3.2 Clase 5 y 6 - SQL (1)

PostgreSQL:

Una instancia de PostgreSQL es una implementación individual de la base de datos PostgreSQL. Cuando instalas PostgreSQL en tu sistema, se crea automáticamente una instancia de PostgreSQL que se ejecuta en segundo plano y proporciona acceso a una base de datos de PostgreSQL.

La instancia de PostgreSQL es responsable de gestionar los recursos del sistema necesarios para ejecutar la base de datos PostgreSQL, como la memoria, el almacenamiento y la CPU. Además, la instancia de PostgreSQL también gestiona la seguridad y la autenticación de los usuarios, y proporciona acceso a la base de datos a través de una variedad de protocolos, como TCP/IP, Unix sockets y Named Pipes.

En resumen, una instancia de PostgreSQL es el motor que alimenta la base de datos de PostgreSQL y proporciona acceso a los datos almacenados en ella.

PgAdmin: (IU)

Es una herramienta de administración de bases de datos para PostgreSQL que permite a los usuarios interactuar con una base de datos de forma visual. Con PgAdmin, puedes realizar tareas como crear, modificar y eliminar bases de datos, tablas, vistas, usuarios y grupos de usuarios, entre otras.

Además, también ofrece una amplia variedad de herramientas de consulta y análisis de datos, lo que permite a los usuarios escribir y ejecutar consultas SQL complejas y analizar los resultados de forma visual.

En resumen, es una herramienta esencial para cualquier persona que trabaje con bases de datos PostgreSQL, ya que proporciona una forma fácil e intuitiva de administrar y analizar los datos almacenados en la base de datos.

Psycopg2:

Es un adaptador de base de datos PostgreSQL para el lenguaje de programación Python. Con Psycopg2, puedes conectarte a una base de datos PostgreSQL desde una aplicación de Python y realizar tareas como crear tablas, insertar y actualizar datos, y ejecutar consultas SQL.

Es compatible con la mayoría de las versiones de Python y PostgreSQL, lo que lo convierte en una herramienta esencial para cualquier persona que trabaje con DDBB en el contexto de una aplicación de Python.

En resumen, Psycopg2 es una herramienta esencial para cualquier persona que desee conectarse a una base de datos PostgreSQL desde una aplicación de Python y realizar tareas de manipulación de datos(DML).

1. I**nstalamos por medio del prompt: pip install psycopg2**

```
import psycopg2
```

```
conn = psycopg2.connect(
host="localhost",
database="mydatabase",
user="myusername",
password="mypassword",
port="5432"
)
```

Los parámetros necesarios para esta conexión son los siguientes:

- `host`: la dirección del host donde se encuentra la base de datos. Si estás trabajando en tu máquina local, usualmente puedes usar "localhost" o "127.0.0.1".
- `database`: el nombre de la base de datos a la que te quieres conectar.
- `user`: el nombre del usuario que tiene acceso a la base de datos.
- `password`: la contraseña del usuario.
- `port`: el número del puerto en el que se encuentra la base de datos. El valor por defecto para PostgreSQL es el puerto 5432.

DBeaver:

DBeaver es una herramienta de administración de bases de datos que permite a los usuarios interactuar con una variedad de bases de datos relacionales, incluyendo PostgreSQL, MySQL, Oracle y Microsoft SQL Server, entre otras.

Con este manejador de bases de datos puedes realizar tareas como crear, modificar y eliminar bases de datos, tablas, vistas, usuarios y grupos de usuarios, y ejecutar consultas SQL complejas.

DBeaver también ofrece una amplia variedad de herramientas de análisis de datos y visualización, lo que permite a los usuarios analizar los datos almacenados en la base

de datos de forma visual.

En resumen, DBeaver es una herramienta esencial para cualquier persona que trabaje con bases de datos relacionales, ya que proporciona una forma fácil e intuitiva de administrar y analizar los datos almacenados en múltiples bases de datos diferentes.
RDS = DDBB

Instalación de los componentes a utilizar para Crear(**CREATE**), Eliminar(**DELETE**), Insertar(**INSERT**) y actualizar datos(**UPDATE**) en la base de datos que crearemos:

1. Descargar postgres.

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads (**Windows x86-64)**

2. Descargar dbeaver.

https://dbeaver.io/files/dbeaver-ce-latest-x86_64-setup.exe (Para windows)

En **seleccion de componentes** colocar solo:
DBeaver Community
Include Java

3. Abrir PGAdmin.
4. Ingresar contraseñas de logeo.
5. click derecho en Databases > Create > DataBase.
6. Colocar un nombre a la DB.
7. Definir otros patrones de ser necesario y dar click en Save.
8. Click en la base de datos > Schemas > Tables > Click derecho > Query Tool
9. Copiamos los siguientes tablas de creacion(DDL Schema):

```
CREATE TABLE customers(
customerid INT primary key,
name VARCHAR(50),
occupation VARCHAR(50),
email VARCHAR(50),
company VARCHAR(50),
phonenumber VARCHAR(20),
```

```
age INT
);

CREATE TABLE agents(
agentid INT primary key,
name VARCHAR(50)
);

CREATE TABLE calls(
callid INT primary key,
agentid INT,
customerid INT,
pickedup SMALLINT,
duration INT,
productsold SMALLINT
);
```

1. Verificamos que las tablas se hayan creado
2. Ahora procedemos a importar la data en las tablas
3. clieck derecho en la tabla > Import/Export Data > Definir configuracion e importar para cada uno
4. Verificar si existe algun problema en caso de tener lios ir a File > Preferences > Paths > Binary Paths > apuntar carpeta bin correctamente
5. Abrir Dbeaver
6. Nueva conexion > PostgreSQL> Definir host y parametros de conexion
7. Si es primera vez descargar el conector de postgresql: https://jdbc.postgresql.org/download/
8. Driver Settings > Libraries > Añadir Archivo > Buscar archivo descargado > Aceptar
9. Probar conexion y ver si todo sale en orden
10. Podemos ejecutar consultas ya y ver nuestros esquemas y tablas en la DB.

Drive de tablas:

Actividades:

1. Listar los nombres de los clientes y sus respectivas ocupaciones, ordenados alfabéticamente por nombre de cliente.

```
SELECT name, occupation
FROM customers
ORDER BY name;
```

2. Listar los nombres de los agentes y sus respectivos IDs, ordenados alfabéticamente por nombre de agente.

```
SELECT name, agentid
FROM agents
ORDER BY name;
```

4. Contar el número de llamadas realizadas por cada agente.

```
SELECT agentid, COUNT(*) AS total_calls
FROM calls
GROUP BY agentid;
```

5. Obtener la duración promedio de todas las llamadas realizadas.

```
SELECT AVG(duration) AS avg_duration
FROM calls;
```

6. Listar los nombres y edades de los clientes mayores de 30 años que han comprado más de 5 productos.

```
SELECT name, age
FROM customers
WHERE age > 30 AND customerid IN
(SELECT customerid FROM calls WHERE productsold > 5);
```

En este último ejercicio, se utiliza una subconsulta en la cláusula WHERE para obtener los IDs de los clientes que cumplan la condición de haber comprado más de 5 productos, y luego se filtran los nombres y edades de estos clientes de la tabla "customers".