Proyecto: Base de Datos

Otoño 2022

Profesor: Felipe López Gamino

Instituto Tecnológico Autónomo de México

Bases de Datos

Miembros del equipo

Díaz Barriga Gómez del Campo Daniel

Juárez Ortiz David Anthoan

Nieto Merodio Javier

Tabla de contenido

1. Introducción

1.1. Definición del problema.

El presente proyecto tiene la intención de elaborar una base de datos relacional de una clínica veterinaria para implementarse en un caso real. El objetivo es que esta base de datos haga consultas sobre los distintos pacientes, consultas, medicamentos y proveedores de la clínica, así como un resgistro óptimo de la información. Para implementarse de forma eficiente y conceptualmente correcta, debe basarse la implementación de dicha base de datos en los conceptos vistos en clase.

El enunciado del problema a realizar (siguiendo el formato de los ejercicios vistos en clase) es el siguiente:

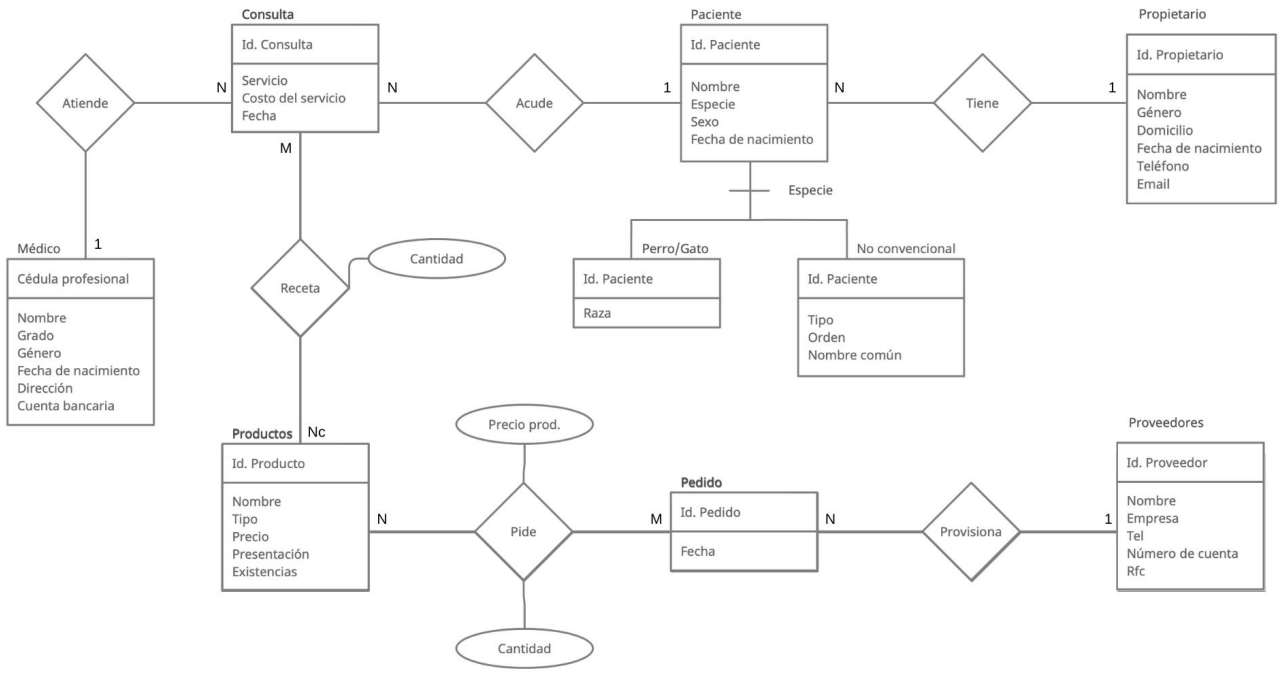
> Una clínica veterinaria desea elaborar un sistema de base de datos para llevar el control de los pacientes, propietarios, personal médico, farmacia con sus medicamentos, materiales y proveedores. El sistema guarda la información de los médicos como: cédula profesional, grado, género, teléfono, fecha de nacimiento, dirección y cuenta bancaria para depositar sueldo.
>
> El sistema registraría a los propietarios de quienes se desea tener el nombre, edad, género, dirección, teléfono, y correo electrónico. Cada propietario puede ser dueño de más de un paciente de los cuales se desea saber el nombre, fecha de nacimiento, sexo y especie; en especial de los perros y gatos se desea saber la raza y en el caso de las mascotas no convencionales se debe obtener el tipo (ave, reptil, mamífero, anfibio o pez), nombre y orden taxonómico.
>
> En el caso de la farmacia debe existir una lista de medicamentos y materiales los cuales deben contar con el nombre, tipo (analgésico, antibiótico, desparasitante, biológicos, material y misceláneos), número de existencias, precio unitario público y presentación (suspensión, tabletas, capsulas, inyectable, polvo).
>
> La información que se pide de los proveedores es el nombre de la empresa, nombre del representante, rfc, teléfono y número de cuenta para transferencia bancaria.
>
> Además, la clínica desea registrar para sus pedidos la fecha en la que fueron realizados. Los pedidos solo tienen un proveedor, pero un proveedor puede atender varios pedidos.

