Nombre: Celic Gabriel Hernández Archundia

Matrícula: 2877240

Ejercicios de Avance 2 Evidencia 1

In [164]:
```
import pandas as pd

pd.set_option('display.max_colwidth', None) # Esta función hace que no haya
# un límite en el ancho de la columna al mostrar datos en un DataFrame de pand
# "pd.set_option" es una función de la biblioteca de pandas de Python
# Se utiliza para estableces opciones de configuración globales para la visual

# 'display' se usa para la visualización o presentación de los datos en un DF
# 'max_colwidth' establece el ancho máximo permitido en cada columna
# 'None' hará que pandas muestre todas las columnas en su ancho completo sin t
# Sin el "None", pandas truncará las columnas que excedan ese ancho máximo
# Truncar: En este contexto, cortar o reducir algo a una longitud o tamaño esp
# Display: En este contexto, es la visualización o presentación de los datos e

pd.options.display.float_format = '{:,.2f}'.format # Esto es un formato de vis
# "{:,.2f}" es un especificador de formato para números decimales.
    # ":" Inicia la especificación del formato
    # ",.2f" indica que se desea el número con dos decimales y separador de mi
    # La coma endica la parte de dividir los miles con ella
    # El .2f debe ser la parte de los decimales
```

In [6]:
```
pd.__version__  # Código para saber la versión de pandas
```

Out[6]: '1.4.4'

In [8]:
```
df = pd.read_csv("casosCovid0202.csv") # df -> ahora vale lo que le implementar
# Es decir, la tabla de casosCovid0202 que le mandamos a leer con pandas
df.head() # Imprime los primeros elementos de la tabla
```

Out[8]:

	cve_ent	poblacion	nombre	26-02-2020	27-02-2020	28-02-2020	29-02-2020	01-03-2020	02-03-2020	03-03-2020	...	23-01-2023
0	1	1434635	AGUASCALIENTES	0	0	0	0	0	0	0	...	176
1	2	3634868	BAJA CALIFORNIA	0	0	0	0	0	0	0	...	83
2	3	804708	BAJA CALIFORNIA SUR	0	0	0	0	0	0	0	...	53
3	4	1000617	CAMPECHE	0	0	0	0	0	0	0	...	11
4	7	5730367	CHIAPAS	0	0	0	1	0	0	0	...	5

5 rows × 1075 columns

Pequeño ejercicio para entender los "for" de abajo...

In [10]:
```
df["26-02-2020"] + 1000 # Aquí solo invocamos una columna de la tabla "df"
# Y cuando le sumamos el 1000, se lo sumamos a toda la columna "26-02-2020"
```

Out[10]:
```
0     1000
1     1000
2     1000
3     1000
4     1000
5     1000
6     1000
7     1000
8     1000
9     1000
10    1000
11    1000
12    1000
13    1000
14    1000
15    1000
16    1000
17    1000
18    1000
19    1000
20    1000
21    1000
22    1000
23    1000
24    1000
25    1000
26    1000
27    1000
28    1000
29    1000
30    1000
31    1000
32    1000
Name: 26-02-2020, dtype: int64
```

Variables Generales

In [31]:
```
columnas = list(df.columns) # "columnas" ahora es una lista que contiene todas
# de "df"
dias = columnas[3:] # "dias" ahora vale lo que hay en la variable "columnas"
# Con el [3:] estamos indicando que tome únicamente del 3er elemento de la tab
# adelante
min_dia = dias[0] # "min_dia" Valdrá lo que haya en el primer elemento
# del arreglos dias[]
max_dia = dias[-1] # "min_dia" Valdrá lo que haya en el último elemento
# del arreglos dias[]

#columnas
dias
```

```
 '10-04-2020',
 '11-04-2020',
 '12-04-2020',
 '13-04-2020',
 '14-04-2020',
 '15-04-2020',
 '16-04-2020',
 '17-04-2020',
 '18-04-2020',
 '19-04-2020',
 '20-04-2020',
 '21-04-2020',
 '22-04-2020',
 '23-04-2020',
 '24-04-2020',
 '25-04-2020',
 '26-04-2020',
 '27-04-2020',
 '28-04-2020',
 '29-04-2020',
```

