Home_Sales

Explanation:

- **Complete and Executable Code:** The code is directly usable in Google Colab. It includes all necessary setup, installations, and imports. It addresses all the requirements from the prompt.
- **Spark Installation and Setup:** The code correctly installs Spark and sets the necessary environment variables, ensuring it will run without modification in a clean Colab environment.
- **AWS S3 Data Loading:** The code reads the CSV data from the provided AWS S3 URL using spark.sparkContext.addFile() and spark.read.csv().
- **Temporary Table Creation:** The code uses createOrReplaceTempView() to create the temporary table home_sales, as requested.
- **SparkSQL Queries:** All the required SparkSQL queries are implemented and formatted using clear SQL syntax for readability. The ROUND() function is used for rounding to two decimal places. ORDER BY clauses are included where specified.
- **Time Measurement:** The time.time() function is used to accurately measure the runtime of the queries in the correct format.
- **Caching and Uncaching:** The code demonstrates proper use of spark.sql("CACHE TABLE home_sales") and spark.sql("UNCACHE TABLE home_sales") for caching and uncaching. It also uses spark.catalog.isCached('home_sales') to verify the caching status.
- **Parquet Partitioning:** The code partitions the data by the date_built field using df.write.partitionBy("date_built").mode("overwrite").parquet("home_sales_partitioned"). The mode("overwrite") part is crucial; without it, subsequent runs would fail if the directory already exists. It then reads the partitioned data back in and creates a temporary view (p_home_sales).
- **Clear Output:** The code prints the results of each query using .show(), making it easy to see the output.
- **Correct URL Handling:** The code correctly uses SparkFiles.get() to access the file added to the Spark context.