

GFPI-F-135 REALIZA EL PROCESO DE LIMPIEZA DE DATOS LIMPIEZA DE DATOS CON PYTHON

ACTIVIDADES POR DESARROLLAR:

1. Datos faltantes
2. Columnas irrelevantes (que no responden al problema a solucionar)
3. Filas repetidas
4. Valores extremos o atípicos (outliers)
5. Errores tipográficos.
6. Formatos de fechas
7. Imputación de datos perdidos



Para realizar el taller se debe descargar el DATASET [“COVID19-JULIO2020”](#), que corresponde a los datos del ICFES SABERPRO de 2012 calendario A.

Situaciones para realizar [limpieza en los datos](#):

- Datos faltantes
 - Columnas irrelevantes (que no responden al problema a solucionar)
 - Filas repetidas
 - Valores extremos o atípicos (OUTLIERS)
 - Errores tipográficos.
 - Formatos de fechas
-
- **Remove duplicados o datos irrelevantes:** Tener datos duplicados sucede en la etapa de recolección de datos. Al tener diversas fuentes, se busca juntar los datos, lo que puede resultar en tener duplicados y se deben descartar filas repetidas. Los

datos irrelevantes no influyen o no impactan al problema que se está intentando solucionar.

- **Corregir errores estructurales:** Cuando observan nomenclaturas extrañas, errores tipográficos o gramaticales.
- **Corregir outliers:** o valor atípico es aquel que se está por encima o por debajo del rango normal de valores de la variable que se está estudiando.
- **Imputar:** es el proceso estadístico de sustituir valores faltantes o incorrectos con valores nuevos mediante técnicas de estimación.
 - Eliminación de datos
 - Imputación sin cálculos
 - Imputación básica
 - Usando la librería SKLearn
 - Usando distribuciones
 - Usando interpolación
- **Manejo de datos faltantes:** Son aquellos vacíos de datos en la información recolectada. Existen opciones para tratar los datos faltantes:
 - **Eliminar todos los registros que contengan datos faltantes en algún campo:** Siempre y cuando los registros a eliminar sean mínimos. Si son un porcentaje importante de los datos, más del 10%, es mejor considerar otra alternativa.
 - **Reemplazar los datos faltantes por valores basados en otras observaciones:** Existen varias técnicas para esto (utilizar knn o algoritmo de vecinos cercanos, predecir los datos faltantes, etc.), lo más común es reemplazar por el promedio, o por la mediana (en el caso que el promedio esté sesgado por algunos valores dentro de los datos).
 - Prueba de las correlaciones dicotomizadas: donde se construye una variable dicotomizada asignando cero a los variables ausente en las variables que contengan valores ausentes y el valor de 1 a las presentes; dentro las [variables dicotómicas](#) tenemos entre otras:
 - Género: Masculino o Femenino
 - Coin Flip: cara o cruz
 - Tipo de propiedad: residencial o comercial



- Estado de atleta: profesional o aficionado
- Resultados del examen: aprobado o reprobado
- Si al realizar la correlación y su resultado es no significativa, podemos decir que son aleatorios y se podrá imputar de forma directa o su eliminación siempre y cuando no contengan valores significativos en las demás variables que afecten el objeto de estudio. Se recomienda la lectura de datos perdido en este [enlace](#) en especial el ejemplo de la figura 10-6 de la pagina 339.
- Tenemos algunas imputaciones directas que son el proceso de la estimación de valores ausentes en valores validos en la muestra, tales como:
 - Imputación por las características de la distribución (desviación típica y las correlaciones)
 - Imputación por valores estimados sobre la base de otra información existente en la muestra, por ejemplo, falta el tipo de documento de identidad podemos relacionarla con su fecha de nacimiento.
 - También podemos imputar por:
 - El método de sustitución por la media o mediana
 - El método por un valor constante
 - El método por interpolación lineal
 - El método de imputación por regresión: se utiliza para predecir los valores ausentes en relación con otras variables, este método tiene como desventaja que subestima la varianza de la distribución suponiendo que la variables con datos ausentes tienen correlaciones estanciales con otras variables.
 - El método de imputación múltiple: es la combinación de varios métodos anteriormente mencionados.

Se recomienda realizar la lectura de [Guía Quartz de limpieza de datos](#).

Comandos utilizados en Python

Detección de datos nulos

IsNull: Nos permite detectar datos nulos, simplificando este proceso independientemente de la dimensión de nuestra base de datos.



Notnull: Es la indicación lógica contraria, y la forma de llamar a la función es similar.

Limpieza de Datos

Dropna: Elimina las filas que contienen datos nulos.

Fillna: Rellena los valores nulos con un valor predeterminado.

Bfill: Este método de fillna rellena los datos nulos, con base al valor de la siguiente fila.

- *Ventajas:* Este método es mejor que asignar un valor arbitrario a los datos, además garantiza que los datos se mantengan dentro de un rango específico. Es útil en bases de datos de gran extensión y con poca proporción de datos nulos.
- *Desventajas:* Se debe asegurar que el número de datos nulos NO sea significativo, para que el coeficiente de variación y otras medidas de dispersión no sufran de un sesgo muy grande.