Cada que un paciente acuda a la clínica se registra una consulta donde se encuentra el servicio proporcionado (consulta, revisión, cirugía, vacunación, desparasitación, manejo medico), costo del servicio y fecha, una consulta puede generar el registro de una receta la cual se surte en la clínica con el nombre del medicamento y su cantidad. Cabe aclarar, no todas los servicios generan un receta.

2. Diseño de la solución

2.1. Diagrama de entidad-vínculo

Con base en el enunciado anterior, el modelo entidad vínculo que proponemos es el siguiente:



2.2. Esquema relacional

Finalmente, a partir del modelo entidad vínculo anterior, se derivó el modelo relacional que exponemos a continuación. Este, sigue los estándares vistos en clase respecto a su escritura y orden de aparición de los enunciados. Cabe aclarar, los nombres de las entidades y las relaciones se cambiaron al crear las tablas en PG Admin, para hacer las consultas más sencillas. No obstante, en este proyecto escrito se mantiene la misma consistencia de nombres entre el modelo Entidad

Vínculo y el Modelo Relacional (a excepción de los atributos, que se simplificaron para no ocupar demasiados renglones).

Médico(CedProf, NomMed, Grado, GenMed, FechaMed, DirMed, CuentaBan)

Propietario(IdProp, NomProp, GenProp, Dom, FechaProp, TelProp, EmailProp)

Proveedores(IdProv, NomProd, Emp, TelProd, NumCuenta, RFC)

Productos(IdProd, NomProd, Tipo, Precio, Pres, Exist)

Pedido(IdPed, FechaPed, IdProv(FK))

Consulta(IdCons, Serv, Costo, FechaCons, IdPac(FK), CedProf(FK))

Paciente(IdPac, NomPac, Esp, Sexo, FechaPac, IdProp(FK))

PerroGato(IdPac(FK), Raza)

NoConvencional(IdPac(FK), Tipo, Orden, NomCom)

Receta(IdProd(FK), IdCons(FK), Cant)

Pide(IdPed(FK), IdProd(FK), PrecioPide, CantPide)

3. Consultas (primera parte)

A continuación, se expondrán las consultas que se hicieron en PgAdmin para probar nuestra base de datos ya construida, junto con la explicación de qué regresa cada una.
Consultas en la base de datos PG Admin - Proyecto Bases de Datos

A -- Una que involucre varias tablas y order by:
Para cada propietario, obtener su nombre, el nombre de su mascota, y el costo de cada consulta a la que lo ha llevado.
Ordenar ascendentemente por nombre del propietario y descendientemente por el costo de cada consulta

```
select NomProp, NomPac, Costo
```

```
from Cons c, Prop p, Paciente m
where m.IdProp=p.IdProp and c.IdPac=m.IdPac
order by NomProp asc, Costo desc;
```

B -- Una con manejo de fechas.

Mostrar el nombre del paciente, el servicio, y el médico de todas las consultas del año pasado.

```
select NomPac, NomMed, Serv
From Paciente p, Med m, Cons c
Where p.IdPac = c.IdPac and c.CedProf = m.CedProf
        and extract(year from FechaCons) = extract(year from now()) - 1;
```

C -- Una con not in y subconsulta.

Mostrar el nombre y precio de los productos con precio mayor a 50 que no han estado en un pedido

```
Select NomProd, Precio
From Producto
Where IdProd not in (select IdProd from Pide)
        and Precio<50;
```

D -- Una con intersección.

