

Conexión MySQL Python



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



Claro, te puedo ayudar con eso. Para conectarte a una base de datos MySQL en Python, necesitarás utilizar un conector de MySQL. Uno de los más comunes es `mysql-connector-python`. Aquí tienes un ejemplo básico de cómo puedes conectarte a una base de datos MySQL:

1. Instalar el conector:

Primero, necesitas instalar el paquete `mysql-connector-python`. Puedes hacerlo usando pip:

```
bash
```

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

2. Escribir el código para conectarte a la base de datos:

Aquí hay un ejemplo de cómo conectarte a una base de datos MySQL, ejecutar una consulta y cerrar la conexión:

```
python
```

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

def connect_to_database():
    try:
        # Establecer la conexión
        connection = mysql.connector.connect(
            host='localhost',          # Cambia esto por tu host
            database='test_db',        # Cambia esto por tu nombre de base de datos
            user='root',               # Cambia esto por tu nombre de usuario
            password='password'        # Cambia esto por tu contraseña
        )

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

            # Crear un cursor para ejecutar consultas
            cursor = connection.cursor()
            cursor.execute('SELECT * FROM tu_tabla') # Cambia esto por tu consulta

            # Obtener los resultados
            rows = cursor.fetchall()
            for row in rows:
                print(row)
```

```
except Error as e:
    print(f'Error al conectar a MySQL: {e}')

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print('Conexión cerrada')

if __name__ == '__main__':
    connect_to_database()
```

Asegúrate de ajustar los valores para `host`, `database`, `user`, `password` y la consulta SQL según tu configuración.

3. Ejecutar el script:

Guarda el código en un archivo `.py` y ejecútalo. Si todo está configurado correctamente, deberías ver los resultados de tu consulta y mensajes de estado sobre la conexión.

Si tienes alguna pregunta o necesitas ayuda adicional con los detalles, ¡dime!



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



Memoria actualizada

Para consultar los registros de la tabla `curso`, convertirlos en un DataFrame y exportarlos a un archivo Excel, puedes usar `pandas` además del conector MySQL. Aquí tienes un ejemplo de cómo hacerlo:

1. Instalar las dependencias necesarias:

Necesitarás `pandas` y `openpyxl` para manejar los DataFrames y exportar a Excel. Puedes instalarlos con pip:

```
bash
```

```
pip install pandas openpyxl
```

2. Escribir el código para consultar la tabla y exportar a Excel:

```
python
```

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

```
def query_to_dataframe():
    try:
        # Establecer la conexión
        connection = mysql.connector.connect(
            host='localhost',      # Cambia esto por tu host
            database='test_db',    # Cambia esto por tu nombre de base de datos
            user='root',           # Cambia esto por tu nombre de usuario
            password='password'    # Cambia esto por tu contraseña
        )

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

            # Crear un cursor para ejecutar consultas
            query = 'SELECT * FROM curso' # Cambia esto por tu consulta
            df = pd.read_sql(query, connection)

            # Exportar a Excel
            df.to_excel('curso_registros.xlsx', index=False, engine='openpyxl')
            print('Datos exportados a Excel con éxito')

        except Error as e:
            print(f'Error al conectar a MySQL: {e}')

        finally:
            if connection.is_connected():
                connection.close()
                print('Conexión cerrada')