In [15]:
```
meses = [] # Creamos un arreglo llamado "meses"

for dia in dias: # Recorremos el arreglo dias elemento por elemento
    mes = dia[3:] # mes derá igual al elemento de "dia" pero sin contar los pr
    if mes not in meses: # Si lo que valga mes no está en el arreglo meses, qu
        meses.append(mes) # Añade los meses al arreglo meses[]

print(meses)
```

```
['02-2020', '03-2020', '04-2020', '05-2020', '06-2020', '07-2020', '08-2020',
'09-2020', '10-2020', '11-2020', '12-2020', '01-2021', '02-2021', '03-2021',
'04-2021', '05-2021', '06-2021', '07-2021', '08-2021', '09-2021', '10-2021',
'11-2021', '12-2021', '01-2022', '02-2022', '03-2022', '04-2022', '05-2022',
'06-2022', '07-2022', '08-2022', '09-2022', '10-2022', '11-2022', '12-2022',
'01-2023', '02-2023']
```

1. Número de casos confirmados por cada mes

In [18]:
```
zero = [] # Se crea un arreglo meses
for i in range(0, len(meses)): # Por cada elemento en el rango de 0 a la canti
    zero.append(0) # Vamos a agregarle un cero al arreglo zero[]

#zero
# Pero hay una forma más sencilla de hacer lo anterior: Las listas de comprens

zero1 = [0 for x in range(len(meses))]

print(zero)
print(zero1)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [20]:
```
# Vamos a hacer un diccionario
def casos_por_mes():
    zeros = [0 for x in range(len(meses))] # Lo mismo que arriba pero comprimi
    resultados = dict(zip(meses,zeros)) # resultados valdrá lo que haya en el
    # zip sirve para unir dos arreglos separados.
    # dict los convierte en diccionarios

    for dia in dias: # Por cada elemento "dia" dentro del arreglo dias[]
        mes = dia[3:] # la variable "mes" valdrá lo que valga un elemento del
        # Sin tomar en cuenta los primeros 3 caracteres
        resultados[mes] = resultados[mes] + df[dia] # El diccionario "resultad
        # Cuando "resultados" esté en el elemento indicado por la variable "mes
        # la clave "resultados" valdrá dicho mes y su valor será df[dia]
        # Es decir, lo que valga "dia" dentro de la tabla de "df"
        # df es la variable que posee el DataFrame

    return resultados

res = casos_por_mes() # res vale el resultado de la función
#res # Imprime el diccionario
res_df = pd.DataFrame(res) # res_df vale lo que se le proporcione al nuevo obj
# creado con el método pd.DataFrame(), en este caso, el diccionario de "res"

pd.concat([ df["nombre"], res_df], axis = 'columns') # Con el axis lo pegamos
# por columnas, o sea verticalmente
# concat: Sirve para unir dos o más dataframes
```

21	QUERETARO	0	38	139	892	1193	1936	2705	2613	5048
22	QUINTANA ROO	0	113	862	1112	2079	4207	2466	1522	1229
23	SAN LUIS POTOSI	0	44	122	880	2563	8617	7490	4806	5866
24	SINALOA	0	88	1227	3139	5506	4377	3289	2751	2769
25	SONORA	0	37	411	3661	8343	11777	7284	4364	4130
26	TABASCO	0	75	1179	3535	7436	10991	5707	3457	2198
27	TAMAULIPAS	0	24	557	1741	6388	10730	6159	4370	3333
28	TLAXCALA	0	12	253	969	1703	2401	1621	965	710
29	VERACRUZ	0	53	874	4296	7298	11376	6485	4762	3318
30	YUCATAN	0	75	595	1385	2852	5955	4593	3452	2860
31	ZACATECAS	0	25	92	247	716	1906	2545	2153	3929
32	Nacional	8	3122	29650	92470	158886	212274	170819	147468	172147

2. El promedio de casos confirmados por cada mes

In [27]:
```
def prom_casos_por_mes():
    zeros = [0 for x in range(len(meses))]
    resultados = dict(zip(meses, zeros))
    num_dias = dict(zip(meses, zeros)) # num_dias será un diccionario que tenga
    # como clave a los "meses", y como valor los "zeros"

    for dia in dias:
        mes = dia[3:] # la variable "mes" valdrá lo que valga un elemento del d
        # Sin tomar en cuenta los primeros 3 caracteres, o sea la fecha. Ejemp
        resultados[mes] = resultados[mes] + df[dia]
        # Ejemplo: 08-2020: 54
        num_dias[mes] = num_dias[mes] + 1
        # num_dias[mes] accede al valor que posea la clave
        # Con el "+ 1" estamos sumando 1 a los ceros iniciales que estaban dent
        #
    #print(num_dias)


    for key in resultados: # Por cada elemento dentro del diccionario "resultad
        # key toma el valor de una de las claves del diccionario
        resultados[key] = resultados[key] // num_dias[key]
        # resultados[key] mostraría el valor de la clave key
        # resultados[key] valdrá la división del valor de la clave key del
        # diccionario "resultados" entre el valor
        # de la clave key del diccionario "num_dias"
        # Conclusión: Está dividiendo los casos de Covid por día entre el
        # número de casos por mes. Y en este caso estamos omitiendo los decima

    return resultados

res = prom_casos_por_mes()
res_df = pd.DataFrame(res)

pd.concat([ df["nombre"], res_df], axis = 'columns')
```

1

Out[27]:

	nombre	02-2020	03-2020	04-2020	05-2020	06-2020	07-2020	08-2020	09-2020	10-2020	...	05-2022	06-2022
0	AGUASCALIENTES	0	1	8	21	50	58	54	54	91	...	61	102
1	BAJA CALIFORNIA	0	6	80	126	149	141	103	93	100	...	32	216
2	BAJA CALIFORNIA SUR	0	1	10	10	35	98	99	84	75	...	50	268
3	CAMPECHE	0	0	4	19	62	78	30	13	11	...	7	77
4	CHIAPAS	0	0	7	64	93	44	20	13	7	...	3	21
5	CHIHUAHUA	0	1	36	73	59	93	97	118	457	...	14	163
6	DISTRITO FEDERAL	0	28	290	760	809	878	863	992	1221	...	405	3396
7	COAHUILA	0	2	14	36	160	278	264	154	276	...	9	200
8	COLIMA	0	0	0	4	15	46	60	46	43	...	12	88
9	DURANGO	0	0	2	14	57	72	78	88	198	...	5	92
10	GUANAJUATO	0	3	9	54	249	438	350	275	280	...	29	276
11	GUERRERO	0	1	11	76	133	164	117	145	84	...	2	55
12	HIDALGO	0	1	11	55	72	101	114	79	89	...	13	155
13	JALISCO	0	4	11	60	196	227	211	234	239	...	49	293
14	MEXICO	0	16	200	566	721	600	504	487	391	...	77	1107
15	MICHOACAN	0	0	12	61	145	134	184	181	126	...	7	64
16	MORELOS	0	0	17	50	46	37	37	26	31	...	13	61
17	NAYARIT	0	0	3	17	40	58	47	33	21	...	13	99
18	NUEVO LEON	0	3	11	44	205	409	338	372	427	...	74	679
19	OAXACA	0	0	6	62	159	165	100	116	109	...	26	210
20	PUEBLA	0	5	25	98	296	324	207	130	123	...	18	189
21	QUERETARO	0	1	4	28	39	62	87	87	162	...	17	183
22	QUINTANA ROO	0	3	28	35	69	135	79	50	39	...	28	364
23	SAN LUIS POTOSI	0	1	4	28	85	277	241	160	189	...	26	280
24	SINALOA	0	2	40	101	183	141	106	91	89	...	76	621
25	SONORA	0	1	13	118	278	379	234	145	133	...	9	218
26	TABASCO	0	2	39	114	247	354	184	115	70	...	8	158
27	TAMAULIPAS	0	0	18	56	212	346	198	145	107	...	18	246
28	TLAXCALA	0	0	8	31	56	77	52	32	22	...	4	37
29	VERACRUZ	0	1	29	138	243	366	209	158	107	...	34	355
30	YUCATAN	0	2	19	44	95	192	148	115	92	...	56	366
31	ZACATECAS	0	0	3	7	23	61	82	71	126	...	4	50
32	Nacional	2	100	988	2982	5296	6847	5510	4915	5553	...	1212	10701

33 rows × 38 columns

In [23]:
```
# Prueba de función
def prom_casos_por_mes():
    zeros = [0 for x in range(len(meses))]
    resultados = dict(zip(meses, zeros))
    num_dias = dict(zip(meses, zeros))