Datos erróneos e irrelevantes.

La reducción de la dimensionalidad produce una representación más compacta y más fácilmente interpretable del concepto de objetivo, centrando la atención del usuario en las variables más relevantes.

Utilizar las siguientes librerías

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Descargar el DATASET, descomprimir, colocar en la variable ruta el lugar donde se descargó el archivo, no olvide de no utilizar “\” sino “/” para construir la ruta.

Cargar el archivo CSV

```
data=pd.read_csv(ruta)
print(data.shape)
```

Realizar el inventario de filas y tipos

```
data.info()
```



```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 400489 entries, 0 to 400488  
Data columns (total 11 columns):  
ID                400489 non-null int64  
FECHA             400489 non-null object  
DIVIPOLA          400489 non-null int64  
CIUDAD            400489 non-null object  
DEPARTAMENTO       400489 non-null object  
ATENCION          399277 non-null object  
EDAD              400489 non-null int64  
SEXO              400489 non-null object  
TIPO              400489 non-null object  
ESTADO            398910 non-null object  
PAIS              920 non-null object  
dtypes: int64(3), object(8)  
memory usage: 33.6+ MB
```

Subniveles de las categorías

```
cols_cat=['PAIS','CIUDAD','SEXO','TIPO','ESTADO','ATENCION','DEPARTAMENTO']  
for col in cols_cat:  
    print(f'Columna {col}: {data[col].nunique()} subniveles')
```

Descubra y arregle el por qué sale un error al ejecutar la sentencia

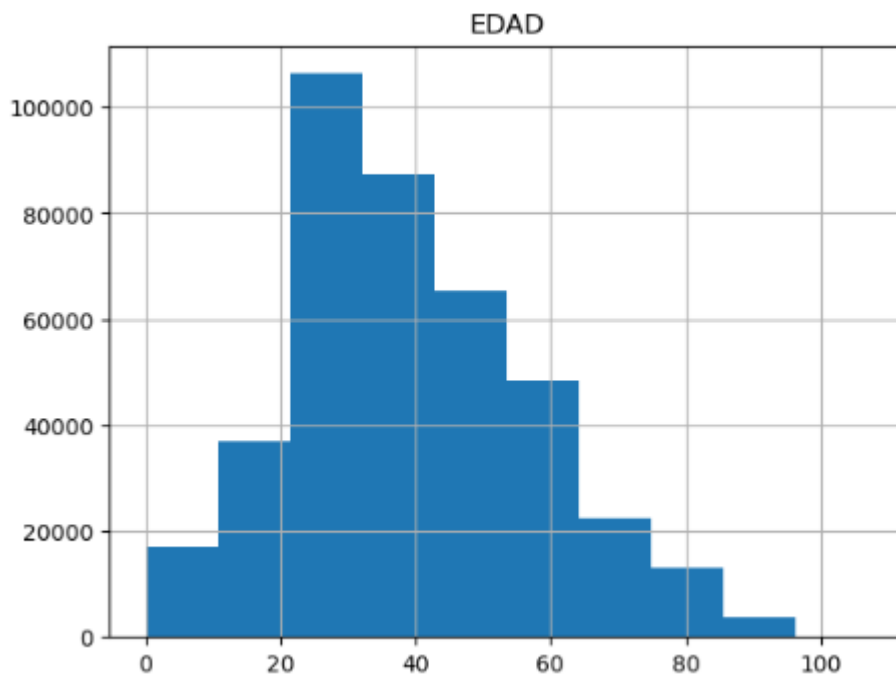
```
Columna PAIS: 49 subniveles  
Columna CIUDAD: 871 subniveles  
Columna SEXO: 4 subniveles  
Columna TIPO: 6 subniveles  
Columna ESTADO: 6 subniveles  
Columna ATENCION: 5 subniveles  
Columna DEPARTAMENTO: 37 subniveles
```

Medidas de tendencias central

```
data.describe()
```

	ID	DIVIPOLA	EDAD
count	400489.000000	400489.000000	400489.000000
mean	207136.763996	24228.489584	39.378647
std	130326.805430	24497.916534	18.054488
min	1.000000	5001.000000	0.000000
25%	100173.000000	8758.000000	27.000000
50%	200302.000000	11001.000000	37.000000
75%	300464.000000	25754.000000	51.000000
max	997851.000000	99524.000000	107.000000

```
data.hist('EDAD')
```



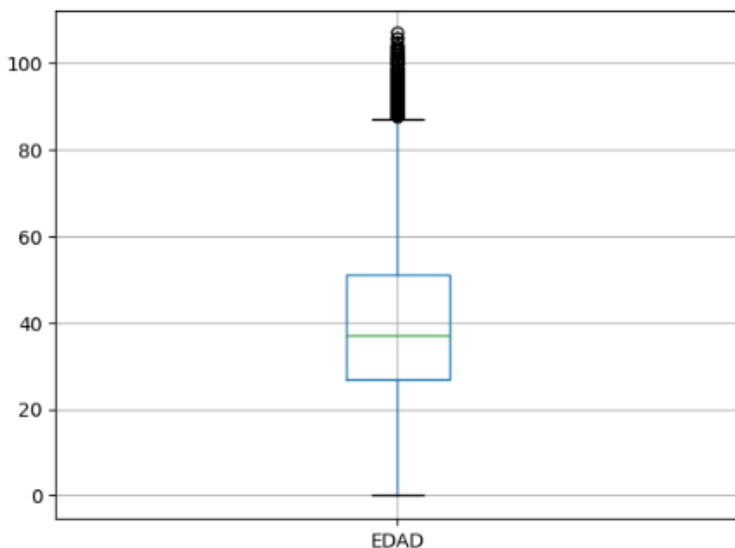
Quitar filas duplicadas

```
print(data.shape)  
data.drop_duplicates(inplace=True)  
print(data.shape)
```

```
(400489, 11)  
(400489, 11)
```

Graficar las cantidades de filas en columnas numéricas, identificar outliers

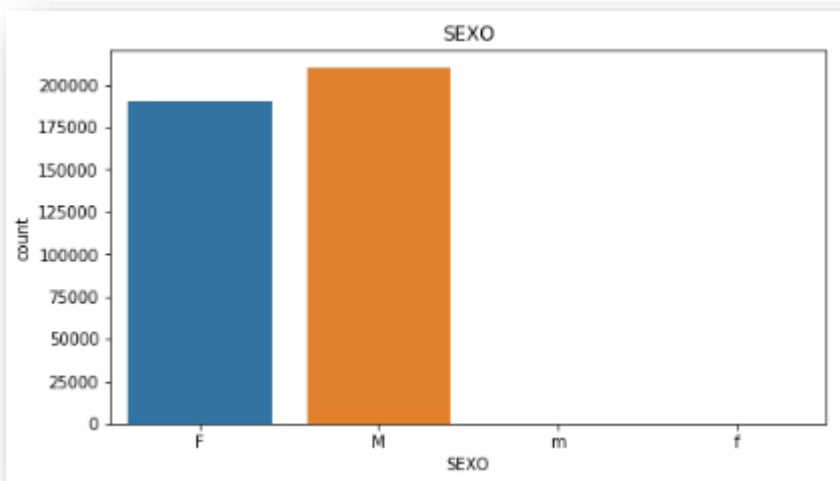
```
boxplot = data.boxplot(column=['EDAD'])
```