    if __name__ == '__main__':
        query_to_dataframe()
```

En este código:

- Utilizamos `pandas` para leer los datos directamente desde la consulta SQL.
- Exportamos el DataFrame a un archivo Excel usando `to_excel()`. El parámetro `index=False` se utiliza para no incluir el índice del DataFrame en el archivo Excel.
- Asegúrate de ajustar el `host`, `database`, `user`, `password`, y la consulta SQL según tu configuración.

3. Ejecutar el script:

Guarda el código en un archivo `.py` y ejecútalo. El script creará un archivo llamado `curso_registros.xlsx` en el mismo directorio que el script, que contendrá los registros de la tabla `curso`.

Si necesitas ajustar algo o tienes más preguntas, ¡déjame saber!



Escribe un código en Python de la tabla anterior que inserte 10 registros diferentes donde los campos son los siguientes "idCurso" que es auto incrementable es decimal, "nombreDescriptivo" es de tipo `tinytext` y "nAsignaturas" es de tipo decimal de tres posiciones



Claro, te mostraré cómo puedes insertar 10 registros diferentes en la tabla `curso` utilizando Python y `mysql-connector-python`. El código incluirá la conexión a la base de datos y la inserción de los registros.

1. Escribir el código para insertar los registros:

python

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

def insert_records():
    try:
        # Establecer la conexión
        connection = mysql.connector.connect(
            host='localhost',      # Cambia esto por tu host
            database='test_db',    # Cambia esto por tu nombre de base de datos
            user='root',          # Cambia esto por tu nombre de usuario
            password='password'    # Cambia esto por tu contraseña
        )

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

            # Crear un cursor para ejecutar consultas
            cursor = connection.cursor()

            # Datos para insertar
            cursos = [
                ("Curso A", 3),
                ("Curso B", 4),
                ("Curso C", 2),
                ("Curso D", 5),
                ("Curso E", 6),
                ("Curso F", 3),
                ("Curso G", 4),
                ("Curso H", 5),
                ("Curso I", 2),
                ("Curso J", 6)
            ]

            # Consulta SQL para insertar datos
            insert_query = """
            INSERT INTO curso (nombreDescriptivo, nAsignaturas)
            VALUES (%s, %s)
            """

            # Ejecutar la inserción de cada registro
            cursor.executemany(insert_query, cursos)
            connection.commit()
            print('10 registros insertados con éxito')

    except Error as e:
        print(f'Error al conectar a MySQL o al insertar datos: {e}')

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print('Conexión cerrada')
```

```
if __name__ == '__main__':
    insert_records()
```

En este código:

- **Conexión:** Establecemos la conexión a la base de datos MySQL.
- **Datos para insertar:** Definimos una lista de tuplas con los datos que queremos insertar en la tabla `curso`. Aquí, cada tupla contiene el `nombreDescriptivo` y `nAsignaturas`.
- **Consulta SQL:** Preparamos una consulta SQL para insertar los registros. Usamos `%s` como marcador de posición para los valores que serán insertados.
- **executemany:** Utilizamos `executemany` para ejecutar la consulta SQL con todos los datos de una vez.
- **Commit:** Llamamos a `connection.commit()` para guardar los cambios en la base de datos.

Asegúrate de ajustar `host`, `database`, `user`, y `password` según tu configuración, y verifica que la tabla `curso` tenga los campos `idCurso`, `nombreDescriptivo`, y `nAsignaturas` como se especifica en tu esquema.

Si necesitas más ayuda o tienes alguna pregunta, ¡avísame!



Escribe un código en Python de la tabla anterior que inserte 10 registros diferentes donde los campos son los siguientes, "nombreDescriptivo" es de tipo tinytext y "nAsignaturas" es de tipo decimal de tres posiciones



Claro, aquí tienes un código en Python que inserta 10 registros en la tabla `curso`, donde los campos son `nombreDescriptivo` (de tipo `TINYTEXT`) y `nAsignaturas` (de tipo `DECIMAL(3,0)`). La columna `idCurso` es auto incrementable, por lo que no necesitas incluirla en la inserción.

1. Código para insertar 10 registros:

python

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

def insert_records():
    try:
        # Establecer la conexión
        connection = mysql.connector.connect(
            host='localhost',          # Cambia esto por tu host
            database='test_db',        # Cambia esto por tu nombre de base de datos
            user='root',               # Cambia esto por tu nombre de usuario
            password='password'        # Cambia esto por tu contraseña
        )

        if connection.is_connected():
            print('Conexión exitosa a la base de datos')
```

```
# Crear un cursor para ejecutar consultas
cursor = connection.cursor()

# Datos para insertar
cursos = [
    ("Introducción a Python", 5),
    ("Bases de Datos", 4),
    ("Estructuras de Datos", 3),
    ("Algoritmos Avanzados", 6),
    ("Desarrollo Web", 4),
    ("Machine Learning", 3),
    ("Redes Neuronales", 5),
    ("Análisis de Datos", 4),
    ("Programación Orientada a Objetos", 6),
    ("Ciberseguridad", 3)
]