    for dia in dias:
        mes = dia[3:]
        resultados[mes] = resultados[mes] + df[dia]
        num_dias[mes] = num_dias[mes] + 1

    #for key in resultados:
    #    resultados[key] = resultados[key] // num_dias[key]

    return resultados, num_dias

res, n = prom_casos_por_mes()
print(n)

pd.concat([ df["nombre"], res_df], axis = 'columns')
```

```
{'02-2020': 4, '03-2020': 31, '04-2020': 30, '05-2020': 31, '06-2020': 30,
'07-2020': 31, '08-2020': 31, '09-2020': 30, '10-2020': 31, '11-2020': 30,
'12-2020': 31, '01-2021': 31, '02-2021': 28, '03-2021': 31, '04-2021': 30,
'05-2021': 31, '06-2021': 30, '07-2021': 31, '08-2021': 31, '09-2021': 30,
'10-2021': 31, '11-2021': 30, '12-2021': 31, '01-2022': 31, '02-2022': 28,
'03-2022': 31, '04-2022': 30, '05-2022': 31, '06-2022': 30, '07-2022': 31,
'08-2022': 31, '09-2022': 30, '10-2022': 31, '11-2022': 30, '12-2022': 31,
'01-2023': 31, '02-2023': 1}
```

Out[23]:

	nombre	02-2020	03-2020	04-2020	05-2020	06-2020	07-2020	08-2020	09-2020	10-2020	...	05-2022	06-202
0	AGUASCALIENTES	0	1	8	21	50	58	54	54	91	...	61	10
1	BAJA CALIFORNIA	0	6	80	126	149	141	103	93	100	...	32	21
2	BAJA CALIFORNIA SUR	0	1	10	10	35	98	99	84	75	...	50	26
3	CAMPECHE	0	0	4	19	62	78	30	13	11	...	7	7

3. Una función que muestre qué día de cada mes se presentó el mayor número de casos confirmados

In [345]:
```
df2 = df
df2 = df2.drop([32])
df2 = df2.drop(["cve_ent", "poblacion","nombre"], axis = 1)
#df2
df3 = df["nombre"]
df3 = df3.drop([32])
df4 = pd.concat([ df3, df2], axis = 'columns')
#df4
fechas = pd.DataFrame(df2.head(0))
fechas
```

Out[345]:

26-02-2020	27-02-2020	28-02-2020	29-02-2020	01-03-2020	02-03-2020	03-03-2020	04-03-2020	05-03-2020	06-03-2020	...	23-01-2023	24-01-2023	25-01-2023	26-01-2023	27 01 202

0 rows × 1072 columns

In [453]:
```
diccionario_df2 = df2.to_dict()
diccionario_estados = df4["nombre"]
lista_fechas = df2.columns.tolist()

#diccionario_estados
diccionario_df2
#lista_fechas
```

Out[453]:
```
{'26-02-2020': {0: 0,
  1: 0,
  2: 0,
  3: 0,
  4: 0,
  5: 0,
  6: 0,
  7: 0,
  8: 0,
  9: 0,
  10: 0,
  11: 0,
  12: 0,
  13: 0,
  14: 0,
  15: 0,
  16: 0,
  17: 0,
  18: 0,
```

In [607]:
```
def mayoresCasosDia():
    for fecha in diccionario_df2.values():
        valorMax = max(fecha.values())
        print(valorMax)

    return

mayoresCasosDia()
```

```
0
2
1
1
1
3
3
3
5
4
4
3
2
5
16
23
26
23
10
```

In [606]:
```
# Función
def mayorNumCasos():
    zeros = [0 for x in range(len(meses))]
    resultados = dict(zip(meses, zeros))
    casosMax = 0
    mayorNumCasosDia = []
    mayorNumCasosMes = []
    dias = df2
    i = 1
    num = 0
    nuevoArreglo = []

    for clave, valor in diccionario_df2.items():
        #print("{0} --> {1}".format(clave, valor))
        for estado, casos in valor.items():
            if casos > casosMax:
                casosMax = casos
        mayorNumCasosDia.append(casosMax)
        casosMax = 0

    dia_casos = dict(zip(dias,mayorNumCasosDia))

    for dia in dias: # Por cada elemento "dia" dentro del arreglo dias[]
        mes = dia[3:] # la variable "mes" valdrá lo que valga un elemento del 
        # Sin tomar en cuenta los primeros 3 caracteres
        if mes in dia and dia[3:5] == "02":
            if dia[6:] == "2020":
                #print(i)
                num = i
                #i+=1

    for fecha, valor in diccionario_df2.items():
        #print("{0} --> {1}".format(clave, valor))
        casosMax = 0
        mes = fecha[3:]

        if mes in fecha and fecha[3:5] == "01":
            if fecha[6:] == "2020":
                if casos > casosMax:
                    casosMax = casos


        if mes in fecha and fecha[3:5] == "02":
            if fecha[6:] == "2020":
                if casos > casosMax:
                    casosMax = casos


        if mes in fecha and fecha[3:5] == "03":
            if fecha[6:] == "2020":
                if casos > casosMax:
                    casosMax = casos

        nuevoArreglo.append(casosMax)
```

```
    for fecha, casos in dia_casos.items():
        casosMax = 0
        mes = fecha[3:]
        if mes in fecha and fecha[3:5] == "01":
            if fecha[6:] == "2020":
                if casos > casosMax:
                    casosMax = casos


    mes_casos = dict(zip(meses,nuevoArreglo))
    return mes_casos
#res = mayorNumCasos()
#res_df = pd.DataFrame(res)

#pd.concat([ df3, res_df], axis = 'columns')
mayorNumCasos()
```

Out[606]:

```
{'02-2020': 28,
 '03-2020': 28,
 '04-2020': 28,
 '05-2020': 28,
 '06-2020': 28,
 '07-2020': 28,
 '08-2020': 28,
 '09-2020': 28,
 '10-2020': 28,
 '11-2020': 28,
 '12-2020': 28,
 '01-2021': 28,
 '02-2021': 28,
 '03-2021': 28,
 '04-2021': 28,
 '05-2021': 28,
 '06-2021': 28,
 '07-2021': 28,
 '08-2021': 28,
 '09-2021': 28,
 '10-2021': 28,
 '11-2021': 28,
 '12-2021': 28,
 '01-2022': 28,
 '02-2022': 28,
 '03-2022': 28,
 '04-2022': 28,
 '05-2022': 28,
 '06-2022': 28,
 '07-2022': 28,
 '08-2022': 28,
 '09-2022': 28,
 '10-2022': 28,
 '11-2022': 28,
 '12-2022': 28,
 '01-2023': 0,
 '02-2023': 0}
```

4. Una función que muestre el porcentaje de la población afectada por mes

In [536]:
```
def porcentaje_casos_por_mes():
    zeros = [0 for x in range(len(meses))]
    resultados = dict(zip(meses, zeros))
    num_dias = dict(zip(meses, zeros))
    casosTotales = 0

    for dia in dias:
        mes = dia[3:]
        resultados[mes] = resultados[mes] + df[dia]
        num_dias[mes] = num_dias[mes] + 1

    for clave, valor in diccionario_df2.items():
        for estado, casos in valor.items():
            #print("{0} --> {1}".format(estado, casos))
            casosTotales = casosTotales + casos
    print(casosTotales)


    for key in resultados: # Por cada elemento dentro del diccionario "resulta
        # key toma el valor de una de las claves del diccionario
        resultados[key] = (resultados[key] / casosTotales) *100

