ETL Project

Real Estate/Walk Score Listings in Calgary

Objective

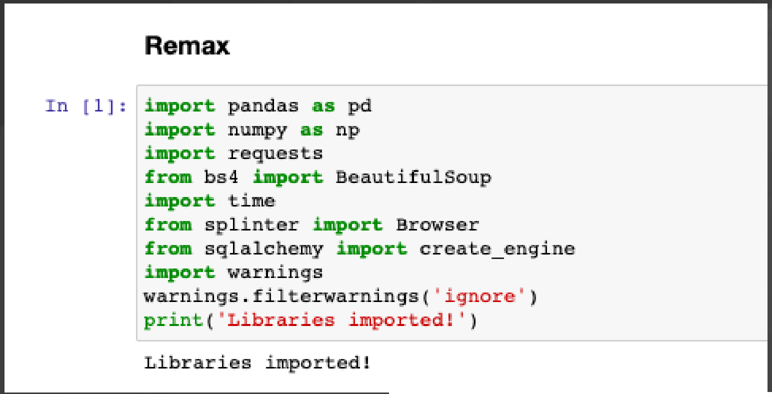
- Extracting Real Estate data in Calgary, Alberta and Walk Scores for corresponding house addresses
- Walk Score: shows a walk/transit/bike score (0-100) for any address to the local downtown
- Transforming retrieved data into easy-to-read tables
- Loading transformed data into relational and non-relational databases for optimal functionality

Data Sources

- Remax Canada: https://www.remax.ca/ab/calgary-real-estate
- Walk Score: https://www.walkscore.com/CA-AB/Calgary

Extract

Scraping Calgary Real Estate Data



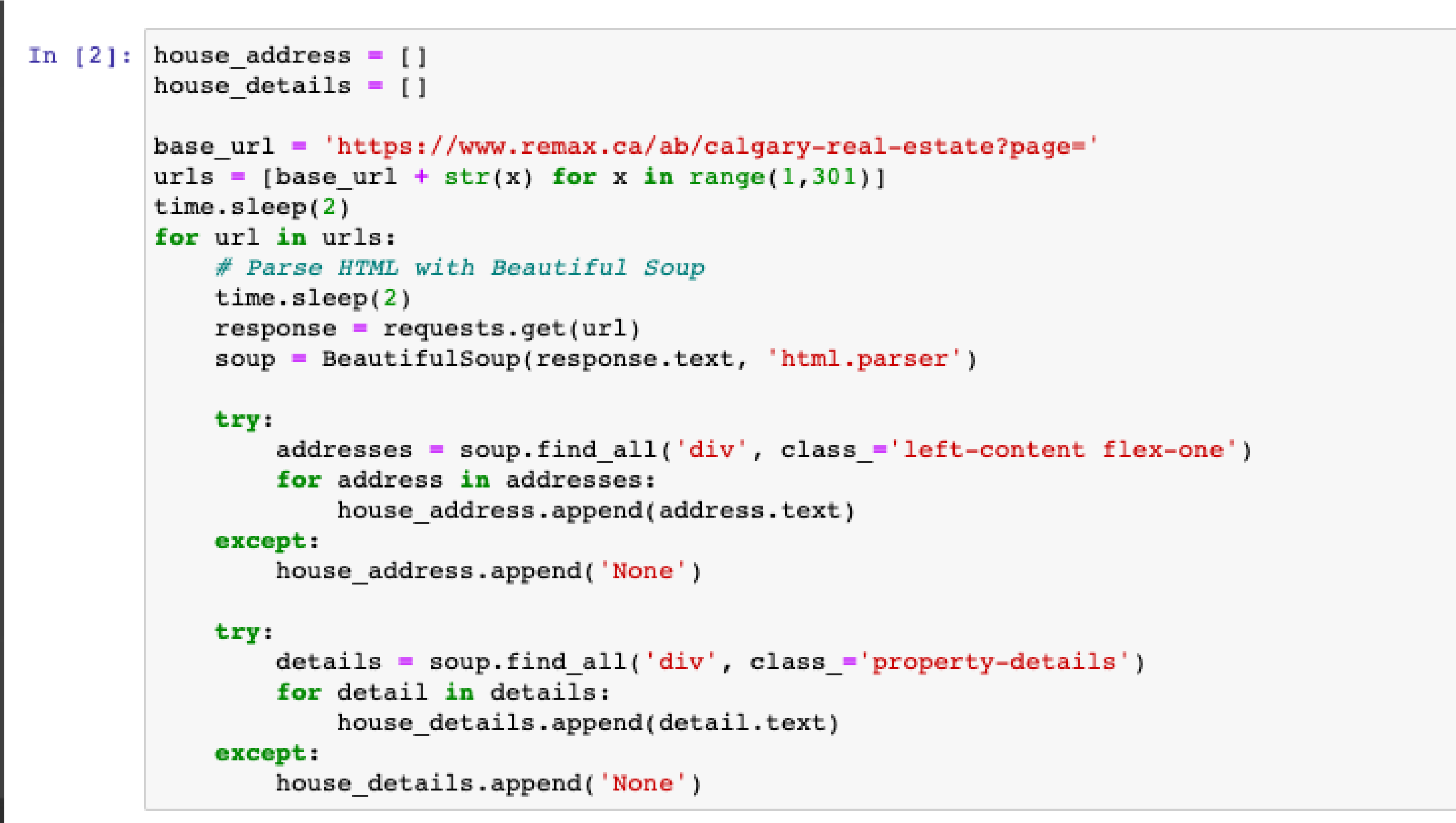
Remax

In [1]:

```
import pandas as pd
import numpy as np
import requests
from bs4 import BeautifulSoup
import time
from splinter import Browser
from sqlalchemy import create_engine
import warnings
warnings.filterwarnings('ignore')
print('Libraries imported!')
```

```
Libraries imported!
```

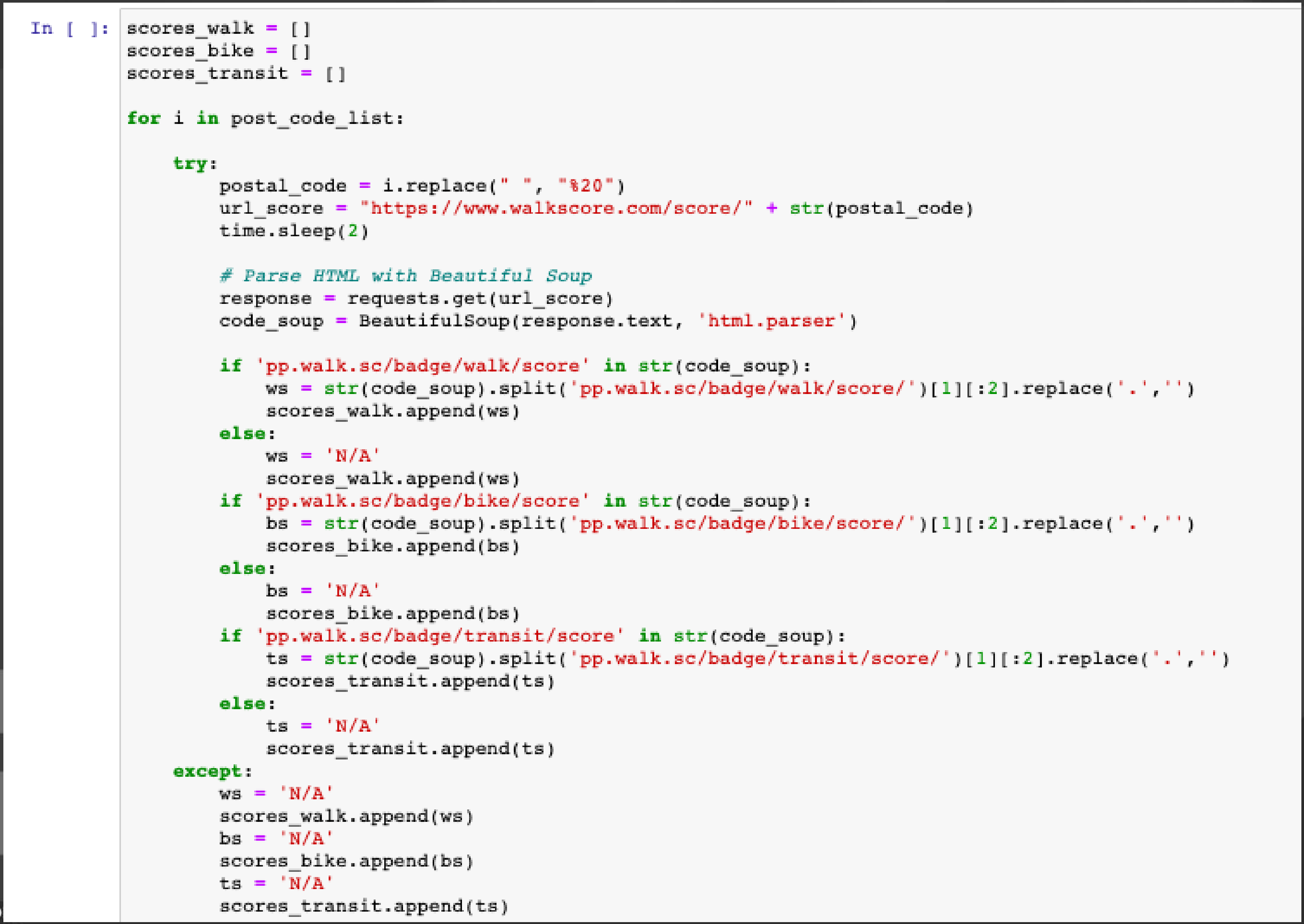
Using BeautifulSoup to scrape property details (house address, house details).



In [2]:

```
house_address = []
house_details = []

base_url = 'https://www.remax.ca/ab/calgary-real-estate?page='
urls = [base_url + str(x) for x in range(1,301)]
time.sleep(2)
for url in urls:
    # Parse HTML with Beautiful Soup
    time.sleep(2)
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    try:
        addresses = soup.find_all('div', class_='left-content flex-one')
        for address in addresses:
            house_address.append(address.text)
    except:
        house_address.append('None')

    try:
        details = soup.find_all('div', class_='property-details')
        for detail in details:
            house_details.append(detail.text)
    except:
        house_details.append('None')
```

Scraping Walk Score Data

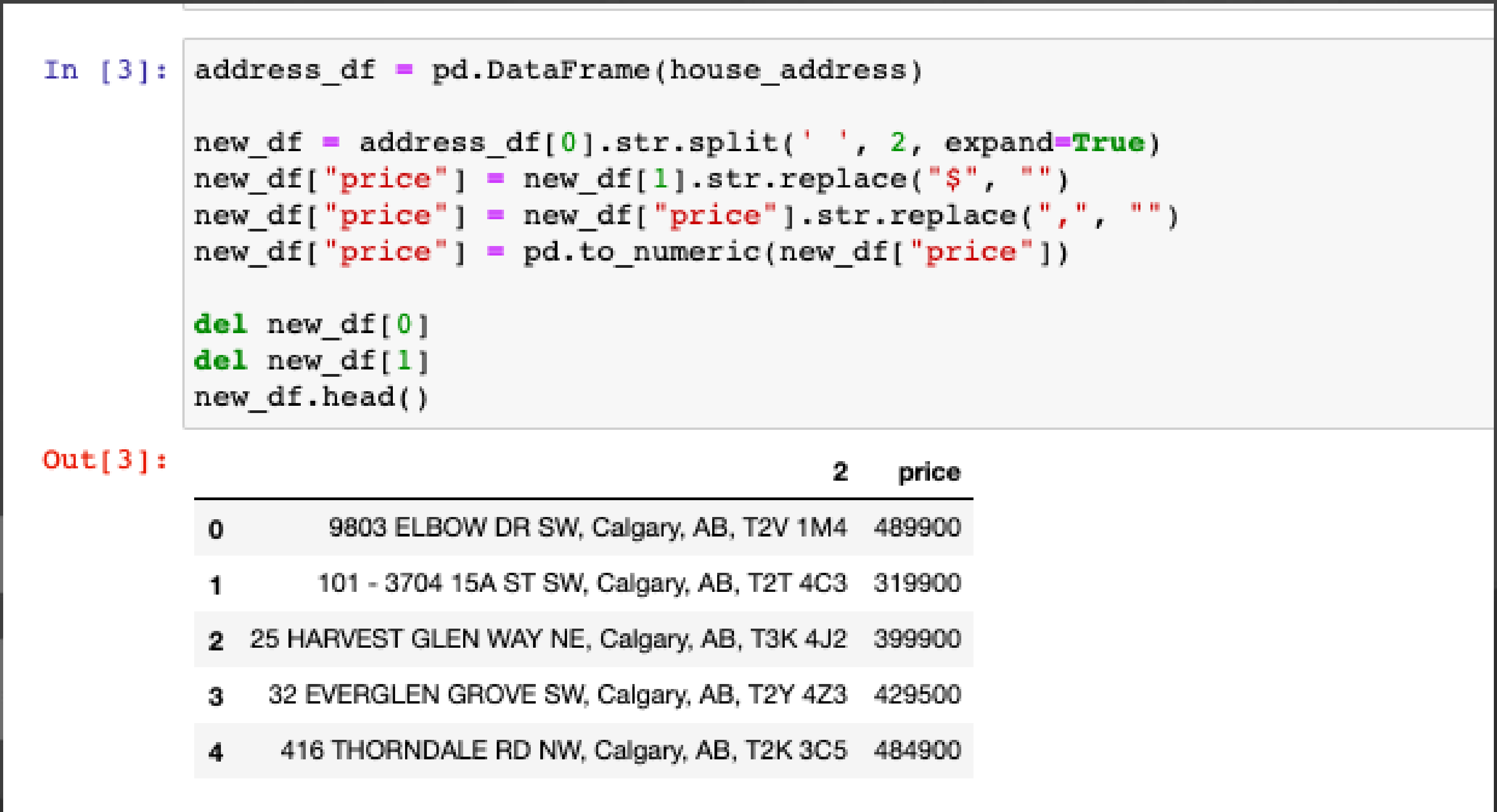


```
In [ ]: scores_walk = []
        scores_bike = []
        scores_transit = []

        for i in post_code_list:

            try:
                postal_code = i.replace(" ", "%20")
                url_score = "https://www.walkscore.com/score/" + str(postal_code)
                time.sleep(2)

                # Parse HTML with Beautiful Soup
                response = requests.get(url_score)
                code_soup = BeautifulSoup(response.text, 'html.parser')

                if 'pp.walk.sc/badge/walk/score' in str(code_soup):
                    ws = str(code_soup).split('pp.walk.sc/badge/walk/score/')[1][:2].replace('.','')
                    scores_walk.append(ws)
                else:
                    ws = 'N/A'
                    scores_walk.append(ws)
                if 'pp.walk.sc/badge/bike/score' in str(code_soup):
                    bs = str(code_soup).split('pp.walk.sc/badge/bike/score/')[1][:2].replace('.','')
                    scores_bike.append(bs)
                else:
                    bs = 'N/A'
                    scores_bike.append(bs)
                if 'pp.walk.sc/badge/transit/score' in str(code_soup):
                    ts = str(code_soup).split('pp.walk.sc/badge/transit/score/')[1][:2].replace('.','')
                    scores_transit.append(ts)
                else:
                    ts = 'N/A'
                    scores_transit.append(ts)
            except:
                ws = 'N/A'
                scores_walk.append(ws)
                bs = 'N/A'
                scores_bike.append(bs)
                ts = 'N/A'
                scores_transit.append(ts)
```

