```
print("helloword")
```

```
helloword
```

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

```
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.3.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (7.2 kB)
Downloading mysql_connector_python-9.3.0-cp311-cp311-manylinux_2_28_x86_64.whl (33.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 33.9/33.9 MB 57.4 MB/s eta 0:00:00
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.3.0
```

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

hostname = "keuyv.h.filess.io"
database = "OlistDataBase_drivingleg"
port = "3307"
username = "OlistDataBase_drivingleg"
password = "3b363f5c4d0a749eb80cf5e58cd7bc213287ca47"

try:
    connection = mysql.connector.connect(host=hostname, database=database, user=username, password=password, port=port)
    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record)

except Error as e:
    print("Error while connecting to MySQL", e)
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

```
/tmp/ipython-input-4-4280410883.py:13: DeprecationWarning: Call to deprecated function get_server_info. Reason:
    The property counterpart 'server_info' should be used instead.

  db_Info = connection.get_server_info()
Connected to MySQL Server version  8.0.36-28
You're connected to database:  ('OlistDataBase_drivingleg',)
MySQL connection is closed
```

```
import pandas as pd
order_payments = pd.read_csv("/content/olist_order_payments_dataset.csv")
order_payments.head()
order_payments.shape
```

```
(103886, 5)
```

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

# Connection details
hostname = "keuyv.h.filess.io"
database = "OlistDataBase_drivingleg"
port = "3307"
username = "OlistDataBase_drivingleg"
password = "3b363f5c4d0a749eb80cf5e58cd7bc213287ca47"

# CSV file path
csv_file_path = "/content/olist_order_payments_dataset.csv"

# Table name where the data will be uploaded
table_name = "olist_order_payments"
```

```
try:
    # Step 1: Establish a connection to MySQL server
    connection = mysql.connector.connect(
        host=hostname,
        database=database,
        user=username,
        password=password,
        port=port
    )
    if connection.is_connected():
        print("Connected to MySQL Server successfully!")

        # Step 2: Create a cursor to execute SQL queries
        cursor = connection.cursor()

        # Step 3: Drop table if it already exists (for clean insertion)
        cursor.execute(f"DROP TABLE IF EXISTS {table_name};")
        print(f"Table `{table_name}` dropped if it existed.")

        # Step 4: Create a table structure to match CSV file
        create_table_query = f"""
        CREATE TABLE {table_name} (
            order_id VARCHAR(50),
            payment_sequential INT,
            payment_type VARCHAR(20),
            payment_installments INT,
            payment_value FLOAT
        );
        """
        cursor.execute(create_table_query)
        print(f"Table `{table_name}` created successfully!")

        # Step 5: Load the CSV data into pandas DataFrame
        data = pd.read_csv(csv_file_path)
        print("CSV data loaded into pandas DataFrame.")

        # Step 6: Insert data in batches of 500 records
        batch_size = 500  # Define the batch size
        total_records = len(data)  # Get total records in the DataFrame

        print(f"Starting data insertion into `{table_name}` in batches of {batch_size} records.")
        for start in range(0, total_records, batch_size):
            end = start + batch_size
            batch = data.iloc[start:end]  # Get the current batch of records

            # Convert batch to list of tuples for MySQL insertion
            batch_records = [
                tuple(row) for row in batch.itertuples(index=False, name=None)
            ]

            # Prepare the INSERT query
            insert_query = f"""
            INSERT INTO {table_name}
            (order_id, payment_sequential, payment_type, payment_installments, payment_value)
            VALUES (%s, %s, %s, %s, %s);
            """

            # Execute the insertion query for the batch
            cursor.executemany(insert_query, batch_records)
            connection.commit()  # Commit after each batch
            print(f"Inserted records {start + 1} to {min(end, total_records)} successfully.")

        print(f"All {total_records} records inserted successfully into `{table_name}`.")

except Error as e:
    # Step 7: Handle any errors
    print("Error while connecting to MySQL or inserting data:", e)

finally:
    # Step 8: Close the cursor and connection
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed.")
```

```
Connected to MySQL Server successfully!
Table `olist_order_payments` dropped if it existed.
Table `olist_order_payments` created successfully!
CSV data loaded into pandas DataFrame.
Starting data insertion into `olist_order_payments` in batches of 500 records.
Inserted records 1 to 500 successfully.
Inserted records 501 to 1000 successfully.
Inserted records 1001 to 1500 successfully.
Inserted records 1501 to 2000 successfully.
Inserted records 2001 to 2500 successfully.
Inserted records 2501 to 3000 successfully.
Inserted records 3001 to 3500 successfully.
Inserted records 3501 to 4000 successfully.
Inserted records 4001 to 4500 successfully.
Inserted records 4501 to 5000 successfully.
Inserted records 5001 to 5500 successfully.
Inserted records 5501 to 6000 successfully.
Inserted records 6001 to 6500 successfully.
Inserted records 6501 to 7000 successfully.
Inserted records 7001 to 7500 successfully.
Inserted records 7501 to 8000 successfully.
Inserted records 8001 to 8500 successfully.
Inserted records 8501 to 9000 successfully.
Inserted records 9001 to 9500 successfully.
Inserted records 9501 to 10000 successfully.
Inserted records 10001 to 10500 successfully.
Inserted records 10501 to 11000 successfully.
Inserted records 11001 to 11500 successfully.
Inserted records 11501 to 12000 successfully.
Inserted records 12001 to 12500 successfully.
Inserted records 12501 to 13000 successfully.
Inserted records 13001 to 13500 successfully.
Inserted records 13501 to 14000 successfully.
Inserted records 14001 to 14500 successfully.
Inserted records 14501 to 15000 successfully.
Inserted records 15001 to 15500 successfully.
Inserted records 15501 to 16000 successfully.
Inserted records 16001 to 16500 successfully.
Inserted records 16501 to 17000 successfully.
Inserted records 17001 to 17500 successfully.
Inserted records 17501 to 18000 successfully.
Inserted records 18001 to 18500 successfully.
Inserted records 18501 to 19000 successfully.
Inserted records 19001 to 19500 successfully.
Inserted records 19501 to 20000 successfully.
Inserted records 20001 to 20500 successfully.
Inserted records 20501 to 21000 successfully.
Inserted records 21001 to 21500 successfully.
Inserted records 21501 to 22000 successfully.
Inserted records 22001 to 22500 successfully.
Inserted records 22501 to 23000 successfully.
Inserted records 23001 to 23500 successfully.
Inserted records 23501 to 24000 successfully.
Inserted records 24001 to 24500 successfully.
Inserted records 24501 to 25000 successfully.
Inserted records 25001 to 25500 successfully.
Inserted records 25501 to 26000 successfully.
Inserted records 26001 to 26500 successfully.
```

MongoDB

```
!pip install pymongo
```

```
Collecting pymongo
  Downloading pymongo-4.13.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (22 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.7.0-py3-none-any.whl.metadata (5.8 kB)
Downloading pymongo-4.13.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 37.4 MB/s eta 0:00:00
Downloading dnspython-2.7.0-py3-none-any.whl (313 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.6/313.6 kB 15.2 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.7.0 pymongo-4.13.2
```

```
# importing module

from pymongo import MongoClient
hostname = "dftt0.h.filess.io"

database = "OlistMongoDb_molecular"
```

```
port= "27018"

username= "OlistMongoDb_molecular"

password= "2aaa152ec7d71276e5e5742e00bdeac7e12aaf98"

uri = "mongodb://"+ username + ":"+ password + "@"+ hostname + ":"+ port + "/"+ database# Connect with the portnumber and hos

client= MongoClient(uri) # Access database

mydatabase= client[database]
```

```
import pandas as pd
from pymongo import MongoClient

# Load the product_category CSV file into a pandas DataFrame
try:
  product_category_df = pd.read_csv("/content/product_category_name_translation.csv")
except FileNotFoundError:
  print("Error: 'product_category_name_translation.csv' not found.")
  exit() # Exit the script if the file is not found


# MongoDB connection details (assuming these are already defined in your script)
hostname = "dftt0.h.filess.io"
database = "OlistMongoDb_molecular"
port = "27018"
username = "OlistMongoDb_molecular"
password = "2aaa152ec7d71276e5e5742e00bdeac7e12aaf98"

uri = "mongodb://" + username + ":" + password + "@" + hostname + ":" + port + "/" + database

try:
    # Establish a connection to MongoDB
    client = MongoClient(uri)
    db = client[database]

    # Select the collection (or create if it doesn't exist)
    collection = db["product_categories"]  # Choose a suitable name for your collection

    # Convert the DataFrame to a list of dictionaries for insertion into MongoDB
    data_to_insert = product_category_df.to_dict(orient="records")

    # Insert the data into the collection
    collection.insert_many(data_to_insert)

    print("Data uploaded to MongoDB successfully!")

except Exception as e:
    print(f"An error occurred: {e}")

finally:
    # Close the MongoDB connection
    if client:
        client.close()
```

```
Data uploaded to MongoDB successfully!
```