Prueba técnica Data Engineer
Mercado Libre

Nombre: Juan David Espitia Aguillón

Objetivo:

Esta prueba consta de tres puntos de desarrollo, cada uno con fines y objetivos diferentes con los cuales buscamos poner a prueba sus habilidades técnicas en diseño y programación.

Stack Tecnológico empleado para la prueba:

1. Lenguaje de programación: Python 3.9
2. Control de Versiones: Git + GitHub + GitFlow
3. Framework: PyCharm
4. Diseños: Draw.io
5. Nube: Google Cloud Platform

Dirección del repositorio de código:
https://github.com/JuanDavidEspitia/meli-tech-test-data-engineer

Punto 1:

a. De acuerdo al set de datos SALARIOS ¿Cuantos cargos estaban ocupados solamente por una persona en 2011?

Inicialmente cargamos los set de datos a dataframes con la librería de Pandas

Dataframe de Salarios

```
****  Dataframe de Salarios ****
            Id       EmployeeName  ...         Agency  Status
0            1     NATHANIEL FORD  ...  San Francisco     NaN
1            2       GARY JIMENEZ  ...  San Francisco     NaN
2            3     ALBERT PARDINI  ...  San Francisco     NaN
3            4  CHRISTOPHER CHONG  ...  San Francisco     NaN
4            5    PATRICK GARDNER  ...  San Francisco     NaN
...        ...                ...  ...            ...     ...
148649  148650      Roy I Tillery  ...  San Francisco     NaN
148650  148651       Not provided  ...  San Francisco     NaN
148651  148652       Not provided  ...  San Francisco     NaN
148652  148653       Not provided  ...  San Francisco     NaN
148653  148654          Joe Lopez  ...  San Francisco     NaN
```

Dataframe de Compras

```
****  Dataframe de Compras ****
                                                Address  ... Purchase Price
0     16629 Pace Camp Apt. 448\nAlexisborough, NE 77...  ...          98.14
1     9374 Jasmine Spurs Suite 508\nSouth John, TN 8...  ...          70.73
2                       Unit 0065 Box 5052\nDPO AP 27450  ...           0.95
3                  7780 Julia Fords\nNew Stacy, WA 45798  ...          78.04
4     23012 Munoz Drive Suite 337\nNew Cynthia, TX 5...  ...          77.82
...                                                  ...  ...            ...
9995      966 Castaneda Locks\nWest Juliafurt, CO 96415  ...          82.21
9996  832 Curtis Dam Suite 785\nNorth Edwardburgh, T...  ...          25.63
9997               Unit 4434 Box 6343\nDPO AE 28026-0283  ...          83.98
9998                0096 English Rest\nRoystad, IA 12457  ...          38.84
9999     40674 Barrett Stravenue\nGrimesville, WI 79682  ...          67.59
```

Imprimimos en pantalla la cantidad de registros por cada uno

```
Cantidad de Registros set de Salarios: 148654
Cantidad de Registros set de Compras: 10000
```

Filtramos el set de datos de salarios para que solo estén los datos pertenecientes al año 2011.

Luego agrupamos el set de datos con los campo JobTitle e ID y obtenemos la cantidad de cargos ocupados por una única persona y posteriormente nos quedamos con aquellos cuyo conteo haya sido de uno solo.

```
                              JobTitle     Id  Cantidad
0                          ACCOUNT CLERK  20766         1
1                     RECREATION LEADER  34168         1
2                     RECREATION LEADER  34095         1
3                     RECREATION LEADER  34112         1
4                     RECREATION LEADER  34120         1
...                                 ...    ...       ...
36154  IS BUSINESS ANALYST - PRINCIPAL   5721         1
36155  IS BUSINESS ANALYST - PRINCIPAL   5429         1
36156  IS BUSINESS ANALYST - PRINCIPAL   5428         1
36157  IS BUSINESS ANALYST - PRINCIPAL   5210         1
36158                       ZOO CURATOR  18779         1
```

La cantidad final es de:

```
[36159 rows x 3 columns]
Cantidad de Cargos que son ocupados por una sola persona: 36159
```

b. De acuerdo al set de datos SALARIOS ¿Cuánta gente tiene la palabra 'MANAGER' en su cargo?

Inicialmente filtramos por el campo JobTitle todos los que contengan la palabra manager y luego los sumamos para obtener la cantidad.

```
**************    Punto 2   **************
Filtramos por el campo JobTitle todos aquellos que contengan la palabra manager
0          True
1         False
2         False
3         False
4         False
          ...
148649    False
148650    False
148651    False
148652    False
148653    False
Name: JobTitle, Length: 148654, dtype: bool
Cantidad de personas cuyo cargo contiene MANAGER: 1030
```

La cantidad de personas con la palabra MANAGER son: 1030

c. De acuerdo al set de datos SALARIOS ¿Cual es el nombre de la persona que menos gana (incluyendo beneficios - TotalPayBenefits)?

   Se debe comparar el menor salario de la columna TotalPayBenefits con la persona que obtiene ese mismo valor

```
**************    Punto 3   **************
Comparamos el menor salario de la columna TotalPayBenefits y lo comparamos con la persona que tiene el menor valor
La persona con el menor salario es:
           Id EmployeeName  ...         Agency  Status
148653  148654    Joe Lopez  ...  San Francisco     NaN

[1 rows x 13 columns]
```

La persona con el menor salario es:
Joe Lopez ID: 148654

d. De acuerdo al set de datos SALARIOS ¿cuál es el salario base (BasePay) promedio de todos los empleados para el año(2012)?

   Filtramos el DF de salarios por el año 2012 y luego con el campo de BasePay (Salario Base) obtenemos el promedio.

```
**************    Punto 4   **************
Filtramos el DF de Salarios por el año 2012 y luego con el campo BasePay sacamos el promedio de salarios
El Salario base promedio es de: 65436.40685742263
```

El Promedio de salario es: 65436,407

e. De acuerdo al set de datos SALARIOS ¿cuál fue la suma total pagada con beneficios por los dos trabajos más populares?.

Hacemos uso del método value_conts() para obtener el número de ocurrencias de cada uno y así obtener el más populares y Posteriormente usamos la sentencia para sumar el total de beneficios pagados de ambos trabajos.

```
**************    Punto 5   **************
Hacemos uso del metodo value_conts() para obtener el numero de ocurrencias de cada uno y asi obtener el mas popular
Transit Operator    7036
Special Nurse       4389
Name: JobTitle, dtype: int64
807419537.5400001
```

El total pagado por los 2 trabajos más populares es: 807419537.540001

f. De acuerdo al set de datos COMPRAS ¿Cuales son los 5 proveedores de correo electrónico más comunes, con cuantos usuarios está asociado cada uno? (hotmail.com,gmai.com, etc)

```
**************    Punto 6   **************
Hacemos un split en el campo de correo separado por el @ para separar el dominio
Posteriormente contamos las ocurrencias de cada dominio y listamos los 5 primeros
hotmail.com     1638
yahoo.com       1616
gmail.com       1605
smith.com         42
williams.com      37
Name: 1, dtype: int64
```

Los 5 correos más comunes son:

Proveedor	Ocurrencias
Hotmail.com	1638
Yahoo.com	1616
Gmail.com	1605
Smith.com	42
Williams.com	37

g. De acuerdo al set de datos COMPRAS ¿Cuantas personas tienen una tarjeta de crédito que expira en 2025?

Con el Campo CC Exp Date lo usamos para validar si contine la cadena /25, que indica que vence en el 2025 y luego usamos la función len para validar la cantidad de registros reportados

```
**********    Punto 7   **********
Con el Campo CC Exp Date lo usamos para validar si contine la cadena /25, que indica que vence en el 2025
                                          Address  ... Purchase Price
4     23012 Munoz Drive Suite 337\nNew Cynthia, TX 5...  ...          77.82
5     7502 Powell Mission Apt. 768\nTravisland, VA 3...  ...          25.15
7     260 Rachel Plains Suite 366\nCastroberg, WV 24...  ...          44.25
13    118 Melton Via Suite 681\nAlexanderbury, FL 32104  ...           8.93
15       31730 Chelsea Crest\nBlakemouth, CT 90395-0620  ...          71.78
...                                                 ...  ...           ...
9955      66539 Potts Forge\nCartershire, AK 58271-5180  ...          84.09
9963  3865 Davis Meadow Suite 915\nPort Stacieview, ...  ...          58.17
9993  7555 Larson Locks Suite 229\nEllisburgh, MA 34...  ...          65.61
9994       6276 Rojas Hollow\nLake Louis, WY 56410-7837  ...          31.85
9996  832 Curtis Dam Suite 785\nNorth Edwardburgh, T...  ...          25.63

[1033 rows x 14 columns]
La cantidad de personas que vencen la tarjeta de credito en 2025 son: 1033
```

La cantidad de personas que vence su tarjeta de crédito en 2025 son: 1033

h. De acuerdo al set de datos COMPRAS ¿Cuántas personas tienen tarjetas Mastercard e hicieron una compra por más de $20?

Debemos realizar dos filtros, el primero sobre el campo CC Provider cuyo valor sea igual a Mastercard y el segundo sobre el campo Purchase Price cuyo valor sea superior a los 20$

```
**********    Punto 8   **********
Realizamos dos filtros
1. Donde el CC Provider sea Mastercard
2. Donde el precio de las compras sea superior a 20
Cantidad de personas con Mastercard y compra superior a los 20$: 651

Process finished with exit code 0
```

La cantidad de personas con Mastercard y compra superior a los 20$ son: 651

i. De acuerdo al set de datos COMPRAS ¿Alguien hizo una compra desde Lot: "90 WT", ¿cual fue el precio de compra de esta transacción?

```
**********    Punto 9   **********
Filtramos en el campo Lot el valor 90 WT y mostramos solamente el precio
   index  Precio
0    513    75.1
```

El precio final es de 75.1

j. De acuerdo al set de datos COMPRAS ¿ Cuánto suma el total de precio de compras para las dos compañías menos populares?, ¿Cuáles son esas dos compañías?

```
**************    Punto 10   **************
Filtramos las dos compañias con mayor ocurrencia en el set de datos
Luego sumamos el campo Purchase Price
Total de precio de las dos compañias mas populares: 121.51
Las compañias son:
Davis, Parker and Rivera    1
Greene Inc                  1
Name: Company, dtype: int64
```

El total de precio de las dos compañías más populares es de: 121.51 y las dos compañías son:

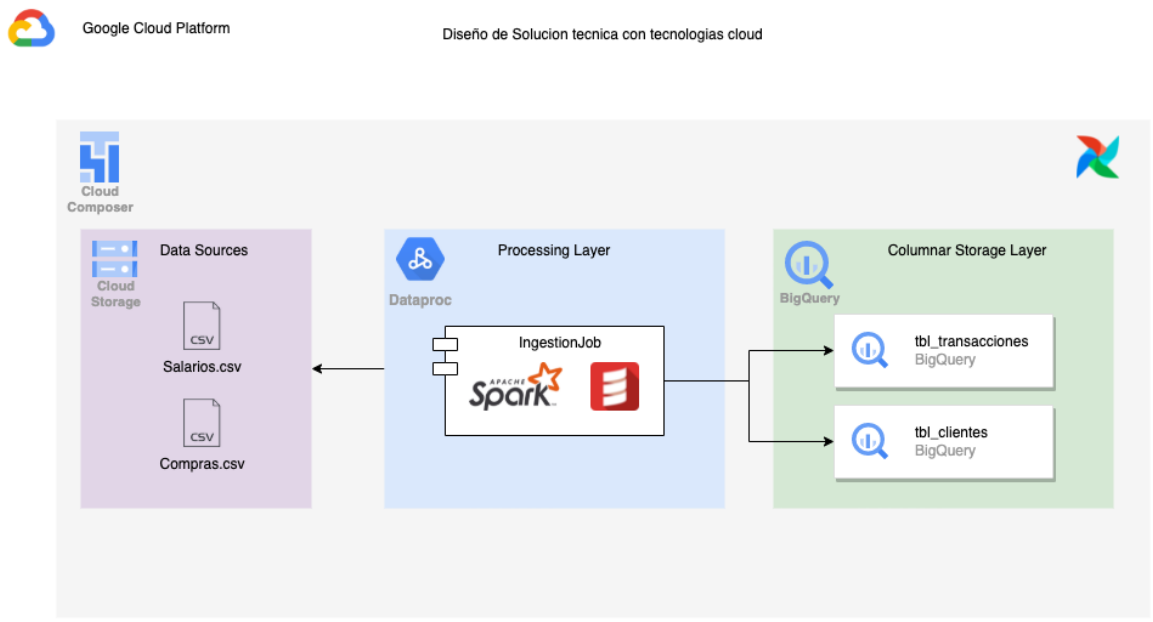
- Davis, Parker and Rivera
- Greene Inc

Punto 2:

Diseño Cloud – GCP

Diseño de la solución con tecnologías Cloud. Para este ejercicio se uso la nube de Google con sus servicios para procesar datos.

La arquitectura general de solución es la siguiente:



Nota:

El diseño está pensado en un pipeline de datos con un alto volumen, por tal razón se necesita de servicios idóneos para la transformación de los datos.

El Componente de IngestionJob, sería un desarrollo construido en el framework de procesamiento de datos Apache Spark, bajo el lenguaje de programación Scala, donde se implementaría la lógica de lectura de los archivos, las transformación de concatenación, hash, homologación y persistencia de los datasets finales en BQ

Si se piensa un diseño con set de datos de menor volumen, podemos reemplazar en la capa de procesamiento por el artefacto de Scala por una Cloud Function, desarrollada con Python y podemos utilizar el código desarrollado para la prueba en el punto 2.

El desarrollo se realizó de la necesidad se implementó en lenguaje Python por efectos de agilidad con la entrega de la prueba técnica

Implementación

El primer paso es establecer las rutas donde se encuentran los datos a ser cargados.

```
# Primero especificamos un patrón de los archivos y lo pasamos como parámetro en la función glob para la ruta
# de datos y datos complementos
print("Function Read Files in List String")
pathDatos = glob.glob('/Users/JuanEspitia/Documents/Github/data-engineer/meli-tech-test-data-engineer/data/punto2/*.csv')
pathComplementos = glob.glob('/Users/JuanEspitia/Documents/Github/data-engineer/meli-tech-test-data-engineer/data/punto2/datos_complementos/*.csv')
print("File List Path1: " + str(pathDatos))
print("File List Path2: " + str(pathComplementos))
```

Resultado:

```
Function Read Files in List String
File List Path1: ['/Users/JuanEspitia/Documents/Github/data-engineer/meli-tech-test-data-engineer/data/punto2/MOCK_DATA_1.csv', '/Users/JuanEspitia/Documents/Github/data-engineer/meli-tech-test-data-engineer/data/pu
File List Path2: ['/Users/JuanEspitia/Documents/Github/data-engineer/meli-tech-test-data-engineer/data/punto2/datos_complementos/MOCK_DATA_18 - copia.csv', '/Users/JuanEspitia/Documents/Github/data-engineer/meli-tec
```

Seguido, creamos una función que itere cada una de las rutas con sus respectivos archivos .csv y vamos cargando uno a uno de los archivos .csv en una lista de dataframes y retornamos la lista de lista de dataframes.

```
# Ahora creamos una funcion que se encargue de leer los archivos para cada carpeta
print("Function Load Files")
def readFiles(csv_files):
  #files = glob.glob(ruta)
  list_data = []
  for filename in csv_files:
      data = pd.read_csv(filename)
      list_data.append(data)
  return list_data
```

```
Function Load Files
****** List Dataframes ******
[       id     nombre  ... numero_identificacion tipo_identificacion
0        1     Freddi  ...            232219919                  CC
1        2      Emmie  ...            359867238                  PP
2        3      Riley  ...            966156703                  PP
3        4      Libby  ...            814981127                  IT
4        5     Marcia  ...            820247909                  PP
..     ...        ...  ...                  ...                 ...
995    996  Corabelle  ...           1012320432                  CC
996    997     Marlin  ...            341301956                  IT
997    998    Garrett  ...            828216055                  IT
998    999     Maible  ...            431179297                  IT
999   1000       Arch  ...            809361061                  CC
```

Luego hacemos uso de la función de Python concat() de dataframes para concatenar los elementos que contiene la lista que nos retorna la función anterior y lo convertimos en un dataframe para cada ruta de datos.

```python
# Hacemos uso de la funcion concat para concatenar
print("****** List Dataframes ******")
df1 = readFiles(pathDatos)
print(df1)
print(" Concat List Dataframes in one Dataframes of Data ")
dfDatos = pd.concat(readFiles(pathDatos), axis=0, ignore_index=True)
print(dfDatos)
print(" Concat List Dataframes in one Dataframes of Data Complements")
dfDatosComple = pd.concat(readFiles(pathComplementos), axis=0, ignore_index=True)
print(dfDatosComple)
```

Output

```
 Concat List Dataframes in one Dataframes of Data Complements
        id      nombre  ... numero_identificacion tipo_identificacion
0        1     Claudie  ...              45809058                  CC
1        2        Alys  ...             611607601                  IT
2        3      Pearle  ...             656812545                  CC
3        4        Glen  ...             234071123                  IT
4        5     Delbert  ...             604466829                  PP
...    ...         ...  ...                  ...                 ...
99995  996   Kristofor  ...             740207390                  CC
99996  997      Lizzie  ...             789619747                  CC
99997  998     Kerwinn  ...             625964278                  PP
99998  999    Kakalina  ...             541933418                  CC
99999 1000  Kristopher  ...             199170383                  CC
```

Luego consolidamos los dataframes de datos y datos_complementarios en un solo dataframe, nuevamente con la función concat()

```python
# Consolidamos los dos dataframes en uno solo
print("Union Datos & Datos_Complement")
dfConsolidate = pd.concat([dfDatos,dfDatosComple])
print(dfConsolidate)
```

Output

```
Union Datos & Datos_Complement
        id      nombre  ... numero_identificacion tipo_identificacion
0        1      Freddi  ...            232219919                  CC
1        2       Emmie  ...            359867238                  PP
2        3       Riley  ...            966156703                  PP
3        4       Libby  ...            814981127                  IT
4        5      Marcia  ...            820247909                  PP
...    ...         ...  ...                  ...                 ...
99995  996   Kristofor  ...            740207390                  CC
99996  997      Lizzie  ...            789619747                  CC
99997  998     Kerwinn  ...            625964278                  PP
99998  999    Kakalina  ...            541933418                  CC
99999 1000  Kristopher  ...            199170383                  CC
```

Posteriormente procedemos a eliminar los registros duplicados del dataframe consolidado mediante la función de drop_duplicates()

```
# Eliminimamos los duplicados del dataframe
print("Drop duplicates in Dataframe")
print(dfConsolidate.duplicated())
dfWithoutDuplicates = dfConsolidate.drop_duplicates()
print("Dataframe without duplicates")
print(dfWithoutDuplicates)
```

Output

```
Drop duplicates in Dataframe
0        False
1        False
2        False
3        False
4        False
         ...
99995     True
99996     True
99997     True
99998     True
99999     True
Length: 109000, dtype: bool
Dataframe without duplicates
```

Luego procedemos a concatenar los campos tipo y numero de documento mediante las siguientes sentencias de código

```
# Procedemos a concatenar el tipo y numero de cedula
print("Concat columns tipo_identificacion and numero_identificacion")
dfWithoutDuplicates['tipo_numero_identificacion'] = dfWithoutDuplicates['tipo_identificacion']+''+dfWithoutDuplicates['numero_identificacion'].apply(str)
print(dfWithoutDuplicates)
```

Y obtenemos la siguiente salida

	id	nombre	...	tipo_identificacion	tipo_numero_identificacion
0	1	Freddi	...	CC	CC232219919
1	2	Emmie	...	PP	PP359867238
2	3	Riley	...	PP	PP966156703
3	4	Libby	...	IT	IT814981127
4	5	Marcia	...	PP	PP820247909
...	...	...	...	...	...
91995	996	Kristofor	...	CC	CC740207390
91996	997	Lizzie	...	CC	CC789619747
91997	998	Kerwinn	...	PP	PP625964278
91998	999	Kakalina	...	CC	CC541933418
91999	1000	Kristopher	...	CC	CC199170383

Posterior a la concatenación de columnas realizamos el proceso de encriptar la columna generada de la concatenación de tipo y numero de documento, esto mediante las siguientes sentencias.

```
# Procedemos a crear el campo hash para la tupla
print("Add column Hash")
dfWithoutDuplicates['hash'] = dfWithoutDuplicates['tipo_numero_identificacion'].apply(lambda x: hashlib.sha256(x.encode()).hexdigest())
print(dfWithoutDuplicates.info())
print(dfWithoutDuplicates)
```

Ahora separamos los dos set de datos requeridos, el de Clientes y el de Transacciones con las especificaciones de campos indicados en el documento de la prueba técnica.

```
# Ahora creamos un dataframe de transacciones
print("Dataframe Transacciones")
dftransacciones = dfWithoutDuplicates.drop(['id','nombre','apellido','email','genero','numero_identificacion','tipo_identificacion','tipo_numero_identificacion'], axis=1)
print(dftransacciones.info())
print(dftransacciones)

# Ahora dejamos el Dataframe de solo clientes
print("Dataframe Clientes")
dfClientes = dfWithoutDuplicates.drop(['valor_tx','numero_identificacion','tipo_identificacion','tipo_numero_identificacion'], axis=1)
print(dfClientes.info())
print(dfClientes)
```

Dataframe transacciones

None

	valor_tx	hash
0	$104571.16	e798c45c2f0238de52899540f1f3a39dca4f0f555bcff1...
1	$260057.30	660988ad205e04622e468c043c55fca81f0305eefc5d9d...
2	$463707.55	16e70d31143d240229c0a09858a2bbeec41d46a9a6576c...
3	$766231.08	10ef61638b2fbbfe63d7bdd0df63b4e509560be2dad3cb...
4	$261899.95	be112370d6353f8b301d2aa0d77a278264dcc6e4090103...
...	...	...
91995	$152793.41	fb6e285097d702403a62f0953de7ca4d8e25320bbe7bf4...
91996	$492511.62	c313a0cb6d7e9392c9f4c35100299abb0ed05c4216073c...
91997	$152805.17	84ecedf2a475090e865cffba26e31acbf903fc6a803c11...
91998	$672227.08	f21e493be7dcfc90f0e7257ab19e7b7ba9367b5e025ec2...
91999	$427363.63	8f6dc236bd944d403588e2d09def676efae8af2ba9a039...

Dataframe de Clientes
Nota: No se visualizan bien todas las columnas

```
None
         id  ...                                               hash
0         1  ...  e798c45c2f0238de52899540f1f3a39dca4f0f555bcff1...
1         2  ...  660988ad205e04622e468c043c55fca81f0305eefc5d9d...
2         3  ...  16e70d31143d240229c0a09858a2bbeec41d46a9a6576c...
3         4  ...  10ef61638b2fbbfe63d7bdd0df63b4e509560be2dad3cb...
4         5  ...  be112370d6353f8b301d2aa0d77a278264dcc6e4090103...
...     ...  ...                                               ...
91995   996  ...  fb6e285097d702403a62f0953de7ca4d8e25320bbe7bf4...
91996   997  ...  c313a0cb6d7e9392c9f4c35100299abb0ed05c4216073c...
91997   998  ...  84ecedf2a475090e865cffba26e31acbf903fc6a803c11...
91998   999  ...  f21e493be7dcfc90f0e7257ab19e7b7ba9367b5e025ec2...
91999  1000  ...  8f6dc236bd944d403588e2d09def676efae8af2ba9a039...
```

Complemento de la salida:

```
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   id        50000 non-null  int64 
 1   nombre    50000 non-null  object
 2   apellido  50000 non-null  object
 3   email     50000 non-null  object
 4   genero    50000 non-null  object
 5   hash      50000 non-null  object
dtypes: int64(1), object(5)
memory usage: 2.7+ MB
```