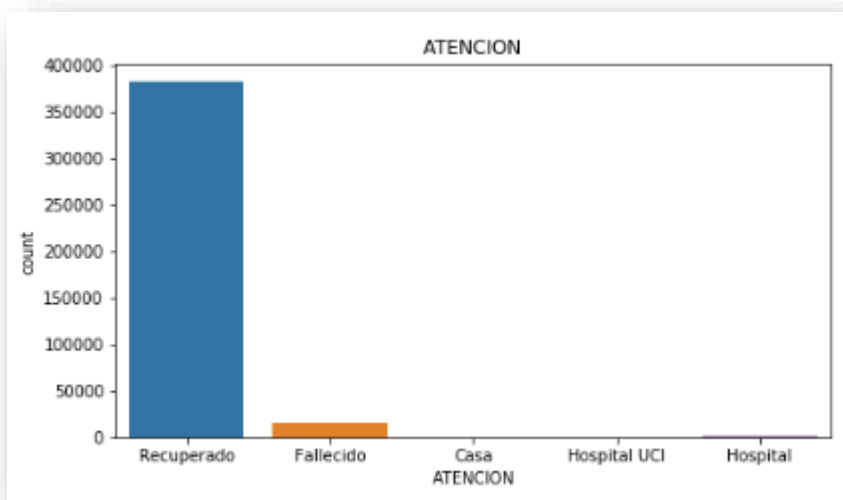


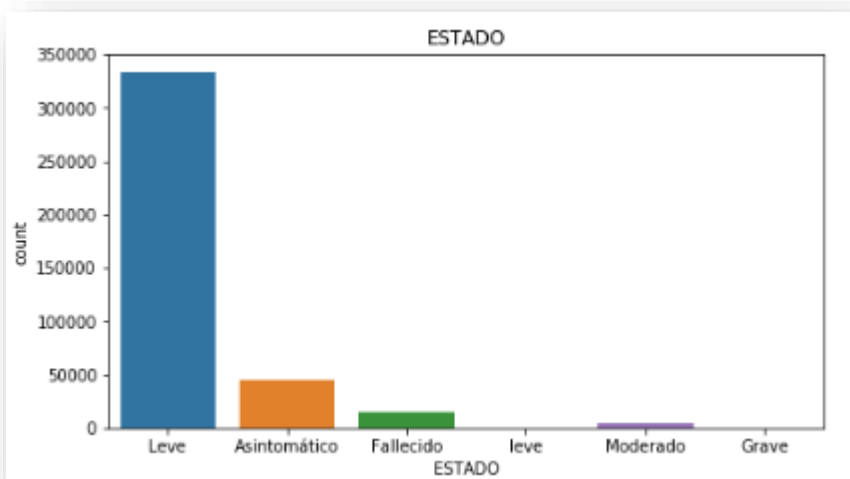
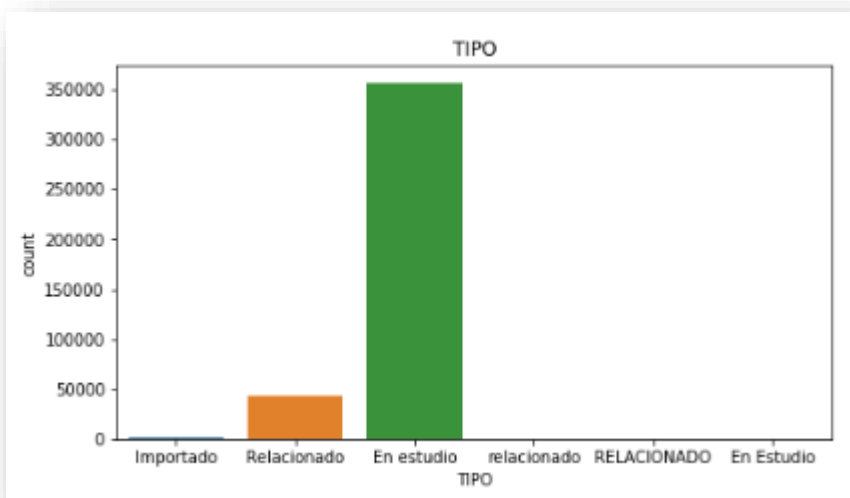
Identificar errores tipográficos:

```
colsnum=['SEXO','ATENCION','TIPO','ESTADO']
fig,aix=plt.subplots(nrows=4,ncols=1,figsize=(8,30))
fig.subplots_adjust(hspace=1)

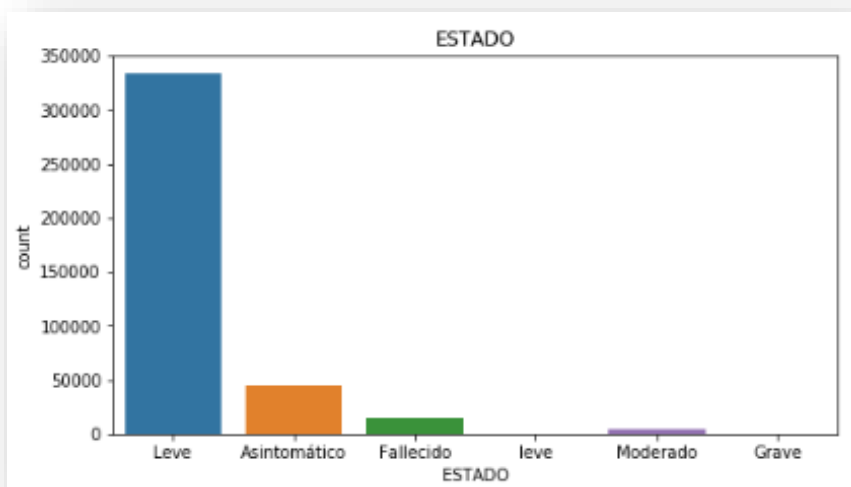
for i,col in enumerate(colsnum):
    aix[i].set_title(col)
    sns.countplot(x=col,data=data,ax=aix[i])
```







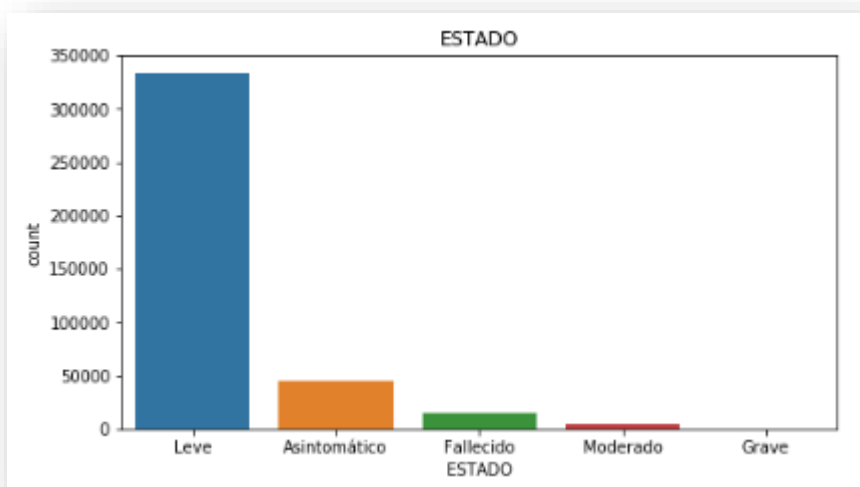
Cambios de errores tipográficos.



En el caso de la variable “Estado”, encontramos valores diferentes que fueron escritos mal, por ejemplo “Leve y leve”.