Transform

Cleaning the Calgary Real Estate Data

First dataframe: Address and Price details



In [3]:

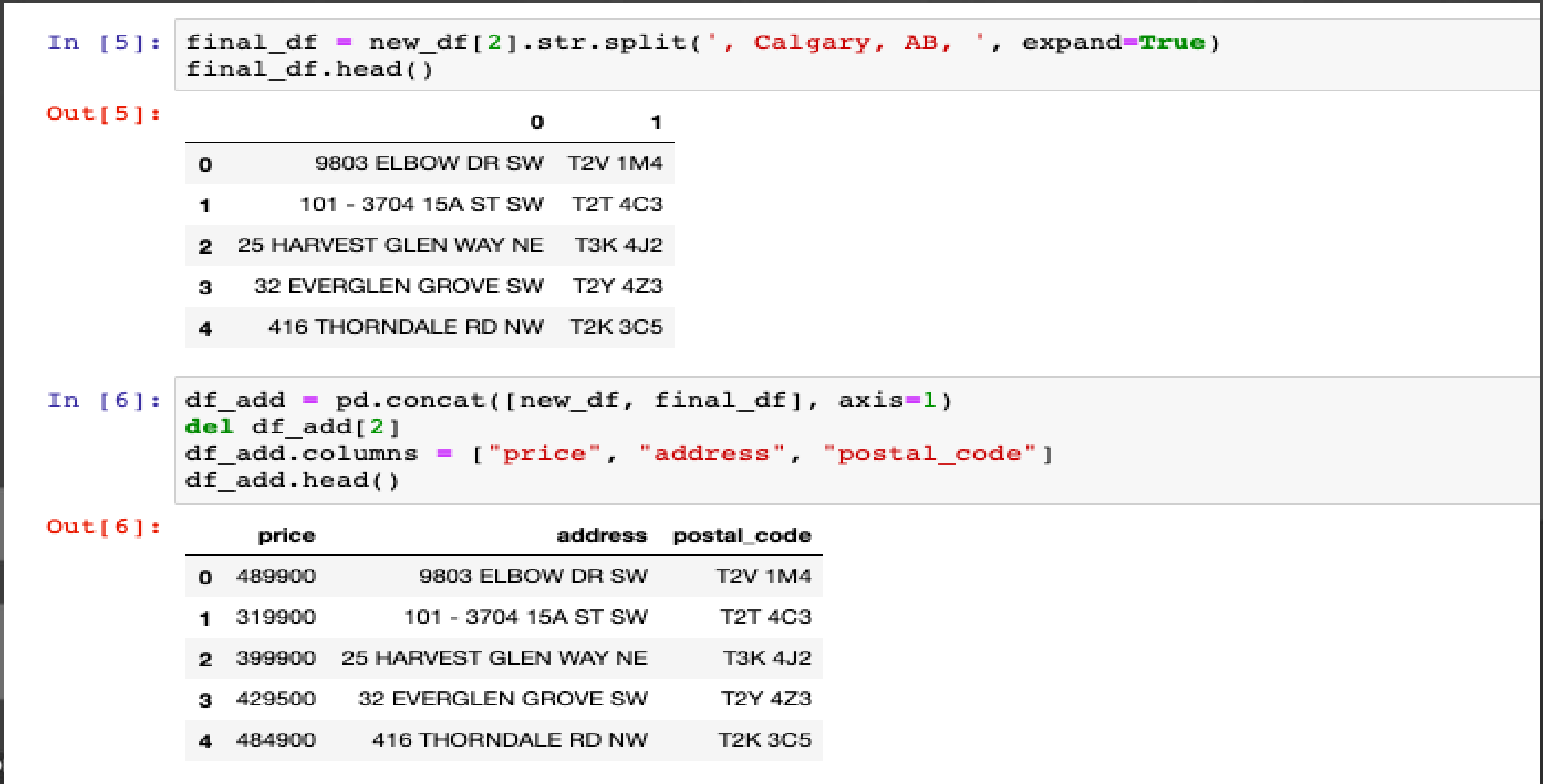
```
address_df = pd.DataFrame(house_address)

new_df = address_df[0].str.split(' ', 2, expand=True)
new_df["price"] = new_df[1].str.replace("$", "")
new_df["price"] = new_df["price"].str.replace(",", "")
new_df["price"] = pd.to_numeric(new_df["price"])

del new_df[0]
del new_df[1]
new_df.head()
```

Out[3]:

```
                                            2    price
0        9803 ELBOW DR SW, Calgary, AB, T2V 1M4   489900
1       101 - 3704 15A ST SW, Calgary, AB, T2T 4C3   319900
2  25 HARVEST GLEN WAY NE, Calgary, AB, T3K 4J2   399900
3    32 EVERGLEN GROVE SW, Calgary, AB, T2Y 4Z3   429500
4     416 THORNDALE RD NW, Calgary, AB, T2K 3C5   484900
```

Cleaning the Calgary Real Estate Data

- Values separated into columns: price, address, postal code, bedrooms, bath, property type
- Price column type changed to integer



In [5]:
```python
final_df = new_df[2].str.split(', Calgary, AB, ', expand=True)
final_df.head()
```

Out[5]:

	0	1
0	9803 ELBOW DR SW	T2V 1M4
1	101 - 3704 15A ST SW	T2T 4C3
2	25 HARVEST GLEN WAY NE	T3K 4J2
3	32 EVERGLEN GROVE SW	T2Y 4Z3
4	416 THORNDALE RD NW	T2K 3C5

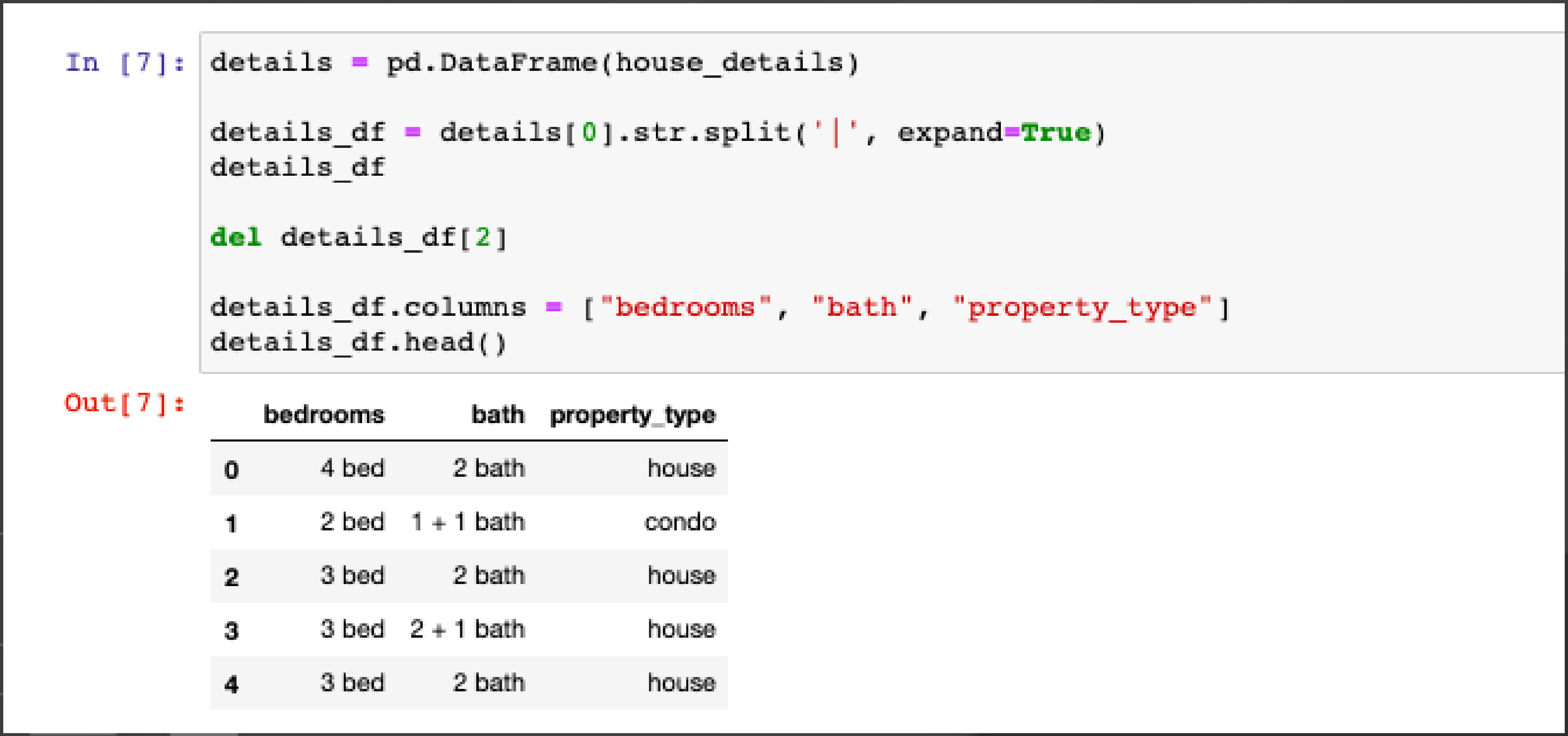
In [6]:
```python
df_add = pd.concat([new_df, final_df], axis=1)
del df_add[2]
df_add.columns = ["price", "address", "postal_code"]
df_add.head()
```