    return resultados

res = porcentaje_casos_por_mes()
res_df = pd.DataFrame(res)

pd.concat([ df["nombre"], res_df], axis = 'columns')
```

```
7381305
```

Out[536]:

	nombre	02-2020	03-2020	04-2020	05-2020	06-2020	07-2020	08-2020	09-2020	10-2020	...	05-2022	06-2022
0	AGUASCALIENTES	0.00	0.00	0.00	0.01	0.02	0.02	0.02	0.02	0.04	...	0.03	0.04
1	BAJA CALIFORNIA	0.00	0.00	0.03	0.05	0.06	0.06	0.04	0.04	0.04	...	0.01	0.09
2	BAJA CALIFORNIA SUR	0.00	0.00	0.00	0.00	0.01	0.04	0.04	0.03	0.03	...	0.02	0.11
3	CAMPECHE	0.00	0.00	0.00	0.01	0.03	0.03	0.01	0.01	0.00	...	0.00	0.03
4	CHIAPAS	0.00	0.00	0.00	0.03	0.04	0.02	0.01	0.01	0.00	...	0.00	0.01
5	CHIHUAHUA	0.00	0.00	0.01	0.03	0.02	0.04	0.04	0.05	0.19	...	0.01	0.07
6	DISTRITO FEDERAL	0.00	0.01	0.12	0.32	0.33	0.37	0.36	0.40	0.51	...	0.17	1.38
7	COAHUILA	0.00	0.00	0.01	0.02	0.07	0.12	0.11	0.06	0.12	...	0.00	0.08
8	COLIMA	0.00	0.00	0.00	0.00	0.01	0.02	0.03	0.02	0.02	...	0.01	0.04
9	DURANGO	0.00	0.00	0.00	0.01	0.02	0.03	0.03	0.04	0.08	...	0.00	0.04
10	GUANAJUATO	0.00	0.00	0.00	0.02	0.10	0.18	0.15	0.11	0.12	...	0.01	0.11
11	GUERRERO	0.00	0.00	0.00	0.03	0.05	0.07	0.05	0.06	0.04	...	0.00	0.02
12	HIDALGO	0.00	0.00	0.00	0.02	0.03	0.04	0.05	0.03	0.04	...	0.01	0.06
13	JALISCO	0.00	0.00	0.00	0.03	0.08	0.10	0.09	0.10	0.10	...	0.02	0.12
14	MEXICO	0.00	0.01	0.08	0.24	0.29	0.25	0.21	0.20	0.16	...	0.03	0.45
15	MICHOACAN	0.00	0.00	0.01	0.03	0.06	0.06	0.08	0.07	0.05	...	0.00	0.03
16	MORELOS	0.00	0.00	0.01	0.02	0.02	0.02	0.02	0.01	0.01	...	0.01	0.03
17	NAYARIT	0.00	0.00	0.00	0.01	0.02	0.02	0.02	0.01	0.01	...	0.01	0.04
18	NUEVO LEON	0.00	0.00	0.00	0.02	0.08	0.17	0.14	0.15	0.18	...	0.03	0.28
19	OAXACA	0.00	0.00	0.00	0.03	0.06	0.07	0.04	0.05	0.05	...	0.01	0.09
20	PUEBLA	0.00	0.00	0.01	0.04	0.12	0.14	0.09	0.05	0.05	...	0.01	0.08
21	QUERETARO	0.00	0.00	0.00	0.01	0.02	0.03	0.04	0.04	0.07	...	0.01	0.07
22	QUINTANA ROO	0.00	0.00	0.01	0.02	0.03	0.06	0.03	0.02	0.02	...	0.01	0.15
23	SAN LUIS POTOSI	0.00	0.00	0.00	0.01	0.03	0.12	0.10	0.07	0.08	...	0.01	0.11
24	SINALOA	0.00	0.00	0.02	0.04	0.07	0.06	0.04	0.04	0.04	...	0.03	0.25
25	SONORA	0.00	0.00	0.01	0.05	0.11	0.16	0.10	0.06	0.06	...	0.00	0.09
26	TABASCO	0.00	0.00	0.02	0.05	0.10	0.15	0.08	0.05	0.03	...	0.00	0.06
27	TAMAULIPAS	0.00	0.00	0.01	0.02	0.09	0.15	0.08	0.06	0.05	...	0.01	0.10
28	TLAXCALA	0.00	0.00	0.00	0.01	0.02	0.03	0.02	0.01	0.01	...	0.00	0.02
29	VERACRUZ	0.00	0.00	0.01	0.06	0.10	0.15	0.09	0.06	0.04	...	0.01	0.14
30	YUCATAN	0.00	0.00	0.01	0.02	0.04	0.08	0.06	0.05	0.04	...	0.02	0.15
31	ZACATECAS	0.00	0.00	0.00	0.00	0.01	0.03	0.03	0.03	0.05	...	0.00	0.02
32	Nacional	0.00	0.04	0.40	1.25	2.15	2.88	2.31	2.00	2.33	...	0.51	4.35

33 rows × 38 columns

5. Describe y desarrolla una función que presente alguna otra información relevante de acuerdo con tus opiniones

In [569]:
```
# Función que muestra el mayor número de casos por día en México.
def mayorNumCasosDia():
    zeros = [0 for x in range(len(meses))]
    resultados = dict(zip(meses, zeros))
    casosMax = 0
    mayorNumCasosDia = []
    mayorNumCasosMes = []
    dias = df2
    i = 1
    num = 0
    nuevoArreglo = []

