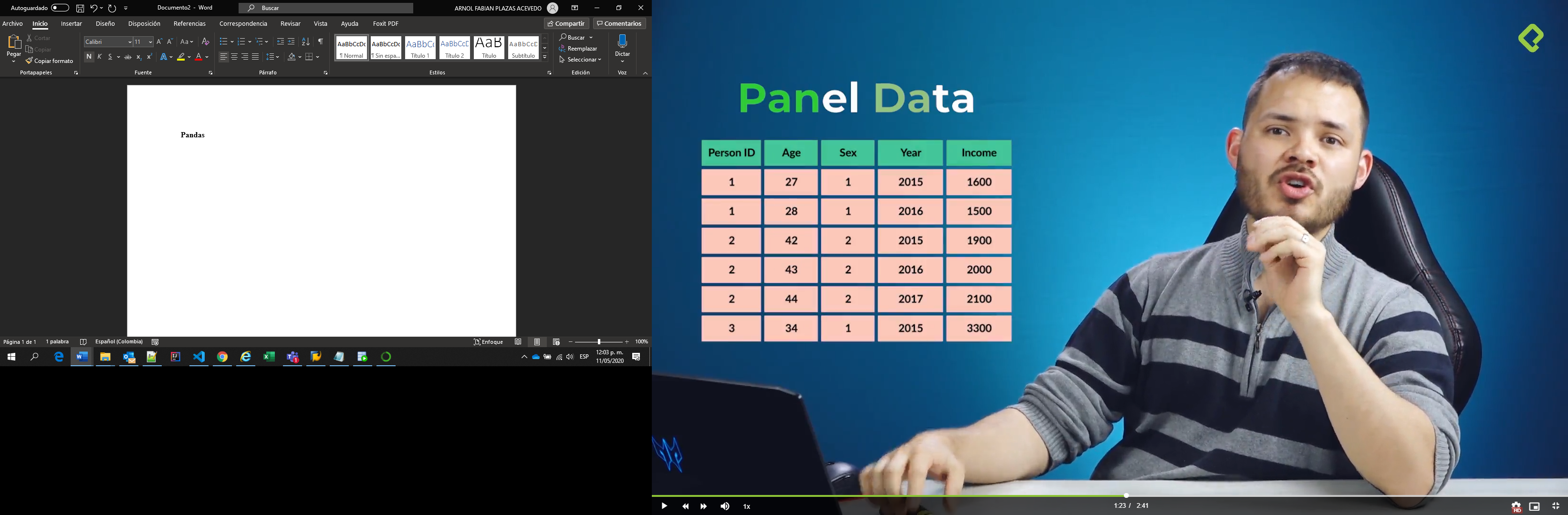
**Pandas**

Inventada por Wes McKinney en el 2008, como una necesidad para analizar grandes volumnes de datos en los mercados financieros.



**Ventajas y desventajas**

* Reduce líneas de código
* Diseñada especialmente para análisis
* API fácil y concisa
* Múltiples funciones
* Compatibilidad con matrices 3D (Numpy)
* Curva de aprendizaje lenta

**Conexión con bases de datos tipo SQL**

Pandas cuenta con una funcionalidad que facilita el acceso a tus bases de datos tipo SQL, para ello te mostrare algunos ejemplos:

**PostgreSQL:**

Valida que tengas la librería psycopg2 usando el comando import. Si no está instalada en tu ambiente, usa el comando !pip install psycopg2 en la terminal de python para instalarlo.

Comenzamos cargando las librerías:

**import** pandas **as** pd

**import** psycopg2

Luego creamos el elemento de conexión con el siguieente código:

conn\_sql = psycopg2.connect(user = "user\_name",

password = "password",

host = "xxx.xxx.xxx.xxx",

port = "5432",

database = "postgres\_db\_name")

Seguido simplemente definimos nuestra query en SQL:

query\_sql = '''

select \*

from table\_name

limit 10

'''

Y creamos nuestro dataframe:

df = pd.read\_sql(query\_sql, sql\_conn)

df.head(5)

**SQL Server:**

Valida que tengas la librería pyodbc usando el comando import, si no está instalada en tu ambiente, usa el comando !pip install pyodbc en la terminal python para instalarlo.

