### CISD43_MongoDB Final

-- Using USA_Housing.csv within MongoDB when reentering MongoDB Compass, find the "housing_sales", click the three dots, select [Open in new tab] th data will open

- By Taichun Chao

## Open MongoDB Compass and:

-

create a new database called "housing_db"

-

Create a collection named housing_saless".

## Import Data:¶

Press "Import Data" and select the fileUSA_Housingl.csv. The file contains columns: Ave. Area House Ageo,Ave. Area Number of Roomse, Ave. Area Number of Bedroomsn,Area Populationy,Pricee,aandAddressy".

In [4]:
```
#!pip install pymongo
```

### Import MongoClient and Pandas

In [7]:
```
#import liibrary
from pymongo import MongoClient
import pandas as pd
```

## Connect to MongoDB

In [10]:
```
#Connect to MonagoDB
client = MongoClient('mongodb://localhost:27017/')
#Select the database
db = client["housing_db"]
#Select the collection
collection = db["housing_sales"]
```

## Count the total number of record

In [13]:
```
total_records = collection.count_documents({})
print("Total number of records: ", total_records)
```

```
Total number of records:  5000
```

## Retrieve the first three records

In [16]:
```
# Retrieve the first three records
results = collection.find().limit(3)

# Print the records
for result in results:
    print(result)
```

```
{'_id': ObjectId('6657e896a2bc56798b9339d6'), 'Avg': {' Area Income': 79545.45857431
678, ' Area House Age': 5.682861321615587, ' Area Number of Rooms': 7.00918814279223
7, ' Area Number of Bedrooms': 4.09}, 'Area Population': 23086.800502686456, 'Pric
e': 1059033.5578701235, 'Address': '208 Michael Ferry Apt. 674\nLaurabury, NE 37010-
5101'}
{'_id': ObjectId('6657e896a2bc56798b9339d7'), 'Avg': {' Area Income': 79248.64245482
568, ' Area House Age': 6.0028998082752425, ' Area Number of Rooms': 6.7308210190949
19, ' Area Number of Bedrooms': 3.09}, 'Area Population': 40173.07217364482, 'Pric
e': 1505890.91484695, 'Address': '188 Johnson Views Suite 079\nLake Kathleen, CA 489
58'}
{'_id': ObjectId('6657e896a2bc56798b9339d8'), 'Avg': {' Area Income': 61287.06717865
6784, ' Area House Age': 5.865889840310001, ' Area Number of Rooms': 8.5127274303750
99, ' Area Number of Bedrooms': 5.13}, 'Area Population': 36882.15939970458, 'Pric
e': 1058987.9878760849, 'Address': '9127 Elizabeth Stravenue\nDanieltown, WI 06482-3
489'}
```

## Use query to find the condition, only print out "price" not whole row

In [19]:
```
# Query to find houses with 4 bedrooms and house age >= 6, then count the total amo
query = {
    'Avg. Area Number of Bedrooms': 4,
    'Avg. Area House Age': {'$gte': 6}
}

# Execute the query and project the price field
#results = collection.find(query, {'Price': 1, '_id': 1}) # 0 means not include
results =collection.find(query,{'Price':1, 'Address':1}).limit(5)
count = collection.count_documents(query)
# Print the prices
for house in results:
   print(f"Price: {house['Price']}, Address: {house['Address']}")
   # print(house['Price'], "_id":0)
print('total item:', count)
```

```
Price: 1520234.2293774572, Address: 905 Lane Pines Suite 348
Brownborough, DE 57196-3319
Price: 1262017.792059947, Address: 089 Wilson Forks Suite 185
East Gina, PA 98785-6132
Price: 1852375.50677484, Address: 22801 Skinner Isle
Mooretown, TN 01957
Price: 1610006.604587974, Address: 338 Karen Prairie Apt. 341
Murphymouth, MI 56611
Price: 1550359.5484654682, Address: 31018 Park Square Suite 876
Normanside, MA 65696-1725
total item: 20
```

## Find hourse price less $200000 and only print out 2 on list

In [22]:
```python
#find the house price is less $80000 and only print 2 on the list
query = {'Price': {"$lt": 200000}}
count = collection.count_documents(query)
results = collection.find(query).limit(2)

print("Total house less than $200000: total num ", count, '\n')
for result in results:
    print(result)
```

```
Total house less than $200000: total num  6

{'_id': ObjectId('6657e896a2bc56798b933add'), 'Avg': {' Area Income': 40366.61629125
728, ' Area House Age': 4.902939589246314, ' Area Number of Rooms': 7.61711809996279
1, ' Area Number of Bedrooms': 5.07}, 'Area Population': 16349.365394310094, 'Pric
e': 152071.87474956046, 'Address': '503 Howard Pass Apt. 427\nFernandezborough, GA 0
2514'}
{'_id': ObjectId('6657e896a2bc56798b933ecd'), 'Avg': {' Area Income': 37971.20756623
529, ' Area House Age': 4.291223903128535, ' Area Number of Rooms': 5.80750952723879
8, ' Area Number of Bedrooms': 3.24}, 'Area Population': 33267.767727560946, 'Pric
e': 31140.517620186045, 'Address': '98398 Terrance Pines\nSouth Joshua, MT 00544-891
9'}
```