Por ultimo mostramos el Top 10 de los clientes con más transacciones realizadas

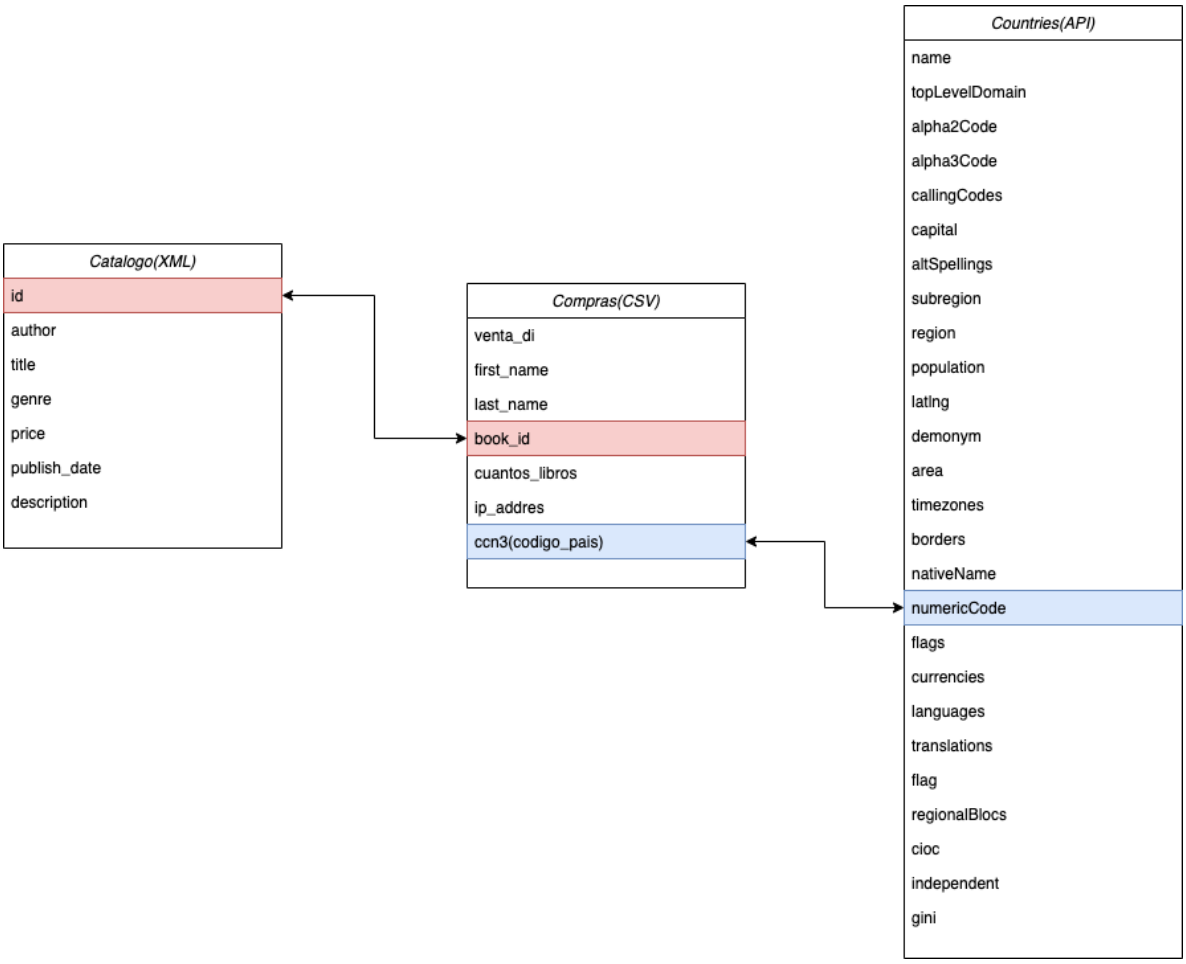
```python
# Por ultimo mostramos un top 10 de cada uno de los dataframes
print("****** Top 10 ******")
print(dfClientes.value_counts().head(10))
print(dftransacciones.value_counts().head(10))
print("*********   TOP 10 BY TRX  *********")
print(dfClientes[dfClientes.hash.isin(dftransacciones['hash'].head(10))])
```

Salida por consola

```
*********   TOP 10 BY TRX  *********
   id    nombre  ...  genero                                               hash
1   1    Freddi  ...  Female  e798c45c2f0238de52899540f1f3a39dca4f0f555bcff1...
2   2     Emmie  ...  Female  660988ad205e04622e468c043c55fca81f0305eefc5d9d...
3   3     Riley  ...    Male  16e70d31143d240229c0a09858a2bbeec41d46a9a6576c...
4   4     Libby  ...  Female  10ef61638b2fbbfe63d7bdd0df63b4e509560be2dad3cb...
5   5    Marcia  ...  Female  be112370d6353f8b301d2aa0d77a278264dcc6e4090103...
6   6    Gerome  ...    Male  b8b2362371f1911ef5a7d8cd4b6a40a3df3d2eb5ba4bb3...
7   7   Lizette  ...  Female  d1a1b15d3d336430e39b0d96ab8ae48b79b45e8a025d8a...
8   8  Rosaleen  ...  Female  9d7f948a12e14a762e4b1f7188c2370dc8c91c4b12d244...
9   9     Datha  ...  Female  66e70311c951c8436dc896cc98ece47a4654a814d271c2...
10 10      Rudd  ...    Male  be4f3c716a15bbee873cf7b34c5abc863cf3c9244aaf24...
```

Punto 3

Lo primero que se debe realizar es un análisis de como están conformados los datos, comprender el dominio de cada uno de estos sets y como se pueden relacionar cada uno de ellos. Por tal razón la primera tarea que se realizo fue aterrizar los datos en un diagrama de relación de estos objetos e identificación de los campos llave para comprender como se pueden relacionar. La siguiente figura muestra la relación de cada uno de estos objetos.



Posteriormente cargamos todos los sets de datos a sus respectivos dataframes, como lo evidencia las imágenes a continuación.

```python
# Leemos el archivo  CSV de compras
# Lo cargamos a DF
dfCompras = pd.read_csv(pathCompras)
print("************       Dataset de Compras      ************")
print(dfCompras.value_counts().head(10))
print(dfCompras.info())
print(dfCompras.describe())
print("Cantidad de registros: " + str(len(dfCompras)))
```

Salida:

Unnamed: 0	venta_id	first_name	last_name	book_id	cuantos_libros	ip_address	ccn3(Codigo_pais)	
0	1	Charita	Fain	bk103	2	50.66.62.43	276.0	1
673	674	Benedetta	Spilsbury	bk111	5	93.220.3.68	840.0	1
660	661	Cyrill	Dugall	bk108	3	181.35.60.176	840.0	1
661	662	Wade	Gavey	bk101	1	243.171.32.223	840.0	1
662	663	Even	Brockest	bk104	3	73.67.161.154	840.0	1
663	664	Ximenes	Belhomme	bk112	1	221.81.182.248	840.0	1
664	665	Ezra	Jurgenson	bk109	4	212.251.7.43	840.0	1
665	666	Tildy	Frost	bk109	2	165.59.113.35	840.0	1
666	667	Norman	Collingham	bk107	2	128.152.46.242	840.0	1
667	668	Zorah	Obeney	bk109	1	132.109.102.75	840.0	1

dtype: int64

Dataset Catálogos

```python
# Leemos el archivo XML
# Debemos instalar el paquete de python lxml comando = pip install lxml
# Uso constantemente el metodo values_counts, por que me permite ver todos los campos de del set de datos
dfCatalogo = pd.read_xml(pathCatalogoXML)
print("************       Dataset de Catalogos      ************")
print(dfCatalogo.value_counts().head(10))
print(dfCatalogo.info())
print(dfCatalogo.describe())
print("Cantidad de registros: " + str(len(dfCatalogo)))
dfCatalogo.to_csv(pathOut + "/Catalogs.csv", index=False, sep='|')
```