Out[6]:

	price	address	postal_code
0	489900	9803 ELBOW DR SW	T2V 1M4
1	319900	101 - 3704 15A ST SW	T2T 4C3
2	399900	25 HARVEST GLEN WAY NE	T3K 4J2
3	429500	32 EVERGLEN GROVE SW	T2Y 4Z3
4	484900	416 THORNDALE RD NW	T2K 3C5

Cleaning the Calgary Real Estate Data

- Second dataframe: House details



In [7]:

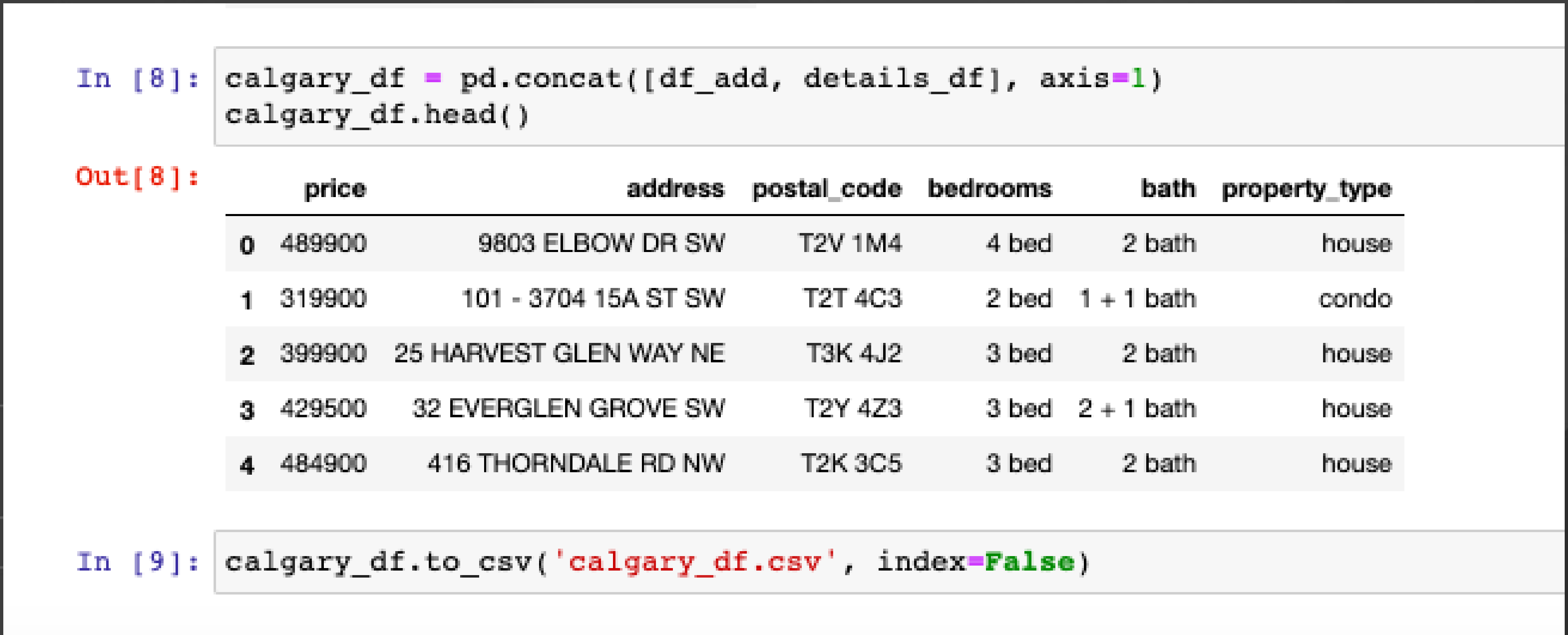
```
details = pd.DataFrame(house_details)

details_df = details[0].str.split('|', expand=True)
details_df

del details_df[2]

details_df.columns = ["bedrooms", "bath", "property_type"]
details_df.head()
```

Out[7]:

	bedrooms	bath	property_type
0	4 bed	2 bath	house
1	2 bed	1 + 1 bath	condo
2	3 bed	2 bath	house
3	3 bed	2 + 1 bath	house
4	3 bed	2 bath	house

Joining the Calgary Real Estate Dataframes

- Concatenating House Address/Price details and House details dataframes.



In [8]:
```
calgary_df = pd.concat([df_add, details_df], axis=1)
calgary_df.head()
```

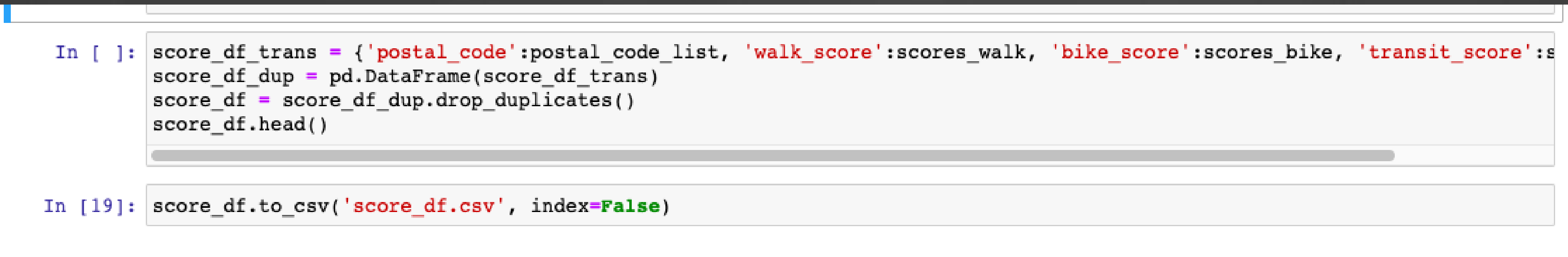
Out[8]:

	price	address	postal_code	bedrooms	bath	property_type
0	489900	9803 ELBOW DR SW	T2V 1M4	4 bed	2 bath	house
1	319900	101 - 3704 15A ST SW	T2T 4C3	2 bed	1 + 1 bath	condo
2	399900	25 HARVEST GLEN WAY NE	T3K 4J2	3 bed	2 bath	house
3	429500	32 EVERGLEN GROVE SW	T2Y 4Z3	3 bed	2 + 1 bath	house
4	484900	416 THORNDALE RD NW	T2K 3C5	3 bed	2 bath	house

