



Instituto Politécnico Nacional

Escuela Superior de Cómputo



## **Práctica 4**

***Proceso de ETL:  
Caso “Precipitación pluvial en CDMX”***

Grupo: 3CV18

Integrantes.  
Cazares Martínez Maximiliano  
Ramos Nieves Adrián

Profesor:  
Roberto Eswart Zagal Flores

## **Introducción**

En la presente práctica se realizará el proceso ETL sobre los datos de la fuente del depósito atmosférico de precipitación pluvial en la CDMX, por lo que se extraerá el contenido de los archivos .xls, teniendo en cuenta el periodo del 2010 al 2019. Para poder analizar dichos datos decidimos que sería mucho más eficiente si utilizamos Python, el cual nos brinda bibliotecas tales como “xlrd” la cual nos permite leer todas las celdas de un documento Excel y manipularlas, del mismo modo la biblioteca “pyodbc” nos ayudará a realizar la conexión con nuestro manejador de base de datos, SQL server. Haciendo uso las bibliotecas previamente mencionadas migraremos todos los datos contenidos en los archivos Excel a nuestra base de datos.

Realizamos los catálogos de la precipitación pluvial donde separamos los indicadores de mediciones, estaciones y la tabla principal en la que tenemos los elementos más relevantes de los archivos. Finalmente, con nuestra base de datos poblada podemos hacer un análisis con la herramienta Tableau y consultas SQL donde extraemos la información más importante para poder ver un análisis más detallado de los datos recabados en este periodo de tiempo.

## **Desarrollo**

### **1. Definición de flujo de trabajo del ETL**

Usamos la biblioteca “xlrd” para extraer los datos de un archivo .xls mediante las funciones predeterminadas de la biblioteca, posteriormente convertimos los datos a nuestra conveniencia, para nuestro caso, utilizamos las biblioteca “xldate\_as\_tuple” y “datetime” para poder convertir nuestro dato de fecha en los archivos .xls a un String y posteriormente en un objeto “datetime” para manipular la fecha y separarla en parámetros tales como el año, mes y día de manera que se puedan visualizar en una variable independiente una de la otra, ya que para el análisis de los datos con la herramienta Tableau sea mucho más fácil buscar los datos ya sea por año, mes o día según la información que requiéramos.

## 2. Catálogos

En las siguientes imágenes se muestran los catálogos creados con su respectivo valor, para poder almacenar los datos provenientes de los archivos Excel.



```
1 create table EstacionesMonitoreo(  
2     Id_estacion int,  
3     Clave_estacion varchar(10),  
4     Nombre_estacion varchar(50),  
5     Delegacion_municipio varchar(50),  
6     Entidad varchar(20)  
7 );
```



```
1 create table Mediciones(  
2     Id_estacion int,  
3     Medicion float,  
4     Fecha varchar(20),  
5     Dia int,  
6     Mes varchar(10),  
7     Anio varchar(5),  
8     DiaSemana varchar(10)  
9 );
```

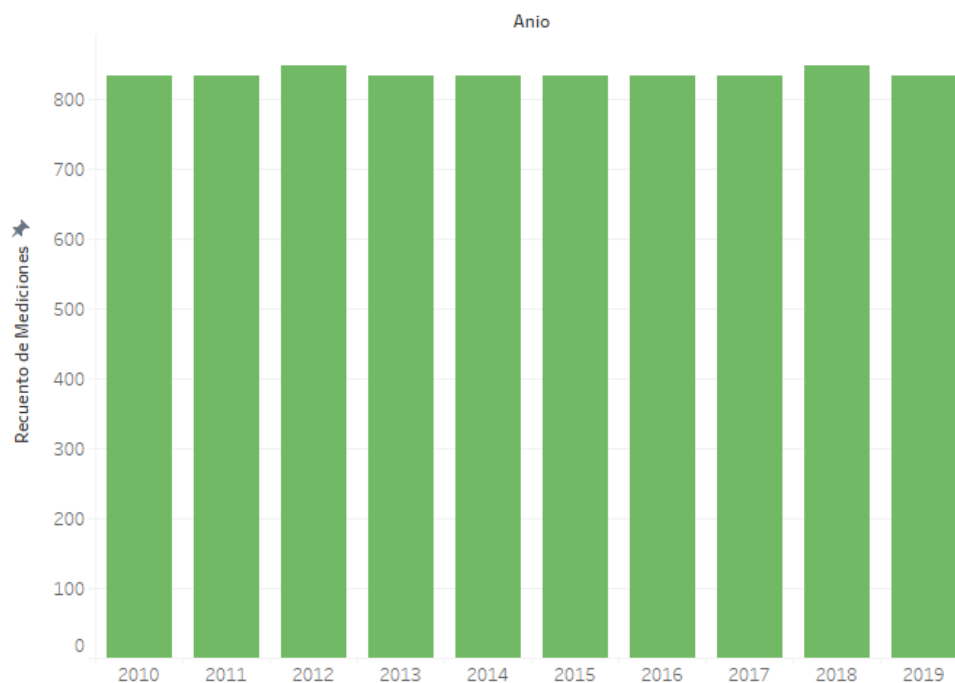


```
1 create table Principal(  
2     Clave_estacion varchar(10),  
3     Localizacion varchar(50),  
4     DiaSemana varchar(10),  
5     Fecha varchar(20),  
6     Medicion float  
7 );
```

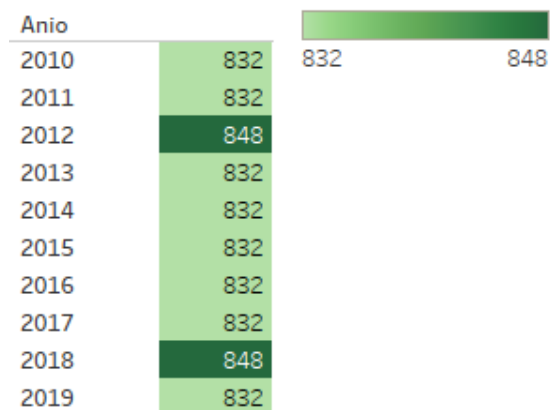
### 3. Exploración de los datos integrados

Las siguientes imágenes muestra la cantidad de registros por año, aquí nos percatamos de que tenemos casi la misma cantidad de registros al año, las variaciones existen debido al número de semanas de cada uno de estos.

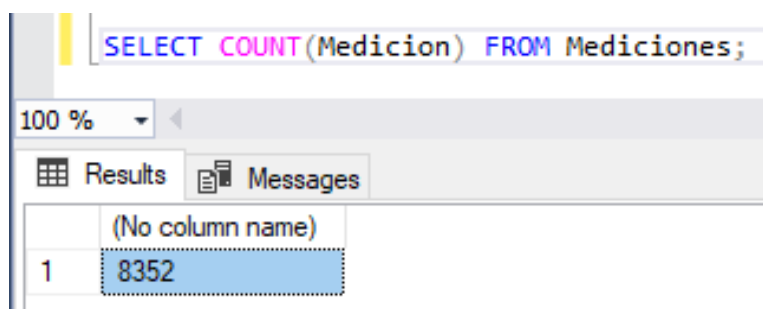
Cantidad de registros por año



Cantidad de registros por año



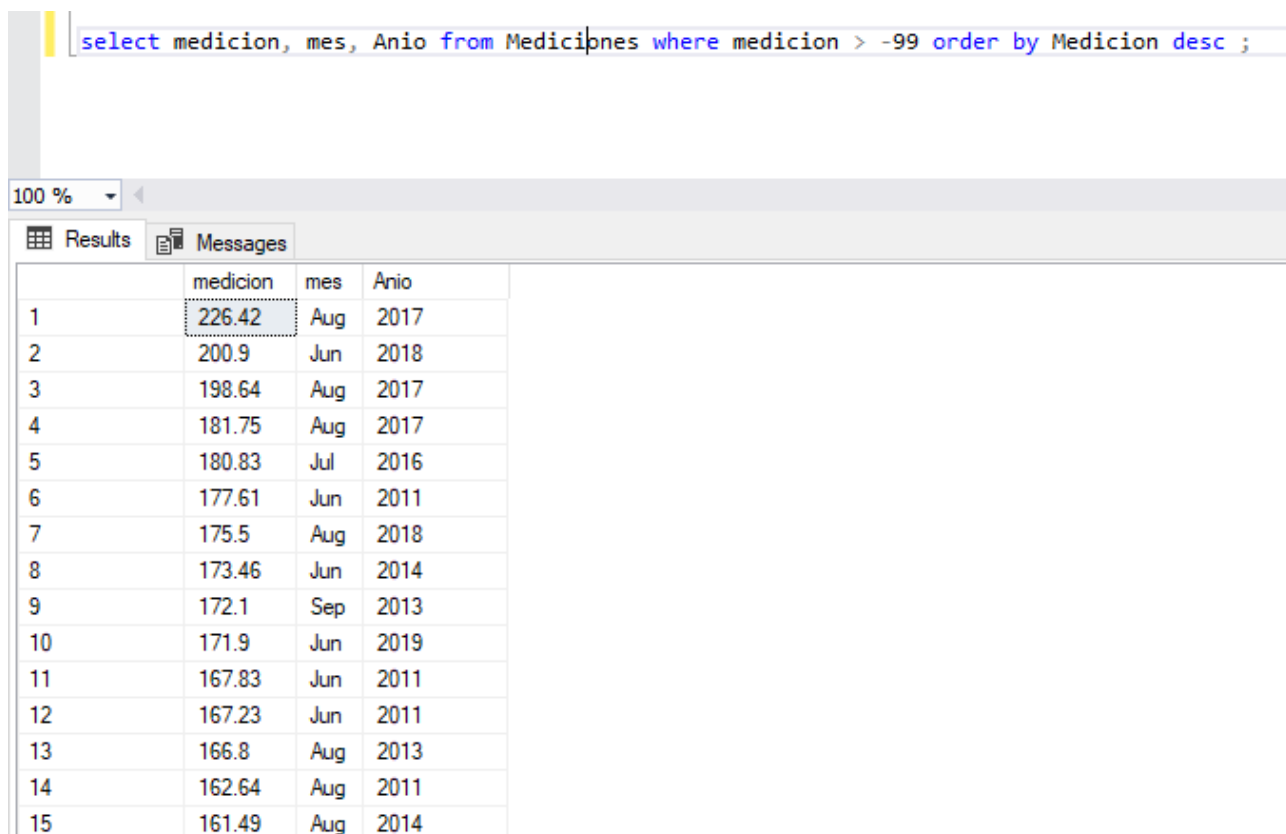
Con una consulta SQL podemos medir la cantidad de mediciones que se hicieron a lo largo del periodo 2010 – 2019, las cuales fueron 8352.



The screenshot shows a SQL query editor with the query: `SELECT COUNT(Medicion) FROM Mediciones;`. Below the query, there are tabs for 'Results' and 'Messages'. The 'Results' tab is active, showing a single row with the value 8352 under the column header '(No column name)'.

	(No column name)
1	8352

En el segundo análisis se encontró mediante una consulta SQL el mes con mayor precipitación en comparación con los demás, en los cuales los primeros puestos se los lleva agosto y junio, algo realmente importante a destacar es que la mayoría de los primeros meses tales como enero, febrero, marzo contienen puros valores vacíos ya que no es común que en esos meses se presentan registros de precipitación, es por eso que no tenemos datos que abarquen esos respectivos meses.



The screenshot shows a SQL query editor with the query: `select medicion, mes, Anio from Mediciones where medicion > -99 order by Medicion desc ;`. Below the query, there are tabs for 'Results' and 'Messages'. The 'Results' tab is active, showing a table with 4 columns: 'medicion', 'mes', and 'Anio'. The table contains 15 rows of data, ordered by the 'medicion' column in descending order.

	medicion	mes	Anio
1	226.42	Aug	2017
2	200.9	Jun	2018
3	198.64	Aug	2017
4	181.75	Aug	2017
5	180.83	Jul	2016
6	177.61	Jun	2011
7	175.5	Aug	2018
8	173.46	Jun	2014
9	172.1	Sep	2013
10	171.9	Jun	2019
11	167.83	Jun	2011
12	167.23	Jun	2011
13	166.8	Aug	2013
14	162.64	Aug	2011
15	161.49	Aug	2014

Por último, tenemos los lugares registrados con mayor valor de precipitación pluvial, en donde nos encontramos con que Tlalpan es el lugar donde más se registran precipitaciones, que haciendo algunos análisis rápidos podemos decir que es un municipio el cual siempre presenta problemas de inundación justamente por este factor de la precipitación pluvial.

```
select EstacionesMonitoreo.Delegacion_municipio, Mediciones.Medicion, Mediciones.Mes, Mediciones.Anio
from Mediciones inner join EstacionesMonitoreo
on Mediciones.Id_estacion = EstacionesMonitoreo.Id_estacion
where Mediciones.medicion > -99 order by Medicion desc
```

	Delegacion_municipio	Medicion	Mes	Anio
1	Tlalpan	226.42	Aug	2017
2	Miguel Hidalgo	200.9	Jun	2018
3	Cuajimalpa de Morelos	198.64	Aug	2017
4	La Magdalena Contreras	181.75	Aug	2017
5	Cuajimalpa de Morelos	180.83	Jul	2016
6	Miguel Hidalgo	177.61	Jun	2011
7	Tlalpan	175.5	Aug	2018
8	Gustavo A. Madero	173.46	Jun	2014
9	Cuajimalpa de Morelos	172.1	Sep	2013
10	Ecatepec de Morelos	171.9	Jun	2019
11	Cuajimalpa de Morelos	167.83	Jun	2011
12	Miguel Hidalgo	167.23	Jun	2011
13	Tlalnepantla de Baz	166.8	Aug	2013
14	Tlalpan	162.64	Aug	2011
15	Miguel Hidalgo	161.49	Aug	2014
16	Miguel Hidalgo	160.3	Jun	2018
17	Tlalpan	158.15	Jun	2011

#### 4. Código

Como primera instancia de nuestro código, creamos un archivo Python con la conexión a la base de datos, la cual importamos a cada archivo .py, que crea una tabla específica. Como primer paso llamamos a la biblioteca “pyodbc”, el cual nos permite utilizar funciones tales como “connect”, que realizará la búsqueda en nuestro manejador para poder conectarse a una base de datos específica mediante el nombre del servidor, el usuario y contraseña. La función mediante un bloque “try – except” nos notificará si ocurre algún error durante la conexión, de lo contrario marcará como conexión exitosa, esta función regresamos nuestra instancia de la conexión para poder usarla en la creación de cada tabla.

```
1 import pyodbc
2
3 def connectionBD():
4     server = 'MAXO'
5     database = 'PPH_2010_2019'
6     user = 'sa'
7     password = 'Sql_Max'
8
9     try:
10        connection = pyodbc.connect('DRIVER={SQL Server};SERVER='+server
11        +';DATABASE='+database+';UID='+user+';PWD='+password)
12        print('connection successfully')
13    except Exception as e:
14        print(f'error during connection: {e}')
15
16    return connection
```

En términos generales para la extracción de los datos ocupamos el mismo algoritmo para todas las tablas, en primer lugar, obtenemos con la función “open\_workbok” una instancia para tener el archivo listo para ser usado, posteriormente obtenemos las hojas que contiene el Excel, para este caso todos nuestros .xls solo tienen una hoja, y mediante la consulta insertaremos los datos previamente obtenidos de los archivos Excel a la base de datos.

```
1 book = xlrd.open_workbook(file)
2 sheet = book.sheet_by_index(0)
3 query = """Insert into
4     Mediciones(Id_estacion, Medicion, Fecha, Dia, Mes, Anio, DiaSemana)
5     values (?, ?, ?, ?, ?, ?, ?);"""
```