# Consulta SQL para insertar datos
insert_query = """
INSERT INTO curso (nombreDescriptivo, nAsignaturas)
VALUES (%s, %s)
"""

# Ejecutar la inserción de cada registro
cursor.executemany(insert_query, cursos)
connection.commit()
print('10 registros insertados con éxito')

except Error as e:
    print(f'Error al conectar a MySQL o al insertar datos: {e}')

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print('Conexión cerrada')

if __name__ == '__main__':
    insert_records()
```

Explicación del código:

- **Conexión:** Establecemos la conexión con la base de datos MySQL usando ``mysql.connector.connect``.
- **Datos para insertar:** Creamos una lista de tuplas con los valores que queremos insertar en la tabla ``curso``. Cada tupla contiene un ``nombreDescriptivo`` y un ``nAsignaturas``.
- **Consulta SQL:** Preparamos una consulta SQL para insertar los datos. Usamos ``%s`` como marcador de posición para los valores que serán insertados.
- ``executemany``: Usamos ``executemany`` para ejecutar la consulta con todos los datos de una vez.
- **Commit:** Llamamos a ``connection.commit()`` para guardar los cambios en la base de datos.

Asegúrate de ajustar ``host``, ``database``, ``user``, y ``password`` según tu configuración. Si la tabla tiene una estructura diferente o si necesitas más ayuda, ¡házmelo saber!



```
Me da el código el siguiente error "C:\Users\marco\OneDrive\Desktop\Curso Python
2024\codigos\M3.2 Bases de datos SQL\M3.2 Bases de datos SQL\.venv\Scripts\python.exe"
"C:\Users\marco\OneDrive\Desktop\Curso Python 2024\codigos\M3.2 Bases de datos SQL\M3.2
Bases de datos SQL\insertRegistroCurso.py"
Conexión exitosa a la base de datos
Error al conectar a MySQL o al insertar datos: 1062 (23000): Duplicate entry '0' for key 'PRIMARY'
Conexión cerrada
```

```
Process finished with exit code 0
```



El error `1062 (23000): Duplicate entry '0' for key 'PRIMARY'` indica que se está intentando insertar un valor duplicado en una columna que tiene una restricción de clave primaria. En este caso, parece que la columna `idCurso`, que es auto incrementable, está causando problemas. Aquí hay algunos pasos para solucionar el problema:

1. **Verifica que la columna `idCurso` esté correctamente configurada como auto incrementable:** Asegúrate de que `idCurso` esté definido en la tabla como `AUTO_INCREMENT`. Puedes verificar esto con una consulta SQL para describir la estructura de la tabla:

```
sql
```

```
DESCRIBE curso;
```

La columna `idCurso` debe tener la propiedad `AUTO_INCREMENT`.

2. **Revisa el estado de la tabla:** Si has estado insertando registros manualmente, es posible que el contador de auto incremento haya sido alterado. Puedes restablecer el valor de auto incremento a su valor correcto con una consulta SQL como esta:

```
sql
```

```
ALTER TABLE curso AUTO_INCREMENT = 1;
```

Esto reinicia el contador de auto incremento. Ten cuidado con esta operación, especialmente si ya tienes datos en la tabla, para evitar posibles conflictos con las claves primarias existentes.

3. **Revisa el código:** Asegúrate de que el código no esté intentando insertar valores en la columna `idCurso`. La columna auto incrementable no debería estar incluida en la lista de campos a insertar. En el código proporcionado, no estamos insertando `idCurso`, pero verifica que no haya errores en el código que puedan estar causando problemas.

Aquí tienes un ejemplo revisado del código para asegurar que no se inserte un valor en `idCurso`:

python

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

def insert_records():
    try:
        # Establecer la conexión
        connection = mysql.connector.connect(
            host='localhost',      # Cambia esto por tu host
            database='test_db',    # Cambia esto por tu nombre de base de datos
            user='root',          # Cambia esto por tu nombre de usuario
            password='password'    # Cambia esto por tu contraseña
        )

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

            # Crear un cursor para ejecutar consultas
            cursor = connection.cursor()

            # Datos para insertar
            cursos = [
                ("Introducción a Python", 5),
                ("Bases de Datos", 4),
                ("Estructuras de Datos", 3),
                ("Algoritmos Avanzados", 6),
                ("Desarrollo Web", 4),
                ("Machine Learning", 3),
                ("Redes Neuronales", 5),
                ("Análisis de Datos", 4),
                ("Programación Orientada a Objetos", 6),
                ("Ciberseguridad", 3)
            ]

            # Consulta SQL para insertar datos
            insert_query = """
            INSERT INTO curso (nombreDescriptivo, nAsignaturas)
            VALUES (%s, %s)
            """

            # Ejecutar la inserción de cada registro
            cursor.executemany(insert_query, cursos)
            connection.commit()
            print('10 registros insertados con éxito')

        except Error as e:
            print(f'Error al conectar a MySQL o al insertar datos: {e}')

        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
                print('Conexión cerrada')

    if __name__ == '__main__':
        insert_records()
```

Si sigues teniendo problemas, proporciona más detalles sobre la estructura de la tabla y cualquier otra configuración relevante, y estaré encantado de ayudarte a resolverlo.



Me sigue dando un error, el primer campo id autoincrementable ese no se puede insertar, solo inserta los campos posteriores dame el código



Entiendo, si el problema persiste y la columna `idCurso` es auto incrementable, entonces no debes intentar insertar un valor en esa columna. Asegúrate de que solo estás insertando valores en las columnas `nombreDescriptivo` y `nAsignaturas`.

Aquí está el código revisado para asegurarse de que solo se están insertando los campos correctos:

python

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

def insert_records():
    try:
        # Establecer la conexión
        connection = mysql.connector.connect(
            host='localhost',          # Cambia esto por tu host
            database='test_db',        # Cambia esto por tu nombre de base de datos
            user='root',               # Cambia esto por tu nombre de usuario
            password='password'        # Cambia esto por tu contraseña
        )

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

            # Crear un cursor para ejecutar consultas
            cursor = connection.cursor()

            # Datos para insertar
            cursos = [
                ("Introducción a Python", 5),
                ("Bases de Datos", 4),
                ("Estructuras de Datos", 3),
                ("Algoritmos Avanzados", 6),
                ("Desarrollo Web", 4),
                ("Machine Learning", 3),
                ("Redes Neuronales", 5),
                ("Análisis de Datos", 4),
                ("Programación Orientada a Objetos", 6),
                ("Ciberseguridad", 3)
            ]

            # Consulta SQL para insertar datos
            insert_query = """
            INSERT INTO curso (nombreDescriptivo, nAsignaturas)
            VALUES (%s, %s)
            """

            # Ejecutar la inserción de cada registro
            cursor.executemany(insert_query, cursos)
            connection.commit()
            print('10 registros insertados con éxito')

        except Error as e:
            print(f'Error al conectar a MySQL o al insertar datos: {e}')

    finally:
```

```
if connection.is_connected():
    cursor.close()
    connection.close()
    print('Conexión cerrada')

if __name__ == '__main__':
    insert_records()
```

Verificaciones adicionales:

1. **Estructura de la tabla:** Asegúrate de que la tabla `curso` esté correctamente definida con `idCurso` como `AUTO_INCREMENT`. Puedes verificar esto con:

sql

```
DESCRIBE curso;
```

La columna `idCurso` debe estar configurada con `AUTO_INCREMENT` y no debe incluirse en las operaciones de inserción.

2. **Restablecimiento del auto incremento:** Si has realizado múltiples inserciones y el contador de auto incremento se ha desincronizado, puedes restablecerlo a un valor adecuado. Sin embargo, esto debe hacerse con cuidado para no sobrescribir valores existentes. Puedes hacerlo con:

sql