In [9]:
```
calgary_df.to_csv('calgary_df.csv', index=False)
```

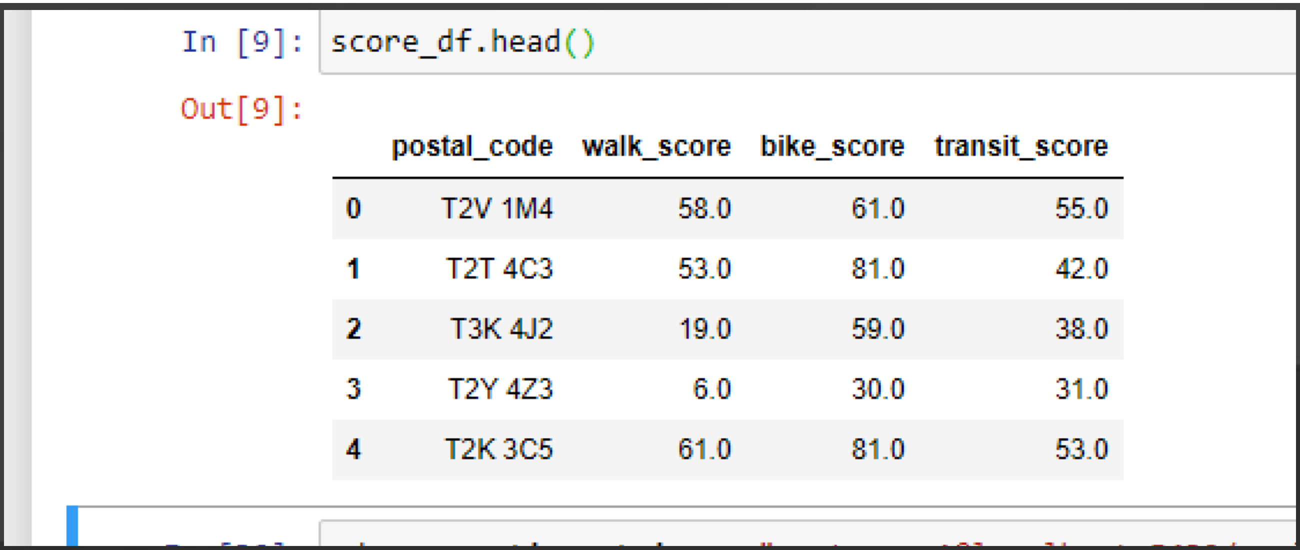
Cleaning the Walk Score Data

- Data converted into dataframe and columns named.



```
In [ ]: score_df_trans = {'postal_code':postal_code_list, 'walk_score':scores_walk, 'bike_score':scores_bike, 'transit_score':s
        score_df_dup = pd.DataFrame(score_df_trans)
        score_df = score_df_dup.drop_duplicates()
        score_df.head()
```

```
In [19]: score_df.to_csv('score_df.csv', index=False)
```



```
In [9]: score_df.head()
```

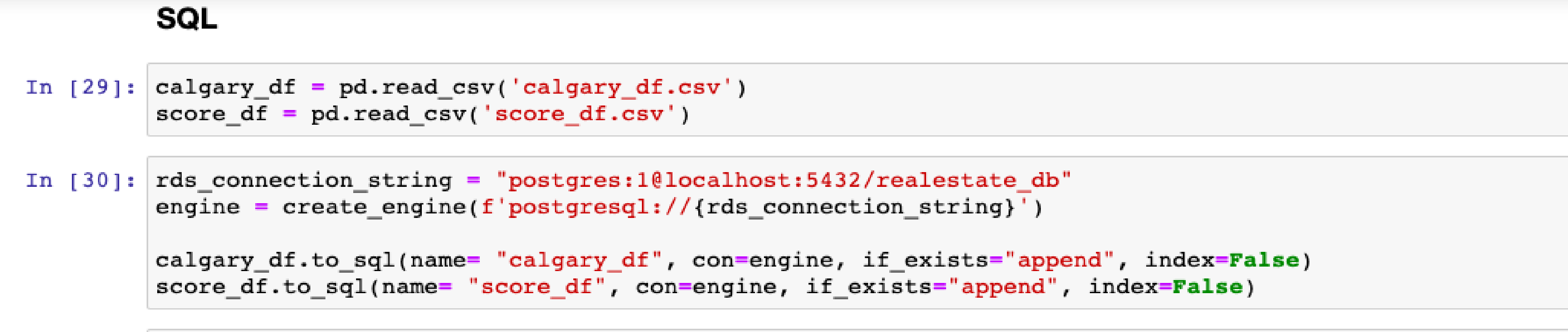
Out[9]:

	postal_code	walk_score	bike_score	transit_score
0	T2V 1M4	58.0	61.0	55.0
1	T2T 4C3	53.0	81.0	42.0
2	T3K 4J2	19.0	59.0	38.0
3	T2Y 4Z3	6.0	30.0	31.0
4	T2K 3C5	61.0	81.0	53.0