Para realizar el cambio:

```
print(data['ESTADO'].unique())
data['ESTADO']=data['ESTADO'].str.replace('leve','Leve',regex=False)
print(data['ESTADO'].unique())
```



MODELO ESTRELLA DEL COVID19

Convertir a CSV la dimensión "DM_ESTADO"

```
DA={  
    'NOMBRE':data['ESTADO']  
}  
DM_ESTADO=pd.DataFrame(DA)  
DM_ESTADO.drop_duplicates(inplace=True)  
  
# agregar columna de consecutivo  
DM_ESTADO['IDESTADO'] = range(1, len(DM_ESTADO) + 1)  
  
# establecer columna "IDESTADO" como índice  
DM_ESTADO.set_index('IDESTADO', inplace=True)  
  
DM_ESTADO=DM_ESTADO.fillna('NA')  
# mostrar DataFrame resultante  
print(DM_ESTADO)  
DM_ESTADO.to_csv('C:/borrar/DM_ESTADO.csv')  
print('Guardado')
```

Convertir a CSV la dimensión “DM_TIPO”

```
DA={
  'NOMBRE':data['TIPO']
}
data['TIPO']=data['TIPO'].str.replace('relacionado','Relacionado',regex=False)
data['TIPO']=data['TIPO'].str.replace('RELACIONADO','Relacionado',regex=False)
data['TIPO']=data['TIPO'].str.replace('En Estudio','En estudio',regex=False)

DM_TIPO=pd.DataFrame(DA)
DM_TIPO.drop_duplicates(inplace=True)

# agregar columna de consecutivo
DM_TIPO['IDTIPO'] = range(1, len(DM_TIPO) + 1)

# establecer columna "IDTIPO" como índice
DM_TIPO.set_index('IDTIPO', inplace=True)

DM_TIPO=DM_TIPO.fillna('NA')
# mostrar DataFrame resultante
print(DM_TIPO)
DM_TIPO.to_csv('C:/borrar/DM_TIPO.csv')
print('Guardado')
```

```
      NOMBRE
IDTIPO
1      Importado
2      Relacionado
3      En estudio
4      relacionado
5      RELACIONADO
6      En Estudio
Guardado
```

Convertir a CSV la dimensión "DM_SEXO"

```
data['SEXO']=data['SEXO'].str.replace('m','M',regex=False)
data['SEXO']=data['SEXO'].str.replace('f','F',regex=False)

DM_SEXO=pd.DataFrame(data['SEXO'])
DM_SEXO.drop_duplicates(inplace=True)

# agregar columna de consecutivo
DM_SEXO['IDSEXO'] = range(1, len(DM_SEXO) + 1)

# establecer columna "IDSEXO" como índice
DM_SEXO.set_index('IDSEXO', inplace=True)

DM_SEXO=DM_SEXO.fillna('NA')
# mostrar DataFrame resultante
print(DM_SEXO)
DM_SEXO.to_csv('C:/borrar/DM_SEXO.csv')
print('Guardado')
```

IDSEXO	NOMBRE
1	F
2	M
3	m
4	f

Guardado

Convertir a CSV la dimensión "DM_ATENCION"

```
data['ATENCION']=data['ATENCION'].str.replace('Hospital UCI','UCI',regex=False)
```

```
DM_ATENCION=pd.DataFrame(data['ATENCION'])
```

```
DM_ATENCION.drop_duplicates(inplace=True)
```

```
# agregar columna de consecutivo
```

```
DM_ATENCION['IDATENCION'] = range(1, len(DM_ATENCION) + 1)
```

```
# establecer columna "IDATENCION" como índice
```

```
DM_ATENCION.set_index('IDATENCION', inplace=True)
```

```
DM_ATENCION=DM_ATENCION.fillna('NA')
```

```
# mostrar DataFrame resultante
```

```
print(DM_ATENCION)
```

```
DM_ATENCION.to_csv('C:/borrar/DM_ATENCION.csv')
```

```
print('Guardado')
```

IDATENCION	NOMBRE
1	Recuperado
2	Fallecido
3	NA
4	Casa
5	Hospital UCI
6	Hospital

Guardado



Convertir a CSV la dimensión “DM_DEPARTAMENTO”

```
DA={  
  'IDDPTO':data['DIVIPOLA']/1000,  
  'NOMBRE':data['DEPARTAMENTO']  
}  
  
DM_DEPARTAMENTO=pd.DataFrame(DA)  
DM_DEPARTAMENTO.drop_duplicates(inplace=True)  
DM_DEPARTAMENTO.set_index('IDDPTO', inplace=True)  
  
# mostrar DataFrame resultante  
print(DM_DEPARTAMENTO)  
DM_DEPARTAMENTO.to_csv('C:/borrar/DM_DEPARTAMENTO.csv')  
print('Guardado')
```



Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

IDDP TO	NOMBRE
11	Bogotá D.C.
76	Valle del Cauca
5	Antioquia
13	Cartagena D.T. y C.
41	Huila
50	Meta
66	Risaralda
54	Norte de Santander
17	Caldas
25	Cundinamarca
8	Barranquilla D.E.
68	Santander
63	Quindío
73	Tolima
19	Cauca
47	Santa Marta D.T. y C.
20	Cesar
88	Archipiélago de San Andrés Providencia y Santa...
85	Casanare
52	Nariño
8	Atlántico
15	Boyacá
23	Córdoba
13	Bolívar
70	Sucre
47	Magdalena
44	La Guajira
76	Buenaventura D.E.
27	Chocó
91	Amazonas
18	Caquetá
86	Putumayo
81	Arauca
97	Vaupés
94	Guainía
99	Vichada
95	Guaviare
Guardado	

Convertir a CSV la dimensión “ DM_CIUADAD”

```
DA={
    'IDCIUDAD':data['DIVIPOLA'],
    'NOMBRE':data['CIUDAD']
}

DM_CIUADAD=pd.DataFrame(DA)
DM_CIUADAD.drop_duplicates(inplace=True)
DM_CIUADAD.set_index('IDCIUDAD', inplace=True)

# mostrar DataFrame resultante
print(DM_CIUADAD)
DM_CIUADAD.to_csv('C:/borrar/DM_CIUADAD.csv')

print('Guardado')
```

IDCIUDAD	NOMBRE
11001	Bogotá D.C.
76111	Guadalajara de Buga
5001	Medellín
5360	Itagüí
13001	Cartagena de Indias
...	...
68176	Chima
76616	Riofrío
5036	Angelópolis
50318	Guamal
19785	Sucre

[937 rows x 1 columns]
Guardado

Convertir a CSV la dimensión “ DM_FECHA”

```
DM_FECHA=pd.DataFrame(data['FECHA'])
DM_FECHA.drop_duplicates(inplace=True)

DM_FECHA['IDFECHA'] = range(1, len(DM_FECHA) + 1)
DM_FECHA.set_index('IDFECHA', inplace=True)

# mostrar DataFrame resultante
print(DM_FECHA)
DM_FECHA.to_csv('C:/borrar/DM_FECHA.csv')
```



IDFECHA	FECHA
1	2020-03-02T00:00:00.000
2	2020-03-06T00:00:00.000
3	2020-03-07T00:00:00.000
4	2020-03-09T00:00:00.000
5	2020-03-10T00:00:00.000
...	...
145	2020-07-18T00:00:00.000
146	2020-07-21T00:00:00.000
147	2020-07-22T00:00:00.000
148	2020-07-31T00:00:00.000
149	2020-07-25T00:00:00.000

[149 rows x 1 columns]

Crea la función fn_dimension para encontrar el ID de una dimensión dada

```
def fn_dimension(modelo,clave,valor):  
    x=modelo[modelo[clave]==valor]  
    if len(x)>0:  
        return x.index[0]  
    else:  
        return -1  
print(fn_dimension(DM_ATENCION,'NOMBRE','Casa'))
```

Reemplazar datos de las tablas dimensiones en la tabla de hecho

```
data.columns = data.columns.str.replace('TIPO', 'IDTIPO') # renombrar la columna TIPO por IDTIPO
data['IDTIPO']=data.IDTIPO.str.replace('Importado','1',regex=False) #cambio en valor Importado por 1
data['IDTIPO']=data.IDTIPO.str.replace('Relacionado','2',regex=False) #cambio en Relacionado por 2

data.columns = data.columns.str.replace('ESTADO', 'IDESTADO') #Cambio el nombre de la columna
data.columns = data.columns.str.replace('DIVIPOLA', 'IDCIUDAD') #Cambio el nombre de la columna
data['IDESTADO']=data.IDESTADO.str.replace('Leve','1',regex=False) ) #cambio en Leve por 1
data['IDESTADO']=data.IDESTADO.str.replace('Asintomático','2',regex=False) ) #cambio en 'Asintomático'por 2
data.columns = data.columns.str.replace('ATENCION', 'IDATENCION') #Cambio el nombre de la columna
data['IDATENCION']=data.IDATENCION.str.replace('Recuperado','1',regex=False) ) #cambio Recuperado por 1
data['IDATENCION']=data.IDATENCION.str.replace('Fallecido','2',regex=False) ) #cambio Fallecido por 2
data['IDATENCION']=data.IDATENCION.str.replace('NA','3',regex=False) ) #cambio NA por 3
data['IDATENCION']=data.IDATENCION.str.replace('Casa','4',regex=False) ) #cambio Casa por 4
data['IDATENCION']=data.IDATENCION.str.replace('UCI','5',regex=False) ) #cambio UCI por 5
data['IDATENCION']=data.IDATENCION.str.replace('Hospital','6',regex=False) ) #cambio Hospital por 6

data.set_index('ID', inplace=True) # coloco ID como columna index
data['DEPARTAMENTO']=data['IDCIUDAD']//1000 # tomo los dos dígitos de IDCIUDAD
data.columns = data.columns.str.replace('DEPARTAMENTO', 'IDDPPTO') # renombro la columna

del(data['CIUDAD']) #Borro la columna CIUDAD
data.to_csv('C:/borrar/TH_COVID19.csv') #Creo el archivo CSV
```

Revisamos los subniveles

```
cols_cat=['PAIS','CIUDAD','SEXO','IDTIPO','IDESTADO','IDATENCION','DEPARTAMENTO']
for col in cols_cat:
    print(f'Columna {col}: {data[col].nunique()} subniveles')
```

```
Columna PAIS: 49 subniveles
Columna CIUDAD: 871 subniveles
Columna SEXO: 2 subniveles
Columna IDTIPO: 3 subniveles
Columna IDESTADO: 5 subniveles
Columna IDATENCION: 5 subniveles
Columna DEPARTAMENTO: 37 subniveles
```

Cargo los dataframe

```
data=pd.read_csv('c:/borrar/TH_COVID19.csv')
pais=pd.read_csv('c:/borrar/DM_PAIS.csv')
```

Busco el id del país



```
z1=pais['IDPAIS'] #carga la columna idpais
z2=pais['NOMBRE'] #carga la columna del nombre del pais

for idpais,nombre in zip(z1,z2): # recorro y empaqueto los dos vectores
    data['PAIS']=data.PAIS.str.replace(nombre,str(idpais),regex=False) #reemplazo el nombre del pais con el idpais

data
```

Creamos la tabla de hechos

```
data.columns = data.columns.str.replace('PAIS', 'IDPAIS') # se renombra el pais
data.set_index('ID', inplace=True)
data.to_csv('C:/borrar/TH_COVID19.csv') # escribir el CSV de la tabla de hecho
```

Crear categorías dependiendo de la edad

```
rangos=[0,5,10,18,50,100,110]
nombrer=['A','B','C','D','E','F']
data['GEDAD']=pd.cut(data['EDAD'],rangos,labels=nombrer)
```

Verificar cuantos hay en categoría "A"

```
data[data['GEDAD']=='A'].count()
```

```
IDFECHA      7018
IDCIUDAD      7018
IDDPTO        7018
IDATENCION    7001
EDAD          7018
SEXO          7018
IDTIPO        7018
IDESTADO      6998
IDPAIS        7018
GEDAD         7018
dtype: int64
```

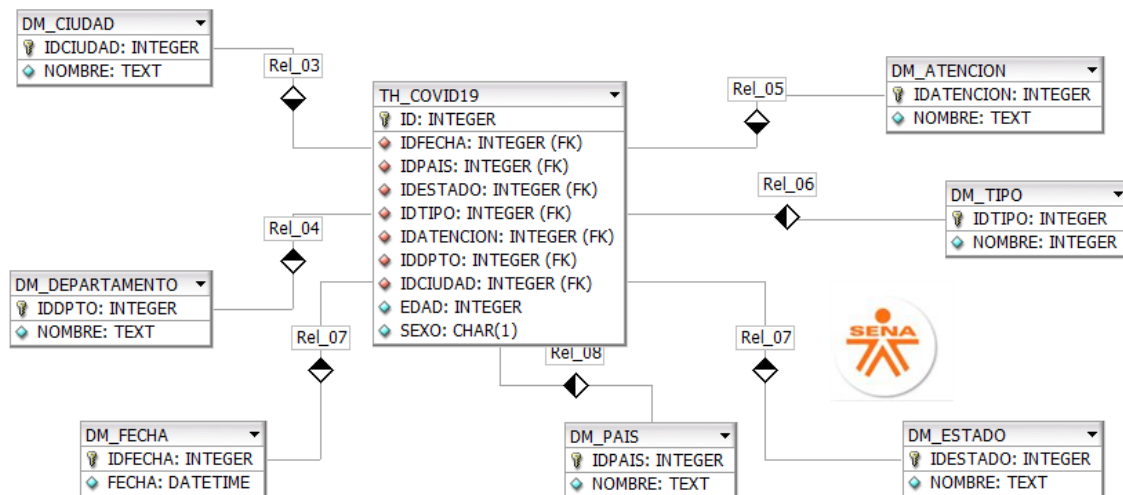
Enlazamos la tabla de hecho con la dimensión "DM_DEPARTAMENTO"

```
mdpto=pd.read_csv('./DM_DEPARTAMENTO.csv')

z1=mdpto['IDDPTO']
z2=mdpto['NOMBRE']

for iddpto,nom in zip(z1,z2):
    data['DEPARTAMENTO']=data.DEPARTAMENTO.str.replace(nom,str(iddpto),regex=False)
data.rename(columns={'FECHA':'IDFECHA','DEPARTAMENTO':'IDDPTO'},inplace=True)
data.to_csv('./TH_COVID19.csv')
```

MODELO ESTRELLA COVID19



Cargar los datos generados en una base de datos MySQL llamada “COVID19”, mediante un programa Python, de acuerdo al modelo relacional presentado.