    for clave, valor in diccionario_df2.items():
        #print("{0} --> {1}".format(clave, valor))
        for estado, casos in valor.items():
            if casos > casosMax:
                casosMax = casos
        mayorNumCasosDia.append(casosMax)
        casosMax = 0

    dia_casos = dict(zip(dias,mayorNumCasosDia))

    return dia_casos

res = mayorNumCasosDia()
```

Out[569]:
```
{'26-02-2020': 0,
 '27-02-2020': 2,
 '28-02-2020': 1,
 '29-02-2020': 1,
 '01-03-2020': 1,
 '02-03-2020': 3,
 '03-03-2020': 3,
 '04-03-2020': 3,
 '05-03-2020': 5,
 '06-03-2020': 4,
 '07-03-2020': 4,
 '08-03-2020': 3,
 '09-03-2020': 2,
 '10-03-2020': 5,
 '11-03-2020': 16,
 '12-03-2020': 23,
 '13-03-2020': 26,
 '14-03-2020': 23,
 '15-03-2020': 10,
```

In [588]:
```
# Función que remarca con un color la celda que se desea.
def colorin (x):
    df2 = x.copy()
    df2.loc[:,:] = 'background-color: '
    df2.iloc[1,1] = 'background-color: pink'

    return df2
```

In [589]:
```
df2.style.apply(colorin, axis=None)
```

Out[589]:

	26-02-2020	27-02-2020	28-02-2020	29-02-2020	01-03-2020	02-03-2020	03-03-2020	04-03-2020	05-03-2020	06-03-2020	07-03-2020	08-03-2020	09-03-2020	10-03-2020	11-03-2020
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	2	0	0	0	0	1	3	5	1	2	3	2	3	16
7	0	0	1	0	0	0	1	0	0	0	0	0	0	0	2
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	1	3	3	3	4	4	4	2	1	1	4
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	1	1	1	0	2	2	2	1	0	0	2	1	5	3
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
17	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	1	0	0	0	0	0	4	5
19	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	1	0	1	0	0	0	0	2	2

In [587]:
```
# Función que remarca con color los estados que presentaron 0 casos de infecci
def highlight_cells(value):
    if value == 0:
        return "background-color: yellow"

df2.style.applymap(highlight_cells)
```

Out[587]:

	26-02-2020	27-02-2020	28-02-2020	29-02-2020	01-03-2020	02-03-2020	03-03-2020	04-03-2020	05-03-2020	06-03-2020	07-03-2020	08-03-2020	09-03-2020	10-03-2020	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	2	0	0	0	0	1	3	5	1	2	3	2	3	
7	0	0	1	0	0	0	1	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	