Name: Papadopoulos Nikolaos

Course: Large Scale Data Management

Instructor: Dr. L.Sidirourgos

Introduction

In the semester project we were tasked with following a series of steps to gain firsthand experience with the process of building up a large-scale database management system/distributed OLAP system, loading data, querying that data, and eventually performing an analytical task.

This task mandated we install PySpark and the open-source column store MonetDB (With Python UDF support). PySpark is the Python API for Apache Spark, an open source, distributed computing framework and set of libraries for real-time, large-scale data processing. MonetDB is an open-source column-oriented relational database management system (RDBMS). It is designed to provide high performance on complex queries against large databases, such as combining tables with hundreds of columns and millions of rows. MonetDB has been applied in high-performance applications for online analytical processing, data mining, geographic information system and many other areas.

The dataset released was about real-estate (Data collected by Zillow for the Boston, MA area). The dataset consisted of title, address, city, state, postal code, price, facts and features, real estate provider and URL columns of which some were not used at all and were removed in the coding part.

Set-Up

Installing PySpark was fairly easy as only an installation using the pip install command was needed and then the PySpark environment was ready to be used using the import function in the python notebook. Then a session was to be created using the built-in function and after a few setting tweaks (That would optimize performance when using Pandas/NumPy data) everything was set up correctly to start the coding part.

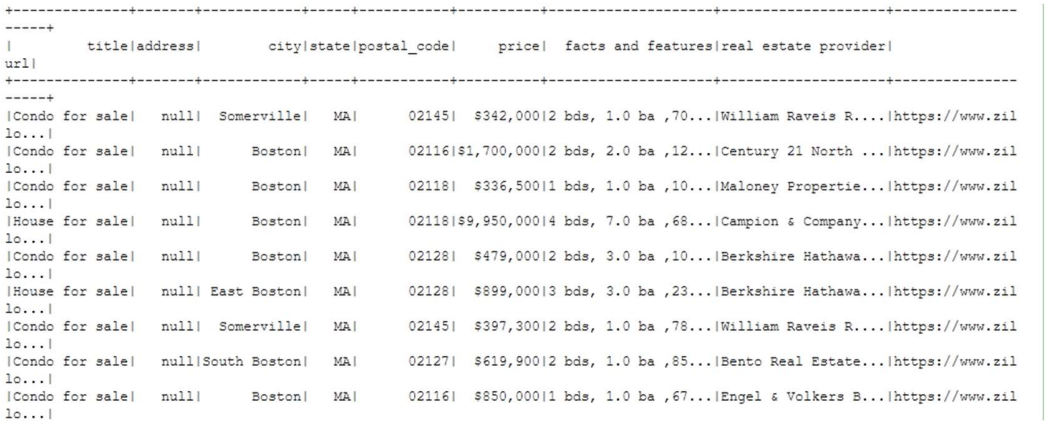
Setting up the Monet Database proved to be more challenging than expected as there were some errors that needed to be combated with but were not easy to find the solution for online. For my workstation, a registry change was needed as to correct the path that the database pulled the needed python version. (Because of previous installations/environments set up there was a mix-up with the versions of

Python installed and I experienced difficulty making the program grab the correct python version).Apart from that, downloading and running the installer set almost everything up and I only had to start the server by running a .bat file and then running the MonetDB client. After inputting the credentials given in the website the MonetDB environment was all set and the analysis/coding part followed.

Coding

PySpark

I first read the data using spark.read.csv(Equivalent of the pandas.read.csv) and having Headers set as True so the dataframe is properly read and I get a table with the following form:



```
+--------------+-------+------------+-----+-----------+----------+--------------------+--------------------+--------------
-----+
|         title|address|        city|state|postal_code|     price|  facts and features|real estate provider|
url|
+--------------+-------+------------+-----+-----------+----------+--------------------+--------------------+--------------
-----+
|Condo for sale|   null|  Somerville|   MA|      02145|  $342,000|2 bds, 1.0 ba ,70...|William Raveis R....|https://www.zil
lo...|
|Condo for sale|   null|      Boston|   MA|      02116|$1,700,000|2 bds, 2.0 ba ,12...|Century 21 North ...|https://www.zil
lo...|
|Condo for sale|   null|      Boston|   MA|      02118|  $336,500|1 bds, 1.0 ba ,10...|Maloney Propertie...|https://www.zil
lo...|
|House for sale|   null|      Boston|   MA|      02118|$9,950,000|4 bds, 7.0 ba ,68...|Campion & Company...|https://www.zil
lo...|
|Condo for sale|   null|      Boston|   MA|      02128|  $479,000|2 bds, 3.0 ba ,10...|Berkshire Hathawa...|https://www.zil
lo...|
|House for sale|   null| East Boston|   MA|      02128|  $899,000|3 bds, 3.0 ba ,23...|Berkshire Hathawa...|https://www.zil
lo...|
|Condo for sale|   null|  Somerville|   MA|      02145|  $397,300|2 bds, 1.0 ba ,78...|William Raveis R....|https://www.zil
lo...|
|Condo for sale|   null|South Boston|   MA|      02127|  $619,900|2 bds, 1.0 ba ,85...|Bento Real Estate...|https://www.zil
lo...|
|Condo for sale|   null|      Boston|   MA|      02116|  $850,000|1 bds, 1.0 ba ,67...|Engel & Volkers B...|https://www.zil
lo...|
```

Starting with task number 1 which asks that we extract the number of bedrooms by implementing a UDF that processes the 'facts and features' column. The column is of the form:

X bds, Y.0 baths ,V Sqft

Using the command pyspark.sql.functions.udf (I've renamed the first part to F so as to shortcut it / Hence it is used as F.udf) and a lambda function I split each cell (Which was a string) on commas (,) and got the first sub-string which consisted of 'X bds'. I then re-split the sub-string on spaces and got the first element which was the number of bedrooms. Now that I've created the required function I use it on the dataframe storing all the outputs in a column (using the df.withColumn command which runs the function on every cell of the column).Similarly for question 2 and 3(Extract Bathrooms & Sqft) I split again on the commas, pick the appropriate sub-string and then re-split on the needed string(Number). When extracting bathrooms there was an extra space after the comma so I had to work around that by getting the second element instead. I then fill all None values with 0s as I then proceed to turn all new columns from strings to integers as they will be used in the following tasks.

Onto our next question, we were asked to extract the type and I've accomplished that by creating a udf function with lambda that has nested ifs which will return the title of the type of the estate. For example, if the type had 'house' in that column then it assigned its type to 'house'. Similarly, when extracting offer I classified all offers using a nested if which produced one of the 4 needed offer-types(`sale`, `rent`, `sold`, `forclose`). For example, if the title had forclosure in it, it would be classified under the forclose category.

Next task was to filter out listings that were not for sale and that was done using a simple filter function df = df[df.offer=='Sale'] where df would be our dataframe and 'Sale' would be the type of house that was produced from the previous task.

We were then tasked with creating a UDF that parses the price(which was a string) from the price column and return the price as an integer. This task was done via regular expression operation and a lambda function in the form of : int(re.sub("[^0-9]", "",x))) where x was the cell's value.

Then we had to filter out listings with more than 10 bedrooms,out of those filter out listings with price greater than 20000000 and lower than 100000 and lastly filter out all offers that are not houses from the previous output. This was accomplished easily as I had already turned the strings into integers from the steps above.

As our final task, we were required to calculate average price per sqft for houses for sale grouping them by the number of bedrooms and that was done via the command:

df.groupBy("beds").agg(F.avg(df.prices/df.sqft))

Which grouped by number of bedrooms(groupBy) and calculated the average price per sqft for houses for sale(Using the pyspark.sql.functions command)

MonetDB

When working with MonetDB, after starting the server and the client a loader had to be created so as to read the dataframe properly. After initializing the loader, a table was created using it and the dataframe was set. The output was a table of the form:

title	city	state	postal_code	price	facts and features	real estate provider
Condo for sale	Somerville	MA	2145	$342,000	2 bds, 1.0 ba ,705 sqft	William Raveis R.E. & Home Services
Condo for sale	Boston	MA	2116	$1,700,000	2 bds, 2.0 ba ,1228 sqft	Century 21 North East
Condo for sale	Boston	MA	2118	$336,500	1 bds, 1.0 ba ,1000 sqft	Maloney Properties, Inc.
House for sale	Boston	MA	2118	$9,950,000	4 bds, 7.0 ba ,6836 sqft	Campion & Company Fine Homes Real Estate
Condo for sale	Boston	MA	2128	$479,000	2 bds, 3.0 ba ,1000 sqft	Berkshire Hathaway HomeServices Commonwealth Real Estate
House for sale	East Boston	MA	2128	$899,000	3 bds, 3.0 ba ,2313 sqft	Berkshire Hathaway HomeServices Warren Residential
Condo for sale	Somerville	MA	2145	$397,300	2 bds, 1.0 ba ,780 sqft	William Raveis R.E. & Home Services
Condo for sale	South Boston	MA	2127	$619,900	2 bds, 1.0 ba ,856 sqft	Bento Real Estate Group, Inc.
Condo for sale	Boston	MA	2116	$850,000	1 bds, 1.0 ba ,675 sqft	Engel & Volkers Boston
Condo for sale	Boston	MA	2114	$649,900	2 bds, 1.0 ba ,511 sqft	Leading Edge Real Estate

For the first question I create a function that initializes a list called results and then gets a string and splits it twice (First on commas and gets the first element then on space and gets the first element again/Similar to what I did in PySpark). I run the function on every cell of the required column and I've appended every output to my results list and as a last step I return the results list. I then create similar functions for questions 2 and 3 since they work on the same column (Facts & Features).

For the next question (4) I create a function that takes a string and returns a string based on the type of the estate. Title column is of the form:

Condo for sale/House for sale/Townhouse

That is accomplished via nested ifs that checks whether, for example, the word construction is in that string (After lowercasing it so as to avoid any capitalization mistakes) and if that Boolean returns true then the type of that estate would be classified under 'construction'. I consider houses and townhouses

to be in the same category so I include both under the houses class. Also, I have added an extra category named Other for types that were not included in the first/Missing Values.

For question 5 I create a function that takes a string and returns a string based on the offer type of the estate. Again, working on the title column, we get cells of the form:

Condo for sale/Foreclosure/New Construction

This is done via nested ifs that work on the string we received as input. For example, I classify the estate as for 'Sale' if it has sale in the string (After lowercasing again to avoid any caps mismatch).

I then create a table with all the filters above and add an extra filter that only includes the offers that are up for sale which was required in the next task adding a where (offer='Sale') in my create table statement.

I then create a function that takes the price (which was a string) and turns it into an integer.

The price column is of the form:

$342,000/$1,700,000/etc.

I first get the string after the first character because that is the dollar symbol and then replace commas and + and any extra space with nothing ('') so as to remove any extra symbols. Out of the newly created table I run the function above and get the prices as integers and create a new table called PriceValues to store these prices.

As for the next 3 filtering questions, I create one table in order to save space which has all the filters required in the form of:

```
create table final_filters as select *  from test where beds <=10 and (PriceValue > 100000) and (PriceValue < 20000000) and (type='House')
```

The above statement filters all estates where number of beds is under 10 and Price is between 100000 and 20000000 and have their type is classified under the House category.

As for the last question I run a select command that gets the average of the pricevalue divided by the Sqft from the final_filters table and have them grouped by number of beds.

```
select avg(pricevalue/Sqft) as avg_price_per_sqft,beds from final_filters group by beds;
```

Results for both MonetDB and PySpark are the same which means that given the decisions I had made on filtering both worked similarly. (±1 which might be due to rounding).

Conclusion

The dataset was part of Zillow's data on real estate and was most probably used to analyze and check for any patterns or get a better insight on the real estate information regarding the Boston,MA area. Even though we only had a few columns and around 100.000 rows and that may not be considered as 'Big Data', it was beneficial to tamper with both PySpark and MonetDB and have some hands-on

experience when it comes to data analysis. Data analytics technologies and techniques give organizations a way to analyze data sets and gather new information. Business intelligence (BI) queries answer basic questions about business operations and performance. Organizations can use big data analytics systems and software to make data-driven decisions that can improve business-related outcomes.

MonetDB **achieves significant speed up compared to more traditional designs by innovations at all layers of a DBMS** and PySpark **is a Python-based API for utilizing the Spark framework in combination with Python** (It is a Big Data computational engine).

A first experience with large scale data processing is valuable because regular python usage is not able to handle vast amount of data as it is mainly a programming language whereas MonetDB and PySpark are used in Big Data Analytics and can accomplish the aforementioned task.