Load

Loading Data to Databases

- Building connection to PostgreSQL/MongoDB and loading transformed data

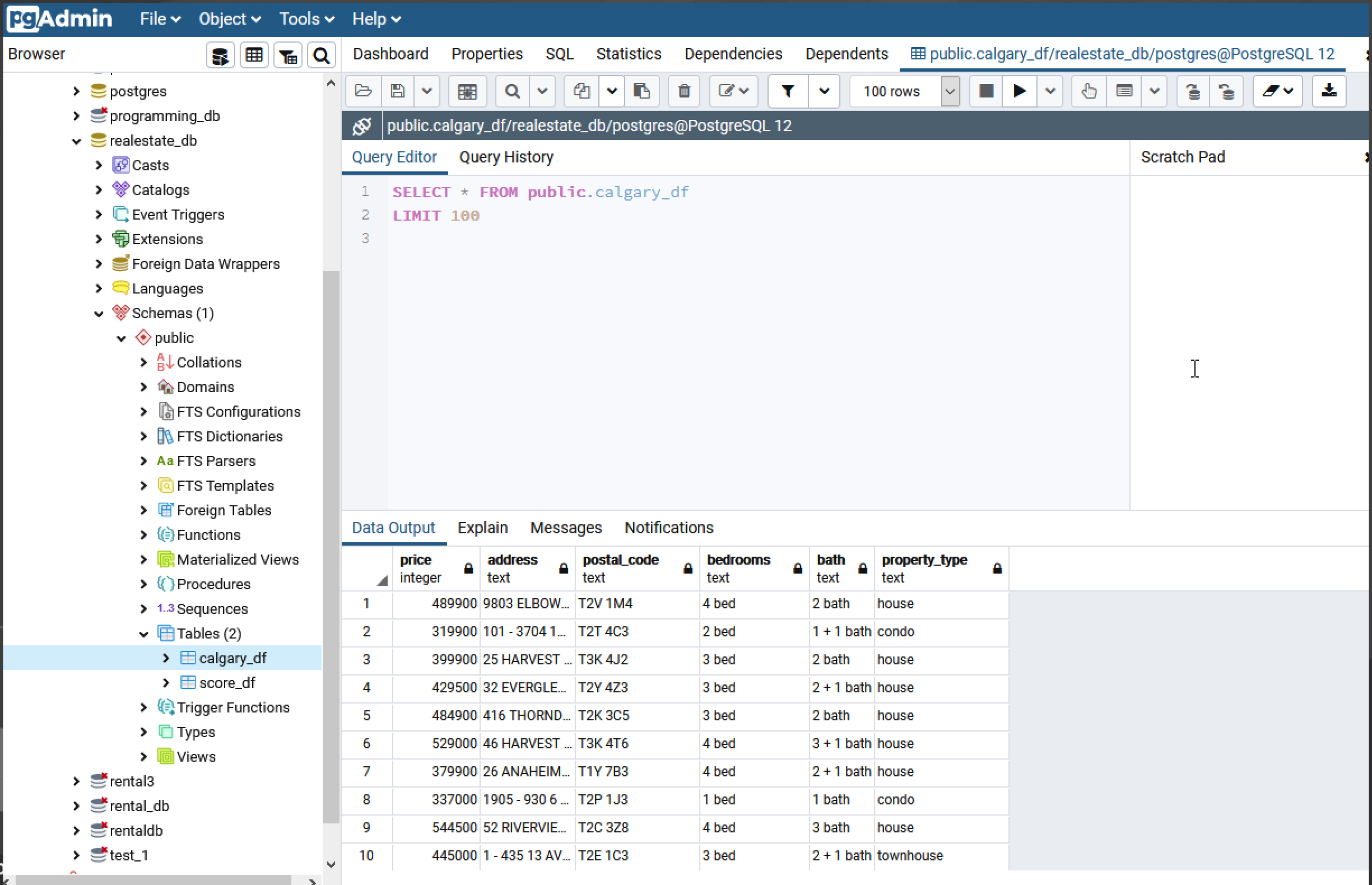


SQL

```
In [29]: calgary_df = pd.read_csv('calgary_df.csv')
         score_df = pd.read_csv('score_df.csv')
```

```
In [30]: rds_connection_string = "postgres:1@localhost:5432/realestate_db"
         engine = create_engine(f'postgresql://{rds_connection_string}')

         calgary_df.to_sql(name= "calgary_df", con=engine, if_exists="append", index=False)
         score_df.to_sql(name= "score_df", con=engine, if_exists="append", index=False)
```

PostgreSQL Database

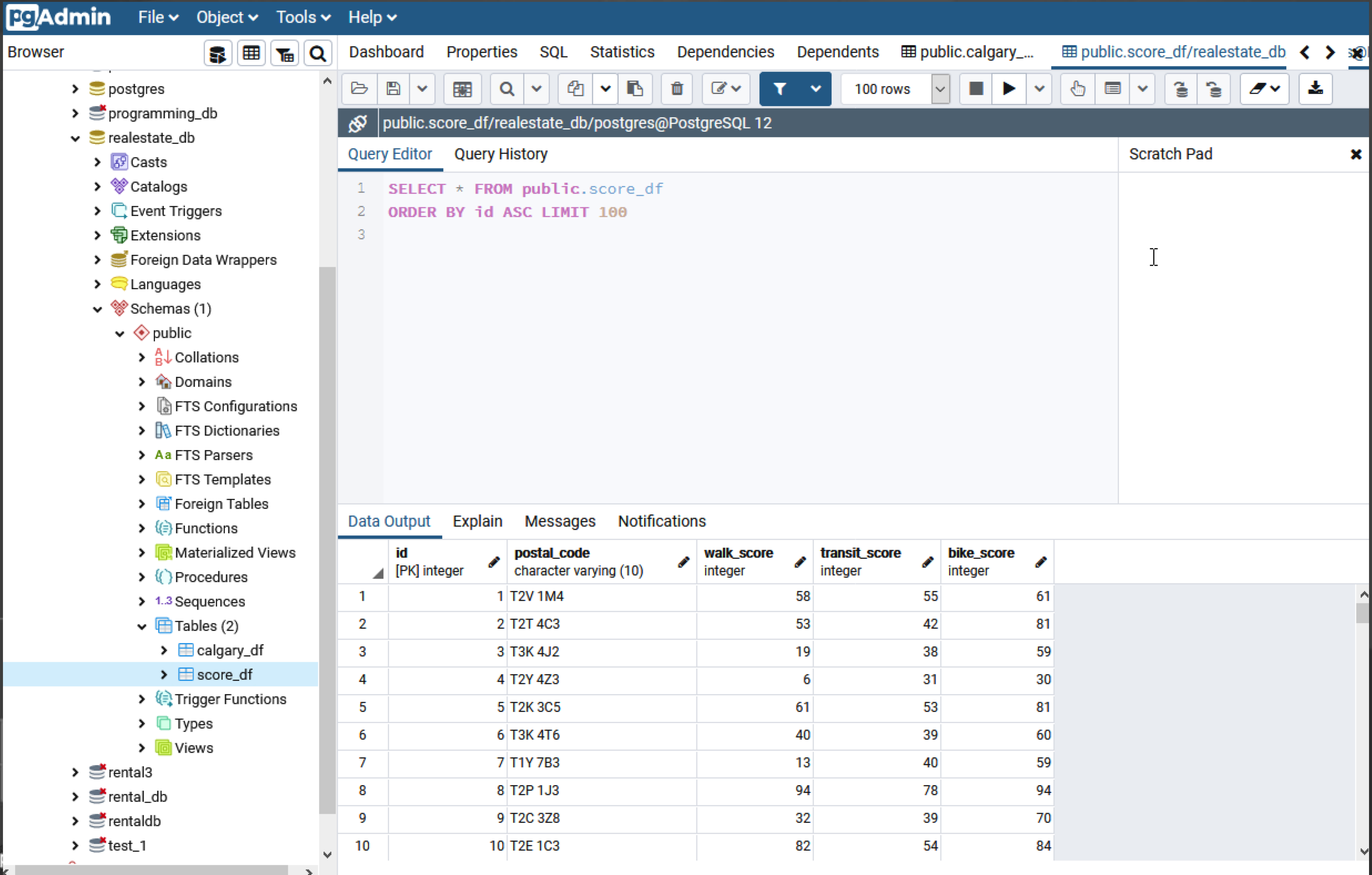


```sql
SELECT * FROM public.calgary_df
LIMIT 100
```

	price integer	address text	postal_code text	bedrooms text	bath text	property_type text
1	489900	9803 ELBOW...	T2V 1M4	4 bed	2 bath	house
2	319900	101 - 3704 1...	T2T 4C3	2 bed	1 + 1 bath	condo
3	399900	25 HARVEST ...	T3K 4J2	3 bed	2 bath	house
4	429500	32 EVERGLE...	T2Y 4Z3	3 bed	2 + 1 bath	house
5	484900	416 THORND...	T2K 3C5	3 bed	2 bath	house
6	529000	46 HARVEST ...	T3K 4T6	4 bed	3 + 1 bath	house
7	379900	26 ANAHEIM...	T1Y 7B3	4 bed	2 + 1 bath	house
8	337000	1905 - 930 6 ...	T2P 1J3	1 bed	1 bath	condo
9	544500	52 RIVERVIE...	T2C 3Z8	4 bed	3 bath	house
10	445000	1 - 435 13 AV...	T2E 1C3	3 bed	2 + 1 bath	townhouse

fpp

PostgreSQL Database

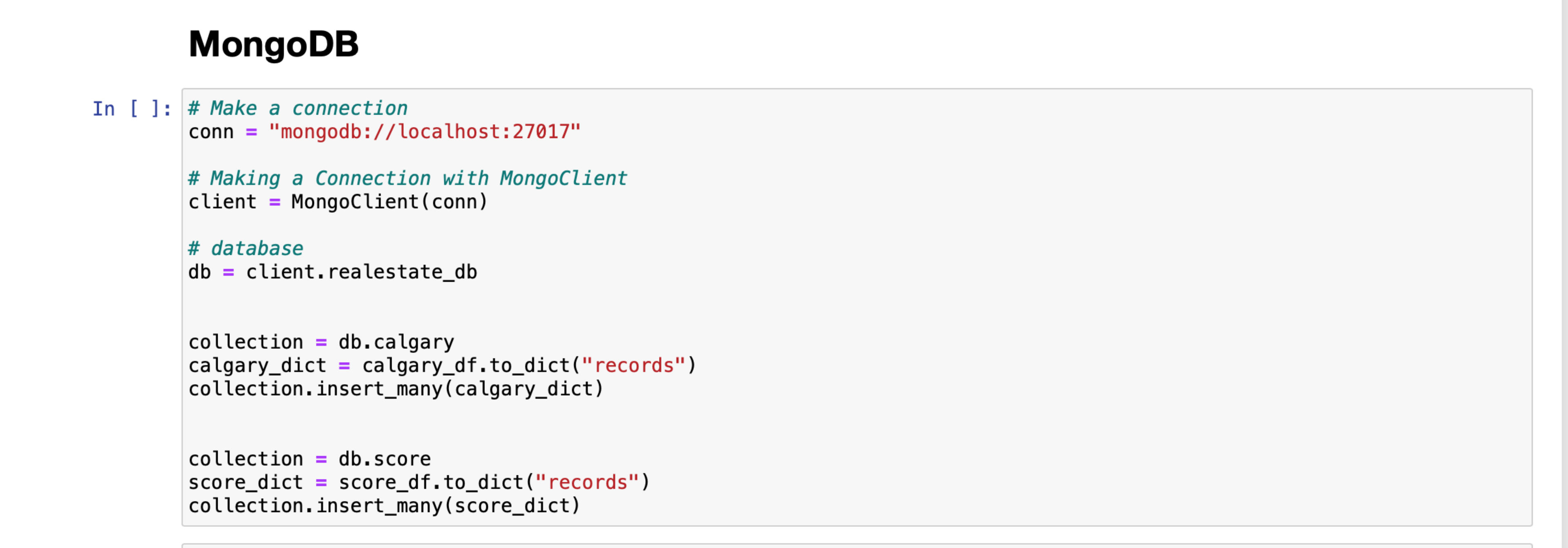


public.score_df/realestate_db/postgres@PostgreSQL 12

```sql
SELECT * FROM public.score_df
ORDER BY id ASC LIMIT 100
```

	id [PK] integer	postal_code character varying (10)	walk_score integer	transit_score integer	bike_score integer
1	1	T2V 1M4	58	55	61
2	2	T2T 4C3	53	42	81
3	3	T3K 4J2	19	38	59
4	4	T2Y 4Z3	6	31	30
5	5	T2K 3C5	61	53	81
6	6	T3K 4T6	40	39	60
7	7	T1Y 7B3	13	40	59
8	8	T2P 1J3	94	78	94
9	9	T2C 3Z8	32	39	70
10	10	T2E 1C3	82	54	84

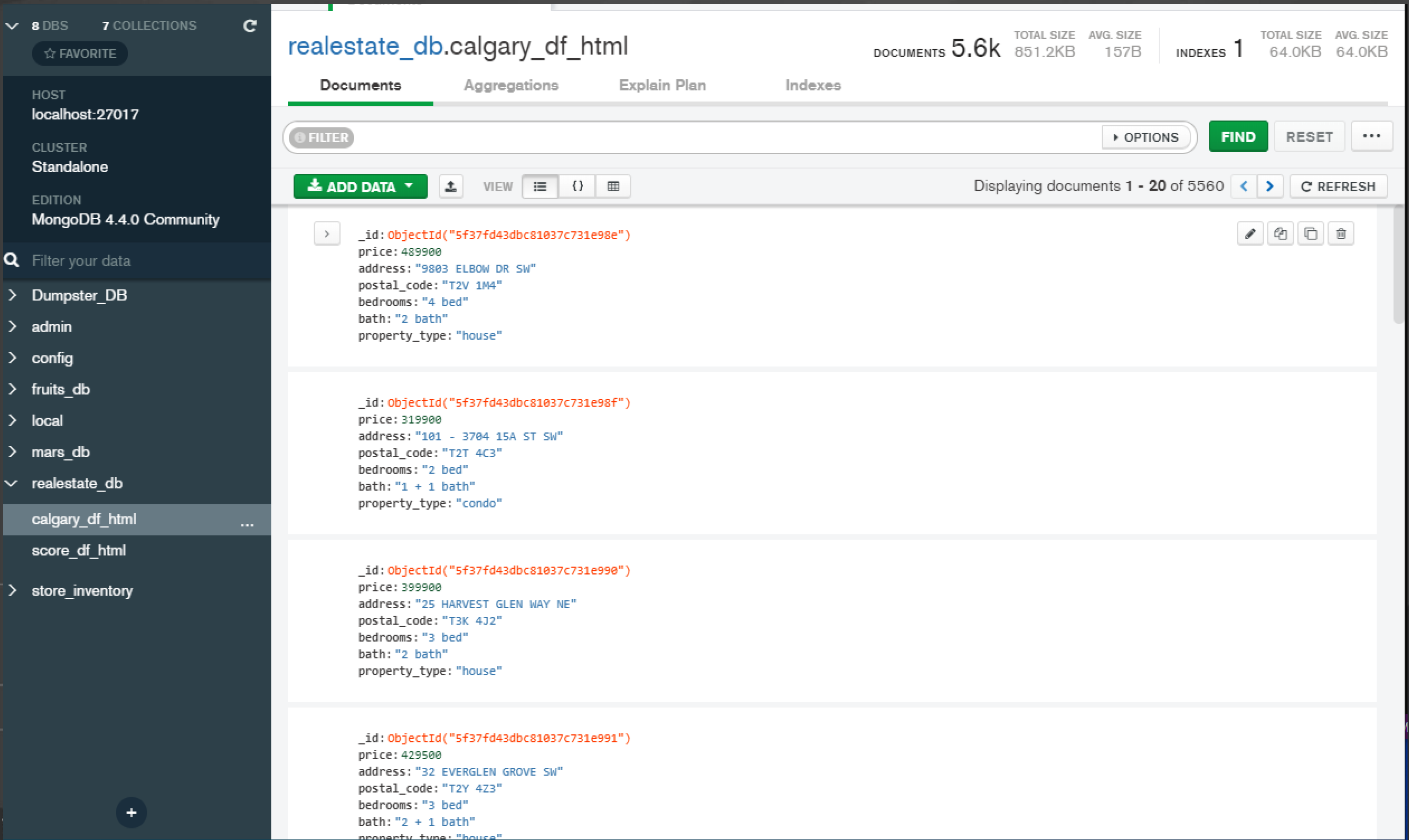
MongoDB Database



MongoDB

```
In [ ]: # Make a connection
conn = "mongodb://localhost:27017"

# Making a Connection with MongoClient
client = MongoClient(conn)

# database
db = client.realestate_db


collection = db.calgary
calgary_dict = calgary_df.to_dict("records")
collection.insert_many(calgary_dict)


collection = db.score
score_dict = score_df.to_dict("records")
collection.insert_many(score_dict)
```

MongoDB Database



HOST
localhost:27017

CLUSTER
Standalone

EDITION
MongoDB 4.4.0 Community

- Dumpster_DB
- admin
- config
- fruits_db
- local
- mars_db
- realestate_db
  - calgary_df_html
  - score_df_html
- store_inventory