Mostrar el nombre de la empresa de los proveedores que hayan hecho un pedido de productos tipo ANT y BIO (de ambas).

```
Select Empresa
From Proveedores a, Pedido p, Producto m, Pide pi
Where m.IdProd = pi.IdProd and pi.IdPedido=p.IdPedido  and a.IdProv = p.IdProv
        and TipoProd = 'ANT'
intersect
Select Empresa
```

```
From Proveedores a, Pedido p, Producto m, Pide pi
Where m.IdProd = pi.IdProd and pi.IdPedido=p.IdPedido  and a.IdProv = p.IdProv
        and TipoProd = 'BIO'
```

E -- Una con unión.

Seleccionar el nombre de los propietarios que han pagado una consulta mayor a 500 pesos, o que son hombres (una, otra o ambas)

```
Select NomProp
From Prop p, Paciente m, Cons c
where m.IdProp=p.IdProp and c.IdPac=m.IdPac
        and Costo > 500
Union
Select NomProp
From Prop
Where GenProp = 'M'
```

F -- Una con agrupamiento

Para cada médico, contar cuántas consultas ha dado y mostrar su nombre

```
Select NomMed, count(*)
From Med m, Cons c
Where m.CedProf = c.CedProf
Group by NomMed
```

G -- Una con agrupamiento y having

Mostrar el nombre de cada médico con un promedio general de costo de sus consultas superior a 450 pesos. Mostrar también dicho promedio.

```
Select NomMed, avg(Costo) "Promedio de costo de consultas"
From Med m, Cons c
```

```
Where m.CedProf = c.CedProf
Group by NomMed
Having avg(Costo) > 450
```

H -- Una que contenga where (con al menos una condición que no sea sólo equijunta), group by y having.

Mostrar el nombre de los productos que fueron recetados más de una vez el año pasado, y mostrar cuántas veces se recetó dicho año

```
Select NomProd, count(*) "Número de veces que se recetó"
From Receta r, Cons c, Producto p
Where p.IdProd = r.IdProd and r.IdCons = c.IdCons
        and extract(year from FechaCons) = extract(year from now()) - 1
Group by NomProd
Having count(*)>1
```

I -- Una que contenga el máximo, o mínimo, de un conjunto de grupos (similar, por ejemplo, a una consulta: ciudades que tengan el máximo de agencias).

Mostrar el nombre de los pacientes que más veces han ido a una consulta, así como cuántas veces fueron

```
select nompac, count(nompac)
from paciente p, cons c
where p.idpac = c.idpac
group by nompac
having count(nompac)=(select max(visitas)
                                    from (select idpac, count(idpac) visitas
                                            from cons
                                            group by idpac)as foo)
```

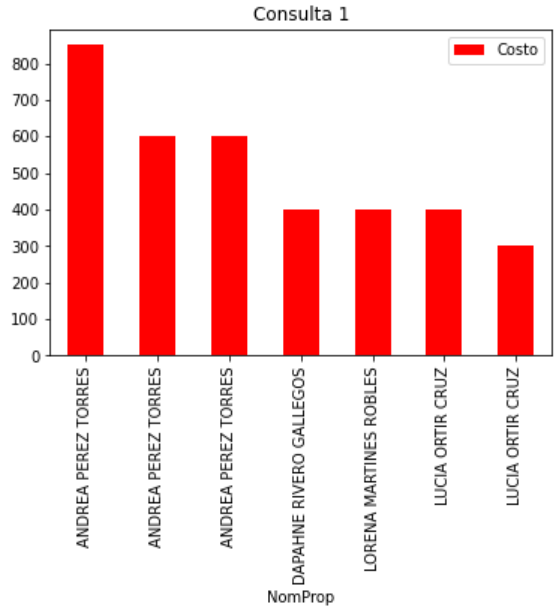
4. Consultas (segunda parte)

Ahora, mostraremos las consultas que hicimos en Python (habiendo conectado SQL a Python) y las gráficas que pudimos hacer con dichos resultados.

