DAMG 6210: Data Management and Database Design - Final Database Project

Northeastern University

Project Title: Realty to Reality – The search for your dream home ends now!

Team Members:

- Shashank Dongre, 002747740, dongre.s@northeastern.edu, https://github.com/ShashankDongre
- Riya Virani, 002747048, virani.r@northeastern.edu, https://github.com/riya-virani
- Ojasvi Patel, 002793770, patel.oj@northeastern.edu, https://github.com/Ojasvi1329

- GitHub URL: https://github.com/ShashankDongreNeu/DMDD_Final_Project.git

Technologies used:

- MySQL – Relational Database Management System
- Python - Web Scraping, Data wrangling, Data cleaning, Data auditing
- IDE – PyCharm, Google Collab Notebooks, MySQL Workbench
- Libraries – pandas, NumPy, seaborn, regular expression
- MS Excel – Data extraction, basic formatting, csv, data import/export
- Kaggle: https://www.kaggle.com/datasets/polartech/real-estate-information-in-boston
- MS Office - Project operation and execution

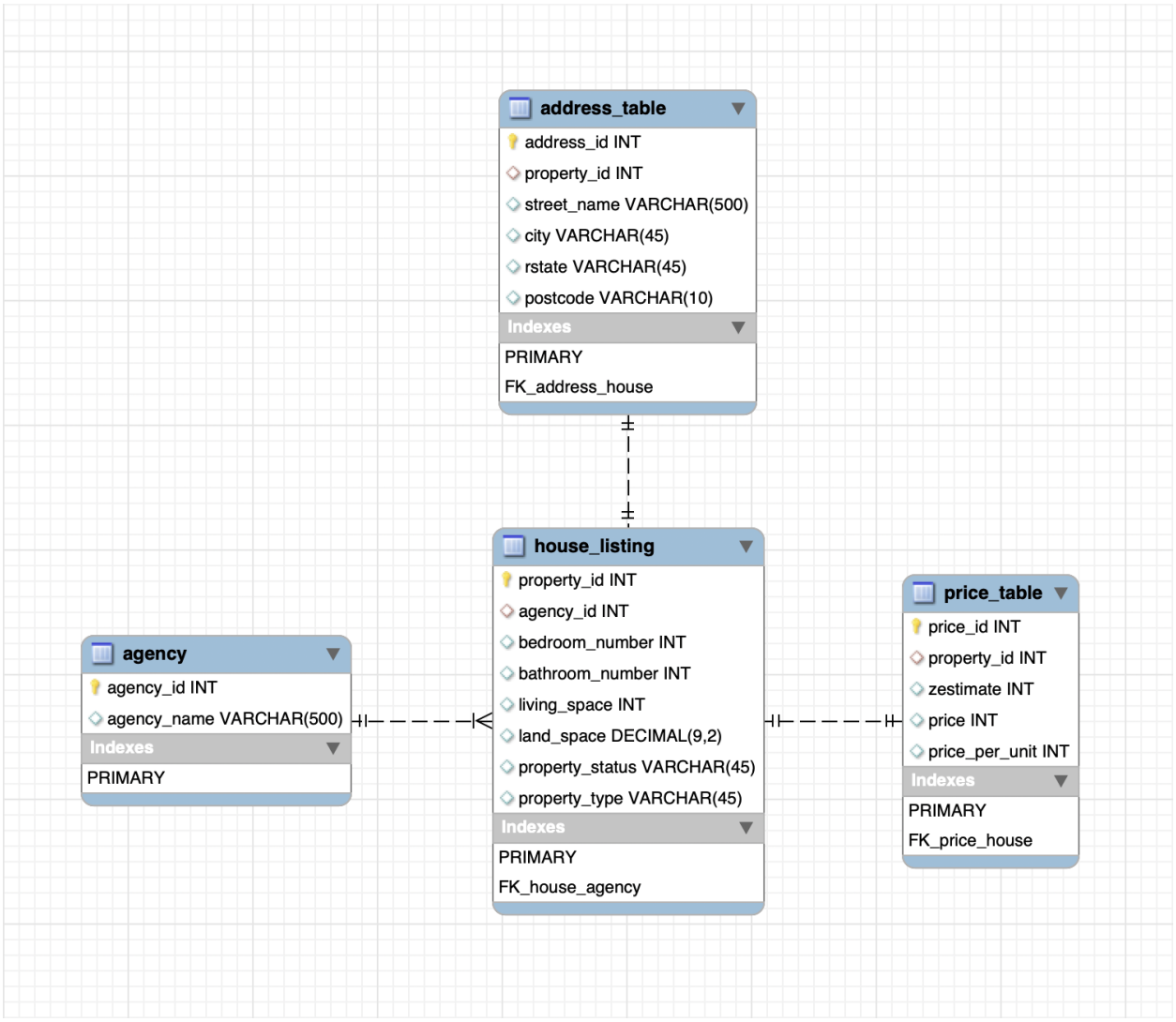
Project Description, Design Considerations, and Project Scope Definition:

Our project topic majorly deals with the sale and listing of properties in the Boston, MA geographical region. The initial questions that we were seeking to find answers to with our database were kept into consideration throughout the entire length of the project. The domain of our choice was Real Estate, as we were seeking to find answers to the question related to the selling of houses in our city. But as time progressed, we slightly shifted our focus from the domain topic to the actual implementation of data management, various technologies, and resources available on the internet helpful in database creation, and effective maintenance of the created database.

We initially framed several questions which would then become our use cases in the later stage. The Twitter scrapper was specific to the listings on Twitter. As we wanted to expose ourselves to the world of data management and database design, we thought the twitter scraping bot proved to help create the foundation of the entire project. It gave us a structure and some foundation on which we could utilize the powerful libraries of python like pandas, matplotlib, seaborn etc. to perform data extraction, cleaning, basic data analysis and data visualization.

The assignment on data cleaning, munging, and data wrangling, also proved to be a great learning opportunity to play with real-world datasets from the web. Again, here we could explore the vast opportunities and functionalities of python and how powerful it can be for data analysis. The data cleaning was performed using python's pandas library. The initial dataset contained several null values, missing values, and inappropriate values in certain columns. We handled the cleaning part using pandas' data cleaning abilities. But in the process, we lost some of the data. Our team decided to move ahead with the cleaned, good-quality data instead of large datasets with inappropriate formatting and missing values. Ultimately, for the creation of the database, we need a level of data which achieves our initial project objectives of data management and the creation of efficient database design.

Entity – Relationship Diagram of the Model:



Explanation on some of the design decisions:

● The ‘House_Listing’ table contains details extracted from Zillow’s webpages of various house listings either by a seller or an agency. In this table the unique attribute i.e., ‘property_id’ is the primary key whereas ‘agency_id’ is the foreign key. This table has data which is essential while buying a house. We carefully extracted the limited number of attributes as data is the most valuable asset of Zillow and hence, they try hard to hide as much data as possible on a single page using various drop-down arrangements and hidden UI features of a webpage. This table contains the following desired attributes: property_id, agency_id, bedroom_number, bathroom_number, living_space, land_space, property_status, and property_type.

● Zestimate: Zillow's assessment of a house's market worth is based on the Zestimate® home valuation methodology. A Zestimate considers the specifics of the house, the neighborhood, and market trends in addition to data from the MLS, public sources, and user submissions. It cannot be used in place of an appraisal since it is not an appraisal. The Zestimate for on-market properties has a median error rate of 3.2% throughout the country, while the Zestimate for off-market homes has a median error rate of 7.52%. The regional data accessibility will determine how accurate the Zestimate is for a certain residence. Some places have access to more specific housing data, such square footage and the number of bedrooms or bathrooms, whereas others do not. The Zestimate number will be more precise the more data there are.

● The ‘Agency’ table is a simple table containing the details regarding the agency name and it has a unique ‘agency_id’ as its primary key. Since this table is linking the agency name to the other tables, it need not have to have a foreign key for this purpose.

●. The ‘Price_table’ contains the extracted details regarding the price and Zestimate of the listed property. In this table, ‘price_id’ is assigned as a primary key. The unique attribute ‘property_id’ is referenced in the address table using the foreign key. This table contains price details such as property_id, price, price per unit of the land, and Zillow’s propriety attribute named ‘Zestimate’.

●. The ‘Address_table’ contains the extracted details regarding the address of the listed property. In this table, ‘address_id’ is assigned as a primary key. The unique attribute ‘property_id’ is referenced in the address table using the foreign key. This table contains address details such as street_name, rstate, city, and postcode.

Observations on the Audit Validity, Accuracy, Completeness, Consistency, Uniformity:

The data that we scraped from Zillow's webpage is specific to Boston city. Kindly note that this data belongs to the time when we scraped it from Zillow's website. This may contain some old listings as well. Some listings may not be present during the time of the review. The basic formatting of the data was conducted in MS Excel, which again filtered data for Boston city in several other states such as there are 16 cities in the United States named as 'Boston'. This implied that the number of records drastically got reduced and hence we were reduced to almost 750 records for Boston, Massachusetts. Hence, for the sake of scope and limitations of the project, our team decided to focus on the creation and maintenance of the database related to Boston, Massachusetts region only. This implied considering only the data, which is relevant for our analysis and queries, making our process of database creation more focused and specific to the domain use cases.

All data cleaning, munging, wrangling was performed in google collab notebook using python and its powerful libraries. The detail code can be found in the python notebook. Below are some of the code snippets from the google collab notebook and the explanation on some of the results:

Python scripts and code snippets from previous assignments:

In the twitter scrapper assignment, we created a bot that would scrape twitter data from twitter using their developer APIs. The config file that contained the twitter API keys and authentication tokens. We used *tweepy*, a popular python package used by many researchers to interact with the twitter API. We scraped few hundred tweets using our specific keywords and analyze that data to form domain-specific use cases. The code editor used was PyCharm. Below there is a code snippet from the twitter scraping assignment.

```python
import configparser
import pandas as pd
import tweepy

# read configs
config = configparser.ConfigParser()
config.read('config.ini')

api_key = config['twitter']['api_key']
api_key_secret = config['twitter']['api_key_secret']

access_token = config['twitter']['access_token']
access_token_secret = config['twitter']['access_token_secret']


# authentication
auth = tweepy.OAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)
# keyword search
keyword = '#houseforsale'
limit=300

tweets = tweepy.Cursor(api.search_tweets, q=keyword, count=100, tweet_mode='extended').items(limit)

# create DataFrame
columns = ['User', 'Tweet', 'Tweet_id', 'Created_At']
data = []

for tweet in tweets:
    data.append([tweet.user.screen_name, tweet.full_text, tweet.id_str, tweet.created_at])

df = pd.DataFrame(data, columns=columns)

# print(df)
df.to_csv('tweets_hashtag_houseforsale_3.csv')
```

In the data scraping, wrangling, cleaning, munging assignment, we were looking for a dataset specific to our domain and geographic location. Luckily, we found one on GitHub which had enough columns for us to perform the required analysis. The dataset contained thousands of records, however, for our specific use cases and geographic locations, we only could use seven hundred plus records. The team discussed this issue with the teaching assistant and concluded that we should proceed with this dataset only as it would be helpful in achieving the course and project objectives.

All data cleaning, munging, wrangling was performed in Google Collab notebook using python and its powerful libraries. Below are some code snippets from the data scraping, wrangling, cleaning, munging assignment.

In [9]:
```
# Data Cleaning 2
data.drop(data.loc[data['zestimate'] == ' '].index, inplace =True)
# Data Cleaning 3
data.drop(data.loc[data['zestimate'] == 'N.A.'].index, inplace =True)
data
```

Out[9]:

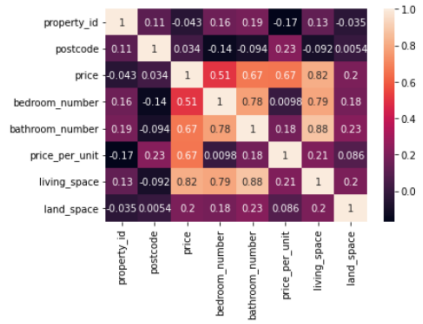
	property_id	address	street_name	city	rstate	postcode	price	zestimate	bedroom_number	bathroom_number	price_per_unit	living_space	land_space	prope
0	59169678	8 Gloucester St APT 12, Boston, MA 02115	Gloucester St APT 12	Boston	MA	2115	1175000	1202100	2.0	2.0	1019.0	1153.0	1306.8	
3	59101641	31 Homestead St #1, Boston, MA 02121	Homestead St #1	Boston	MA	2121	455000	461100	3.0	2.0	369.0	1232.0	1306.8	
7	59171381	386 Riverway APT 2, Boston, MA 02115	Riverway APT 2	Boston	MA	2115	495000	590100	1.0	1.0	529.0	934.0	934.0	
8	189924486	87-89 Nottinghill Rd, Boston, MA 02135	89 Nottinghill Rd	Boston	MA	2135	999000	999000	4.0	4.0	403.0	2478.0	2613.6	MULT
12	118162434	301 Chestnut Ave #1, Boston, MA 02130	Chestnut Ave #1	Boston	MA	2130	799000	814700	2.0	2.0	700.0	1141.0	1306.8	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
702	189975964	352R Blue Hill Ave #10, Boston, MA 02121	Blue Hill Ave #10	Boston	MA	2121	449900	456500	3.0	2.0	407.0	1104.0	1306.8	
703	333606848	304-306 Cummins Hwy #1, Boston, MA 02131	306 Cummins Hwy #1	Boston	MA	2131	450000	461800	2.0	1.0	443.0	1015.0	1015.0	
708	2146690846	39 Old Morton St #2, Boston, MA 02126	Old Morton St #2	Boston	MA	2126	719800	719800	2.0	4.0	492.0	1463.0	7840.8	
711	118324465	4 Woodworth St #1, Boston, MA 02122	Woodworth St #1	Boston	MA	2122	669000	669000	2.0	3.0	375.0	1784.0	1784.0	

In [10]:
```
corelation = data.corr()
print(corelation.columns)
```

```
Index(['property_id', 'postcode', 'price', 'bedroom_number', 'bathroom_number',
       'price_per_unit', 'living_space', 'land_space'],
      dtype='object')
```

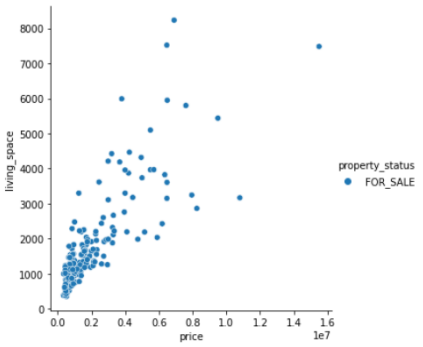
In [11]:
```
sns.heatmap(corelation, xticklabels=corelation.columns, yticklabels=corelation.columns, annot=True)
```

Out[11]:



In [13]:
```
sns.relplot(x= 'price', y='living_space', hue='property_status', data=data)
```

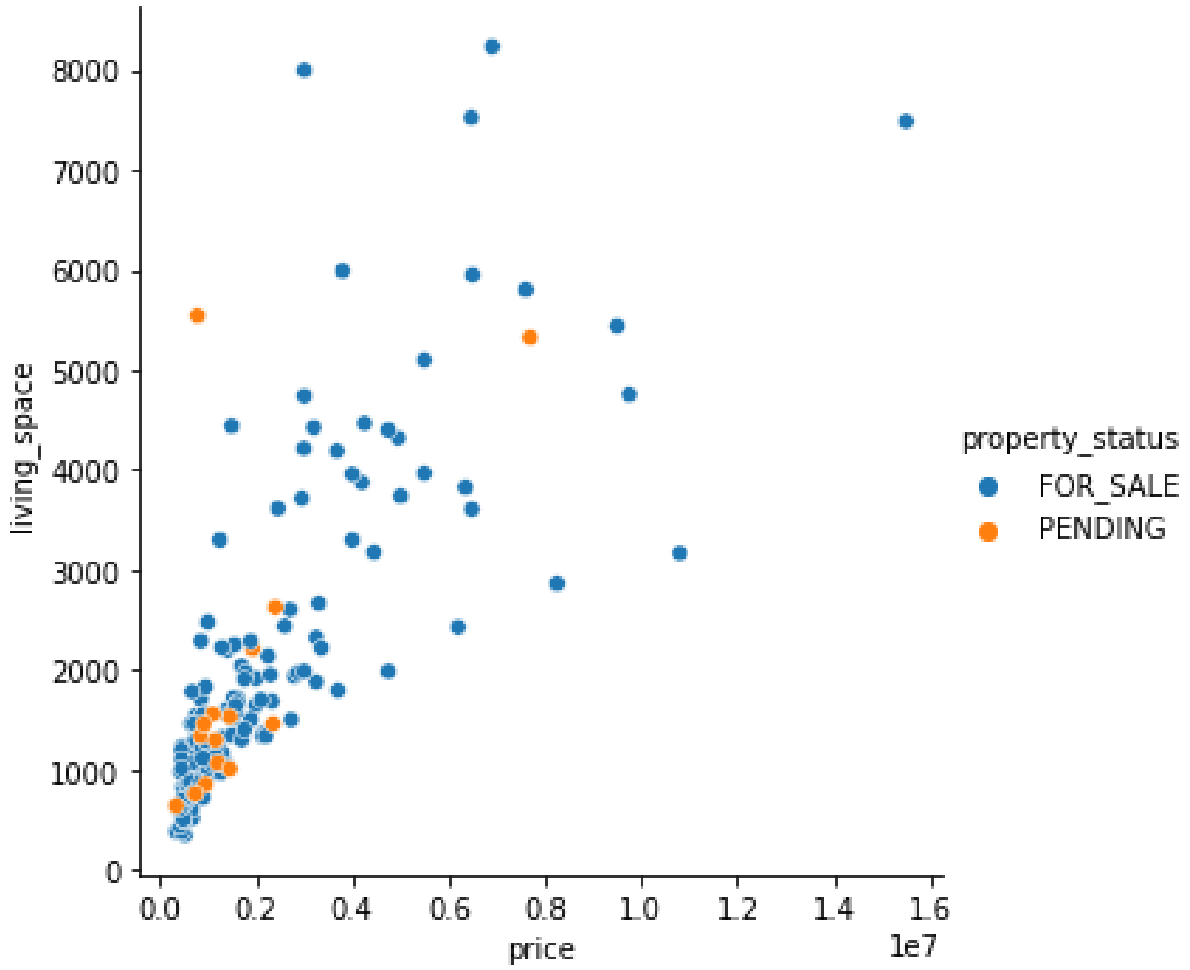
Out[13]:



In [14]:
```
data.describe()
```

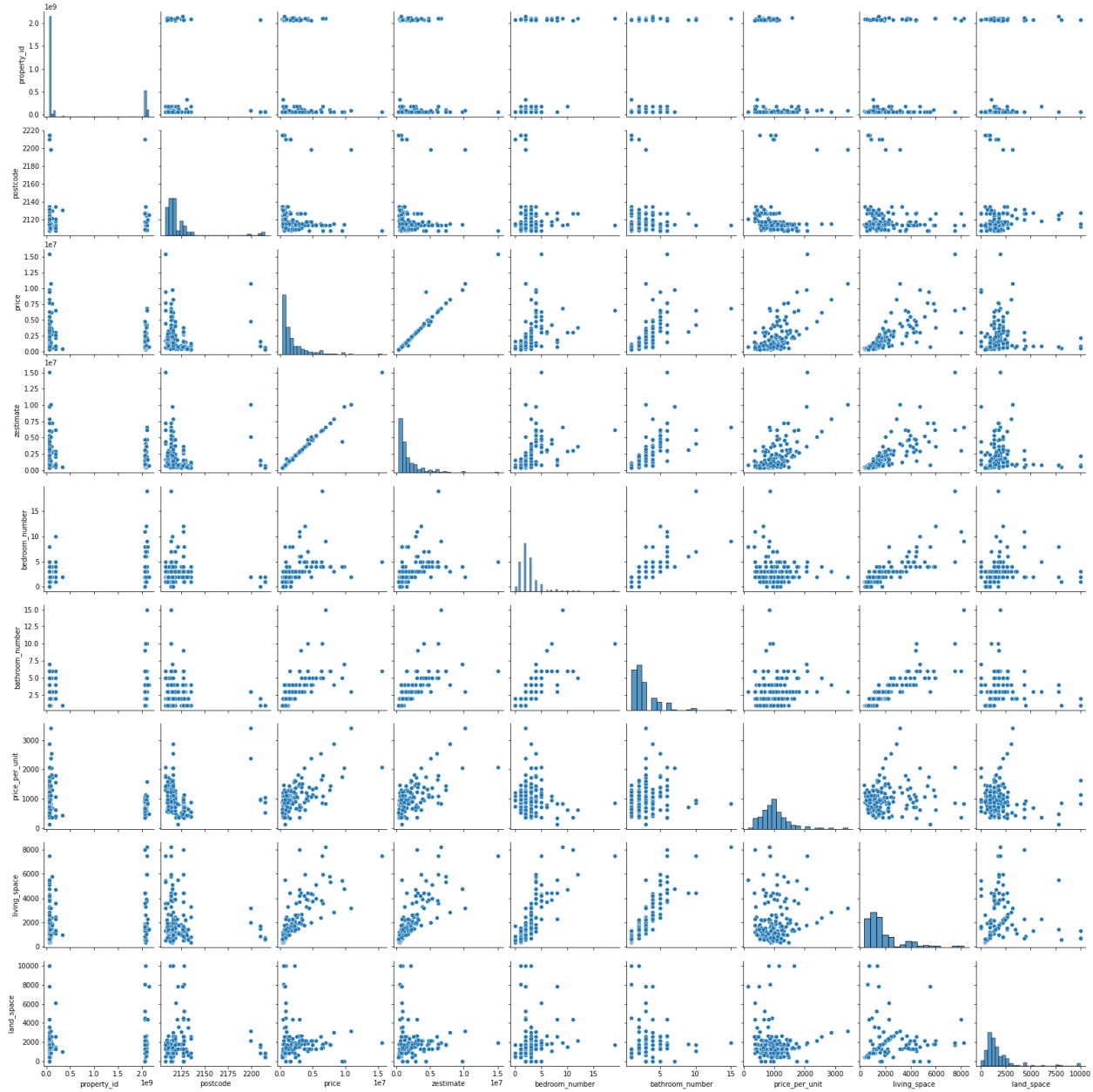
Out[14]:

	property_id	postcode	price	bedroom_number	bathroom_number	price_per_unit	living_space	land_space
count	2.010000e+02	201.000000	2.010000e+02	201.000000	201.000000	201.000000	201.000000	201.000000
mean	6.516190e+08	2123.985075	1.958021e+06	2.412935	2.472637	1060.099502	1717.756219	1540.538607
std	8.936469e+08	24.301127	2.110385e+06	1.906209	1.757981	486.414381	1342.670922	1197.219520
min	5.909188e+07	2108.000000	3.399990e+05	0.000000	0.000000	350.000000	352.000000	0.030000
25%	5.917523e+07	2114.000000	7.100000e+05	1.000000	1.000000	746.000000	913.000000	871.200000
50%	8.185724e+07	2116.000000	1.075000e+06	2.000000	2.000000	963.000000	1265.000000	1302.000000
75%	2.067781e+09	2127.000000	2.295000e+06	3.000000	3.000000	1247.000000	1975.000000	1784.000000
max	2.146691e+09	2215.000000	1.550000e+07	19.000000	15.000000	3409.000000	8232.000000	10018.800000



The above graph is known as a relationship plot, which states the degree of relationship between two chosen attributes. Here, we wanted to view the relationship between price and living space for properties in Boston, MA. The x-axis is for the price (exponential component, USD) and the y-axis is denoted by the living space (sqft) attribute. As we can see the cluster is bigger at, the bottom of the graph which could indicate that most of the properties below 3000 sqft in living space have comparatively less prices as compared to the rest. The properties available for sale are colour coded as blue and the pending properties are colour coded as orange. As we can see there are not many orange properties lying in the higher quartiles of the graph meaning that not many expensive properties are pending to be sold and the expensive ones are for sale as of the

date the data was extracted.



Processes involved in Creation of the Database:

The selection of the domain *Real Estate* for database creation was something of a group consensus, our individual interest, and passion in the domain. In addition to this, the members were excited to see the effects of current global economic condition and the sudden hike in rented properties has affected the selling of properties. With these ideas in our minds, we framed some use cases and went ahead with the subsequent assignments.

We ultimately had to update our use cases and approach in creating the database from scratch pertaining to be more domain specific. The third assignment paved the way for us to get our hands dirty in the MySQL Workbench and try various use cases and queries. We designed the use cases in such a way that they answer the specific questions from the dataset as well as the combination of several other possible iterations of the parameters chosen for the analysis. So, in a way, we tried to achieve our objectives initially set from the project expectations and also created something that can used for data analysis further. The project contains SQL files, tables, and data which can further be used for even more detailed visualization using a powerful visualization tool like Tableau. The integration of this tool with MySQL required us to buy the premium version named Tableau Desktop, which comes with an annual fee but extends support for hundreds of database connectivity options.

For the import of the data into MySQL, we created the required schema and made a parent table which would contain the same number and naming of the columns and their data types. We then imported the data into MySQL using the import tools. In the process we lost some data because of incompatible datatypes and some import tool errors. But the data that got imported was good enough to begin our project.

We then created the respective tables according to the mapping of our data model and inserted the required data into the respective tables using INSERT and SELECT queries. We then altered the datatypes and applied the appropriate constraints of primary and foreign keys to the tables, thereby creating the groundwork for forming relationships. We then applied joins and other aggregate functions in our use cases to further narrow down our analysis. The following are the code snippets of the above process:

SQL - CREATE TABLE STATEMENTS

```
CREATE TABLE `address_table` (
  `address_id` int NOT NULL AUTO_INCREMENT,
  `property_id` int DEFAULT NULL,
  `street_name` varchar(500) DEFAULT NULL,
  `city` varchar(45) DEFAULT NULL,
  `rstate` varchar(45) DEFAULT NULL,
  `postcode` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`address_id`),
  );


CREATE TABLE `agency` (
  `agency_id` int NOT NULL AUTO_INCREMENT,
  `agency_name` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`agency_id`)
);


CREATE TABLE `house_listing` (
  `property_id` int NOT NULL,
  `agency_id` int DEFAULT NULL,
  `bedroom_number` int DEFAULT NULL,
  `bathroom_number` int DEFAULT NULL,
  `living_space` int DEFAULT NULL,
  `land_space` decimal(9,2) DEFAULT NULL,
  `property_status` varchar(45) DEFAULT NULL,
  `property_type` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`property_id`),
);


CREATE TABLE `price_table` (
  `price_id` int NOT NULL AUTO_INCREMENT,
  `property_id` int DEFAULT NULL,
  `zestimate` int DEFAULT NULL,
  `price` int DEFAULT NULL,
  `price_per_unit` int DEFAULT NULL,
  PRIMARY KEY (`price_id`),
);
```

SQL - INSERT DATA STATEMENTS

```
INSERT INTO house_listing (property_id, bedroom_number, bathroom_number, living_space,
land_space, property_status, property_type)
SELECT property_id, bedroom_number, bathroom_number, living_space, land_space, property_status,
property_type  FROM df;


INSERT INTO price_table(property_id, zestimate, price, price_per_unit)
SELECT property_id, zestimate, price, price_per_unit from df;


INSERT INTO address_table(property_id, address, street_name, city, rstate, postcode)
SELECT property_id, address, street_name, city, rstate, postcode from df;


INSERT INTO agency (agency_name)
SELECT distinct agency_name from df;


UPDATE house_listing h, df d
SET h.agency_id =
(SELECT a.agency_id from agency a
JOIN df d ON d.agency_name=a.agency_name
WHERE h.property_id = d.property_id)
WHERE h.property_id = d.property_id;
```

SQL – ADDING FOREIGN KEY CONSTRAINTS

```
alter table address_table
add constraint FK_address_house
foreign key (property_id) references house_listing(property_id);

alter table price_table
add constraint FK_price_house
foreign key (property_id) references house_listing(property_id);


alter table house_listing
add constraint FK_house_agency
foreign key (agency_id) references agency(agency_id);
```

Database Tables:

Table 1: Agency table

```
1 • select * from agency;
2
3
4
5
```

100% 22:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

agency_id	agency_name
1	Engel & Volkers Boston
2	Unlimited Sotheby's International Realty
3	Longwood Residential, LLC
4	Donnelly + Co.
5	McCormack & Scanlan Real Estate
6	Compass
7	Keller Williams Realty Boston-Metro I Back Bay
8	Symphony Properties
9	Midtown Properties, Inc.
10	New Wave Boston Real Estate, LLC
11	Advisors Living - Back Bay
12	Penrose Realty
13	Red Tree Real Estate
14	Leading Edge Real Estate
15	Linnane Real Estate
16	Sprogis & Neale Real Estate
17	eXp Realty
18	Centre Realty Group
19	MGS Group Real Estate LTD
20	Benjamin Realty
21	Setrakian Realty Co.
22	Realty Executives Boston West
23	Coldwell Banker Realty - South End
24	William Raveis R.E. & Home Services
25	Gibson Sotheby's International Realty
26	Coldwell Banker Realty - Charlestown
27	Arris Realty
28	Prevu Real Estate LLC
29	Coldwell Banker Realty - Brookline

Table 2: House Listing table

```
6 •  select * from house_listing;
7
```

100% 19:6

Result Grid Filter Rows: Search Edit: Export/Import:

property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_status	property_type
59091876	12	2	1	832	871.20	FOR_SALE	CONDO
59101062	34	3	3	2254	982.00	FOR_SALE	SINGLE_FAMILY
59101641	2	3	2	1232	1306.80	FOR_SALE	CONDO
59101725	50	2	2	980	1788.00	FOR_SALE	SINGLE_FAMILY
59101748	37	2	1	1161	1306.80	FOR_SALE	CONDO
59106078	17	4	2	1830	3103.00	FOR_SALE	MULTI_FAMILY
59109510	24	3	2	1116	3600.00	FOR_SALE	SINGLE_FAMILY
59160085	67	3	2	1464	3484.80	FOR_SALE	SINGLE_FAMILY
59160973	68	2	2	700	4356.00	FOR_SALE	SINGLE_FAMILY
59161881	6	8	3	5542	7840.00	PENDING	MULTI_FAMILY
59163175	44	2	2	1205	1306.80	FOR_SALE	CONDO
59163178	59	0	1	636	636.00	FOR_SALE	CONDO
59163215	59	2	2	1296	1306.80	FOR_SALE	CONDO
59163389	1	1	1	542	542.00	FOR_SALE	CONDO
59163504	14	1	1	871	871.20	FOR_SALE	CONDO
59163526	49	1	1	908	871.20	FOR_SALE	CONDO
59163896	43	2	3	1973	1973.00	FOR_SALE	CONDO
59164354	33	3	2	2226	2226.00	FOR_SALE	CONDO
59164404	6	1	1	771	871.20	FOR_SALE	CONDO
59164504	31	1	1	577	577.00	FOR_SALE	CONDO
59165005	24	2	2	949	949.00	FOR_SALE	CONDO
59165029	33	1	1	705	871.20	FOR_SALE	CONDO
59165356	65	0	1	508	508.00	FOR_SALE	CONDO
59165806	25	2	2	1151	1306.80	FOR_SALE	CONDO
59166308	46	5	6	7482	1923.00	FOR_SALE	SINGLE_FAMILY