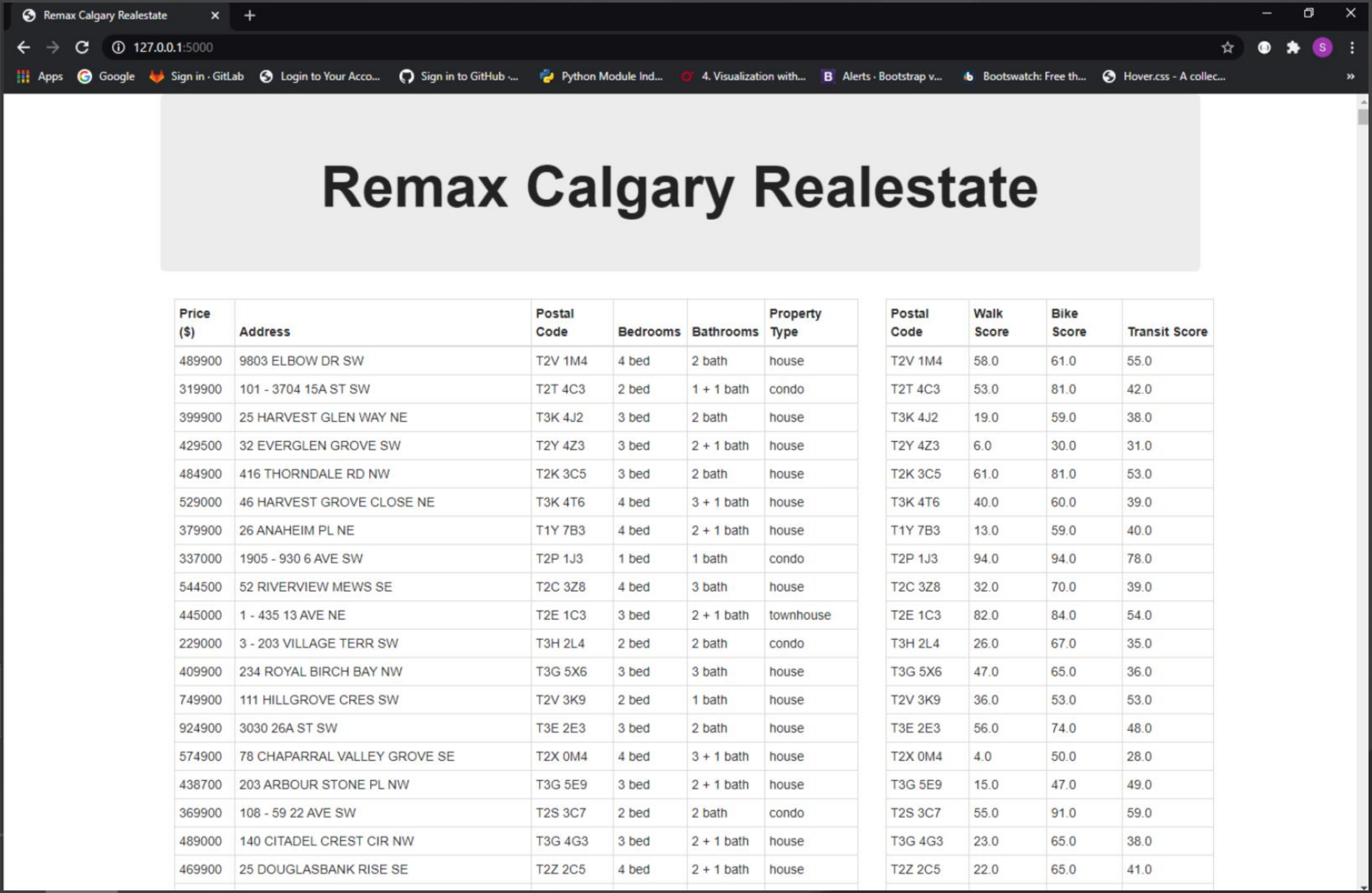
realestate_db.calgary_df_html

DOCUMENTS 5.6k TOTAL SIZE 851.2KB AVG. SIZE 157B INDEXES 1 TOTAL SIZE 64.0KB AVG. SIZE 64.0KB

Displaying documents 1 - 20 of 5560

```
_id: ObjectId("5f37fd43dbc81037c731e98e")
price: 489900
address: "9803 ELBOW DR SW"
postal_code: "T2V 1M4"
bedrooms: "4 bed"
bath: "2 bath"
property_type: "house"

_id: ObjectId("5f37fd43dbc81037c731e98f")
price: 319900
address: "101 - 3704 15A ST SW"
postal_code: "T2T 4C3"
bedrooms: "2 bed"
bath: "1 + 1 bath"
property_type: "condo"

_id: ObjectId("5f37fd43dbc81037c731e990")
price: 399900
address: "25 HARVEST GLEN WAY NE"
postal_code: "T3K 4J2"
bedrooms: "3 bed"
bath: "2 bath"
property_type: "house"

_id: ObjectId("5f37fd43dbc81037c731e991")
price: 429500
address: "32 EVERGLEN GROVE SW"
postal_code: "T2Y 4Z3"
bedrooms: "3 bed"
bath: "2 + 1 bath"
property_type: "house"
```

Converted Database into a Web Based Application



Remax Calgary Realestate

Price ($)	Address	Postal Code	Bedrooms	Bathrooms	Property Type
489900	9803 ELBOW DR SW	T2V 1M4	4 bed	2 bath	house
319900	101 - 3704 15A ST SW	T2T 4C3	2 bed	1 + 1 bath	condo
399900	25 HARVEST GLEN WAY NE	T3K 4J2	3 bed	2 bath	house
429500	32 EVERGLEN GROVE SW	T2Y 4Z3	3 bed	2 + 1 bath	house
484900	416 THORNDALE RD NW	T2K 3C5	3 bed	2 bath	house
529000	46 HARVEST GROVE CLOSE NE	T3K 4T6	4 bed	3 + 1 bath	house
379900	26 ANAHEIM PL NE	T1Y 7B3	4 bed	2 + 1 bath	house
337000	1905 - 930 6 AVE SW	T2P 1J3	1 bed	1 bath	condo
544500	52 RIVERVIEW MEWS SE	T2C 3Z8	4 bed	3 bath	house
445000	1 - 435 13 AVE NE	T2E 1C3	3 bed	2 + 1 bath	townhouse
229000	3 - 203 VILLAGE TERR SW	T3H 2L4	2 bed	2 bath	condo
409900	234 ROYAL BIRCH BAY NW	T3G 5X6	3 bed	3 bath	house
749900	111 HILLGROVE CRES SW	T2V 3K9	2 bed	1 bath	house
924900	3030 26A ST SW	T3E 2E3	3 bed	2 bath	house
574900	78 CHAPARRAL VALLEY GROVE SE	T2X 0M4	4 bed	3 + 1 bath	house
438700	203 ARBOUR STONE PL NW	T3G 5E9	3 bed	2 + 1 bath	house
369900	108 - 59 22 AVE SW	T2S 3C7	2 bed	2 bath	condo
489000	140 CITADEL CREST CIR NW	T3G 4G3	3 bed	2 + 1 bath	house
469900	25 DOUGLASBANK RISE SE	T2Z 2C5	4 bed	2 + 1 bath	house

Postal Code	Walk Score	Bike Score	Transit Score
T2V 1M4	58.0	61.0	55.0
T2T 4C3	53.0	81.0	42.0
T3K 4J2	19.0	59.0	38.0
T2Y 4Z3	6.0	30.0	31.0
T2K 3C5	61.0	81.0	53.0
T3K 4T6	40.0	60.0	39.0
T1Y 7B3	13.0	59.0	40.0
T2P 1J3	94.0	94.0	78.0
T2C 3Z8	32.0	70.0	39.0
T2E 1C3	82.0	84.0	54.0
T3H 2L4	26.0	67.0	35.0
T3G 5X6	47.0	65.0	36.0
T2V 3K9	36.0	53.0	53.0
T3E 2E3	56.0	74.0	48.0
T2X 0M4	4.0	50.0	28.0
T3G 5E9	15.0	47.0	49.0
T2S 3C7	55.0	91.0	59.0
T3G 4G3	23.0	65.0	38.0
T2Z 2C5	22.0	65.0	41.0