

## M3.2 BD Sql



Ayúdame a crear un programa de Python para conectarse a una MySQL



Claro, puedo ayudarte a crear un programa en Python para conectarse a una base de datos MySQL. A continuación te muestro un ejemplo básico usando la librería `mysql-connector-python`. Primero, asegúrate de tener instalada la librería. Si no la tienes, puedes instalarla con el siguiente comando:

```
bash
```

```
pip install mysql-connector-python
```

Luego, puedes usar el siguiente código para conectarte a tu base de datos MySQL:

```
python
```

```
import mysql.connector
from mysql.connector import Error

def connect_to_mysql(host, user, password, database):
    try:
        # Conectar a la base de datos MySQL
        connection = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )

        if connection.is_connected():
            db_info = connection.get_server_info()
            print(f"Conectado a MySQL Server versión {db_info}")
            cursor = connection.cursor()
            cursor.execute("SELECT DATABASE();")
            record = cursor.fetchone()
            print(f"Conectado a la base de datos: {record[0]}")

            # Aquí puedes agregar más operaciones de base de datos

            return connection

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")
        return None

def close_connection(connection):
    if connection.is_connected():
```

```

        connection.close()
        print("Conexión a MySQL cerrada.")

# Ejemplo de uso:
host = 'localhost'
user = 'tu_usuario'
password = 'tu_contraseña'
database = 'tu_base_de_datos'

connection = connect_to_mysql(host, user, password, database)

# Si necesitas realizar operaciones, aquí es donde lo harías

# Cerrar la conexión al finalizar
close_connection(connection)

```

## Explicación del código:

### 1. Importación de librerías:

- ``mysql.connector`` para manejar la conexión.
- ``Error`` para capturar excepciones relacionadas con MySQL.

### 2. Función ``connect_to_mysql``:

- Recibe los parámetros ``host``, ``user``, ``password`` y ``database``.
- Intenta establecer la conexión a la base de datos con los parámetros proporcionados.
- Si la conexión es exitosa, imprime la versión del servidor y la base de datos conectada.

### 3. Función ``close_connection``:

- Cierra la conexión a la base de datos si está abierta.

### 4. Ejemplo de uso:

- Configura los parámetros de conexión y establece la conexión llamando a ``connect_to_mysql``.
- Realiza cualquier operación adicional antes de cerrar la conexión.

Este es un programa básico para conectarte a una base de datos MySQL. Puedes adaptarlo según tus necesidades, como agregar funciones para realizar consultas, inserciones, actualizaciones, etc. Si necesitas ayuda con algo más específico, no dudes en decirme.



Ahora quiero consultar los registros de la tabla `curso` y que los convierta en formato DataFrame y los exporte a un Excel



Para consultar los registros de la tabla ``curso``, convertirlos en un ``DataFrame`` y exportarlos a un archivo Excel, puedes seguir los pasos que te mostraré a continuación. Usaremos las librerías ``pandas`` para manejar el DataFrame y ``openpyxl`` para exportar a Excel. Primero, asegúrate de tener instaladas las siguientes librerías:

```
bash
```

```
pip install mysql-connector-python pandas openpyxl
```

Luego, el código sería el siguiente:

```
python
```

```
import mysql.connector
import pandas as pd
from mysql.connector import Error

def connect_to_mysql(host, user, password, database):
    try:
        connection = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )

        if connection.is_connected():
            print("Conexión exitosa a la base de datos")
            return connection

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")
        return None

def fetch_data_to_dataframe(connection, query):
    try:
        # Ejecutar la consulta y obtener los datos en un DataFrame
        df = pd.read_sql(query, connection)
        return df
    except Error as e:
        print(f"Error al realizar la consulta: {e}")
        return None

def export_dataframe_to_excel(df, file_name):
    try:
        # Exportar el DataFrame a un archivo Excel
        df.to_excel(file_name, index=False)
        print(f"Datos exportados exitosamente a {file_name}")
    except Exception as e:
        print(f"Error al exportar a Excel: {e}")

def close_connection(connection):
    if connection.is_connected():
        connection.close()
        print("Conexión a MySQL cerrada.")

# Ejemplo de uso:
host = 'localhost'
user = 'tu_usuario'
password = 'tu_contraseña'
database = 'tu_base_de_datos'

# Conectar a la base de datos
connection = connect_to_mysql(host, user, password, database)

if connection:
```

```
# Definir la consulta SQL
query = "SELECT * FROM curso"

# Obtener los datos en un DataFrame
df = fetch_data_to_dataframe(connection, query)

if df is not None:
    # Exportar el DataFrame a un archivo Excel
    export_dataframe_to_excel(df, "curso_data.xlsx")

# Cerrar la conexión a la base de datos
close_connection(connection)
```

## Explicación del código:

### 1. Función `fetch\_data\_to\_dataframe`:

- Ejecuta la consulta SQL pasada como parámetro y convierte los resultados en un `DataFrame` de pandas.

### 2. Función `export\_dataframe\_to\_excel`:

- Toma el `DataFrame` y lo exporta a un archivo Excel utilizando el método `to\_excel`.

### 3. Consulta y Exportación:

- Primero, el código se conecta a la base de datos MySQL.
- Luego, ejecuta la consulta SQL `SELECT \* FROM curso` y obtiene los resultados como un `DataFrame`.
- Finalmente, exporta los resultados a un archivo Excel llamado `curso\_data.xlsx`.

Este código se puede adaptar para cualquier consulta SQL y exportar los resultados a Excel. Si tienes alguna tabla diferente o una consulta más específica, solo necesitas cambiar la consulta en la variable `query`.



Ahora quiero insertar un registro en la tabla curso



Para insertar un registro en la tabla `curso`, puedes utilizar el siguiente código. Este código utiliza la misma conexión a MySQL y la librería `mysql-connector-python` para ejecutar una consulta de inserción.