Table 3: Address table

```
16 • select * from address_table;
17
18
19
```

100% 29:16

Result Grid Filter Rows: Search Edit: Export/Import:

address_id	property_id	street_name	city	rstate	postcode
3	59171381	386 Riverway APT 2	Boston	MA	2115
4	189924486	87-89 Nottinghill Rd	Boston	MA	2135
5	118162434	301 Chestnut Ave #1	Boston	MA	2130
6	59188486	402 Marlborough St APT 1	Boston	MA	2115
7	81854345	65 Highland St #2	Boston	MA	2119
8	59168062	30 Peterborough St APT 34	Boston	MA	2215
9	81858046	12 Stoneholm St APT 315	Boston	MA	2115
10	81857998	108 Peterborough St APT 6A	Boston	MA	2215
11	190005995	820-824 Huntington Ave	Boston	MA	2115
12	59091876	7 Saybrook St #1	Boston	MA	2135
13	20855795…	5 Braemore Rd APT 5	Boston	MA	2135
14	20851598…	117 Sutherland Rd #117	Boston	MA	2135
15	59212696	36 Symphony Rd APT 4A	Boston	MA	2115
16	59177283	65 Burbank St APT 14	Boston	MA	2115
17	59172993	138 Saint Botolph St	Boston	MA	2115
18	59177003	51 Hemenway St APT 1	Boston	MA	2115
19	59169052	27 Hereford St	Boston	MA	2115
20	59169848	352 Marlborough St	Boston	MA	2115
21	20666744…	126 Marion St	Boston	MA	2128
22	59173708	61 W Cedar St #2	Boston	MA	2114
23	59171698	79 Dartmouth St #2	Boston	MA	2116
24	59164404	100 Fulton St #3K	Boston	MA	2109
25	59211266	1387 Washington St APT 504	Boston	MA	2118
26	59165005	2 Clarendon St APT 102	Boston	MA	2116
27	20928041…	8 Howell St #1	Boston	MA	2125

Table 4: Price table

```
11 •  select * from price_table;
12
```

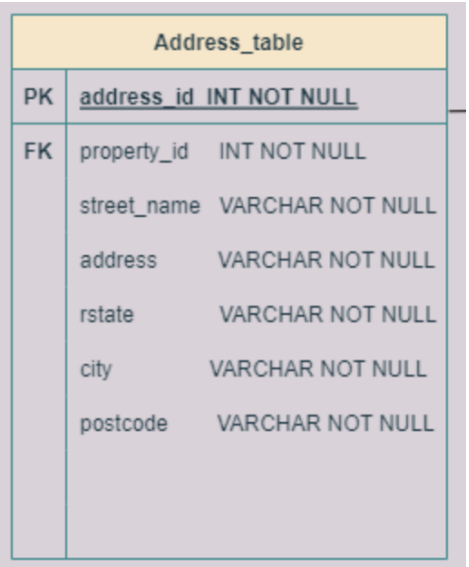
100% 27:11

Result Grid Filter Rows: Search Edit: Export/Import:

price_id	property_id	zestimate	price	price_per_unit
1	59169678	1202100	1175000	1019
2	59101641	461100	455000	369
3	59171381	590100	495000	529
4	189924486	999000	999000	403
5	118162434	814700	799000	700
6	59188486	1178600	1179000	1023
7	81854345	739000	739000	479
8	59168062	586000	585000	887
9	81858046	518800	519000	1474
10	81857998	649700	649000	1046
11	190005995	2879000	2999000	632
12	59091876	431000	469000	563
13	20855795...	355900	339999	883
14	20851598...	1249000	1249000	378
15	59212696	1036500	1049000	920
16	59177283	496000	499000	1087
17	59172993	5278400	5495000	1077
18	59177003	735700	750000	870
19	59169052	4575300	4999999	1337
20	59169848	9750000	9750000	2050
21	20666744...	1489000	1489000	335
22	59173708	869000	869000	1158
23	59171698	1026000	1075000	1113
24	59164404	776700	710000	920
25	59211266	1114200	995000	851

Database Normalization:

Table 1 – Address_Table



1st Normal Form:

1. It has primary key with minimal attributes. (PK = address_id)
2. The values in each column of a table are NOT atomic (Multi-value attributes are present).
3. There are no repeating groups.

The address_table is not in 1st Normal Form.

To convert this table to 1NF, we dropped the address column, because the multi-value data in that column was already present in 3 different columns – street_name, city, rstate, and postcode.

2nd Normal Form:

1. After the conversion, the table will be in 1st Normal form.
2. All non-key attributes are fully dependent on Primary key.

The address_table table is in 2nd Normal form.

3rd Normal Form:

1. It is in the 2nd Normal Form.
2. It has no transitive dependencies.

The address_table is in 3rd Normal form. Therefore, the address_table table is now Normalized.

```
1 •  select * from address_table;
2
3
4
5
```

100% 1:5

Result Grid | Filter Rows: Search | Edit: | Export/Import:

address_id	property_id	address	street_name	city	rstate	postcode
1	59169678	8 Gloucester St APT 12, Boston, MA 02115	8 Gloucester St APT 12	Boston	MA	2115
2	59101641	31 Homestead St #1, Boston, MA 02121	31 Homestead St #1	Boston	MA	2121
3	59171381	386 Riverway APT 2, Boston, MA 02115	386 Riverway APT 2	Boston	MA	2115
4	189924486	87-89 Nottinghill Rd, Boston, MA 02135	87-89 Nottinghill Rd	Boston	MA	2135
5	118162434	301 Chestnut Ave #1, Boston, MA 02130	301 Chestnut Ave #1	Boston	MA	2130
6	59188486	402 Marlborough St APT 1, Boston, MA 02115	402 Marlborough St APT 1	Boston	MA	2115
7	81854345	65 Highland St #2, Boston, MA 02119	65 Highland St #2	Boston	MA	2119
8	59168062	30 Peterborough St APT 34, Boston, MA 02215	30 Peterborough St APT 34	Boston	MA	2215
9	81858046	12 Stoneholm St APT 315, Boston, MA 02115	12 Stoneholm St APT 315	Boston	MA	2115
10	81857998	108 Peterborough St APT 6A, Boston, MA 02215	108 Peterborough St APT 6A	Boston	MA	2215
11	190005995	820-824 Huntington Ave, Boston, MA 02115	820-824 Huntington Ave	Boston	MA	2115
12	59091876	7 Saybrook St #1, Boston, MA 02135	7 Saybrook St #1	Boston	MA	2135
13	20855795...	5 Braemore Rd APT 5, Boston, MA 02135	5 Braemore Rd APT 5	Boston	MA	2135
14	20851598...	117 Sutherland Rd #117, Boston, MA 02135	117 Sutherland Rd #117	Boston	MA	2135
15	59212696	36 Symphony Rd APT 4A, Boston, MA 02115	36 Symphony Rd APT 4A	Boston	MA	2115
16	59177283	65 Burbank St APT 14, Boston, MA 02115	65 Burbank St APT 14	Boston	MA	2115
17	59172993	138 Saint Botolph St, Boston, MA 02115	138 Saint Botolph St	Boston	MA	2115
18	59177003	51 Hemenway St APT 1, Boston, MA 02115	51 Hemenway St APT 1	Boston	MA	2115
19	59169052	27 Hereford St, Boston, MA 02115	27 Hereford St	Boston	MA	2115
20	59169848	352 Marlborough St, Boston, MA 02115	352 Marlborough St	Boston	MA	2115

Data before 1NF is shown above

```
1 •  select * from address_table;
2
3
4
5
```

100% 28:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

address_id	property_id	street_name	city	rstate	postcode
1	59169678	8 Gloucester St APT 12	Boston	MA	2115
2	59101641	31 Homestead St #1	Boston	MA	2121
3	59171381	386 Riverway APT 2	Boston	MA	2115
4	189924486	87-89 Nottinghill Rd	Boston	MA	2135
5	118162434	301 Chestnut Ave #1	Boston	MA	2130
6	59188486	402 Marlborough St APT 1	Boston	MA	2115
7	81854345	65 Highland St #2	Boston	MA	2119
8	59168062	30 Peterborough St APT 34	Boston	MA	2215
9	81858046	12 Stoneholm St APT 315	Boston	MA	2115
10	81857998	108 Peterborough St APT 6A	Boston	MA	2215
11	190005995	820-824 Huntington Ave	Boston	MA	2115
12	59091876	7 Saybrook St #1	Boston	MA	2135
13	20855795...	5 Braemore Rd APT 5	Boston	MA	2135
14	20851598...	117 Sutherland Rd #117	Boston	MA	2135
15	59212696	36 Symphony Rd APT 4A	Boston	MA	2115
16	59177283	65 Burbank St APT 14	Boston	MA	2115
17	59172993	138 Saint Botolph St	Boston	MA	2115
18	59177003	51 Hemenway St APT 1	Boston	MA	2115
19	59169052	27 Hereford St	Boston	MA	2115
20	59169848	352 Marlborough St	Boston	MA	2115

Data after 3NF is shown above

Table structure after normalization of address_table

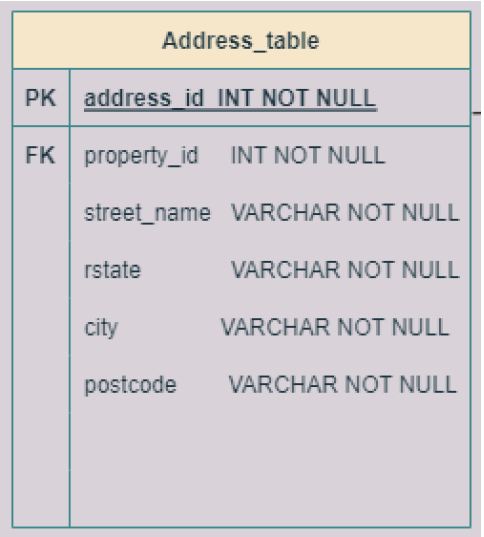
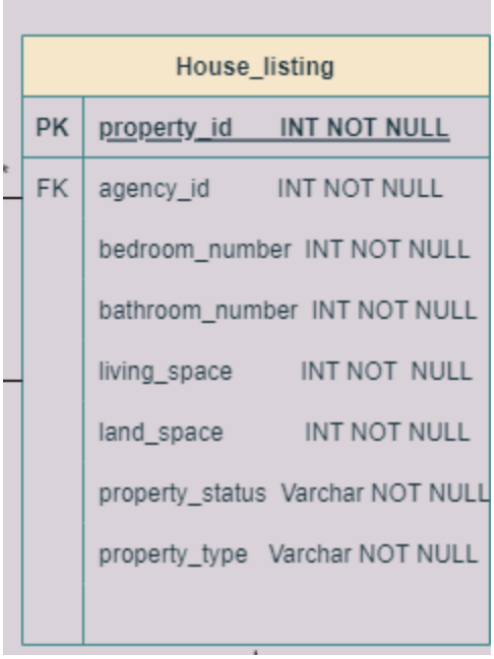


Table 2 – House_Listing



1st Normal Form:

1. It has primary key with minimal attributes. (PK = property_id)

2. The values in each column of a table are atomic (No multi-value attributes are present).
3. There are no repeating groups.

The house_listing table is in 1st Normal Form.

2nd Normal Form:

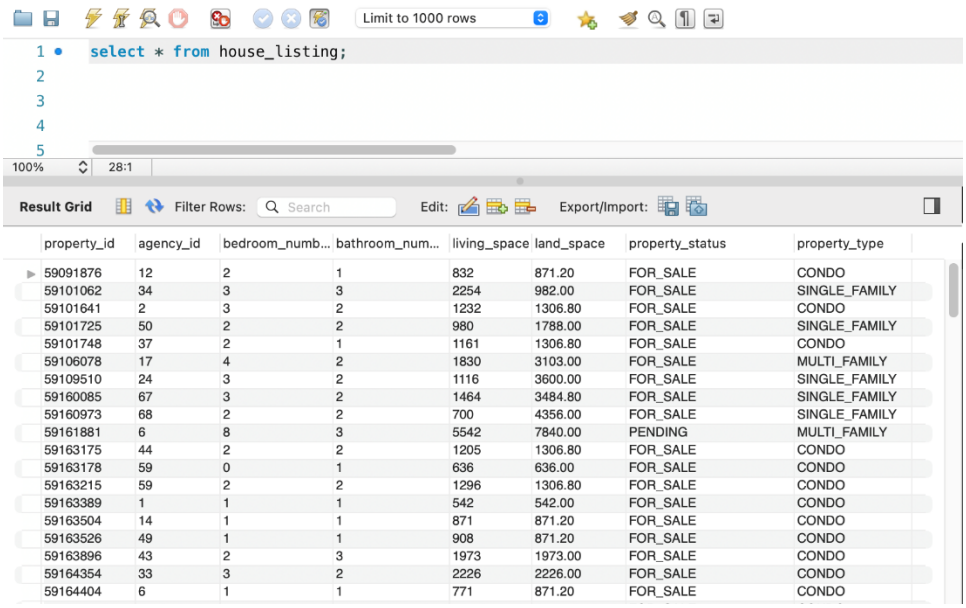
1. After the conversion, the table will be in 1st Normal form.
2. All non-key attributes are fully dependent on Primary key.

The house_listing table is in 2nd Normal form.

3rd Normal Form:

1. It is in the 2nd Normal Form.
2. It has no transitive dependencies.

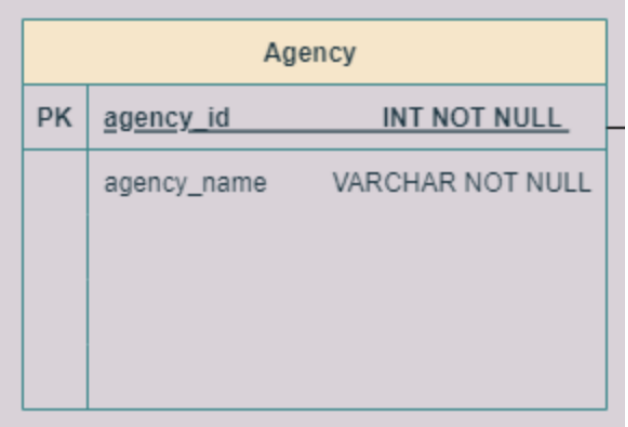
The house_listing is in 3rd Normal form. Therefore, the house_listing table is Normalized.



property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_status	property_type
59091876	12	2	1	832	871.20	FOR_SALE	CONDO
59101062	34	3	3	2254	982.00	FOR_SALE	SINGLE_FAMILY
59101641	2	3	2	1232	1306.80	FOR_SALE	CONDO
59101725	50	2	2	980	1788.00	FOR_SALE	SINGLE_FAMILY
59101748	37	2	1	1161	1306.80	FOR_SALE	CONDO
59106078	17	4	2	1830	3103.00	FOR_SALE	MULTI_FAMILY
59109510	24	3	2	1116	3600.00	FOR_SALE	SINGLE_FAMILY
59160085	67	3	2	1464	3484.80	FOR_SALE	SINGLE_FAMILY
59160973	68	2	2	700	4356.00	FOR_SALE	SINGLE_FAMILY
59161881	6	8	3	5542	7840.00	PENDING	MULTI_FAMILY
59163175	44	2	2	1205	1306.80	FOR_SALE	CONDO
59163178	59	0	1	636	636.00	FOR_SALE	CONDO
59163215	59	2	2	1296	1306.80	FOR_SALE	CONDO
59163389	1	1	1	542	542.00	FOR_SALE	CONDO
59163504	14	1	1	871	871.20	FOR_SALE	CONDO
59163526	49	1	1	908	871.20	FOR_SALE	CONDO
59163896	43	2	3	1973	1973.00	FOR_SALE	CONDO
59164354	33	3	2	2226	2226.00	FOR_SALE	CONDO
59164404	6	1	1	771	871.20	FOR_SALE	CONDO

Data of house_listing table is shown above

Table 3 – Agency



1st Normal Form:

4. It has primary key with minimal attributes. (PK = agency_id)
5. The values in each column of a table are atomic (No multi-value attributes are present).
6. There are no repeating groups.

The agency table is in 1st Normal Form.

2nd Normal Form:

3. After the conversion, the table will be in 1st Normal form.
4. All non-key attributes are fully dependent on Primary key.

The agency table is in 2nd Normal form.

3rd Normal Form:

3. It is in the 2nd Normal Form.
4. It has no transitive dependencies.

The agency is in 3rd Normal form. Therefore, the agency table is Normalized.

```
1 • select * from agency;
2
3
4
5
```

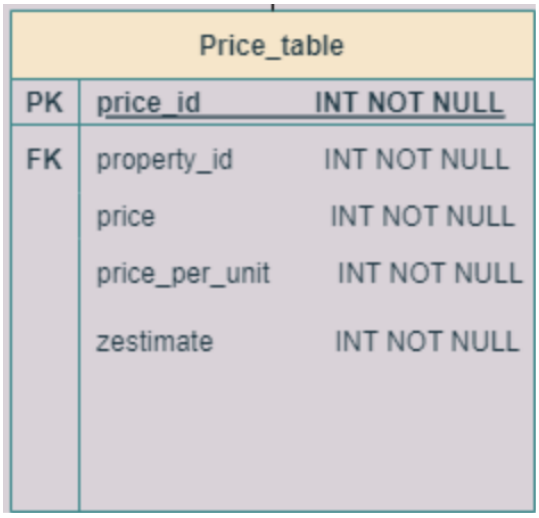
100% 21:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

agency_id	agency_name
1	Engel & Volkers Boston
2	Unlimited Sotheby's International Realty
3	Longwood Residential, LLC
4	Donnelly + Co.
5	McCormack & Scanlan Real Estate
6	Compass
7	Keller Williams Realty Boston-Metro I Back Bay
8	Symphony Properties
9	Midtown Properties, Inc.
10	New Wave Boston Real Estate, LLC
11	Advisors Living - Back Bay
12	Penrose Realty
13	Red Tree Real Estate
14	Leading Edge Real Estate
15	Linnane Real Estate
16	Sprogis & Neale Real Estate
17	eXp Realty
18	Centre Realty Group
19	MGS Group Real Estate LTD
20	Benjamin Realty

Data of agency table is shown above

Table 4 – Price_Table



1st Normal Form:

7. It has primary key with minimal attributes. (PK = price_id)
8. The values in each column of a table are atomic (No multi-value attributes are present).
9. There are no repeating groups.

The price_table is in 1st Normal Form.

2nd Normal Form:

5. After the conversion, the table will be in 1st Normal form.
6. All non-key attributes are fully dependent on Primary key.

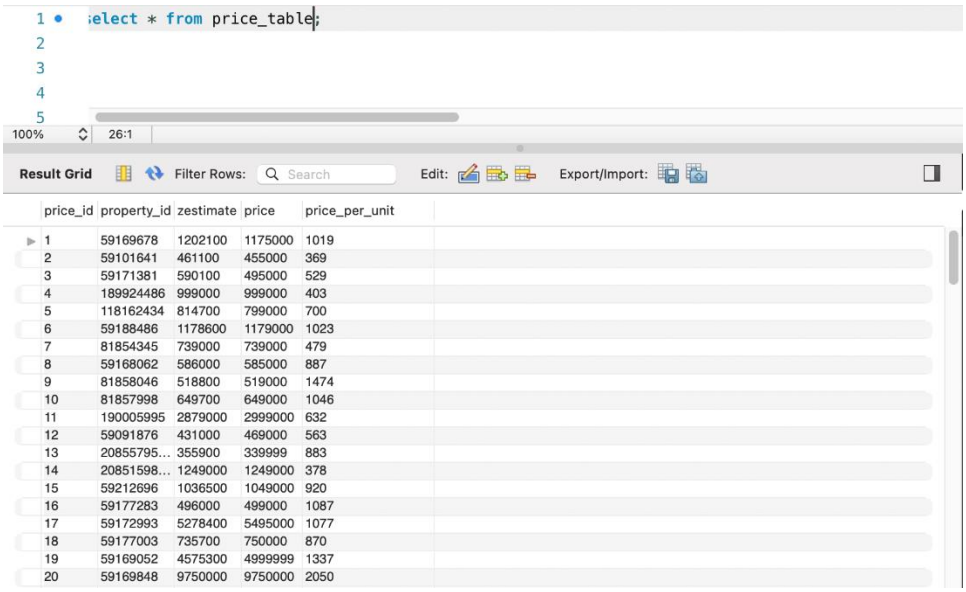
The price_table is in 2nd Normal form.

3rd Normal Form:

5. It is in the 2nd Normal Form.
6. It has no transitive dependencies.

The price_table is in 3rd Normal form.

Therefore, the agency table is Normalized.



price_id	property_id	zestimate	price	price_per_unit
1	59169678	1202100	1175000	1019
2	59101641	461100	455000	369
3	59171381	590100	495000	529
4	189924486	999000	999000	403
5	118162434	814700	799000	700
6	59188486	1178600	1179000	1023
7	81854345	739000	739000	479
8	59168062	586000	585000	887
9	81858046	518800	519000	1474
10	81857998	649700	649000	1046
11	190005995	2879000	2999000	632
12	59091876	431000	469000	563
13	20855795...	355900	339999	883
14	20851598...	1249000	1249000	378
15	59212696	1036500	1049000	920
16	59177283	496000	499000	1087
17	59172993	5278400	5495000	1077
18	59177003	735700	750000	870
19	59169052	4575300	4999999	1337
20	59169848	9750000	9750000	2050

Data of price_table is shown above

Snippets from Database – Sample Outputs:

Use Case 1: Finding properties pertaining to our specific search criteria

```
1   -- 1
2 • select * from house_listing
3   where bedroom_number<4 and living_space>2000;
4
5
6
```

100% 46:3

Result Grid | Filter Rows: Search | Edit: | Export/Import:

property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_status	property_type
59101062	34	3	3	2254	982.00	FOR_SALE	SINGLE_FAMILY
59164354	33	3	2	2226	2226.00	FOR_SALE	CONDO
59170902	6	3	3	2323	2323.00	FOR_SALE	CONDO
59171846	33	3	3	2220	2220.00	FOR_SALE	CONDO
59172798	25	3	5	3177	1306.80	FOR_SALE	SINGLE_FAMILY
59173636	76	3	4	2625	435.00	PENDING	SINGLE_FAMILY
59175907	47	3	2	2138	871.20	FOR_SALE	SINGLE_FAMILY
59183569	3	3	3	2213	2400.00	PENDING	CONDO
59211877	25	3	4	2864	3049.20	FOR_SALE	CONDO
60112593	26	3	4	2200	1306.80	FOR_SALE	SINGLE_FAMILY
61319753	24	3	5	2667	2667.00	FOR_SALE	CONDO
81855959	16	3	3	2040	2178.00	FOR_SALE	CONDO
90141331	19	2	3	3168	3168.00	FOR_SALE	CONDO
190073274	19	2	3	2438	2613.60	FOR_SALE	CONDO
2068770560	6	3	4	3616	1742.40	FOR_SALE	CONDO
2071564666	33	3	4	2603	2613.60	FOR_SALE	CONDO
2085159874	13	3	3	3300	2362.00	FOR_SALE	CONDO
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Use Case 2: Find the properties that are owned by a “single-family”

```
11    -- 2
12 •  select * from house_listing
13    where property_type like '%single_family%';
14
15
16
```

100% 19:13

Result Grid Filter Rows: Search Edit: Export/Import:

property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_status	property_type
59101725	50	2	2	980	1788.00	FOR_SALE	SINGLE_FAMILY
59109510	24	3	2	1116	3600.00	FOR_SALE	SINGLE_FAMILY
59160085	67	3	2	1464	3484.80	FOR_SALE	SINGLE_FAMILY
59160973	68	2	2	700	4356.00	FOR_SALE	SINGLE_FAMILY
59166308	46	5	6	7482	1923.00	FOR_SALE	SINGLE_FAMILY
59169052	18	5	4	3737	1742.40	FOR_SALE	SINGLE_FAMILY
59169456	6	3	3	1941	650.00	FOR_SALE	SINGLE_FAMILY
59169848	19	4	7	4754	1.00	FOR_SALE	SINGLE_FAMILY
59172798	25	3	5	3177	1306.80	FOR_SALE	SINGLE_FAMILY
59172993	16	5	5	5098	2178.00	FOR_SALE	SINGLE_FAMILY
59173220	1	5	4	4188	1306.80	FOR_SALE	SINGLE_FAMILY
59173446	46	4	5	3966	1306.80	FOR_SALE	SINGLE_FAMILY
59173451	24	5	5	4318	2178.00	FOR_SALE	SINGLE_FAMILY
59173636	76	3	4	2625	435.00	PENDING	SINGLE_FAMILY
59174060	25	5	4	3300	1306.80	FOR_SALE	SINGLE_FAMILY
59174448	63	5	6	3825	1344.00	FOR_SALE	SINGLE_FAMILY
59174916	36	4	6	5322	1742.00	PENDING	SINGLE_FAMILY
59175907	47	3	2	2138	871.20	FOR_SALE	SINGLE_FAMILY
59180675	46	5	4	4212	0.03	FOR_SALE	SINGLE_FAMILY
59183902	51	3	2	1199	871.20	FOR_SALE	SINGLE_FAMILY
60112593	26	3	4	2200	1306.80	FOR_SALE	SINGLE_FAMILY
113389948	33	4	6	5799	1814.00	FOR_SALE	SINGLE_FAMILY
189950691	19	4	5	3607	1458.00	FOR_SALE	SINGLE_FAMILY
2068405707	25	2	3	1554	0.07	FOR_SALE	SINGLE_FAMILY
2102089728	40	3	3	1585	900.00	FOR_SALE	SINGLE_FAMILY
2103780872	16	4	5	5948	1801.00	FOR_SALE	SINGLE_FAMILY
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Use Case 3: Find the status of properties that are “pending”

```
17    -- 3
18 •  select * from house_listing
19    where property_status like '%pending%';
20
21
22
```

100% 40:19

Result Grid Filter Rows: Search Edit: Export/Import:

property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_status	property_type
59161881	6	8	3	5542	7840.00	PENDING	MULTI_FAMILY
59168275	76	1	1	643	435.00	PENDING	CONDO
59173636	76	3	4	2625	435.00	PENDING	SINGLE_FAMILY
59174573	4	2	1	1300	1306.00	PENDING	CONDO
59174916	36	4	6	5322	1742.00	PENDING	SINGLE_FAMILY
59183569	3	3	3	2213	2400.00	PENDING	CONDO
59188434	42	2	2	1559	1742.00	PENDING	CONDO
59212026	5	3	3	1531	1496.00	PENDING	CONDO
63042601	7	2	3	1462	1452.00	PENDING	CONDO
67715620	73	2	2	1068	871.00	PENDING	CONDO
71546584	2	2	2	1460	1460.00	PENDING	CONDO
81857945	74	1	1	760	871.00	PENDING	CONDO
90136786	75	2	2	1011	871.00	PENDING	CONDO
90138053	11	1	1	856	871.00	PENDING	CONDO
189938322	1	1	2	1341	1306.00	PENDING	CONDO
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Use Case 4: Find the prices of all listed houses in a particular postcode

```
23    -- 4
24 •  select p.price, a.postcode from price_table p
25    join address_table a on a.property_id = p.property_id
26    where postcode = '2215';
27
28
```

100% 25:26

Result Grid Filter Rows: Search Export:

price	postcode
585000	2215
649000	2215
690000	2215
340451	2215

Use Case 5: Find all condos with exactly 2 bathrooms.

```
30    -- 5
31    select p.price, h.bathroom_number, h.property_type from price_table p
32    join house_listing h on p.property_id = h.property_id
33    where h.bathroom_number=2 and h.property_type like '%condo%';
34
35
```

0% 8:32

Result Grid | Filter Rows: Search | Export:

price	bathroom_num...	property_ty...
741500	2	CONDO
1290000	2	CONDO
989000	2	CONDO
934900	2	CONDO
1175000	2	CONDO
1075000	2	CONDO
1275000	2	CONDO
1365000	2	CONDO
2100000	2	CONDO
1699000	2	CONDO
1225000	2	CONDO
1749000	2	CONDO
579000	2	CONDO
1575000	2	CONDO
1150000	2	CONDO
1100000	2	CONDO
1179000	2	CONDO
995000	2	CONDO
2200000	2	CONDO
1049000	2	CONDO
1299000	2	CONDO
2295000	2	CONDO
1349000	2	CONDO
950000	2	CONDO
1200000	2	CONDO
1498000	2	CONDO
699000	2	CONDO

Use Case 6: Find the Zestimate of multi-family houses**.**

```
36    -- 6
37 •  select p.zestimate, h.property_type from price_table p
38    join house_listing h on p.property_id = h.property_id
39    where h.property_type like '%multi_family%';
40
41
```

100% 14:38

Result Grid Filter Rows: Search Export:

zestimate	property_type
951500	MULTI_FAMILY
786300	MULTI_FAMILY
999000	MULTI_FAMILY
2879000	MULTI_FAMILY
850000	MULTI_FAMILY
2999000	MULTI_FAMILY
4750000	MULTI_FAMILY
1489000	MULTI_FAMILY
1367300	MULTI_FAMILY
3149500	MULTI_FAMILY
4065000	MULTI_FAMILY
2950000	MULTI_FAMILY
3678600	MULTI_FAMILY
4766400	MULTI_FAMILY
6188400	MULTI_FAMILY
6591000	MULTI_FAMILY
1683500	MULTI_FAMILY

Use Case 7: Find the addresses of the listed houses with area greater than 1300 sqft.

```
42    -- 7
43 •  select a.street_name, a.postcode, h.land_space from address_table a
44    join house_listing h on h.property_id = a.property_id
45    where h.land_space>1300;
46
47
```

0% 16:44

Result Grid | Filter Rows: Search | Export:

street_name	postcode	land_space
54 Monadnock St #2	2125	1306.80
3 Magnolia Pl	2125	3103.00
84 Everdean St	2122	3600.00
47 Keystone St	2132	3484.80
28 Bungalow Rd	2132	4356.00
151 Townsend St	2121	7840.00
6 Whittier Pl APT 2D	2114	1306.80
6 Whittier Pl APT 4N	2114	1306.80
20 Rowes Wharf A...	2110	1973.00
100 Fulton St APT...	2109	2226.00
80 Broadway St #2A	2116	1306.80
17 Louisburg Sq	2108	1923.00
27 Hereford St	2115	1742.40
8 Gloucester St AP...	2115	1306.80
242 Beacon St APT 8	2116	2178.00
191 Commonwealt...	2116	2323.00
180 Beacon St #5G	2116	2220.00
50 Brimmer St	2108	1306.80
138 Saint Botolph St	2115	2178.00
129 Pinckney St	2114	1306.80
22 Charles River Sq	2114	1306.80
16 Charles River Sq	2114	2178.00
77 Myrtle St	2114	1306.80
54 Pinckney St	2114	1344.00
43 Commercial Wh...	2110	1306.80
55 Commercial Wh...	2110	1306.00
48 Beacon St APT...	2108	1306.80

Use Case 8: Find the addresses of the listed houses which are under-valued according to Zillow.

```
48    -- 8
49 •  select p.property_id, a.street_name, a.city, a.rstate, a.postcode, p.price, p.zestimate from price_table p
50    join address_table a on a.property_id = p.property_id
51    where p.zestimate < p.price;
52
53
```

)% 29:51

esult Grid Filter Rows: Search Export:

property_id	street_name	city	rstate	postcode	price	zestimate
190005995	820-824 Huntington Ave	Boston	MA	2115	2999000	2879000
59091876	7 Saybrook St #1	Boston	MA	2135	469000	431000
59212696	36 Symphony Rd APT 4A	Boston	MA	2115	1049000	1036500
59177283	65 Burbank St APT 14	Boston	MA	2115	499000	496000
59172993	138 Saint Botolph St	Boston	MA	2115	5495000	5278400
59177003	51 Hemenway St APT 1	Boston	MA	2115	750000	735700
59169052	27 Hereford St	Boston	MA	2115	4999999	4575300
59171698	79 Dartmouth St #2	Boston	MA	2116	1075000	1026000
59165005	2 Clarendon St APT 102	Boston	MA	2116	989000	980500
20928041...	8 Howell St #1	Boston	MA	2125	849000	737500
60112593	7 Charles St	Boston	MA	2129	1399000	1198900
59174060	77 Myrtle St	Boston	MA	2114	3995000	3843600
67715677	25 Channel Center St U...	Boston	MA	2210	1498000	1481100
123845214	580 Washington St #3	Boston	MA	2111	3250000	2976000
59169456	28 Irving St	Boston	MA	2114	2795000	2753100
90136195	99-105 Broad St #8B	Boston	MA	2110	1495000	1345200
59174085	25 Revere St APT 1	Boston	MA	2114	1275000	1269100
81857238	44 Prince St APT 504	Boston	MA	2113	1995000	1961700
59101062	4 Lexington St	Boston	MA	2129	1549000	1537600
59175437	85 E India Row APT 25G	Boston	MA	2110	1225000	1223700
81857648	80 Broad St UNIT 808	Boston	MA	2110	1299000	1278500
59173451	16 Charles River Sq	Boston	MA	2114	4950000	4810200
189950372	43 Phillips St UNIT 13	Boston	MA	2114	2150000	2128700
21020897...	6 Kemble Pl	Boston	MA	2127	1585000	1567200
21076828...	7 Anderson St	Boston	MA	2114	6470000	6188400
2109488118	37 Joy St	Boston	MA	2114	6900000	6591000
20797797...	47 Revere St	Boston	MA	2114	4250000	4065000

Use Case 9: Find the average Zestimate for a given zip code.

```
54      -- 9
55  •   select avg(p.zestimate) as average_zestimate, a.postcode from price_table p
56      join address_table a on a.property_id = p.property_id
57      group by a.postcode;
58
59      -- 10
```

00% 21:56

Result Grid Filter Rows: Search Export:

average_zestima...	postcode
2541883.3333	2115
567966.6667	2121
758725.0000	2135
814700.0000	2130
680233.3333	2119
579825.0000	2215
823822.2222	2128
2805280.7692	2114
2206031.8182	2116
1326416.6667	2109
1855842.8571	2118
643500.0000	2125
1368250.0000	2129
817075.0000	2113
1292733.3333	2210
1221766.8333	2111
1655660.0000	2110
1881284.6154	2127
5627357.1429	2108
7625000.0000	2199
521466.6667	2132
461800.0000	2131
719800.0000	2126
779100.0000	2122
1683500.0000	2120

Use Case 10: Find the addresses of the listed houses which are over-valued according to Zillow.

```
59    -- 10
60 •  select a.street_name, a.postcode, p.zestimate, p.price from price_table p
61    join address_table a on a.property_id = p.property_id
62    where p.zestimate > p.price;
63
64
```

100% 8:61

Result Grid Filter Rows: Search Export:

street_name	postcode	zestimate	price
386 Riverway APT 2	2115	590100	495000
301 Chestnut Ave #1	2130	814700	799000
30 Peterborough St APT 34	2215	586000	585000
108 Peterborough St APT 6A	2215	649700	649000
5 Braemore Rd APT 5	2135	355900	339999
100 Fulton St #3K	2109	776700	710000
1387 Washington St APT 504	2118	1114200	995000
8 Whittier Pl APT 11D	2114	557400	534900
58 E Springfield St #5	2118	745800	699000
304 Shawmut Ave	2118	4766400	4200000
26 Stillman St APT 2-1	2113	1331800	1299000
21 Father Francis Gilday St...	2118	968100	899000
162 Salem St APT 5	2113	537500	525000
12 Worcester Sq APT 1	2118	1907000	1695000
300 Commercial St APT 512	2109	555300	539900
3 Magnolia Pl	2125	951500	950000
54 Monadnock St #2	2125	445600	444900
43 Upton St APT 2	2118	686400	675000
158 F St APT 2	2127	538000	529000
210 South St UNIT 115	2111	825000	824999
151 Tremont St APT 6C	2111	650100	649000
6 Goodwin Pl APT 1	2114	463700	460000
5 Lawrence St	2116	2361600	2250000
330 E St APT 5	2127	901300	889000
42 Beach St #8C	2111	647400	638000
8 Whittier Pl APT 14F	2114	635300	624900
79 Wayland St	2125	439400	425000

Use Case 11: Find all listed houses within the specified area and conditions.

```
65    -- 11
66 •  select h.property_id, bedroom_number, bathroom_number, living_space, land_space,
67        property_status, property_type, price_per_unit from house_listing h
68    join price_table p on p.property_id = h.property_id
69    where price_per_unit between 300 and 1500;
70
```

00% 7:66

Result Grid Filter Rows: Search Export:

property_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_status	property_type	price_per_unit
59171381	1	1	934	934.00	FOR_SALE	CONDO	529
189924486	4	4	2478	2613.60	FOR_SALE	MULTI_FAMILY	403
118162434	2	2	1141	1306.80	FOR_SALE	CONDO	700
59188486	1	2	1152	1306.80	FOR_SALE	CONDO	1023
81854345	3	2	1540	1742.40	FOR_SALE	CONDO	479
59168062	1	1	659	871.20	FOR_SALE	CONDO	887
81858046	1	1	352	435.60	FOR_SALE	CONDO	1474
81857998	1	1	620	620.00	FOR_SALE	CONDO	1046
190005995	10	6	4740	1500.00	FOR_SALE	MULTI_FAMILY	632
59091876	2	1	832	871.20	FOR_SALE	CONDO	563
2085579554	1	1	385	435.60	FOR_SALE	CONDO	883
2085159874	3	3	3300	2362.00	FOR_SALE	CONDO	378
59212696	3	2	1140	1306.80	FOR_SALE	CONDO	920
59177283	1	1	459	459.00	FOR_SALE	CONDO	1087
59172993	5	5	5098	2178.00	FOR_SALE	SINGLE_FAM...	1077
59177003	2	1	862	862.00	FOR_SALE	CONDO	870
59169052	5	4	3737	1742.40	FOR_SALE	SINGLE_FAM...	1337
2066674445	8	6	4440	2613.60	FOR_SALE	MULTI_FAMILY	335
59173708	2	1	750	871.20	FOR_SALE	CONDO	1158
59171698	2	2	965	965.00	FOR_SALE	CONDO	1113
59164404	1	1	771	871.20	FOR_SALE	CONDO	920
59211266	2	2	1168	1306.80	FOR_SALE	CONDO	851
59165005	2	2	949	949.00	FOR_SALE	CONDO	1042
2092804145	3	2	1716	1742.40	FOR_SALE	CONDO	494
59163504	1	1	871	871.20	FOR_SALE	CONDO	614
60112593	3	4	2200	1306.80	FOR_SALE	SINGLE_FAM...	635
90136038	2	1	725	871.20	FOR_SALE	CONDO	964

Use Case 12: Find top 3 most expensive properties under a specific postcode

```
71    -- 12
72    select price, postcode from price_table p
73    join address_table a on a.property_id = p.property_id
74    where postcode = '2115'
75    order by price desc
76    limit 3;
77
```

100% 6:74

Result Grid Filter Rows: Search Export: Fetch rows:

price	postcode
9750000	2115
5495000	2115
4999999	2115

Use Case 13: Find address and other details of the listed houses which are neither under-valued nor over-valued, according to Zillow's index 'Zestimate'.

```
78    -- 13
79    select street_name, city, rstate, postcode, bedroom_number, bathroom_number, living_space, zestimate, price from a
80    join house_listing h on h.property_id = a.property_id
81    join price_table p on p.property_id = a.property_id
82    where zestimate = price;
83
84
```

100% 1:78

Result Grid | Filter Rows: Search | Export: | Fetch rows:

price	postcode
9750000	2115
5495000	2115
4999999	2115

Use Case 14: Find top 3 popular agencies with highest number of listings.

```
85    -- 14
86    select count(h.agency_id) as famous_agency, agency_name from house_listing h
87    join agency a on a.agency_id = h.agency_id
88    group by h.agency_id
89    order by famous_agency desc
90    limit 3;
91
```

100% 4:88

Result Grid | Filter Rows: Search | Export: | Fetch rows:

famous_agency	agency_name
29	Compass
15	Gibson Sotheby's International Realty
10	Coldwell Banker Realty - Back Bay

Use Case 15: Find property and agency details of the desired properties

```
-- 15
select h.property_id, bedroom_number, bathroom_number, living_space, agency_name from house_listing h
join agency a on a.agency_id = h.agency_id
where bedroom_number>3;
```

100% 1:92

Result Grid Filter Rows: Search Export:

property_id	bedroom_numb...	bathroom_num...	living_space	agency_name
59166308	5	6	7482	Campion & Company Fine Homes Real Estate
59169052	5	4	3737	Centre Realty Group
59169848	4	7	4754	MGS Group Real Estate LTD
59172993	5	5	5098	Sprogis & Neale Real Estate
59173220	5	4	4188	Engel & Volkers Boston
59173446	4	5	3966	Campion & Company Fine Homes Real Estate
59173451	5	5	4318	William Raveis R.E. & Home Services
59174060	5	4	3300	Gibson Sotheby's International Realty
59174448	5	6	3825	Berkshire Hathaway HomeServices Warren...
59174642	4	5	5438	Campion & Company Fine Homes Real Estate
59174916	4	6	5322	Broad Street Boutique Realty LLC
59180675	5	4	4212	Campion & Company Fine Homes Real Estate
67709760	4	2	1295	Coldwell Banker Realty - Back Bay
102988140	4	5	2428	Gibson Sotheby's International Realty
113389948	4	6	5799	Coldwell Banker Realty - Back Bay
189924486	4	4	2478	Donnelly + Co.
189950691	4	5	3607	MGS Group Real Estate LTD
190005995	10	6	4740	Advisors Living - Back Bay
190039888	5	3	2288	Signal Real Estate
2066046605	11	6	7993	Boston Realty Advisors
2066559353	4	4	4400	Keller Williams Realty Boston-Metro | Back Bay
2066674445	8	6	4440	Benjamin Realty
2067537844	4	3	1547	Coldwell Banker Realty - Milton
2067630781	6	9	4425	Compass
2079779704	7	10	4466	Compass
2096000383	4	4	3958	Sprogis & Neale Real Estate

Use Case 16: Find property and agency details of the pending properties (not yet sold or bought).

```
99    -- 16
100 • select h.property_id, bedroom_number, bathroom_number, living_space, property_status, agency_name from house_listi
101   join agency a on a.agency_id = h.agency_id
102   where property_status = 'PENDING';
103
```

100% 1:104

Result Grid Filter Rows: Search Export:

property_id	bedroom_numb...	bathroom_num...	living_space	property_status	agency_name
59161881	8	3	5542	PENDING	Compass
59168275	1	1	643	PENDING	Redfin Corp.
59173636	3	4	2625	PENDING	Redfin Corp.
59174573	2	1	1300	PENDING	Donnelly + Co.
59174916	4	6	5322	PENDING	Broad Street Boutique Realty LLC
59183569	3	3	2213	PENDING	Longwood Residential, LLC
59188434	2	2	1559	PENDING	Luxury Residential Group, LLC
59212026	3	3	1531	PENDING	McCormack & Scanlan Real Estate
63042601	2	3	1462	PENDING	Keller Williams Realty Boston-Metro | Back Bay
67715620	2	2	1068	PENDING	Keller Williams Realty Cambridge
71546584	2	2	1460	PENDING	Unlimited Sotheby's International Realty
81857945	1	1	760	PENDING	Carpenter and Company, Inc.
90136786	2	2	1011	PENDING	Boston Realty Net
90138053	1	1	856	PENDING	Advisors Living - Back Bay
189938322	1	2	1341	PENDING	Engel & Volkers Boston

Use Case 17: Find property which costs more than $7000000

```
106     -- 17
107 •   select a.street_name, a.city, a.rstate, a.postcode, p.price from address_table a
108     join price_table p on p.property_id = a.property_id
109     where p.price>7000000;
110
111
```

100% 1:111

Result Grid Filter Rows: Search Export:

street_name	city	rstate	postcode	price
352 Marlborough St	Boston	MA	2115	9750000
17 Louisburg Sq	Boston	MA	2108	15500000
69 Hancock St	Boston	MA	2114	7599000
52 Beacon St	Boston	MA	2108	9500000
1 Huntington Ave PENTHOUSE 1802	Boston	MA	2116	8250000
776 Boylston St UNIT W11A	Boston	MA	2199	10800000
9 Chestnut St	Boston	MA	2108	7695000

Use Case 18: Find all properties which has living area greater than 500 sqft.

```
110
111  -- 18
112  select p.price, p.zestimate, h.living_space, h.land_space from price_table p
113  join house_listing h on h.property_id = p.property_id
114  where h.living_space>500;
115
116
```

100% 1:110

Result Grid Filter Rows: Search Export:

price	zestimate	living_space	land_space
469000	431000	832	871.20
1549000	1537600	2254	982.00
455000	461100	1232	1306.80
425000	439400	980	1788.00
444900	445600	1161	1306.80
950000	951500	1830	3103.00
895000	889200	1116	3600.00
649500	686000	1464	3484.80
459000	477900	700	4356.00
780000	786300	5542	7840.00
679900	679700	1205	1306.80
475000	461900	636	636.00
741500	727900	1296	1306.80
439000	424600	542	542.00
534900	557400	871	871.20
624900	635300	908	871.20
2875000	2706600	1973	1973.00
1290000	1260500	2226	2226.00
710000	776700	771	871.20
539900	555300	577	577.00
989000	980500	949	949.00
799000	815000	705	871.20
484000	484800	508	508.00
934900	934300	1151	1306.80
155000...	15063000	7482	1923.00
585000	586000	659	871.20

Use cases specific to our domain of interest:

Use Case 1: Finding properties pertaining to our specific search criteria

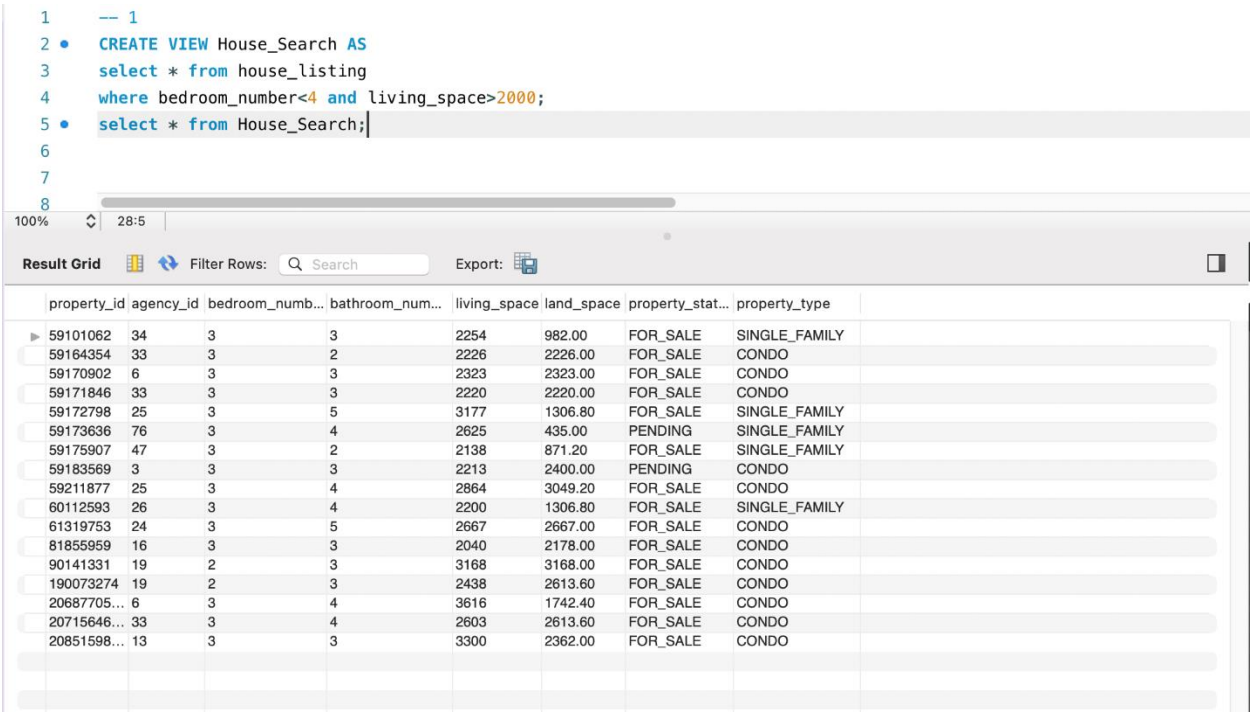
Precondition: The listings should be listed on Zillow's website.

Steps: We select all listings from our master house listing table. We select prices and post code from two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the postcode criteria using the WHERE clause. Please note that these query conditions can be slightly altered producing search results for houses and their prices for various zip code and locations within Boston.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW House_Search AS
select * from house_listing
where bedroom_number<4 and living_space>2000;
select * from House_Search;



property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_stat...	property_type
59101062	34	3	3	2254	982.00	FOR_SALE	SINGLE_FAMILY
59164354	33	3	2	2226	2226.00	FOR_SALE	CONDO
59170902	6	3	3	2323	2323.00	FOR_SALE	CONDO
59171846	33	3	3	2220	2220.00	FOR_SALE	CONDO
59172798	25	3	5	3177	1306.80	FOR_SALE	SINGLE_FAMILY
59173636	76	3	4	2625	435.00	PENDING	SINGLE_FAMILY
59175907	47	3	2	2138	871.20	FOR_SALE	SINGLE_FAMILY
59183569	3	3	3	2213	2400.00	PENDING	CONDO
59211877	25	3	4	2864	3049.20	FOR_SALE	CONDO
60112593	26	3	4	2200	1306.80	FOR_SALE	SINGLE_FAMILY
61319753	24	3	5	2667	2667.00	FOR_SALE	CONDO
81855959	16	3	3	2040	2178.00	FOR_SALE	CONDO
90141331	19	2	3	3168	3168.00	FOR_SALE	CONDO
190073274	19	2	3	2438	2613.60	FOR_SALE	CONDO
20687705...	6	3	4	3616	1742.40	FOR_SALE	CONDO
20715646...	33	3	4	2603	2613.60	FOR_SALE	CONDO
20851598...	13	3	3	3300	2362.00	FOR_SALE	CONDO

Use Case 2: Find the properties that are owned by a “single-family”

Precondition: The listings should be listed on Zillow’s website.

Steps: We select all listings from our master house listing table. We select prices and post code from two tables and perform JOIN operation on the unique attribute ‘property_id’. We then specify the postcode criteria using the WHERE clause. Please note that these query conditions can be slightly altered producing search results for houses and their prices for various zip code and locations within Boston.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Family_Type AS
select * from house_listing
where property_type like '%single_family%';

```
8   -- 2
9   CREATE VIEW Family_Type AS
10  select * from house_listing
11  where property_type like '%single_family%';
12  select * from Family_Type;
13
14
15
```

100% 27:12

Result Grid Filter Rows: Search Export:

property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_stat...	property_type
59109510	24	3	2	1116	3600.00	FOR_SALE	SINGLE_FAMILY
59160085	67	3	2	1464	3484.80	FOR_SALE	SINGLE_FAMILY
59160973	68	2	2	700	4356.00	FOR_SALE	SINGLE_FAMILY
59166308	46	5	6	7482	1923.00	FOR_SALE	SINGLE_FAMILY
59169052	18	5	4	3737	1742.40	FOR_SALE	SINGLE_FAMILY
59169456	6	3	3	1941	650.00	FOR_SALE	SINGLE_FAMILY
59169848	19	4	7	4754	1.00	FOR_SALE	SINGLE_FAMILY
59172798	25	3	5	3177	1306.80	FOR_SALE	SINGLE_FAMILY
59172993	16	5	5	5098	2178.00	FOR_SALE	SINGLE_FAMILY
59173220	1	5	4	4188	1306.80	FOR_SALE	SINGLE_FAMILY
59173446	46	4	5	3966	1306.80	FOR_SALE	SINGLE_FAMILY
59173451	24	5	5	4318	2178.00	FOR_SALE	SINGLE_FAMILY
59173636	76	3	4	2625	435.00	PENDING	SINGLE_FAMILY
59174060	25	5	4	3300	1306.80	FOR_SALE	SINGLE_FAMILY
59174448	63	5	6	3825	1344.00	FOR_SALE	SINGLE_FAMILY
59174916	36	4	6	5322	1742.00	PENDING	SINGLE_FAMILY
59175907	47	3	2	2138	871.20	FOR_SALE	SINGLE_FAMILY
59180675	46	5	4	4212	0.03	FOR_SALE	SINGLE_FAMILY
59183902	51	3	2	1199	871.20	FOR_SALE	SINGLE_FAMILY
60112593	26	3	4	2200	1306.80	FOR_SALE	SINGLE_FAMILY
113389948	33	4	6	5799	1814.00	FOR_SALE	SINGLE_FAMILY

Use Case 3: Find the status of properties that are “pending”

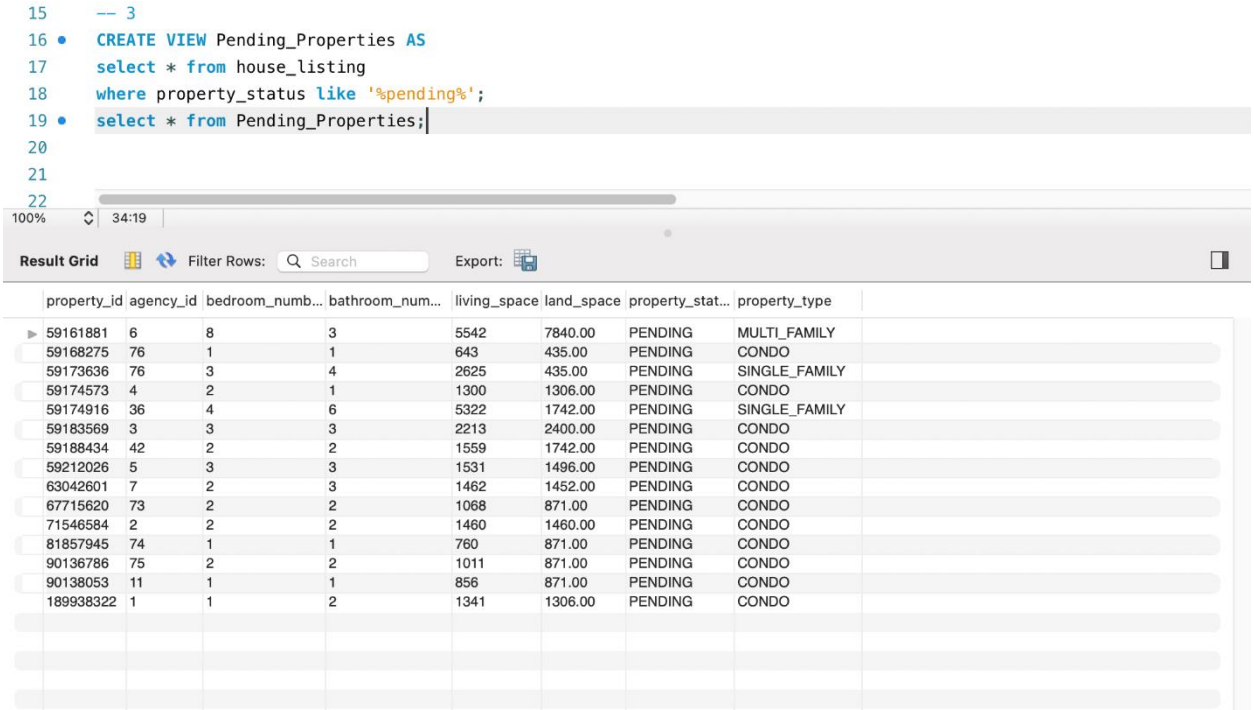
Precondition: The listings should be listed on Zillow’s website.

Steps: We select all listings from our master house listing table. We select prices and post code from two tables and perform JOIN operation on the unique attribute ‘property_id’. We then specify the postcode criteria using the WHERE clause. Please note that these query conditions can be slightly altered producing search results for houses and their prices for various zip code and locations within Boston.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

```
CREATE VIEW Pending_Properties AS
select * from house_listing
where property_status like '%pending%';
```



```
15    -- 3
16 •  CREATE VIEW Pending_Properties AS
17    select * from house_listing
18    where property_status like '%pending%';
19 •  select * from Pending_Properties;
20
21
22
```

100% 34:19

Result Grid | Filter Rows: Search | Export:

property_id	agency_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_stat...	property_type
59161881	6	8	3	5542	7840.00	PENDING	MULTI_FAMILY
59168275	76	1	1	643	435.00	PENDING	CONDO
59173636	76	3	4	2625	435.00	PENDING	SINGLE_FAMILY
59174573	4	2	1	1300	1306.00	PENDING	CONDO
59174916	36	4	6	5322	1742.00	PENDING	SINGLE_FAMILY
59183569	3	3	3	2213	2400.00	PENDING	CONDO
59188434	42	2	2	1559	1742.00	PENDING	CONDO
59212026	5	3	3	1531	1496.00	PENDING	CONDO
63042601	7	2	3	1462	1452.00	PENDING	CONDO
67715620	73	2	2	1068	871.00	PENDING	CONDO
71546584	2	2	2	1460	1460.00	PENDING	CONDO
81857945	74	1	1	760	871.00	PENDING	CONDO
90136786	75	2	2	1011	871.00	PENDING	CONDO
90138053	11	1	1	856	871.00	PENDING	CONDO
189938322	1	1	2	1341	1306.00	PENDING	CONDO

Use Case 4: Find the prices of all listed houses in a particular postcode

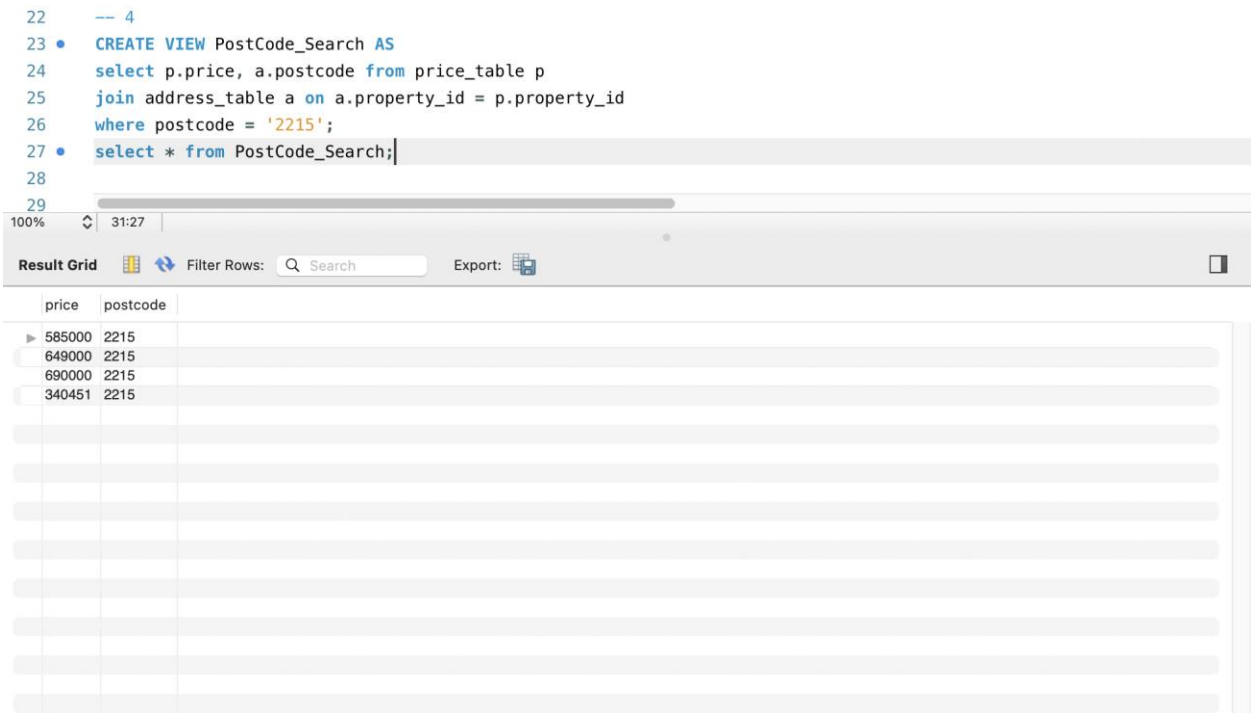
Precondition: The listings should be listed on Zillow's website.

Steps: We select all listings from our master house listing table. We select prices and post code from two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the postcode criteria using the WHERE clause. Please note that these query conditions can be slightly altered producing search results for houses and their prices for various zip code and locations within Boston.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW PostCode_Search AS
select p.price, a.postcode from price_table p
join address_table a on a.property_id = p.property_id
where postcode = '2215';



Use Case 5: Find all condos with exactly 2 bathrooms.

Precondition: The listings should be listed on Zillow's website.

Steps: We select prices, number of bathrooms from two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the number of bathroom and house type criteria using the WHERE clause. Please note that these query conditions can be slightly altered producing a combination of search results for various types of houses and bathroom numbers.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW House_Details AS
select p.price, h.bathroom_number, h.property_type from price_table p
join house_listing h on p.property_id = h.property_id
where h.bathroom_number=2 and h.property_type like '%condo%';

```
29    -- 5
30 •  CREATE VIEW House_Details AS
31    select p.price, h.bathroom_number, h.property_type from price_table p
32    join house_listing h on p.property_id = h.property_id
33    where h.bathroom_number=2 and h.property_type like '%condo%';
34 •  select * from House_Details;
35
36
```

100% 29:34

Result Grid Filter Rows: Search Export:

price	bathroom_num...	property_ty...
741500	2	CONDO
1290000	2	CONDO
989000	2	CONDO
934900	2	CONDO
1175000	2	CONDO
1075000	2	CONDO
1275000	2	CONDO
1365000	2	CONDO
2100000	2	CONDO
1699000	2	CONDO
1225000	2	CONDO
1749000	2	CONDO
579000	2	CONDO
1575000	2	CONDO
1150000	2	CONDO
1100000	2	CONDO
1179000	2	CONDO
995000	2	CONDO
2200000	2	CONDO
1049000	2	CONDO
1299000	2	CONDO

Use Case 6: Find the Zestimate of multi-family houses.

Precondition: The listings should be listed on Zillow's website with reasonable Zestimate score.

Steps: We select Zestimate, property type from two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the property type criteria using the WHERE clause. Please note that these query conditions can be slightly altered producing a combination of search results for various house types.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Property_type AS
select p.zestimate, h.property_type from price_table p
join house_listing h on p.property_id = h.property_id
where h.property_type like '%multi_family%';

```
36    -- 6
37 •  CREATE VIEW Property_type AS
38    select p.zestimate, h.property_type from price_table p
39    join house_listing h on p.property_id = h.property_id
40    where h.property_type like '%multi_family%';
41 •  select * from Property_type;
42
43
```

100% 29:41

Result Grid | Filter Rows: Search | Export:

zestimate	property_type
951500	MULTI_FAMILY
786300	MULTI_FAMILY
999000	MULTI_FAMILY
2879000	MULTI_FAMILY
850000	MULTI_FAMILY
2999000	MULTI_FAMILY
4750000	MULTI_FAMILY
1489000	MULTI_FAMILY
1367300	MULTI_FAMILY
3149500	MULTI_FAMILY
4065000	MULTI_FAMILY
2950000	MULTI_FAMILY
3678600	MULTI_FAMILY
4766400	MULTI_FAMILY
6188400	MULTI_FAMILY
6591000	MULTI_FAMILY
1683500	MULTI_FAMILY

Use Case 7: Find the addresses of the listed houses with area greater than 1300 sqft.

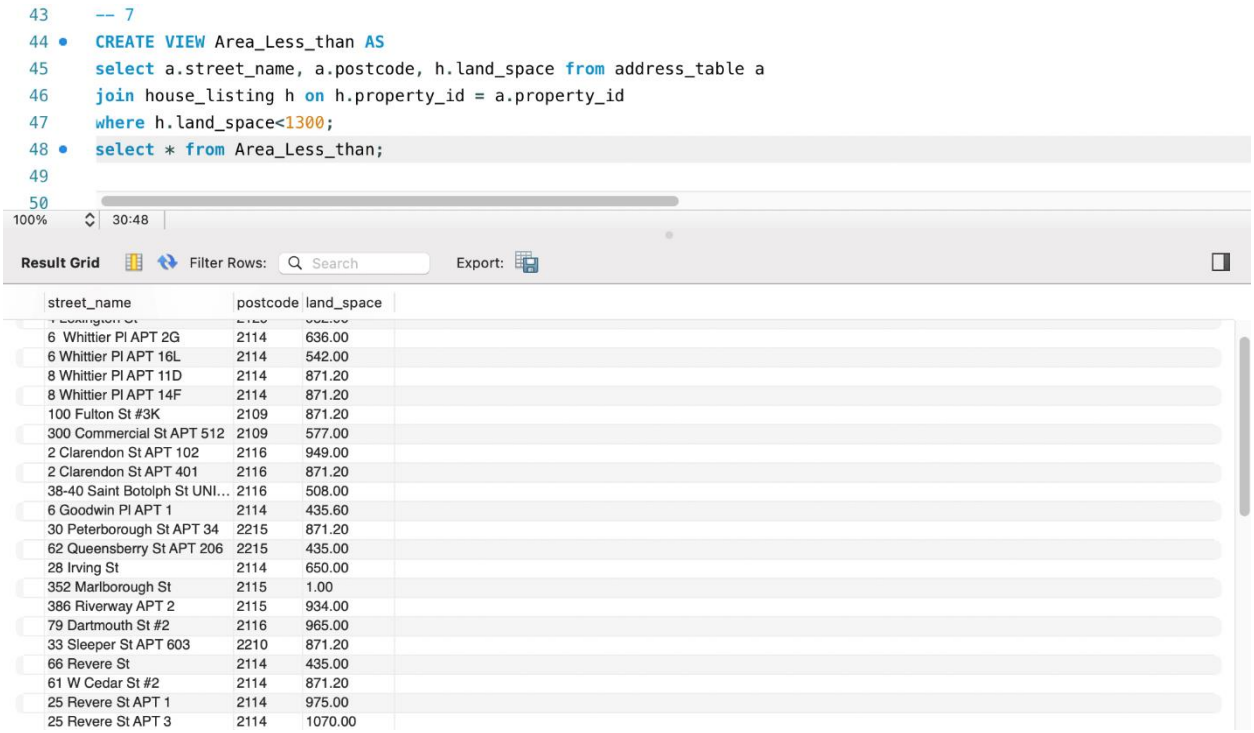
Precondition: The listings should be listed on Zillow's website.

Steps: We select street name, postcode, and area attributes from two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify 'land_space' attribute with suitable value using the WHERE clause. Please note that these query conditions can be slightly altered producing a combination of search results for various areas and locations.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Area_Less_than AS
select a.street_name, a.postcode, h.land_space from address_table a
join house_listing h on h.property_id = a.property_id
where h.land_space<1300;



street_name	postcode	land_space
6 Whittier Pl APT 2G	2114	636.00
6 Whittier Pl APT 16L	2114	542.00
8 Whittier Pl APT 11D	2114	871.20
8 Whittier Pl APT 14F	2114	871.20
100 Fulton St #3K	2109	871.20
300 Commercial St APT 512	2109	577.00
2 Clarendon St APT 102	2116	949.00
2 Clarendon St APT 401	2116	871.20
38-40 Saint Botolph St UNI...	2116	508.00
6 Goodwin Pl APT 1	2114	435.60
30 Peterborough St APT 34	2215	871.20
62 Queensberry St APT 206	2215	435.00
28 Irving St	2114	650.00
352 Marlborough St	2115	1.00
386 Riverway APT 2	2115	934.00
79 Dartmouth St #2	2116	965.00
33 Sleeper St APT 603	2210	871.20
66 Revere St	2114	435.00
61 W Cedar St #2	2114	871.20
25 Revere St APT 1	2114	975.00
25 Revere St APT 3	2114	1070.00

Use Case 8: Find the addresses of the listed houses which are under-valued according to Zillow.

Precondition: The listings should be listed on Zillow's website.

Steps: We select the unique attribute named 'property_id', address, price and Zestimate attributes from two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the condition of Zestimate being less than the listed price using the WHERE clause. Please note that these query conditions can be slightly altered producing a combination of search results for various areas and locations. This would also return the list of over-valued properties as well.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Undervalued_Properties AS
select p.property_id, a.street_name, a.city, a.rstate, a.postcode, p.price, p.zestimate from price_table p
join address_table a on a.property_id = p.property_id
where p.zestimate < p.price;

```
50    -- 8
51 •  CREATE VIEW Undervalued_Properties AS
52    select p.property_id, a.street_name, a.city, a.rstate, a.postcode, p.price, p.zestimate from price_table p
53    join address_table a on a.property_id = p.property_id
54    where p.zestimate < p.price;
55 •  select * from Undervalued_Properties;
56
57
```

100% 38:55

Result Grid Filter Rows: Search Export:

property_id	street_name	city	rstate	postcode	price	zestimate
190005995	820-824 Huntington Ave	Boston	MA	2115	2999000	2879000
59091876	7 Saybrook St #1	Boston	MA	2135	469000	431000
59212696	36 Symphony Rd APT 4A	Boston	MA	2115	1049000	1036500
59177283	65 Burbank St APT 14	Boston	MA	2115	499000	496000
59172993	138 Saint Botolph St	Boston	MA	2115	5495000	5278400
59177003	51 Hemenway St APT 1	Boston	MA	2115	750000	735700
59169052	27 Hereford St	Boston	MA	2115	4999999	4575300
59171698	79 Dartmouth St #2	Boston	MA	2116	1075000	1026000
59165005	2 Clarendon St APT 102	Boston	MA	2116	989000	980500
20928041...	8 Howell St #1	Boston	MA	2125	849000	737500
60112593	7 Charles St	Boston	MA	2129	1399000	1198900
59174060	77 Myrtle St	Boston	MA	2114	3995000	3843600
67715677	25 Channel Center St U...	Boston	MA	2210	1498000	1481100
123845214	580 Washington St #3	Boston	MA	2111	3250000	2976000
59169456	28 Irving St	Boston	MA	2114	2795000	2753100
90136195	99-105 Broad St #8B	Boston	MA	2110	1495000	1345200
59174085	25 Revere St APT 1	Boston	MA	2114	1275000	1269100
81857238	44 Prince St APT 504	Boston	MA	2113	1995000	1961700
59101062	4 Lexington St	Boston	MA	2129	1549000	1537600
59175437	85 E India Row APT 25G	Boston	MA	2110	1225000	1223700
81857648	80 Broad St UNIT 808	Boston	MA	2110	1299000	1278500

Use Case 9: Find the average Zestimate for a given zip code.

Precondition: The listings should be listed on Zillow's website and have a reasonable Zestimate score.

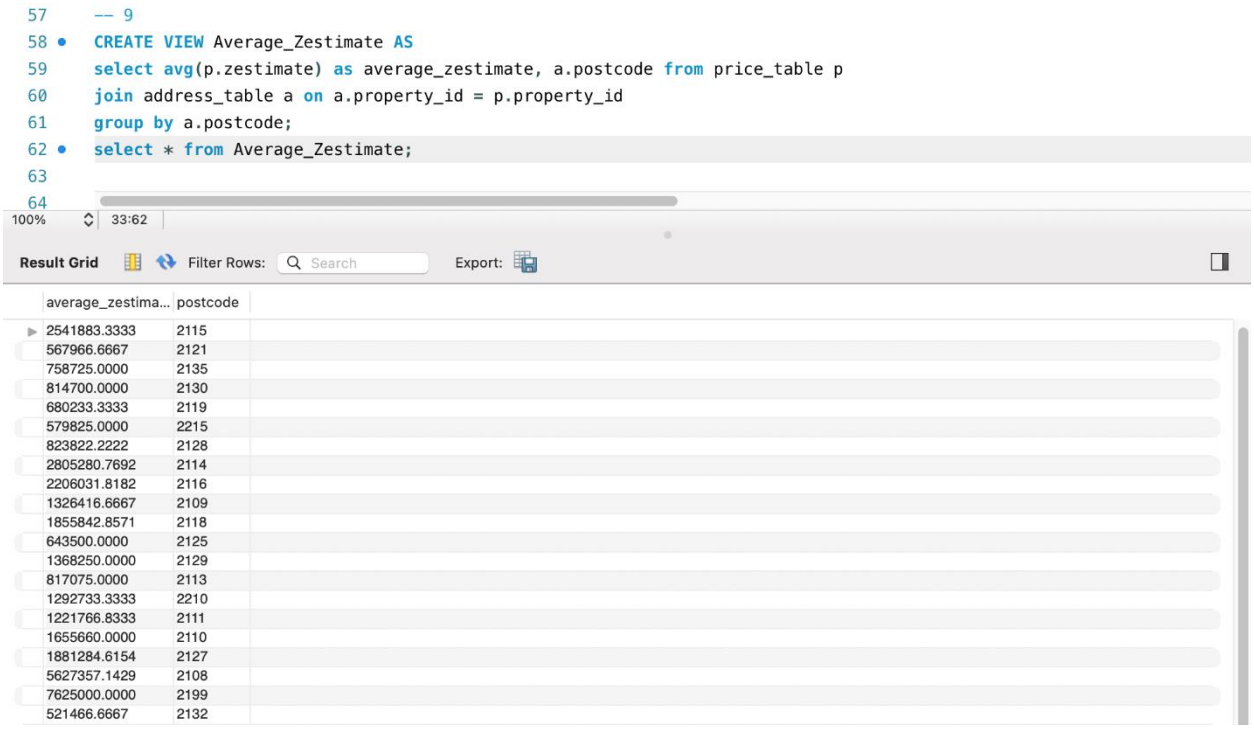
Steps: In this query we use the aggregate function named 'AVG'.

We select the Zestimate and apply the average aggregate function as average_zestimate, postcode from two tables and perform JOIN operation on the unique attribute 'property_id'. We then use the 'GROUP BY' aggregate function on postcode. Please note that these query conditions can be slightly altered producing using other aggregate functions such as 'MIN', 'MAX', 'COUNT', etc.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

```
CREATE VIEW Average_Zestimate AS
select avg(p.zestimate) as average_zestimate, a.postcode from price_table p
join address_table a on a.property_id = p.property_id
group by a.postcode;
```



```
57    -- 9
58 •  CREATE VIEW Average_Zestimate AS
59    select avg(p.zestimate) as average_zestimate, a.postcode from price_table p
60    join address_table a on a.property_id = p.property_id
61    group by a.postcode;
62 •  select * from Average_Zestimate;
63
64
```

100% 33:62

Result Grid | Filter Rows: Search | Export:

average_zestima...	postcode
2541883.3333	2115
567966.6667	2121
758725.0000	2135
814700.0000	2130
680233.3333	2119
579825.0000	2215
823822.2222	2128
2805280.7692	2114
2206031.8182	2116
1326416.6667	2109
1855842.8571	2118
643500.0000	2125
1368250.0000	2129
817075.0000	2113
1292733.3333	2210
1221766.8333	2111
1655660.0000	2110
1881284.6154	2127
5627357.1429	2108
7625000.0000	2199
521466.6667	2132

Use Case 10: Find the addresses of the listed houses which are over-valued according to Zillow.

Precondition: The listings should be listed on Zillow's website with a reasonable Zestimate score.

Steps: We select the unique attribute named 'property_id', address, price and Zestimate attributes from two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the condition of Zestimate being greater than the listed price using the WHERE clause. Please note that these query conditions can be slightly altered producing a combination of search results for various areas and locations. This would also return the list of under-valued properties as well.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

```
CREATE VIEW Overvalued_Properties AS
select a.street_name, a.postcode, p.zestimate, p.price from price_table p
join address_table a on a.property_id = p.property_id
where p.zestimate > p.price;
```

```
64    -- 10
65 •  CREATE VIEW Overvalued_Properties AS
66    select a.street_name, a.postcode, p.zestimate, p.price from price_table p
67    join address_table a on a.property_id = p.property_id
68    where p.zestimate > p.price;
69 •  select * from Overvalued_Properties;
70
71
```

100% 37:69

Result Grid Filter Rows: Search Export:

street_name	postcode	zestimate	price
386 Riverway APT 2	2115	590100	495000
301 Chestnut Ave #1	2130	814700	799000
30 Peterborough St APT 34	2215	586000	585000
108 Peterborough St APT 6A	2215	649700	649000
5 Braemore Rd APT 5	2135	355900	339999
100 Fulton St #3K	2109	776700	710000
1387 Washington St APT 504	2118	1114200	995000
8 Whittier Pl APT 11D	2114	557400	534900
58 E Springfield St #5	2118	745800	699000
304 Shawmut Ave	2118	4766400	4200000
26 Stillman St APT 2-1	2113	1331800	1299000
21 Father Francis Gilday St...	2118	968100	899000
162 Salem St APT 5	2113	537500	525000
12 Worcester Sq APT 1	2118	1907000	1695000
300 Commercial St APT 512	2109	555300	539900
3 Magnolia Pl	2125	951500	950000
54 Monadnock St #2	2125	445600	444900
43 Upton St APT 2	2118	686400	675000
158 F St APT 2	2127	538000	529000
210 South St UNIT 115	2111	825000	824999
151 Tremont St APT 6C	2111	650100	649000

Use Case 11: Find all listed houses within the specified area and conditions.

Precondition: The listings should be listed on Zillow's website.

Steps: We select the unique attribute named 'property_id', bedroom_number, bathroom_number, living_space, and land_space attributes two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the condition of land price per unit area between $300 and $1500 per square feet using the 'WHERE' clause. Please note that these query conditions can be slightly altered producing a combination of search results for various locations and parameters.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Price_Range AS
select h.property_id, bedroom_number, bathroom_number, living_space, land_space,
property_status, property_type, price_per_unit from house_listing h
join price_table p on p.property_id = h.property_id
where price_per_unit between 300 and 1500;

```
71    -- 11
72 •  CREATE VIEW Price_Range AS
73    select h.property_id, bedroom_number, bathroom_number, living_space, land_space,
74    property_status, property_type, price_per_unit from house_listing h
75    join price_table p on p.property_id = h.property_id
76    where price_per_unit between 300 and 1500;
77 •  select * from Price_Range;
78
```

100% 27:77

Result Grid Filter Rows: Search Export:

property_id	bedroom_numb...	bathroom_num...	living_space	land_space	property_stat...	property_type	price_per_u...
59171381	1	1	934	934.00	FOR_SALE	CONDO	529
189924486	4	4	2478	2613.60	FOR_SALE	MULTI_FAMILY	403
118162434	2	2	1141	1306.80	FOR_SALE	CONDO	700
59188486	1	2	1152	1306.80	FOR_SALE	CONDO	1023
81854345	3	2	1540	1742.40	FOR_SALE	CONDO	479
59168062	1	1	659	871.20	FOR_SALE	CONDO	887
81858046	1	1	352	435.60	FOR_SALE	CONDO	1474
81857998	1	1	620	620.00	FOR_SALE	CONDO	1046
190005995	10	6	4740	1500.00	FOR_SALE	MULTI_FAMILY	632
59091876	2	1	832	871.20	FOR_SALE	CONDO	563
20855795...	1	1	385	435.60	FOR_SALE	CONDO	883
20851598...	3	3	3300	2362.00	FOR_SALE	CONDO	378
59212696	3	2	1140	1306.80	FOR_SALE	CONDO	920
59177283	1	1	459	459.00	FOR_SALE	CONDO	1087
59172993	5	5	5098	2178.00	FOR_SALE	SINGLE_FAM...	1077
59177003	2	1	862	862.00	FOR_SALE	CONDO	870
59169052	5	4	3737	1742.40	FOR_SALE	SINGLE_FAM...	1337
20666744...	8	6	4440	2613.60	FOR_SALE	MULTI_FAMILY	335
59173708	2	1	750	871.20	FOR_SALE	CONDO	1158
59171698	2	2	965	965.00	FOR_SALE	CONDO	1113
59164404	1	1	771	871.20	FOR_SALE	CONDO	920

Use Case 12: Find top 3 most expensive properties under a specific postcode.

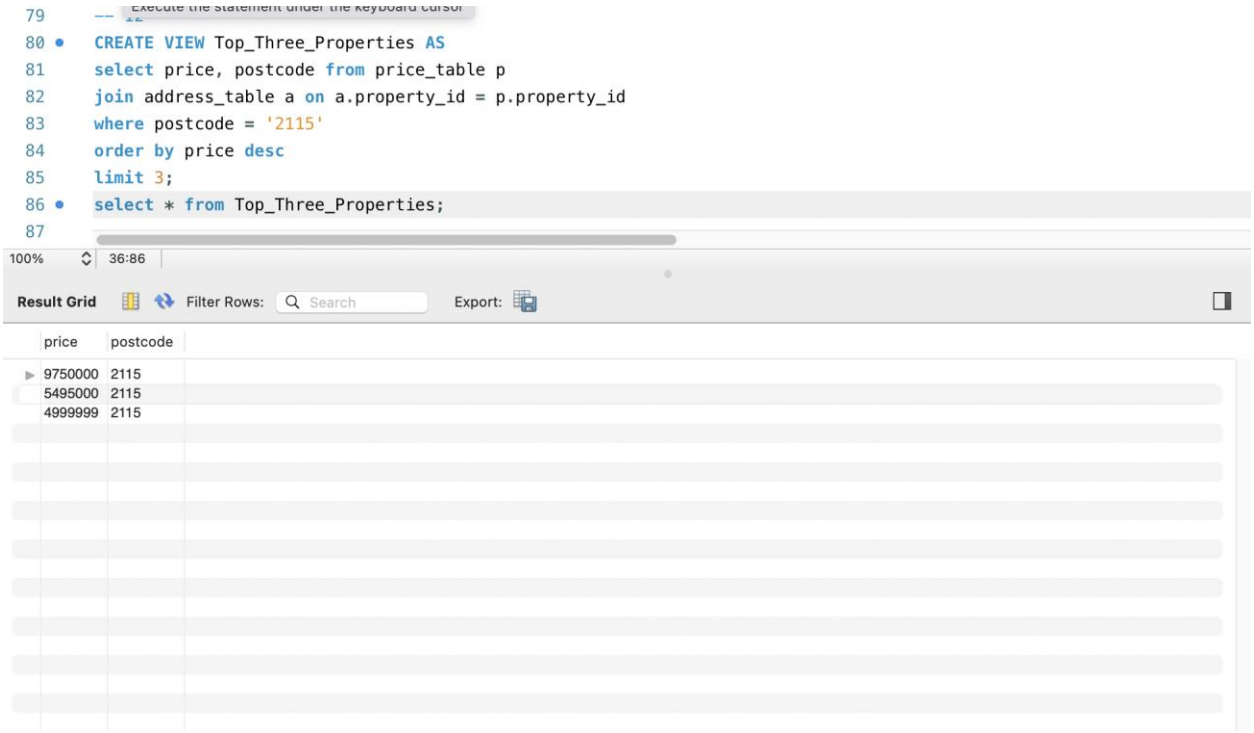
Precondition: The listings should be listed on Zillow's website.

Steps: We select the unique attribute named 'property_id', postcode, and price attributes two tables and perform JOIN operation on the unique attribute 'property_id'. We then specify the condition of the specific postcode using the 'WHERE' clause. We then use the 'ORDER BY' statement to arrange these in descending order. We then limit the results to 3 to view the top three most expensive house listings available. Please note that these query conditions can be slightly altered producing a combination of search results for various locations and parameters.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

```
CREATE VIEW Top_Three_Properties AS
select price, postcode from price_table p
join address_table a on a.property_id = p.property_id
where postcode = '2115'
order by price desc
limit 3;
```



Use Case 13: Find address and other details of the listed houses which are neither under-valued nor over-valued, according to Zillow's index 'Zestimate'.

Precondition: The listings should be listed on Zillow's website and have a reasonable Zestimate score.

Steps: We select the unique attribute named 'property_id', address, bedroom_number, bathroom_number, living_space, Zestimate, price attributes from three tables and perform JOIN operation on the unique attribute 'property_id' in both join operations. The first join is between 'house_listing' and 'address' tables whereas the second join is between 'price_table' and 'address' table. We then specify the condition of the specific in which Zestimate equals the price of the property using the 'WHERE' clause. Please note that these query conditions can be slightly altered producing a combination of search results for various locations and parameters.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Same_Price AS
select street_name, city, rstate, postcode, bedroom_number, bathroom_number, living_space, zestimate, price from address_table a
join house_listing h on h.property_id = a.property_id
join price_table p on p.property_id = a.property_id
where zestimate = price;

```
88    -- 13
89 •  CREATE VIEW Same_Price AS
90    select street_name, city, rstate, postcode, bedroom_number, bathroom_number, living_space, zestimate, price
91    from address_table a
92    join house_listing h on h.property_id = a.property_id
93    join price_table p on p.property_id = a.property_id
94    where zestimate = price;
95 •  select * from Same_Price;
96
```

100% 1:91

Result Grid Filter Rows: Search Export:

street_name	city	rstate	postcode	bedroom_numb...	bathroom_num...	living_space	zestimate	price
117 Sutherland Rd #117	Boston	MA	2135	3	3	3300	1249000	1249000
352 Marlborough St	Boston	MA	2115	4	7	4754	9750000	9750000
126 Marion St	Boston	MA	2128	8	6	4440	1489000	1489000
61 W Cedar St #2	Boston	MA	2114	2	1	750	869000	869000
12 Isabella St APT 1	Boston	MA	2116	2	3	1124	1265000	1265000
25 Revere St APT 3	Boston	MA	2114	3	2	1070	1365000	1365000
202 Maverick St #105	Boston	MA	2128	1	1	605	519000	519000
23 Bradford St #2	Boston	MA	2118	3	3	1918	1995000	1995000
1313 Washington St A...	Boston	MA	2118	3	3	1918	1995000	1995000
400 Stuart St #19A	Boston	MA	2116	3	3	1795	3695000	3695000
2 Rollins St APT D604	Boston	MA	2118	1	2	1049	1349000	1349000
134 Fulton St #5	Boston	MA	2109	2	2	1609	1395000	1395000
12 O St	Boston	MA	2127	6	6	3714	2950000	2950000
28-30 Mercer St	Boston	MA	2127	11	6	7993	2999000	2999000
226 Saratoga St FLO...	Boston	MA	2128	3	4	1441	689000	689000
204 E Eagle St FLOO...	Boston	MA	2128	2	2	839	559000	559000
538 Tremont St #1-4	Boston	MA	2118	4	4	4400	4750000	4750000
204 E Eagle St FLOO...	Boston	MA	2128	2	2	833	609000	609000
194 E St #1	Boston	MA	2127	2	1	800	699000	699000
45 Lewis St UNIT 406	Boston	MA	2128	1	1	754	865000	865000

Use Case 14: Find top 3 popular agencies with highest number of listings.

Precondition: The listings should be listed on Zillow's website. The listings must have at least one agency associated to it.

Steps: We select the name of the agency and use aggregate function 'COUNT' on the attribute 'agency_id' as 'famous_agency' from two tables and perform JOIN operation on the attribute 'agency_id'. We then use the aggregate function named 'GROUP BY' to group all agencies with their unique ids. The query then performs the descending arrangement of the names of the agencies using the 'ORDER BY' statement. We limit the number of results to three as we have initially asked to view these three most popular agencies. Please note that these query conditions can be slightly altered producing a combination of search results for other parameters and arrangements.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

```
CREATE VIEW Famous_Agency AS
select count(h.agency_id) as famous_agency, agency_name from house_listing h
join agency a on a.agency_id = h.agency_id
group by h.agency_id
order by famous_agency desc
limit 3;
```

```
79
80 •  CREATE VIEW Top_Three_Properties AS
81    select price, postcode from price_table p
82    join address_table a on a.property_id = p.property_id
83    where postcode = '2115'
84    order by price desc
85    limit 3;
86 •  select * from Top_Three_Properties;
87
```

100% 36:86

Result Grid | Filter Rows: Search | Export:

price	postcode
9750000	2115
5495000	2115
4999999	2115

Use Case 15: Find property and agency details of the desired properties.

Precondition: The listings should be listed on Zillow's website.

Steps: We select the unique attribute named 'property_id', bedroom_number, bathroom_number, living_space, agency_name from two tables and perform JOIN operation on the attribute 'agency_id'. We then specify the condition of number of bedrooms greater than 3 using the 'WHERE' clause. Please note that these query conditions can be slightly altered producing a combination of search results for various number of bedrooms or bathrooms among many other possible combinations.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Greater_Than_Three AS
select h.property_id, bedroom_number, bathroom_number, living_space, agency_name from house_listing h
join agency a on a.agency_id = h.agency_id
where bedroom_number>3;

```
106     -- 15
107 •   CREATE VIEW Greater_Than_Three AS
108     select h.property_id, bedroom_number, bathroom_number, living_space, agency_name from house_listing h
109     join agency a on a.agency_id = h.agency_id
110     where bedroom_number>3;
111 •   select * from Greater_Than_Three;
112
```

100% 34:111

Result Grid Filter Rows: Search Export:

property_id	bedroom_numb...	bathroom_num...	living_space	agency_name
59166308	5	6	7482	Campion & Company Fine Homes Real Estate
59169052	5	4	3737	Centre Realty Group
59169848	4	7	4754	MGS Group Real Estate LTD
59172993	5	5	5098	Sprogis & Neale Real Estate
59173220	5	4	4188	Engel & Volkers Boston
59173446	4	5	3966	Campion & Company Fine Homes Real Estate
59173451	5	5	4318	William Raveis R.E. & Home Services
59174060	5	4	3300	Gibson Sotheby's International Realty
59174448	5	6	3825	Berkshire Hathaway HomeServices Warren...
59174642	4	5	5438	Campion & Company Fine Homes Real Estate
59174916	4	6	5322	Broad Street Boutique Realty LLC
59180675	5	4	4212	Campion & Company Fine Homes Real Estate
67709760	4	2	1295	Coldwell Banker Realty - Back Bay
102988140	4	5	2428	Gibson Sotheby's International Realty
113389948	4	6	5799	Coldwell Banker Realty - Back Bay
189924486	4	4	2478	Donnelly + Co.
189950691	4	5	3607	MGS Group Real Estate LTD
190005995	10	6	4740	Advisors Living - Back Bay
190039888	5	3	2288	Signal Real Estate
20660466...	11	6	7993	Boston Realty Advisors

Use Case 16: Find property and agency details of the pending properties (not yet sold or bought).

Precondition: The listings should be listed on Zillow's website.

Steps: We select the attribute named 'agency_id', the unique attribute named 'property_id', bedroom_number, bathroom_number, living_space, property_status, agency_name from two tables and perform JOIN operation on the attribute 'agency_id'. We then specify the condition of the status of property as pending using the 'WHERE' clause.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Pending_Properties_of_an_agency AS
select h.property_id, bedroom_number, bathroom_number, living_space, property_status, agency_name from house_listing h
join agency a on a.agency_id = h.agency_id
where property_status = 'PENDING';

```
114     -- 16
115 •   CREATE VIEW Pending_Properties_of_an_agency AS
116     select h.property_id, bedroom_number, bathroom_number, living_space, property_status, agency_name from house_listing h
117     join agency a on a.agency_id = h.agency_id
118     where property_status = 'PENDING';
119 •   select * from Pending_Properties_of_an_agency;
120
```

100% 47:119

Result Grid Filter Rows: Search Export:

property_id	bedroom_numb...	bathroom_num...	living_space	property_stat...	agency_name
59161881	8	3	5542	PENDING	Compass
59168275	1	1	643	PENDING	Redfin Corp.
59173636	3	4	2625	PENDING	Redfin Corp.
59174573	2	1	1300	PENDING	Donnelly + Co.
59174916	4	6	5322	PENDING	Broad Street Boutique Realty LLC
59183569	3	3	2213	PENDING	Longwood Residential, LLC
59188434	2	2	1559	PENDING	Luxury Residential Group, LLC
59212026	3	3	1531	PENDING	McCormack & Scanlan Real Estate
63042601	2	3	1462	PENDING	Keller Williams Realty Boston-Metro I Back Bay
67715620	2	2	1068	PENDING	Keller Williams Realty Cambridge
71546584	2	2	1460	PENDING	Unlimited Sotheby's International Realty
81857945	1	1	760	PENDING	Carpenter and Company, Inc.
90136786	2	2	1011	PENDING	Boston Realty Net
90138053	1	1	856	PENDING	Advisors Living - Back Bay
189938322	1	2	1341	PENDING	Engel & Volkers Boston

Use Case 17: Find property which costs more than $7000000

Precondition: The listings should be listed on Zillow's website.

Steps: We select the unique attribute named 'property_id', address, price from two tables and perform JOIN operation on the attribute 'property_id'. We then specify the condition of the price more than $7000000 using the 'WHERE' clause.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Expensive_Properties AS
select a.street_name, a.city, a.rstate, a.postcode, p.price from address_table a
join price_table p on p.property_id = a.property_id
where p.price>7000000;

```
122    -- 17
123 •  CREATE VIEW Expensive_Properties AS
124    select a.street_name, a.city, a.rstate, a.postcode, p.price from address_table a
125    join price_table p on p.property_id = a.property_id
126    where p.price>7000000;
127 •  select * from Property_Details;
128
```

100% 32:127

Result Grid Filter Rows: Search Export:

price	zestimate	living_space	land_space
455000	461100	1232	1306.80
425000	439400	980	1788.00
444900	445600	1161	1306.80
950000	951500	1830	3103.00
895000	889200	1116	3600.00
649500	686000	1464	3484.80
459000	477900	700	4356.00
780000	786300	5542	7840.00
679900	679700	1205	1306.80
475000	461900	636	636.00
741500	727900	1296	1306.80
439000	424600	542	542.00
534900	557400	871	871.20
624900	635300	908	871.20
2875000	2706600	1973	1973.00
1290000	1260500	2226	2226.00
710000	776700	771	871.20
539900	555300	577	577.00
989000	980500	949	949.00

Use Case 18: Find all properties which has living area greater than 500 sqft.

Precondition: The listings should be listed on Zillow's website and have a reasonable Zestimate score.

Steps: We select the unique attribute named 'property_id', price, Zestimate, living_space, land_space from two tables and perform JOIN operation on the attribute 'property_id'. We then specify the condition of living space greater than 500 sqft using the 'WHERE' clause.

System Responses: The system instantly returns all such results that meet our search criteria.

View Statement Query:

CREATE VIEW Property_Details AS
select p.price, p.zestimate, h.living_space, h.land_space from price_table p
join house_listing h on h.property_id = p.property_id
where h.living_space>500;

```
130    -- 18
131 •  CREATE VIEW Property_Details AS
132    select p.price, p.zestimate, h.living_space, h.land_space from price_table p
133    join house_listing h on h.property_id = p.property_id
134    where h.living_space>500;
135 •  select * from Property_Details;
136
137
```

100% 32:135

Result Grid Filter Rows: Search Export:

price	zestimate	living_space	land_space
455000	461100	1232	1306.80
425000	439400	980	1788.00
444900	445600	1161	1306.80
950000	951500	1830	3103.00
895000	889200	1116	3600.00
649500	686000	1464	3484.80
459000	477900	700	4356.00
780000	786300	5542	7840.00
679900	679700	1205	1306.80
475000	461900	636	636.00
741500	727900	1296	1306.80
439000	424600	542	542.00
534900	557400	871	871.20
624900	635300	908	871.20
2875000	2706600	1973	1973.00
1290000	1260500	2226	2226.00
710000	776700	771	871.20
539900	555300	577	577.00
989000	980500	949	949.00
799000	815000	705	871.20