## Find the number of Avg. Area Number of Bedrooms >= 5. in all the document with using a query variable, a result variable, and a dataframe

In [25]:
```python
# Query to count the number of houses with 5 or more bedrooms
query = {'Avg. Area Number of Bedrooms': {'$gte': 5}}

# Execute the query and count the results
count = collection.count_documents(query)

# Print the count
print(f'Number of houses with 5 or more bedrooms: {count}')
```

```
Number of houses with 5 or more bedrooms: 1214
```

## Find a house's bedroom more than 5 and the house age less than 3 years, the price can not over $500000

- Convert the result to Pandas DataFrame

In [28]:
```
query ={'Avg. Area Number of Bedrooms':{"$gt" : 5},
     'Avg. Area House Age': {'$lt': 3},
     "Price":{'$lt':500000}
   }
results = collection.find(query)
count =collection.count_documents(query)

#The following printout will continue
print(list(results))
print('total item:', count)
```

```
[{'_id': ObjectId('6657e896a2bc56798b933e08'), 'Avg': {' Area Income': 65016.2238106
4409, ' Area House Age': 2.644304186036705, ' Area Number of Rooms': 8.3063040817626
76, ' Area Number of Bedrooms': 6.05}, 'Area Population': 15902.582017185101, 'Pric
e': 414571.22293662146, 'Address': '584 Rick Cove\nLeeberg, ND 15540-8557'}, {'_id':
ObjectId('6657e896a2bc56798b934032'), 'Avg': {' Area Income': 71721.42137723006, ' A
rea House Age': 2.6830429033311556, ' Area Number of Rooms': 7.583527004255256, ' Ar
ea Number of Bedrooms': 6.26}, 'Area Population': 10704.821908621432, 'Price': 39544
0.2021544269, 'Address': '92426 Bennett Islands\nClintonberg, PW 44484'}]
total item: 2
```

In [ ]:
```
```

## To find Maximum and minimum number, setup pipeline

The following structure is called pipeline

- 1 every item put in " "
- 2 use aggregate,
- 3 result need to put in list ex. list(result)
- 4. "_id": none, none can be ""

In [32]:
```
# Define the aggregation pipeline to find max and min prices
pipeline = [
    {
        "$group": {
            "_id": "",
            "max_price": {"$max": "$Price"},
            "min_price": {"$min": "$Price"}
        }
    }
]

# Execute the aggregation
result = list(collection.aggregate(pipeline))

# Print the results
if result:
    print("Max price:", result[0]["max_price"])
    print("Min price:", result[0]["min_price"])
```

```
else:
    print("No data found")
```

```
Max price: 2469065.5941747027
Min price: 15938.657923287848
```

## Put data in dataframe as in pandas

In [35]:
```
data = collection.find({})
# Convert the result to a pandas DataFrame
df = pd.DataFrame(list(data))

# Display the DataFrame
df.head()
```

Out[35]:

| | _id | Avg | Area Population | Price | |
|---|---|---|---|---|---|
| **0** | 6657e896a2bc56798b9339d6 | {' Area Income': 79545.45857431678, ' Area Hou... | 23086.800503 | 1.059034e+06 | 208 Micha 674\nL |
| **1** | 6657e896a2bc56798b9339d7 | {' Area Income': 79248.64245482568, ' Area Hou... | 40173.072174 | 1.505891e+06 | 188 Jo Suit K |
| **2** | 6657e896a2bc56798b9339d8 | {' Area Income': 61287.067178656784, ' Area Ho... | 36882.159400 | 1.058988e+06 | 91 Stravenue\r |
| **3** | 6657e896a2bc56798b9339d9 | {' Area Income': 63345.24004622798, ' Area Hou... | 34310.242831 | 1.260617e+06 | USS Barn |
| **4** | 6657e896a2bc56798b9339da | {' Area Income': 59982.197225708034, ' Area Ho... | 26354.109472 | 6.309435e+05 | USNS Ray |

In [37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   _id              5000 non-null   object 
 1   Avg              5000 non-null   object 
 2   Area Population  5000 non-null   float64
 3   Price            5000 non-null   float64
 4   Address          5000 non-null   object 
dtypes: float64(2), object(3)
memory usage: 195.4+ KB
```

**Tasks: Retrieve Data:**

Write queries to retrieve the following information:

- Total number of records in the collection.
- total_records = collection.count_documents({})
- print("Total number of records:", total_records)
- Find the maximum and minimum values by using the pipeline
- combine two conditions in the same queries
- use "$gt","lte" in query

db.collection.countDocuments(query, options) Example:

- db. collection.countDocument({}) : To count the number of all documents in the collection.
- count the InvoiceDate great than the date '01/01/2024', limit 100 times

-- collection.countDocuments({"InvoiceDate":{$gt: new Date('01/01/2024')}}, {limit: 100})

## Find the most popular product

In [42]:
```
#Conclusion:  USA-housing most of the items numerical, not so many thing to play ar
```

In [ ]: