**Module 22 Challenge – Home\_Sales**

In this challenge, you'll use your knowledge of SparkSQL to determine key metrics about home sales data. Then you'll use Spark to create temporary views, partition the data, cache and uncache a temporary table, and verify that the table has been uncached.

**Before You Begin**

1. Create a new repository for this project called, Home\_Sales. Do not add this homework to an existing repository.
2. Clone the new repository to your computer.
3. Push your changes to GitHub.

Instructions

1. Rename the Home\_Sales\_starter\_code.ipynb file as Home\_Sales.ipynb.
2. Import the necessary PySpark SQL functions for this assignment.
3. Read the home\_sales\_revised.csv data in the starter code into a Spark DataFrame.
4. Create a temporary table called home\_sales.
5. Answer the following questions using SparkSQL:

* What is the average price for a four-bedroom house sold for each year? Round off your answer to two decimal places.
* What is the average price of a home for each year it was built that has three bedrooms and three bathrooms? Round off your answer to two decimal places.
* What is the average price of a home for each year that has three bedrooms, three bathrooms, two floors, and is greater than or equal to 2,000 square feet? Round off your answer to two decimal places.
* What is the "view" rating for homes costing more than or equal to $350,000? Determine the run time for this query, and round off your answer to two decimal places.

1. Cache your temporary table home\_sales.
2. Check if your temporary table is cached.
3. Using the cached data, run the query that filters out the view ratings with an average price of greater than or equal to $350,000. Determine the runtime and compare it to uncached runtime.
4. Partition by the "date\_built" field on the formatted parquet home sales data.
5. Create a temporary table for the parquet data.
6. Run the query that filters out the view ratings with an average price of greater than or equal to $350,000. Determine the runtime and compare it to uncached runtime.
7. Uncache the home\_sales temporary table.
8. Verify that the home\_sales temporary table is uncached using PySpark.
9. Download your Home\_Sales.ipynb file and upload it into your "Home\_Sales" GitHub repository.

**Requirements**

1. A Spark DataFrame is created from the dataset. (5 points)
2. A temporary table of the original DataFrame is created. (10 points)
3. A query is written that returns the average price, rounded to two decimal places, for a four-bedroom house that was sold in each year. (5 points)
4. A query is written that returns the average price, rounded to two decimal places, of a home that has three bedrooms and three bathrooms. (5 points)
5. A query is written that returns the average price of a home with three bedrooms, three bathrooms, two floors, and is greater than or equal to 2,000 square feet for each year built rounded to two decimal places. (5 points)
6. A query is written that returns the view rating for the average price for homes that are greater than or equal to $350,000, rounded to two decimal places. (The output shows the run time for this query.) (10 points)
7. A cache of the temporary "home\_sales" table is created and validated. (10 points)
8. The query from step 6 is run on the cached temporary table, and the run time is computed. (10 points)
9. A partition of the home sales dataset by the "date\_built" field is created, and the formatted parquet data is read. (10 points)
10. A temporary table of the parquet data is created. (10 points)
11. The query from step 6 is run on the parquet temporary table, and the run time is computed. (10 points)
12. The "home\_sales" temporary table is uncached and verified. (10 points)

**Submission**

You are required to submit the URL of your GitHub repository for grading.