Salida:

```
************       Dataset de Catalogos      ************
```

id	author	title	genre	price	publish_date	description	
bk101	Gambardella, Matthew	XML Developer's Guide	Computer	44.95	2000-10-01	An in-depth look at creating applications \n	wi
bk102	Ralls, Kim	Midnight Rain	Fantasy	5.95	2000-12-16	A former architect battles corporate zombies, \n	
bk103	Corets, Eva	Maeve Ascendant	Fantasy	5.95	2000-11-17	After the collapse of a nanotechnology \n	socie
bk104	Corets, Eva	Oberon's Legacy	Fantasy	5.95	2001-03-10	In post-apocalypse England, the mysterious \n	a
bk105	Corets, Eva	The Sundered Grail	Fantasy	5.95	2001-09-10	The two daughters of Maeve, half-sisters, \n	ba
bk106	Randall, Cynthia	Lover Birds	Romance	4.95	2000-09-02	When Carla meets Paul at an ornithology \n	conf
bk107	Thurman, Paula	Splish Splash	Romance	4.95	2000-11-02	A deep sea diver finds true love twenty \n	thou
bk108	Knorr, Stefan	Creepy Crawlies	Horror	4.95	2000-12-06	An anthology of horror stories about roaches,\n	
bk109	Kress, Peter	Paradox Lost	Science Fiction	6.95	2000-11-02	After an inadvertant trip through a Heisenberg\n	
bk110	O'Brien, Tim	Microsoft .NET: The Programming Bible	Computer	36.95	2000-12-09	Microsoft's .NET initiative is explored in \n	c

Dataset Countries (API Rest)

```python
# leemos ahora la API Rest
# https://restcountries.com/v3.1/all
# https://restcountries.com/v2/all
# https://restcountries.com/#api-endpoints-v3-subregion
res = requests.get("https://restcountries.com/v2/all")
j = res.json()
dfRestCountries = pd.read_json(json.dumps(j))
print("************       Dataset de Paises      ************")
print(dfRestCountries)
print(dfRestCountries.info())
dfRestCountries.to_csv(pathOut + "/Countries.csv", index=False)
```

Salida:

```
*********       Dataset de Paises      ************
                 name topLevelDomain alpha2Code  ... cioc independent  gini
          Afghanistan          [.af]         AF  ...  AFG        True   NaN
        Åland Islands          [.ax]         AX  ...  NaN       False   NaN
              Albania          [.al]         AL  ...  ALB        True  33.2
              Algeria          [.dz]         DZ  ...  ALG        True  27.6
       American Samoa          [.as]         AS  ...  ASA       False   NaN
                  ...            ...        ...  ...  ...         ...   ...
    Wallis and Futuna          [.wf]         WF  ...  NaN       False   NaN
       Western Sahara          [.eh]         EH  ...  NaN       False   NaN
                Yemen          [.ye]         YE  ...  YEM        True  36.7
               Zambia          [.zm]         ZM  ...  ZAM        True  57.1
             Zimbabwe          [.zw]         ZW  ...  ZIM        True  50.3
```

Luego entender cómo se relacionan cada uno de ellos, notamos que debemos hacer unas transformaciones de limpieza de datos en algunos campos, como es el caso del campo ccn3(Codigo del Pais) del Dataset de compras, como se puede ver en la siguiente imagen.

```
Unnamed: 0  venta_id  first_name  last_name   book_id  cuantos_libros  ip_address       ccn3(Codigo_pais)
0           1         Charita     Fain        bk103    2               50.66.62.43      276.0               1
673         674       Benedetta   Spilsbury   bk111    5               93.220.3.68      840.0               1
660         661       Cyrill      Dugall      bk108    3               181.35.60.176    840.0               1
661         662       Wade        Gavey       bk101    1               243.171.32.223   840.0               1
662         663       Even        Brockest    bk104    3               73.67.161.154    840.0               1
663         664       Ximenes     Belhomme    bk112    1               221.81.182.248   840.0               1
664         665       Ezra        Jurgenson   bk109    4               212.251.7.43     840.0               1
665         666       Tildy       Frost       bk109    2               165.59.113.35    840.0               1
666         667       Norman      Collingham  bk107    2               128.152.46.242   840.0               1
667         668       Zorah       Obeney      bk109    1               132.109.102.75   840.0               1
dtype: int64
```

Ahora bien procedo a castear el tipo de dato de float64 a un tipo de dato Entero para realizar los cruces de tablas y en el mismo bloque de código renombro la variable para que tenga el mismo nombre de la llave del set de Countries. Esto mediante las siguientes sentencias de código.

```python
# Comenzamos con la transformacion en el dataset de
dfCompras['numericCode'] = dfCompras["ccn3(Codigo_pais)"].fillna(0).apply(np.int64)
# Borramos la columna con la siguiente sentencia sin necesidad de reasignarel df
dfCompras.drop('ccn3(Codigo_pais)', axis=1, inplace=True)
print(dfCompras.head(3))
print(dfCompras.dtypes)
```

Salida:

```
Casteo de Float64 a Int64 y renombrado de campo
   Unnamed: 0  venta_id first_name  ... cuantos_libros     ip_address  numericCode
0           0         1    Charita  ...              2    50.66.62.43          276
1           1         2       Issi  ...              4   27.8.160.236          276
2           2         3     Tiffie  ...              4  72.193.238.62          276

[3 rows x 8 columns]
Unnamed: 0         int64
venta_id           int64
first_name        object
last_name         object
book_id           object
cuantos_libros     int64
ip_address        object
numericCode        int64
dtype: object
```

Ahora cruzamos dos sets de datos, el set de compras con el set de países para asi tener el detalle de la compra en que país se realizó. De esta forma el Set de datos nos queda más nutrido de información. Esto lo hacemos mediante la sentencia de reduce(), que es útil cuando necesita aplicar una función a un iterable y reducirlo a un solo valor acumulativo. Y la lambda con dos variables que corresponden al mismo nombre del campo llave para cruzar los dos dataframes.

```python
# Ahora cruzamos la los sets de datos para tener una sabana de datos mas nutrida de informacion
# Usamos la sentencia left para no excluir aquellos registros del set de compras que no tienen codigo de pais
dfMerged = reduce(lambda x,y: pd.merge(x,y, on='numericCode', how='left'), [dfCompras, dfRestCountries])
print(dfMerged)
```

Salida:

```
     Unnamed: 0  venta_id first_name  ... cioc independent  gini
0             0         1    Charita  ...  GER        True  31.9
1             1         2       Issi  ...  GER        True  31.9
2             2         3     Tiffie  ...  GER        True  31.9
3             3         4      Trudy  ...  GER        True  31.9
4             4         5    Reinald  ...  GER        True  31.9
..          ...       ...        ...  ...  ...         ...   ...
995         995       996      Aldus  ...  USA        True  41.4
996         996       997      Elsie  ...  USA        True  41.4
997         997       998       Russ  ...  USA        True  41.4
998         998       999      Rorie  ...  USA        True  41.4
999         999      1000    Ingunna  ...  USA        True  41.4
```

Ahora hacemos el renombrado del campo llave del Set de Catalogos, para que sea fácilmente identificado por el mismo nombre del campo por el que aparece en el Set de Compras. Esto lo hacemos mediante la siguiente función.

```
# Ahora realizamos el renombrado del campo llave en el set de catalogos
print(dfCatalogo.columns)
print("Rename column ID by BOOK_ID")
dfCatalogo.rename(columns={'id': 'book_id'}, inplace=True)
print(dfCatalogo.columns)
```

Salida:

```
[1000 rows x 33 columns]
Index(['id', 'author', 'title', 'genre', 'price', 'publish_date',
       'description'],
      dtype='object')
Rename column ID by BOOK_ID
Index(['book_id', 'author', 'title', 'genre', 'price', 'publish_date',
       'description'],
      dtype='object')
```

Luego hacemos la unión del dataset resultante de merge anterior y lo relacionamos con el Dataframe de catalogos, mediante las líneas de codigo acontinuacion.

```
# Luego de tener el dataset podemos cruzar los dos sets de datos resultantes
dfCompraFull = reduce(lambda x,y: pd.merge(x,y, on='book_id', how='left'), [dfMerged, dfCatalogo])
print(dfCompraFull)
```

Salida

```
     Unnamed: 0  ...                                        description
0             0  ...  After the collapse of a nanotechnology \n     ...
1             1  ...  An in-depth look at creating applications \n  ...
2             2  ...  After the collapse of a nanotechnology \n     ...
3             3  ...  A former architect battles corporate zombies, ...
4             4  ...  When Carla meets Paul at an ornithology \n    ...
..          ...  ...                                               ...
995         995  ...  Microsoft's .NET initiative is explored in \n ...
996         996  ...  The Microsoft MSXML3 parser is covered in \n  ...
997         997  ...  Microsoft Visual Studio 7 is explored in depth...
998         998  ...  Microsoft Visual Studio 7 is explored in depth...
999         999  ...  The Microsoft MSXML3 parser is covered in \n  ...

[1000 rows x 39 columns]
```

Finalmente teniendo el set de datos consolidado con las diferentes fuentes de información procedemos a establecer las preguntas de negocio o que posiblemente pueden surgir para tomar decisiones y que sirvan como insumos para las áreas analíticas.

Consultas de negocio

1. Cual es el nombre de los clientes con las compras más costosas y más economicas.

```
print("****************************************")
print("********** Preguntas de Negocio **********")
print("****************************************")
# Cual es el nombre de de los clientes con las compras mas costosas
dfCompraFull['TotalPrice'] = dfCompraFull.apply(lambda row: (row['price']*row['cuantos_libros']), axis=1)
print(dfCompraFull[['venta_id','first_name','price', 'cuantos_libros', 'TotalPrice']].sort_values(['TotalPrice'], ascending=[False]))
print(dfCompraFull.TotalPrice.max())
```

Salida:

	venta_id	first_name	price	cuantos_libros	TotalPrice
393	394	Benito	49.95	5	249.75
564	565	Roze	49.95	5	249.75
255	256	Grazia	49.95	5	249.75
347	348	Isidor	49.95	5	249.75
352	353	Lamond	49.95	5	249.75
..	...	...	...	...	...
559	560	Sianna	4.95	1	4.95
560	561	Moselle	4.95	1	4.95
60	61	Florella	4.95	1	4.95
63	64	Annissa	4.95	1	4.95
704	705	Jennie	4.95	1	4.95

2. Cual es el país que posee mas usuarios que compran libros?

```
# Cual es el pais con mas usuarios que compran libros
print(dfCompraFull.groupby(['name'])['cuantos_libros'].agg('sum').sort_values(ascending=False).reset_index(name='Cantidad'))
```

Salida:

	name	Cantidad
0	United States of America	1802
1	Canada	598
2	Colombia	299
3	Germany	276

3. El libro más costoso que se vendió en USA.

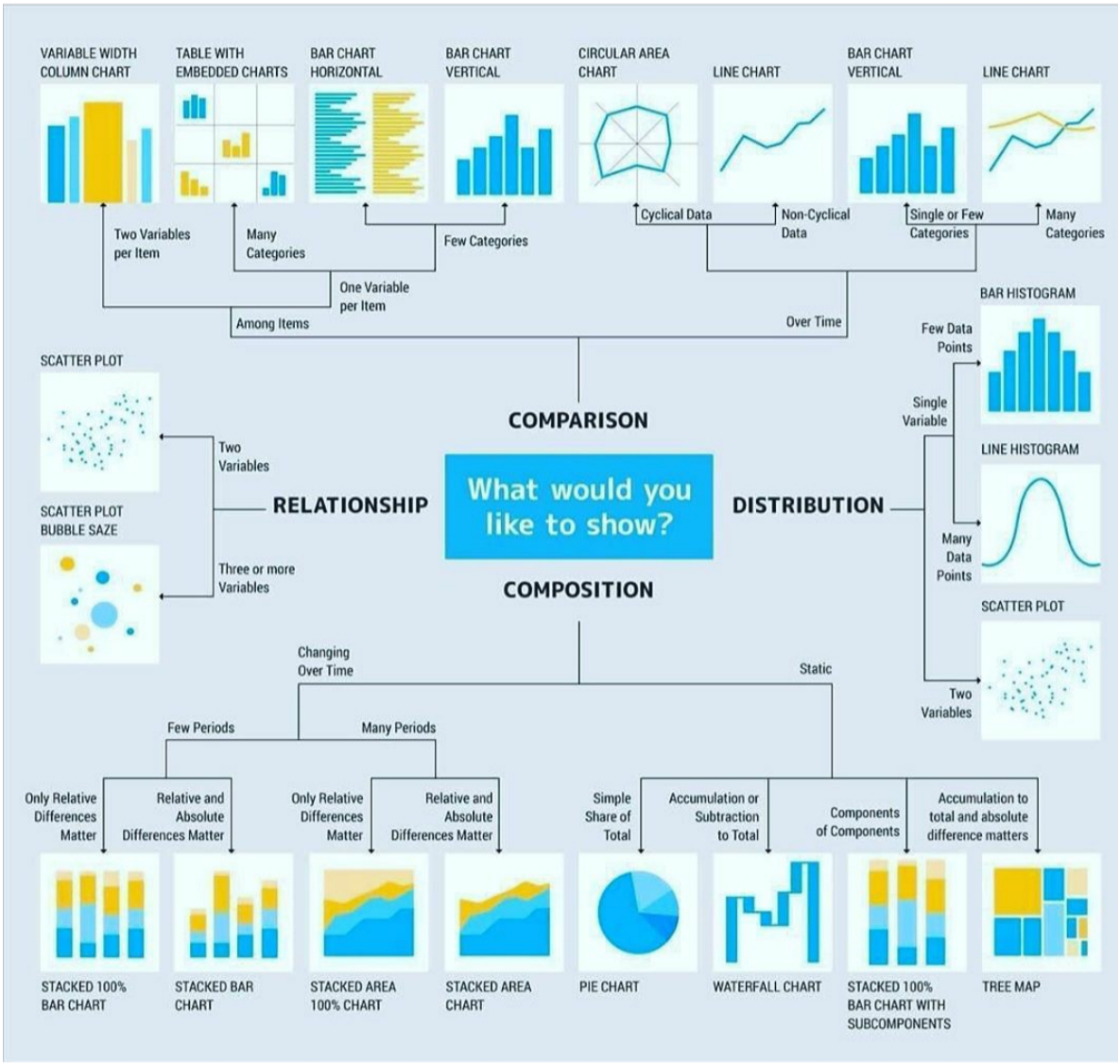
```
# Cual es el libro mas costoso que se vendio en USA
print("El libro mas caro que se vendio en USA fue: ")
dfTest = dfCompraFull[dfCompraFull['alpha3Code'].str.contains('USA', na=False)].sort_values(['price'])
print(dfTest[['title','price']].sort_values(['price'], ascending=[False]))
```

Salida:

```
El libro mas caro que se vendio en USA fue:
                                    title  price
101  Visual Studio 7: A Comprehensive Guide  49.95
```

Mock de Visualizaciones:

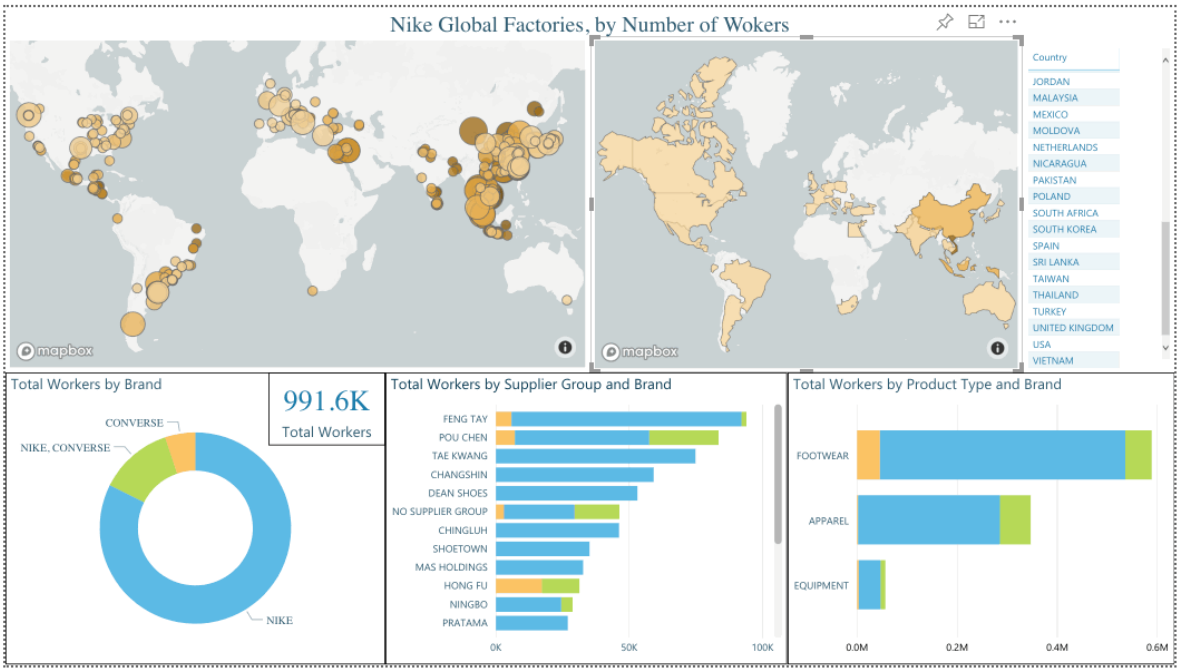
Para los temas de dashboard y reportes que puedo inferir de los datos, principalmente me baso como referencia a esta imagen que me ayuda a identificar el tipo de grafico adecuado para mi necesidad de datos.



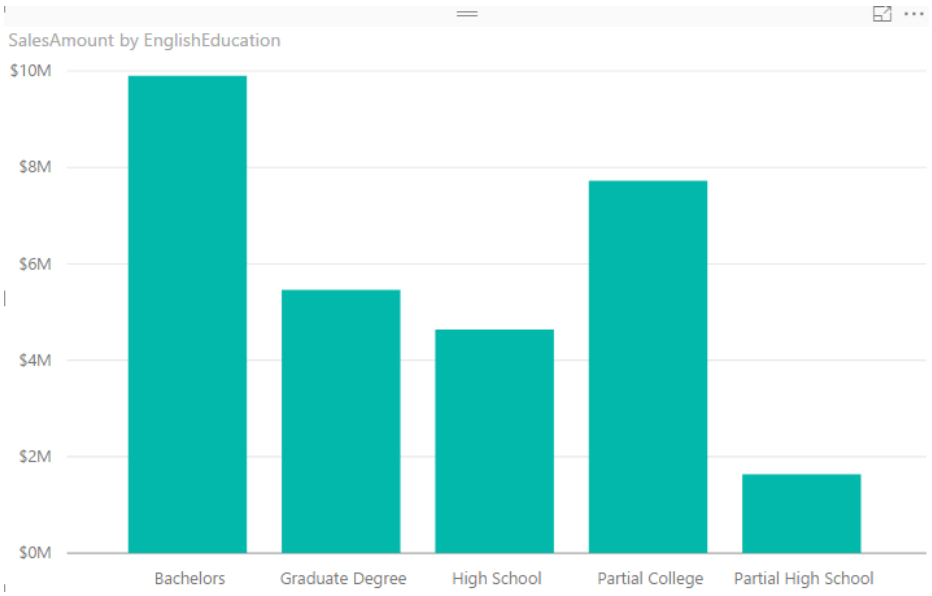
De acuerdo a este grafico determino el tipo de diagrama que mejor me ayuda visualizar mi información.

Como el set de datos hablamos de ventas en países, un gráfico bastante útil que me puede soportar la información que estoy brindando es el diagrama geográfico o un chart Maps,

este me puede brindar información de cantidad de ventas de libros por países, como se puede ver en la siguiente imagen.



Otro grafico bastante interesante es un grafico de barras que me puede ayudar a informar cuales son los géneros de libros que mas les gusta a los lectores. La imagen acontinuacion me puede brindar un apoyo visual de como se puede ilustrar esta información. Imaginando en el eje x los géneros de libros y en el eje Y la cantidad de compras en ese genero.



Visualización de Versionamiento en GitFlow