Consulta 1: Regresa el nombre de los propietarios, sus mascotas, y el precio por cada consulta a la que fueron

```
cadSql= "select NomProp, NomPac, Costo
                from Cons c, Prop p, Paciente m
                where m.IdProp=p.IdProp and c.IdPac=m.IdPac
                order by NomProp asc, Costo desc"
tuplas1 = bd.cons(conn, cadSql)
df1 = pd.DataFrame.from_records(tuplas1, columns = ['NomProp', 'NomPac', 'Costo'])
#f1[['NomProp', 'Costo']].plot('NomProp', kind = 'bar', title = 'Consulta 1', color = 'r')
```

Gráfica consulta 1: solo graficamos los nombres de los propietarios y los costos por cada consulta

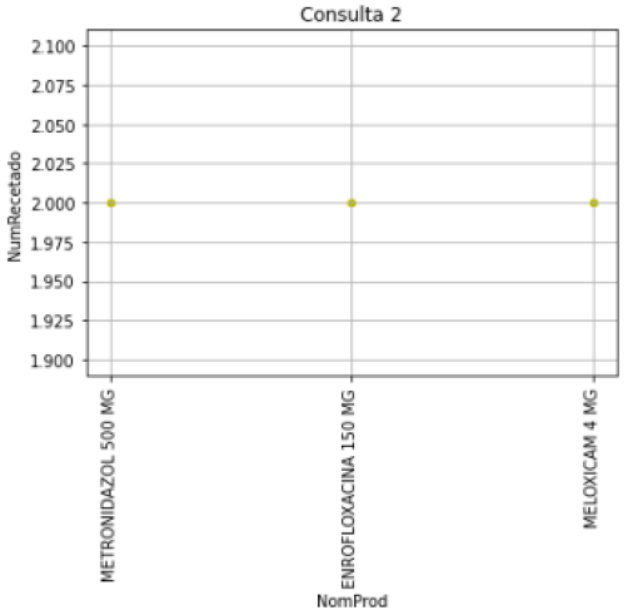


Consulta 2: regresa cuántas veces ha sido recetado cada producto recetado

```
cadSql2 = "Select NomProd, count(*)
        From Receta r, Cons c, Producto p
        Where p.IdProd = r.IdProd and r.IdCons = c.IdCons
```

```
        Group by NomProd
        Having count(*)>1"
tuplas2 = bd.cons(conn, cadSql2)
df2= pd.DataFrame.from_records(tuplas2, columns = ['NomProd', 'NumRecetado'])
#df2.plot(kind = 'scatter', x = 'NomProd', y = 'NumRecetado', rot = 90, color = 'y', grid = True, title = 'Consulta 2')
```

Gráfica consulta 2



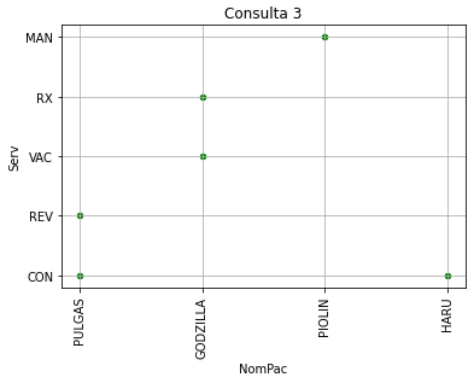
Consulta 3: esta la hicimos en SQL y la exportamos como tabla en csv. Regresa el nombre del paciente y médico, indicando qué servicio se realizó por cada servicio.

	nompac character varying (30)	nommed character varying (30)	serv character varying (3)
1	PULGAS	PEDRO SERRAT LOZA...	CON
2	PULGAS	PEDRO SERRAT LOZA...	REV
3	GODZILLA	AMPARO MOLINA OB...	VAC
4	GODZILLA	FERNANDO CORONA ...	RX
5	PIOLIN	FERNANDO CORONA ...	MAN
6	HARU	AMPARO MOLINA OB...	CON

```
df3 = bib.leeDatosDF('/Users/javi/Desktop/Codigo PFBD/csv/ConsultaB.csv', header = False, nomCols = ['NomPac', 'NomMed', 'Serv'])
```

#df3.plot('NomPac', 'Serv', rot = 90, color = 'g', kind = 'scatter', grid = True, title = 'Consulta 3')

Gráfica consulta 3: excluimos el nombre del médico para que sea más clara la gráfica



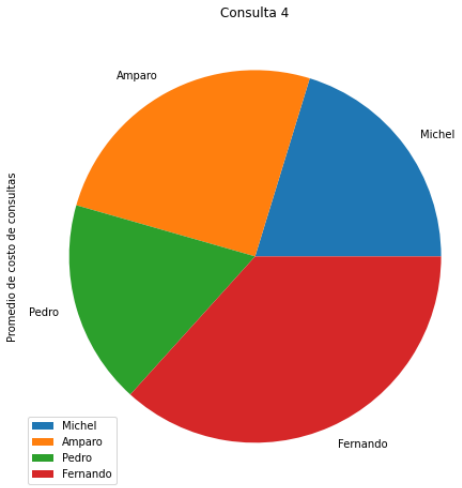
df4 = bib.leeDatosDF('/Users/javi/Desktop/Codigo PFBD/csv/ConsultaG.csv', header = False, nomCols = ['NomMed', 'Promedio de costo de consultas'])

#df4.plot('NomMed', 'Promedio de costo de consultas', rot = 25, kind = 'pie', title = 'Consulta 4', labels = ['Michel','Amparo','Pedro','Fernando'], figsize = (8,8))

Consulta 4: igualmente, la generamos como tabla en PgAdmin y la exportamos como csv. Regresa el nombre de cada médico y el promedio del costo de cada consulta que realizó.

	nommed character varying (30)	Promedio de costo de consultas numeric
1	MICHEL A. LANDERO...	400.0000000000000000
2	AMPARO MOLINA OB...	500.0000000000000000
3	PEDRO SERRAT LOZA...	350.0000000000000000
4	FERNANDO CORONA ...	725.0000000000000000

Gráfica consulta 4



5. Procesos

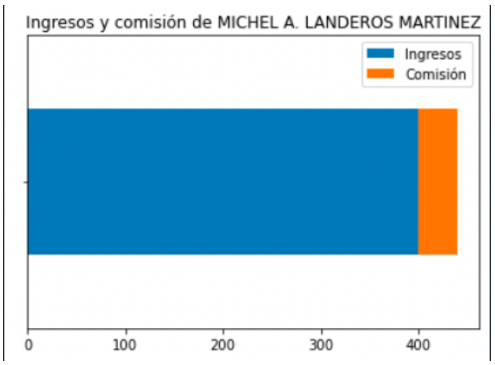
En esta última sección, compartimos los cuatro distintos procesos en Python que hicimos relacionados a consultas con conexión a SQL. Es decir, creamos funciones de Python a las cuales se les puede pasar un parámetro, y estas regresan una gráfica y adicionalmente un resultado impreso.

Proceso 1: Genera comisiones por médico. De tener más de 900 en ingresos, tienen 30% de comisiones. De otra forma, solo 10%. Al final, grafica por cada doctor que se le consulte sus ingresos y la comparación con sus comisiones.

```
def comisiones(nombre):
    cadSqlCom = "select sum(costo) from cons c, med m  where c.cedprof = m.cedprof and nommed
= '" + nombre +"' group by nommed"
    tupCom = bd.cons(conn, cadSqlCom)
    ingresos = tupCom[0][0]
    porc = 0
    if ingresos > 900:
        porc = 0.3
    else:
        porc = 0.1
    comision = ingresos*porc
    ing = [ingresos]
    comi = [comision]
    index = [' ']
    df = pd.DataFrame({'Ingresos': ing,'Comisión': comi}, index=index)
    df.plot.barh(stacked=True, title = "Ingresos y comisión de " + str(nombre))
    return "Nombre: " + nombre + "\nComisión: " + str(comision)

# print(comisiones('MICHEL A. LANDEROS MARTINEZ'))
# print(comisiones('FERNANDO CORONA MONDRAGON '))
```

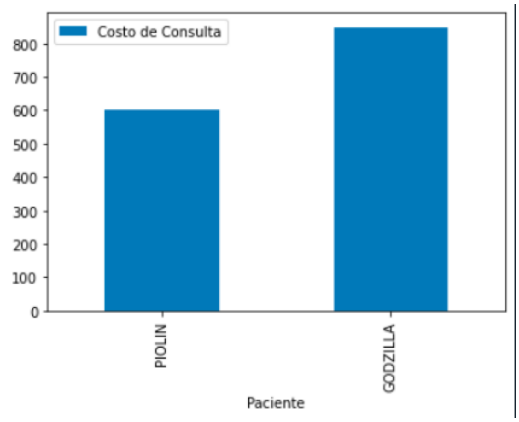
Gráfica de primer proceso: dando como parámetro a Michel



