

```
In [1]: import sqlite3
```

```
In [2]: con = sqlite3.connect('testdb1.sqlite')
cur = con.cursor()

# Create table - AGES
cur.execute('''CREATE TABLE AGES
              ([generated_id] INTEGER PRIMARY KEY,[Age_Key] integer, [Age_Category] text

# Create table - GENDERS
cur.execute('''CREATE TABLE GENDERS
              ([generated_id] INTEGER PRIMARY KEY,[Gender_Key] integer, [Gender] text)''

# Create table - STATES
cur.execute('''CREATE TABLE STATES
              ([generated_id] INTEGER PRIMARY KEY, [State_Key] integer, [State_Name] tex

# Create table - OPINIONS
cur.execute('''CREATE TABLE OPINIONS
              ([generated_id] INTEGER PRIMARY KEY, [Opinion_Key] integer, [Opinion] text

# Create table - RESPONDENTS
cur.execute('''CREATE TABLE RESPONDENTS
              ([generated_id] INTEGER PRIMARY KEY, [Person_Key] integer, [Age_Key] integ

con.commit()
```

```
In [3]: import pandas as pd
```

```
In [4]: con = sqlite3.connect('testdb1.sqlite')
cur = con.cursor()

read_ages = pd.read_csv ('ages.csv')
read_ages.to_sql('AGES', con, if_exists='append', index = False)
# Insert the values from the csv file into the table 'AGES'

read_genders = pd.read_csv ('genders.csv')
read_genders.to_sql('GENDERS', con, if_exists='replace', index = False)
# Replace the values from the csv file into the table 'GENDERS'

read_states = pd.read_csv ('states.csv')
read_states.to_sql('STATES', con, if_exists='replace', index = False)
# Replace the values from the csv file into the table 'STATES'

read_opinions = pd.read_csv ('opinions.csv')
read_opinions.to_sql('OPINIONS', con, if_exists='replace', index = False)
# Replace the values from the csv file into the table 'OPINIONS'

read_respondents = pd.read_csv ('respondents.csv')
read_respondents.to_sql('RESPONDENTS', con, if_exists='replace', index = False)
# Replace the values from the csv file into the table 'RESPONDENTS'
```

```
In [ ]: # When reading the csv:
# - if the path name is complex, can place 'r' before the path string to read any speci
# - Don't forget to put the file name at the end of the path + '.csv'
# - Before running the code, make sure that the column names in the CSV files match wit
#created and in the query below
# - If needed make sure that all the columns are in a TEXT format
```

```
In [5]: con=sqlite3.connect('testdb1.sqlite')
def x(q):
    return pd.read_sql_query(q,con)

q='''SELECT * FROM RESPONDENTS '''

df=x(q)
df
```

```
Out[5]:
```

	Person_Key	Age_Key	Gender_Key	State_Key	Children	Salary	Opinion_Key
<b>0</b>	1	2	2	9	2	63017	5
<b>1</b>	2	2	2	10	3	100302	1
<b>2</b>	3	2	2	2	0	144043	5
<b>3</b>	4	1	2	2	0	36025	4
<b>4</b>	5	2	1	9	0	97543	3
...	...	...	...	...	...	...	...
<b>394</b>	395	2	2	2	0	60715	3
<b>395</b>	396	2	1	10	2	91760	2
<b>396</b>	397	2	2	1	1	82558	1
<b>397</b>	398	2	1	7	1	84880	5
<b>398</b>	399	2	2	4	2	76933	5

399 rows × 7 columns

```
In [6]: q='''SELECT * FROM RESPONDENTS NATURAL JOIN AGES NATURAL JOIN STATES '''

df=x(q)
df
```

```
Out[6]:
```

	Person_Key	Age_Key	Gender_Key	State_Key	Children	Salary	Opinion_Key	generated_id	Age_Ci
<b>0</b>	1	2	2	9	2	63017	5	2	Midd
<b>1</b>	2	2	2	10	3	100302	1	2	Midd
<b>2</b>	3	2	2	2	0	144043	5	2	Midd
<b>3</b>	4	1	2	2	0	36025	4	1	

	Person_Key	Age_Key	Gender_Key	State_Key	Children	Salary	Opinion_Key	generated_id	Age_Ci
<b>4</b>	5	2	1	9	0	97543	3	2	Midd
...	...	...	...	...	...	...	...	...	...
<b>394</b>	395	2	2	2	0	60715	3	2	Midd
<b>395</b>	396	2	1	10	2	91760	2	2	Midd
<b>396</b>	397	2	2	1	1	82558	1	2	Midd
<b>397</b>	398	2	1	7	1	84880	5	2	Midd
<b>398</b>	399	2	2	4	2	76933	5	2	Midd

399 rows × 11 columns

In [7]: `df.to_pickle('./dummy.pkl')`

In [8]: `df1 = pd.read_pickle('./dummy.pkl')`  
df1

Out[8]:

	Person_Key	Age_Key	Gender_Key	State_Key	Children	Salary	Opinion_Key	generated_id	Age_Ci
<b>0</b>	1	2	2	9	2	63017	5	2	Midd
<b>1</b>	2	2	2	10	3	100302	1	2	Midd
<b>2</b>	3	2	2	2	0	144043	5	2	Midd
<b>3</b>	4	1	2	2	0	36025	4	1	
<b>4</b>	5	2	1	9	0	97543	3	2	Midd
...	...	...	...	...	...	...	...	...	...
<b>394</b>	395	2	2	2	0	60715	3	2	Midd
<b>395</b>	396	2	1	10	2	91760	2	2	Midd
<b>396</b>	397	2	2	1	1	82558	1	2	Midd
<b>397</b>	398	2	1	7	1	84880	5	2	Midd
<b>398</b>	399	2	2	4	2	76933	5	2	Midd

399 rows × 11 columns

In [9]: `import os`  
`os.remove('./dummy.pkl')`

```
In [7]: def median(values):
        """Get the median of a list of values

        Args:
            values (iterable of float): A list of numbers

        Returns:
            float
        """
        # Write the median() function
        midpoint = int(len(values) / 2)
        if len(values) % 2 == 0:
            median = (values[midpoint - 1] + values[midpoint]) / 2
        else:
            median = values[midpoint]
        return median
```

```
In [8]: list=[1, 5, 8, 10, 22, 32, 91]
        median(list)
```

Out[8]: 10

```
In [9]: list=[1, 5, 8, 10, 22, 32, 35, 91]
        median(list)
```

Out[9]: 16.0

```
In [10]: def count_letter(content, letter):
        """Count the number of times `letter` appears in `content`.

        Args:
            content (str): The string to search.
            letter (str): The letter to search for.

        Returns:
            int

        # Add a section detailing what errors might be raised
        Raises:
            ValueError: If `letter` is not a one-character string.
        """
        if (not isinstance(letter, str)) or len(letter) != 1:
            raise ValueError("'letter' must be a single character string.")
        return len([char for char in content if char == letter])
```

```
In [11]: count_letter('Happy Pi Day!', 'a')
```

Out[11]: 2

```
In [12]: count_letter('Happy Pi Day!', 'Pi')
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-12-67018246a182> in <module>
----> 1 count_letter('Happy Pi Day!', 'Pi')

<ipython-input-10-4972e87aa9d9> in count_letter(content, letter)
    14     """
    15     if (not isinstance(letter, str)) or len(letter) != 1:
--> 16         raise ValueError("'letter' must be a single character string.")
    17     return len([char for char in content if char == letter])
```

**ValueError:** 'letter' must be a single character string.

```
In [16]: a = int(input('First Integer: '))
         b = int(input('Second Integer: '))
         if b > a:
             print("b is greater than a")
         elif a == b:
             print("a and b are equal")
         else:
             print("a is greater than b")
```

First Integer: 5  
Second Integer: 10  
b is greater than a

```
In [13]: a = int(input('First Integer: '))
         b = int(input('Second Integer: '))
         if a > b: print("a is greater than b")
```

First Integer: 5  
Second Integer: 5

```
In [22]: c = int(input('Third Integer: '))
```

Third Integer: -7

```
In [23]: a = int(input('First Integer: '))
         b = int(input('Second Integer: '))
         print("A") if a > b else print("B")
         #print("A") if a > b else print("=") if a == b else print("B")
```

First Integer: 5  
Second Integer: -1  
A

```
In [24]: #multiple conditions
         if a > b or a > c:
             print("At least one of the conditions is True")

         if (a > b) and (a > c):
             print("Both conditions are True")

         #may need to enclose conditions in parentheses to enforce order of operations or to cla
```

At least one of the conditions is True  
Both conditions are True

```
In [25]: fruits = ["apple", "banana", "cherry"]
         for x in fruits:
             print(x)
             if x == "banana":
                 break
```

```
apple
banana
```

```
In [26]: fruits = ["apple", "banana", "cherry"]
         for x in fruits:
             print(x)
```

```
apple
banana
cherry
```

```
In [27]: fruits = ["apple", "banana", "cherry"]
         for x in fruits:
             if x == "banana":
                 continue
             print(x)
```

```
apple
cherry
```

```
In [29]: for x in range(6):
         print(x)

         for x in range(2, 30, 3):
             print(x)
```

```
0
1
2
3
4
5
2
5
8
11
14
17
20
23
26
29
```

```
In [30]: i = 1
         while i < 6:
             print(i)
             i += 1
         else:
             print("i is no longer less than 6")

         #while is usually used for processes where the number of steps is not known
         #for loops are usually used for a fixed number of steps
```

```
1  
2  
3  
4  
5  
i is no longer less than 6
```

In [ ]:

```
for movie in movies:  
    try:  
        # Find the first occurrence of word  
        print(movie.index("money", 12, 51))  
    except ValueError:  
        print("substring not found")
```