Comenzamos cargando las librerías:

**import** pandas **as** pd

**import** pyodbc

Luego creamos el elemento de conexión con el siguiente código:

driver = '{SQL Server}'

server\_name = 'server\_name'

db\_name = 'database\_name'

user = 'user'

password = 'password'

sql\_conn = pyodbc.connect('''

DRIVER={};SERVER={};DATABASE={};UID={};PWD={};

Trusted\_Connection=yes

'''.format(driver, server\_name, db\_name, user, password))

O si tienes el DSN:

dsn = 'odbc\_datasource\_name'

sql\_conn = pyodbc.connect('''

DSN={};UID={};PWD={};Trusted\_Connection=yes;

'''.format(dsn, user, password))

Seguido simplemente definimos nuestra query en SQL:

query\_sql = 'select \* from table\_name limit 10'

Y creamos nuestro dataframe con:

df = pd.read\_sql(query\_sql, sql\_conn)

df.head(5)

MySQL / Oracle / Otras:

Valida que tengas la librería sqlalchemy usando el comando import, si no está instalada en tu ambiente, usa el comando !pip install sqlalchemy en la terminal de python para instalarlo.

Comenzamos cargando las librerías:

**import** pandas **as** pd

**import** sqlalchemy **as** sql

Escogemos nuestra base de datos, Oracle, MySql o la de tu preferencia:

database\_type = 'mysql'

database\_type = 'oracle'

Luego creamos el elemento de conexión con el siguiente código:

user = 'user\_name'

password = 'password'

host = 'xxx.xxx.xxx.xxx:port'

database = 'database\_name'

conn\_string = '{}://{}:{}@{}/{}'.format(

database\_type, user, password, host, database)

sql\_conn = sql.create\_engine(conn\_string)

Seguido simplemente definimos nuestra query en SQL:

query\_sql = '''

select \*

from table\_name

limit 10

'''

Y creamos nuestro dataframe con:

df = pd.read\_sql(query\_sql, sql\_conn)

df.head(5)

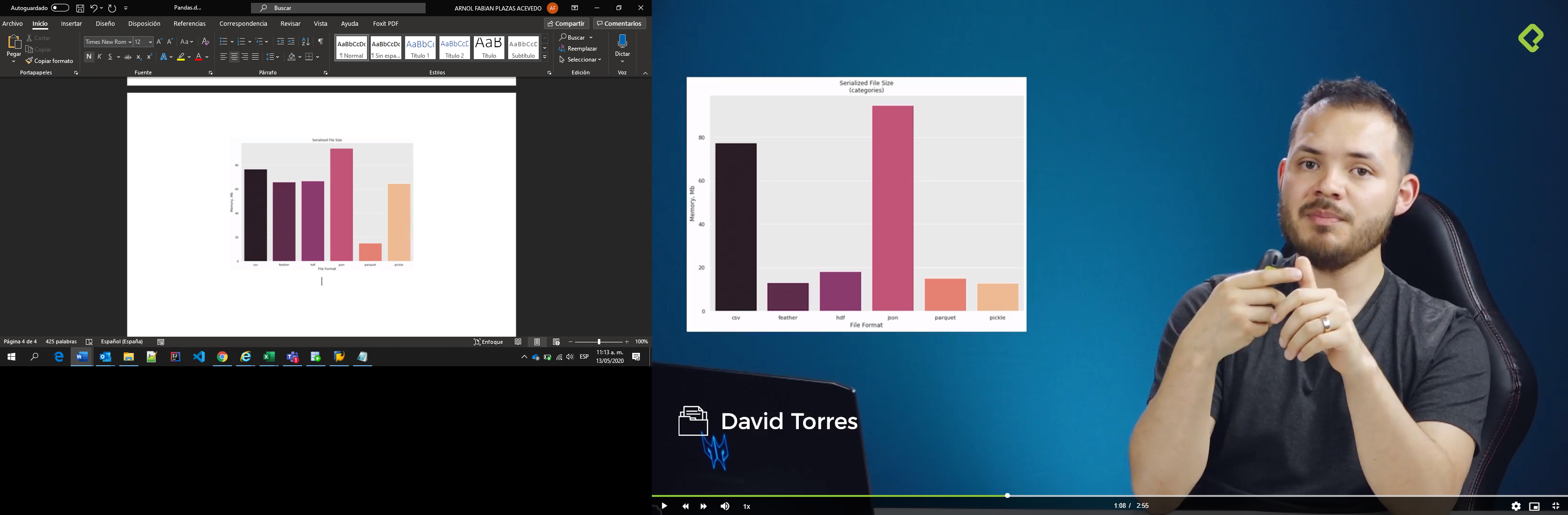
La libreria sqlalchemy también soporta PostgreSQL y otras fuentes de datos.