Proceso 2: regresa, dado el nombre de un médico como parámetro, una gráfica que enseña el nombre de cada paciente que tuvo y cuánto costo dicha consulta. El resultado regresado es una lista de listas, conteniendo el nombre del paciente y su respectivo precio.

```
def medPac(medico):
    funCadSql = "select nompac, costo    from paciente p, med m, cons c where p.idpac = c.idpac and c.cedprof = m.cedprof and m.nommed = '"+medico+"';  "
    res = bd.cons(conn, funCadSql)
    if res == []:
        res = "El médico no ha atendido a ningun paciente, no hay grafica"
    else:
        df = pd.DataFrame.from_records(res, columns=['Paciente', 'Costo de Consulta'])
        df.plot('Paciente', kind ='bar')
    return res
```

Gráfica proceso 2



Proceso 3: regresa, dado un nombre de paciente y dos fechas como intervalo de un periodo, cuántas consultas a las que fue estuvieron dentro de ese periodo y cuántas no.

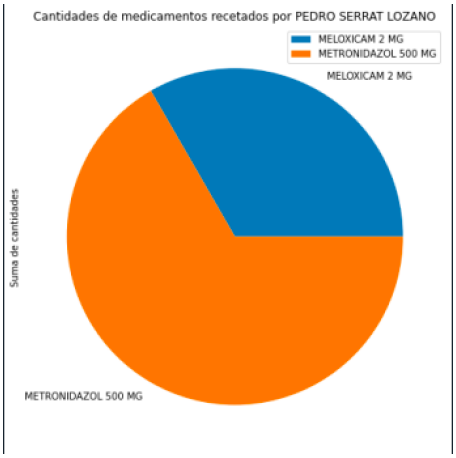
```
def gastoFecha(fechaIni, fechaFin, nom):
    funCadSql = "select sum(costo) from cons c, prop p, paciente pa where c.idpac = pa.idpac and pa.idprop = p.idprop and c.FechaCons between '"+fechaIni +"' and '"+fechaFin+ "' and nomprop = '"+nom+"'"
    res = bd.cons(conn, funCadSql)
    totCadSql = "select sum(costo) from cons c, prop p, paciente pa where c.idpac = pa.idpac and pa.idprop = p.idprop and  nomprop = '"+nom+"'"
    tot = bd.cons(conn, totCadSql)
    por = [tot[0][0] , res[0][0]]
    lista =['Fuera', 'Dentro']
    df = pd.DataFrame({'Periodo': lista, 'Gasto': por})
    df.plot('Periodo', 'Gasto', kind='pie', labels = lista, title="Gastos en el periodo")
    return res[0][0]
```

Gráfica proceso 3: graficamos para gastoFecha('2000-01-01', '2022-01-01', 'LUCIA ORTIR CRUZ').

Proceso 4: el último proceso pide el nombre de un médico, y regresa como resultado la suma de las cantidades que recetó por medicamento.

```
def cant_med(nombre):
    cadSqlCM = "select NomProd, sum(cantidad) from med m, cons c, receta r, producto p where m.cedprof = c.cedprof and c.idcons = r.idcons and r.idprod = p.idprod and m.nommed = '" + nombre + "'  group by nomprod"
    tupCant = bd.cons(conn, cadSqlCM)
    dfcant_med = pd.DataFrame.from_records(tupCant, columns = ['Medicamento', 'Suma de cantidades'])
    lista = dfcant_med['Medicamento'].values
    return dfcant_med.plot('Medicamento', 'Suma de cantidades', kind = 'pie', labels = lista, figsize = (8,8), title = 'Cantidades de medicamentos recetados por ' + str(nombre))
```

Gráfica de proceso 4: al darle el nombre de PEDRO SERRAT LOZANO como parámetro, nos regresa esta gráfica.



6. Conclusiones

Concluyendo el trabajo, queremos compartir que este nos ayudó a ver el impacto que pueden tener las bases de datos. Siendo la base de datos creada una de la clínica veterinaria de Anthoan, pudimos apreciar como nuestro trabajo puede ayudar a las personas a simplificar sus tareas, lo cual fue muy gratificante. Aprendimos mucho sobre Python también; entre la búsqueda de conexiones, navegar por las documentaciones, probar distintos tipos de gráficas con cualquier tipo de parámetros, hizo que nuestra curiosidad nos ayudara a mejorar nuestras habilidades. Por último, creemos que terminamos este proyecto con un gran conocimiento sobre las bases de datos relacionales.