Con dos ciclos for, nos moveremos entre filas y columnas respectivamente recorriendo cada uno de los datos, esto con la intención de revisar los if, los cuales se encargan de cargar los datos en la función values, y al final con la instancia de cursor.execute mandaremos esa información más la consulta con los tipos de datos que queremos insertar en esa tupla. Estos if's los ocupamos para tratar los datos de los archivos Excel, ya que dependiendo del nombre de la columna cambiará la ubicación y se identificará con un id a cada una de las entidades.

```
1 for c in range(1, sheet.ncols):
2     for r in range(1, sheet.nrows):
3         date = datetime(*xldate_as_tuple(sheet.cell(r,0).value, 0)).date()
4         if sheet.cell(0,c).value == 'LOM':
5             values = ('LOM', 'Miguel Hidalgo', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
6         elif sheet.cell(0,c).value == 'TEC':
7             values = ('TEC', 'Gustavo A. Madero', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
8         elif sheet.cell(0,c).value == 'DIC':
9             values = ('DIC', 'Tlalpan', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
10        elif sheet.cell(0,c).value == 'MCM':
11            values = ('MCM', 'Cuauhtémoc', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
12        elif sheet.cell(0,c).value == 'TLA':
13            values = ('TLA', 'Tlalnepantla de Baz ', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
14        elif sheet.cell(0,c).value == 'XAL':
15            values = ('XAL', 'Ecatepec de Morelos', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
16        elif sheet.cell(0,c).value == 'EDL':
17            values = ('EDL', 'Cuajimalpa de Morelos', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
18        elif sheet.cell(0,c).value == 'IBM':
19            values = ('IBM', 'Miguel Hidalgo', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
20        elif sheet.cell(0,c).value == 'NEZ':
21            values = ('NEZ', 'Nezahualcóyotl', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
22        elif sheet.cell(0,c).value == 'MON':
23            values = ('MON', 'Texcoco', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
24        elif sheet.cell(0,c).value == 'EAJ':
25            values = ('EAJ', 'Tlalpan', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
26        elif sheet.cell(0,c).value == 'AJU':
27            values = ('AJU', 'Tlalpan', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
28        elif sheet.cell(0,c).value == 'MPA':
29            values = ('MPA', 'Milpa Alta', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
30        elif sheet.cell(0,c).value == 'SNT':
31            values = ('SNT', 'La Magdalena Contreras', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
32        elif sheet.cell(0,c).value == 'COR':
33            values = ('COR', 'Xochimilco', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
34        elif sheet.cell(0,c).value == 'LAA':
35            values = ('LAA', 'Gustavo A. Madero', date.strftime('%A'), date.strftime('%d/%m/%Y'), sheet.cell(r,c).value)
36
37        cursor.execute(query, values)
```



En la parte final solo repetimos este apartado de acuerdo al número de archivos Excel que tenemos almacenados y realizar la inserción de datos correctamente. Por último, solo cerramos el archivo Excel, y también la conexión que hicimos con el manejador de la base de datos.



```
1  try:
2      for i in range(len(books)):
3          FillTable(books[i])
4          print('All works properly')
5  except Exception as e:
6      print(f'Something is wrong: {e}')
7
8  cursor.commit()
9  cursor.close()
10 connection.close()
```

Este proceso se repite para todas las tablas creadas, lo único que cambia es la lógica que se tendrá en los if's para tratar los datos que necesitamos extraer de cada una de las especificaciones de las tablas.

Algunos de los valores atípicos que encontramos, fueron la increíble acumulación de valores nulos en los diferentes Archivos Excel, pero esto se puede justificar con el hecho de que los 3 primeros meses es raro que pueda llegar a presentar, por lo que, aunque se haga un chequeo semanalmente de las mediciones, no encontrarán nada en ese lapso de tiempo, otra irregularidad fue encontrada en los archivos Excel donde existía una fila sin información pero que contenía una fecha igual a la última medición realizada del mes de diciembre, es por eso que se decidió borrarla.

## **5. Conclusiones**

Las bibliotecas que utilizamos en Python son de mucha ayuda para poder migrar datos de un archivo .xls a un manejador de base de datos, esto permite a que, si no tenemos la información presente dentro de una base de datos, sea mucha más fácil insertarlos de manera rápida con código, a realizarlo manualmente registro por registro. Además, aprendimos lo que implica realizar un ETL desde 0, ya que al no utilizar una herramienta como las vistas en clase y tener que realizar a medida la extracción y transformación de los datos, es comprensible por qué es necesario el proceso de ETL para el análisis de diferentes problemáticas solicitadas por un usuario.