```
ALTER TABLE curso AUTO_INCREMENT = 1;
```

Nota: Solo realiza esto si estás seguro de que no hay datos que se verían afectados.

Si continúas teniendo problemas, revisa los detalles de la tabla en la base de datos para asegurar que esté configurada correctamente y asegúrate de que no haya conflictos con los valores de auto incremento. Si necesitas más ayuda, no dudes en preguntar.



Inserta en la anterior tabla en el registro 7 en "nombreDescriptivo" el valor "Tejido"



Para actualizar un registro específico en la tabla `curso`, donde el `idCurso` es 7 y deseas cambiar el valor de `nombreDescriptivo` a `"Tejido"`, puedes usar una consulta SQL `UPDATE`. Aquí tienes el código para hacerlo en Python:

1. **Código para actualizar el registro:**

python

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

def update_record():
    try:
        # Establecer la conexión
        connection = mysql.connector.connect(
            host='localhost',      # Cambia esto por tu host
            database='test_db',    # Cambia esto por tu nombre de base de datos
            user='root',          # Cambia esto por tu nombre de usuario
            password='password'    # Cambia esto por tu contraseña
        )

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

            # Crear un cursor para ejecutar consultas
            cursor = connection.cursor()

            # Consulta SQL para actualizar el registro
            update_query = """
            UPDATE curso
            SET nombreDescriptivo = %s
            WHERE idCurso = %s
            """

            # Datos para actualizar
            data = ("Tejido", 7)

            # Ejecutar la consulta de actualización
            cursor.execute(update_query, data)
            connection.commit()
            print('Registro actualizado con éxito')

        except Error as e:
            print(f'Error al conectar a MySQL o al actualizar datos: {e}')

        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
                print('Conexión cerrada')

if __name__ == '__main__':
    update_record()
```

Explicación del código:

- **Conexión:** Se establece una conexión con la base de datos MySQL usando `mysql.connector.connect``.
- **Consulta SQL:** Se prepara una consulta SQL `UPDATE`` para cambiar el valor de `nombreDescriptivo`` donde `idCurso`` es igual a 7.
- **Datos:** La tupla `("Tejido", 7)`` se utiliza para reemplazar el valor del campo `nombreDescriptivo`` con `"Tejido"`` en el registro con `idCurso`` igual a 7.
- `execute``: Se ejecuta la consulta de actualización con los datos proporcionados.
- **Commit:** Se guarda el cambio en la base de datos con `connection.commit()``.

Asegúrate de ajustar `host`, `database`, `user`, y `password` según tu configuración. Si hay algún problema o tienes más preguntas, no dudes en decirme.



Dime donde estoy mal en esta sentencia de SQL deseo mostrar un solo registro de tabla profesor donde NIF sea igual a un dato



Para mostrar un solo registro de la tabla `profesor` donde el campo `NIF` sea igual a un dato específico, la sintaxis correcta de la sentencia SQL es:

sql

```
SELECT * FROM profesor WHERE NIF = 'dato';
```

Aquí te explico cada parte:

- `SELECT *`: Esto selecciona todas las columnas del registro.
- `FROM profesor`: Especifica que quieres seleccionar datos de la tabla `profesor`.
- `WHERE NIF = 'dato'`: Filtra los resultados para mostrar solo aquellos registros donde el campo `NIF` sea igual al valor `dato`.

Posibles errores comunes:

1. **Uso incorrecto de comillas:** Si el `NIF` es un dato de tipo `VARCHAR` o `CHAR`, el valor debe estar entre comillas simples (`'dato'`). Si omites las comillas o usas comillas dobles (`"dato"`), podrías obtener un error o un resultado incorrecto.
2. **Especificar un solo registro:** Si `NIF` no es una clave primaria y puede haber múltiples registros con el mismo valor de `NIF`, la consulta podría devolver varios registros. Si solo quieres asegurarte de obtener un solo registro, podrías utilizar `LIMIT 1`:

sql

```
SELECT * FROM profesor WHERE NIF = 'dato' LIMIT 1;
```

Si te sigue dando un error o no obtienes el resultado esperado, por favor compárteme la sentencia exacta que estás usando, y estaré encantado de ayudarte a corregirla.

ChatGPT puede cometer errores. Considera verificar la información importante.