Aquí te muestro un ejemplo de cómo hacerlo:

```
python
```

```
import mysql.connector
from mysql.connector import Error

def connect_to_mysql(host, user, password, database):
    try:
        connection = mysql.connector.connect(
```

```

        host=host,
        user=user,
        password=password,
        database=database
    )

    if connection.is_connected():
        print("Conexión exitosa a la base de datos")
        return connection

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")
        return None

def insert_record(connection, query, data):
    try:
        cursor = connection.cursor()
        cursor.execute(query, data)
        connection.commit() # Confirmar los cambios en la base de datos
        print("Registro insertado exitosamente")
    except Error as e:
        print(f"Error al insertar el registro: {e}")
        connection.rollback() # Deshacer los cambios en caso de error

def close_connection(connection):
    if connection.is_connected():
        connection.close()
        print("Conexión a MySQL cerrada.")

# Ejemplo de uso:
host = 'localhost'
user = 'tu_usuario'
password = 'tu_contraseña'
database = 'tu_base_de_datos'

# Conectar a la base de datos
connection = connect_to_mysql(host, user, password, database)

if connection:
    # Definir la consulta SQL para insertar un registro
    query = "INSERT INTO curso (nombre, duracion, instructor) VALUES (%s, %s, %s)"

    # Datos del nuevo registro
    data = ('Curso de Python', '3 meses', 'Juan Pérez')

    # Insertar el registro en la base de datos
    insert_record(connection, query, data)

    # Cerrar la conexión a la base de datos
    close_connection(connection)

```

## Explicación del código:

### 1. Función `insert\_record`:

- Recibe la conexión a la base de datos, la consulta SQL de inserción y los datos a insertar.
- Ejecuta la consulta utilizando un cursor.
- Confirma los cambios con `connection.commit()`.
- En caso de error, deshace los cambios con `connection.rollback()`.

### 2. Consulta SQL de inserción:

- La consulta `"INSERT INTO curso (nombre, duracion, instructor) VALUES (%s, %s, %s)"` está diseñada para insertar un nuevo registro en la tabla `curso` con los campos `nombre`, `duracion` e `instructor`.
- Los valores a insertar se pasan como una tupla en el argumento `data`.

### 3. Ejemplo de uso:

- Conecta a la base de datos, define la consulta de inserción y los datos del nuevo registro.
- Llama a la función `insert_record` para insertar el registro en la base de datos.
- Finalmente, cierra la conexión.

Puedes ajustar la consulta y los datos según la estructura de tu tabla y el registro que deseas insertar. Si necesitas ayuda con algo más específico, avísame.



Ahora quiero modificar un registro en la tabla curso



Para modificar un registro en la tabla `curso`, puedes usar una consulta SQL de actualización (`UPDATE`). A continuación, te muestro un ejemplo de cómo hacerlo con Python utilizando `mysql-connector-python`.

Aquí está el código para actualizar un registro en la tabla `curso`:

python

```
import mysql.connector
from mysql.connector import Error

def connect_to_mysql(host, user, password, database):
    try:
        connection = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )

        if connection.is_connected():
            print("Conexión exitosa a la base de datos")
            return connection

    except Error as e:
        print(f"Error al conectar a MySQL: {e}")
        return None

def update_record(connection, query, data):
    try:
        cursor = connection.cursor()
        cursor.execute(query, data)
        connection.commit() # Confirmar los cambios en la base de datos
        print("Registro actualizado exitosamente")
    except Error as e:
        print(f"Error al actualizar el registro: {e}")
        connection.rollback() # Deshacer los cambios en caso de error
```

```
def close_connection(connection):
    if connection.is_connected():
        connection.close()
        print("Conexión a MySQL cerrada.")

# Ejemplo de uso:
host = 'localhost'
user = 'tu_usuario'
password = 'tu_contraseña'
database = 'tu_base_de_datos'

# Conectar a la base de datos
connection = connect_to_mysql(host, user, password, database)

if connection:
    # Definir la consulta SQL para actualizar un registro
    query = "UPDATE curso SET nombre = %s, duracion = %s, instructor = %s WHERE id = %s"

    # Datos del registro a actualizar
    # Aquí, el cuarto valor es el ID del registro que deseas actualizar
    data = ('Curso Avanzado de Python', '4 meses', 'Ana Gómez', 1)

    # Actualizar el registro en la base de datos
    update_record(connection, query, data)

    # Cerrar la conexión a la base de datos
    close_connection(connection)
```

## Explicación del código:

### 1. Función `update\_record`:

- Recibe la conexión a la base de datos, la consulta SQL de actualización y los datos a actualizar.
- Ejecuta la consulta utilizando un cursor.
- Confirma los cambios con `connection.commit()`.
- En caso de error, deshace los cambios con `connection.rollback()`.

### 2. Consulta SQL de actualización:

- La consulta `UPDATE curso SET nombre = %s, duracion = %s, instructor = %s WHERE id = %s` está diseñada para actualizar el registro con el `id` especificado en la tabla `curso`.
- Los valores a actualizar se pasan como una tupla en el argumento `data`. El último valor en la tupla es el `id` del registro que deseas actualizar.

### 3. Ejemplo de uso:

- Conecta a la base de datos, define la consulta de actualización y los datos del registro a modificar.
- Llama a la función `update\_record` para actualizar el registro en la base de datos.
- Finalmente, cierra la conexión.

Este código actualizará el registro con el `id` proporcionado. Asegúrate de ajustar la consulta y los datos según la estructura de tu tabla y el registro que deseas modificar. Si necesitas ayuda con algo más específico, no dudes en preguntar.

ChatGPT can make mistakes. Check important info.