Pandas Input - Output — In One Posting …..

```
pip install pandas sqlalchemy pymongo pymysql psycopg2 pyodbc openpyxl pandas-gbq s3fs
```

```
import pandas as pd
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect('database.db')

# Execute SQL query and load data into a DataFrame
data = pd.read_sql('SELECT * FROM tablename', conn)

# Display the first few rows
print(data.head())

import pandas as pd
```

```
import mysql.connector

# Connect to the MySQL database
conn = mysql.connector.connect(
    host='hostname',
    user='username',
    password='password',
    database='databasename'
)

# Execute SQL query and load data into a DataFrame
data = pd.read_sql('SELECT * FROM tablename', conn)

# Display the first few rows
print(data.head())

import pandas as pd
import psycopg2

# Connect to the PostgreSQL database
conn = psycopg2.connect(
    host='hostname',
    user='username',
    password='password',
    database='databasename'
)

# Execute SQL query and load data into a DataFrame
data = pd.read_sql('SELECT * FROM tablename', conn)

# Display the first few rows
print(data.head())

import pandas as pd
import pyodbc

# Connect to the SQL Server database
```

```
conn = pyodbc.connect(
    'DRIVER={SQL Server};
SERVER=hostname;DATABASE=databasename;UID=username;PWD=
password'
)

# Execute SQL query and load data into a DataFrame
data = pd.read_sql('SELECT * FROM tablename', conn)

# Display the first few rows
print(data.head())

import pandas as pd
from sqlalchemy import create_engine

# MySQL
mysql_engine =
create_engine('mysql://username:password@localhost:3306/database_
name')
df = pd.read_sql('SELECT * FROM table_name', mysql_engine)

# PostgreSQL
postgres_engine =
create_engine('postgresql://username:password@localhost:5432/databa
se_name')
df = pd.read_sql('SELECT * FROM table_name', postgres_engine)

# SQLite
sqlite_engine = create_engine('sqlite:///database.db')
df = pd.read_sql('SELECT * FROM table_name', sqlite_engine)

# Microsoft SQL Server
mssql_engine =
create_engine('mssql+pyodbc://username:password@server_name/data
base_name?driver=SQL+Server')
df = pd.read_sql('SELECT * FROM table_name', mssql_engine)
```

```
# Create a connection string
engine =
create_engine('oracle+cx_oracle://username:password@hostname:port/
?service_name=service_name')

# Query the data into a pandas DataFrame
df = pd.read_sql("SELECT * FROM my_table", engine)

# Display the dataframe
print(df)


from pymongo import MongoClient

# Connect to MongoDB
client = MongoClient('mongodb://localhost:27017/')
db = client['database_name']
collection = db['collection_name']

# Convert to DataFrame
df = pd.DataFrame(list(collection.find()))

# Read Excel file
df = pd.read_excel('file.xlsx', sheet_name='Sheet1')

# Read specific columns
df = pd.read_excel('file.xlsx', usecols=['A', 'B'])

# Read multiple sheets
all_sheets = pd.read_excel('file.xlsx', sheet_name=None)
# Returns dict of DataFrames

# Basic CSV read
df = pd.read_csv('file.csv')

# CSV with specific encoding
df = pd.read_csv('file.csv', encoding='utf-8')
```

```
# CSV with different separator
df = pd.read_csv('file.csv', sep=';')

# CSV with specific date format
df = pd.read_csv('file.csv', parse_dates=['date_column'])

# Read JSON file
df = pd.read_json('file.json')

# Read JSON from URL
df = pd.read_json('http://api.example.com/data')

# Read JSON with specific orientation
df = pd.read_json('file.json', orient='records')

# Read tables from HTML
tables = pd.read_html('http://example.com')
df = tables[0]  # Get first table

# Read with specific attributes
df = pd.read_html('file.html', attrs={'id': 'table_id'})

# Read Parquet file
df = pd.read_parquet('file.parquet')

# Read specific columns
df = pd.read_parquet('file.parquet', columns=['col1', 'col2'])

# Read HDF5 file
df = pd.read_hdf('file.h5', 'key')

# Read with where condition
df = pd.read_hdf('file.h5', 'key', where=['col1 > 0'])

from google.cloud import bigquery
```

```
# Create client
client = bigquery.Client()

# Read from BigQuery
query = """
SELECT *
FROM `project.dataset.table`
WHERE date >= '2024-01-01'
"""
df = pd.read_gbq(query, project_id='your-project-id')

# Read with specific data types
df = pd.read_csv('file.csv', dtype={'column1': 'int32', 'column2': 'float64'})

# Handle missing values
df = pd.read_csv('file.csv', na_values=['NA', 'missing'])

# Read specific rows
df = pd.read_csv('file.csv', nrows=1000)

# Skip rows
df = pd.read_csv('file.csv', skiprows=5)

try:
    df = pd.read_csv('file.csv')
except FileNotFoundError:
    print("File not found")
except pd.errors.EmptyDataError:
    print("File is empty")
except pd.errors.ParserError:
    print("Parsing error - check file format")
```