**IO formatos**

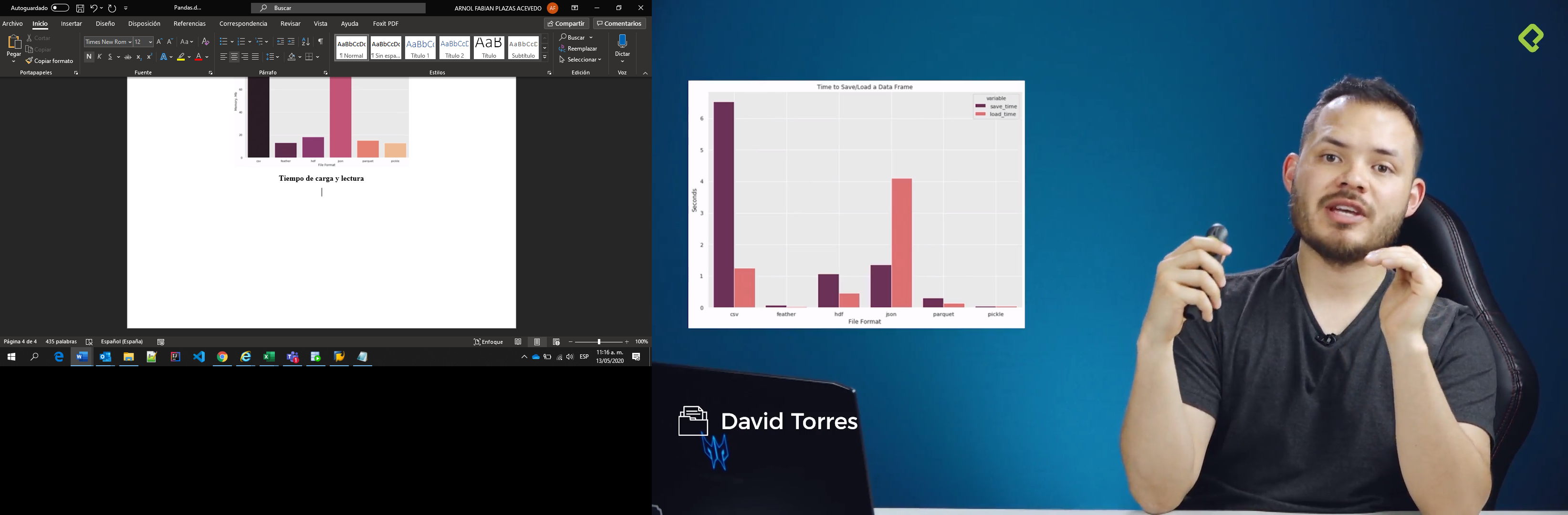
Dataframe con 100.000 registros y 30 campos los cuales 15 son numericos y 15 son formato tipo texto.



**Categorizando las variables tipo texto**



**Tiempo de carga y lectura**



**Consumo de memoria**



**Consumo de memoria categorizando las variables tipo texto**



**Datasets: https://data.worldbank.org/**