Creemos la tabla DM_PAIS en la base de datos COVID19 en MySQL

```
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database COVID19;
Query OK, 1 row affected (0.02 sec)
mysql> USE COVID19
Database changed
```



Crear la tabla "DM_PAIS"

```
mysql> CREATE TABLE DM_PAIS( ID_PAIS INT PRIMARY KEY AUTO_INCREMENT, NOMBRE TEXT );  
Query OK, 0 rows affected (0.03 sec)
```

Construir la sentencia LOAD DATA INFILE así:

```
LOAD DATA INFILE 'c:/Borrar/DM_PAIS.csv' INTO TABLE DM_PAIS  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES  
(ID_PAIS,NOMBRE);
```

Verificamos el cargue de los datos

```
select * from DM_PAIS;
```

La actividad a realizar cargue todos las dimensiones y la tabla de hecho como se explico con la tablas DM_PAIS.

```
CREATE TABLE DM_ATENCION(  
IDATENCION INTEGER PRIMARY KEY AUTO_INCREMENT,  
NOMBRE VARCHAR(30)  
);
```

```
LOAD DATA INFILE 'c:/Borrar/DM_ATENCION.csv' INTO TABLE DM_ATENCION  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES  
(IDATENCION,NOMBRE);
```

```
CREATE TABLE DM_TIPO(  
IDTIPO INTEGER PRIMARY KEY AUTO_INCREMENT,  
NOMBRE VARCHAR(30)  
);
```

```
LOAD DATA INFILE 'c:/Borrar/DM_TIPO.csv' INTO TABLE DM_TIPO  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES  
(IDTIPO,NOMBRE);
```



```
CREATE TABLE DM_ESTADO(  
IDESTADO INTEGER PRIMARY KEY AUTO_INCREMENT,  
NOMBRE VARCHAR(30)  
);  
  
LOAD DATA INFILE 'c:/Borrar/DM_ESTADO.csv' INTO TABLE DM_ESTADO  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES  
(IDESTADO,NOMBRE);
```

```
CREATE TABLE DM_FECHA(  
IDFECHA INTEGER PRIMARY KEY AUTO_INCREMENT,  
FECHA DATETIME  
);  
  
LOAD DATA INFILE 'c:/Borrar/DM_FECHA.csv' INTO TABLE DM_FECHA  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES  
(IDFECHA,FECHA);
```

```
CREATE TABLE DM_DEPARTAMENTO(  
IDDPPTO INTEGER PRIMARY KEY,  
NOMBRE TEXT  
);  
  
LOAD DATA INFILE 'c:/Borrar/DM_DEPARTAMENTO.csv' INTO TABLE DM_DEPARTAMENTO  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES  
(IDDPPTO,NOMBRE);
```

```
CREATE TABLE DM_CIUADAD(  
IDCIUADAD INTEGER PRIMARY KEY,  
NOMBRE TEXT  
);
```



```
LOAD DATA INFILE 'c:/Borrar/DM_CIUADAD.csv' INTO TABLE DM_CIUADAD
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(IDCIUDAD,NOMBRE);
```

```
CREATE TABLE if not exists TH_COVID19(
ID INTEGER PRIMARY KEY,
IDFECHA INTEGER,
IDCIUDAD INTEGER,
IDDPTO INTEGER,
IDATENCION INTEGER,
EDAD INTEGER,
SEXO CHAR(1),
IDTIPO INTEGER,
IDESTADO INTEGER,
IDPAIS INTEGER,
GEDAD CHAR(1)
);
```

```
LOAD DATA INFILE 'c:/Borrar/TH_COVID19.csv' INTO TABLE TH_COVID19
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(ID,IDFECHA,IDCIUDAD,IDDPTO,IDATENCION,EDAD,SEXO,IDTIPO,IDESTADO,IDPAIS,GEDAD);
```

Si usted tiene la data limpia, podrá realizar las siguientes llaves foráneas:

```
ALTER TABLE TH_COVID19 ADD CONSTRAINT ESTADOFK FOREIGN KEY(IDESTADO) REFERENCES
DM_ESTADO(IDESTADO);
ALTER TABLE TH_COVID19 ADD CONSTRAINT ATENCIONFK FOREIGN KEY(IDATENCION) REFERENCES
DM_ATENCION(IDATENCION);
ALTER TABLE TH_COVID19 ADD CONSTRAINT CIUDADFK FOREIGN KEY(IDCIUDAD) REFERENCES
DM_CIUADAD(IDCIUDAD);
ALTER TABLE TH_COVID19 ADD CONSTRAINT DEPARTAMENTOFK FOREIGN KEY(IDDPTO) REFERENCES
DM_DEPARTAMENTO(IDDPTO);
ALTER TABLE TH_COVID19 ADD CONSTRAINT TIPOFK FOREIGN KEY(IDTIPO) REFERENCES DM_TIPO(IDTIPO);
```




Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	José Fernando Galindo Suarez	Instructor	CGMLTI- Teleinformática	13/05/2023

CONTROL DE CAMBIOS (diligenciar únicamente si realizan ajustes al taller)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)	José Fernando Galindo Suarez	Instructor	CGMLTI Teleinformática	23/06/2023	Correcciones generales

Autor: José Fernando Galindo Suárez jgalindos@sena.edu.co 2023

