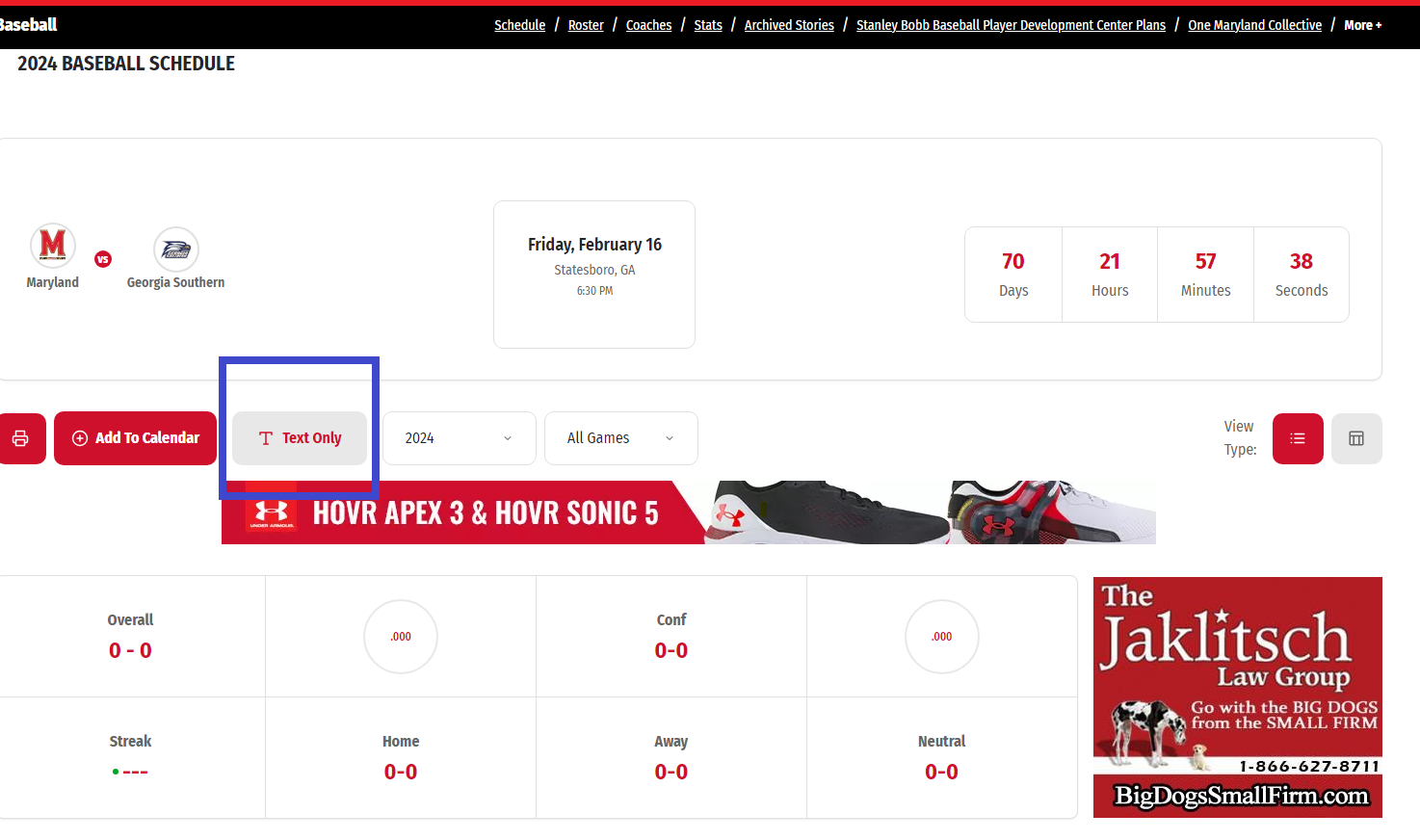
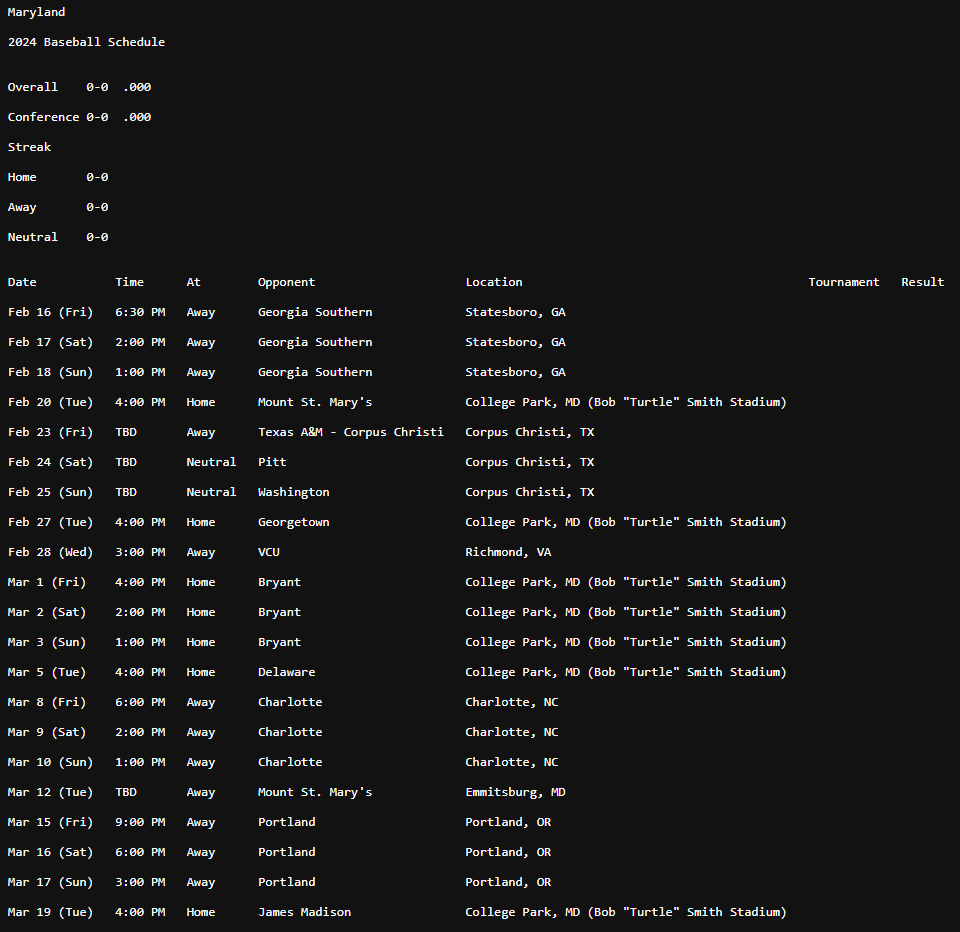
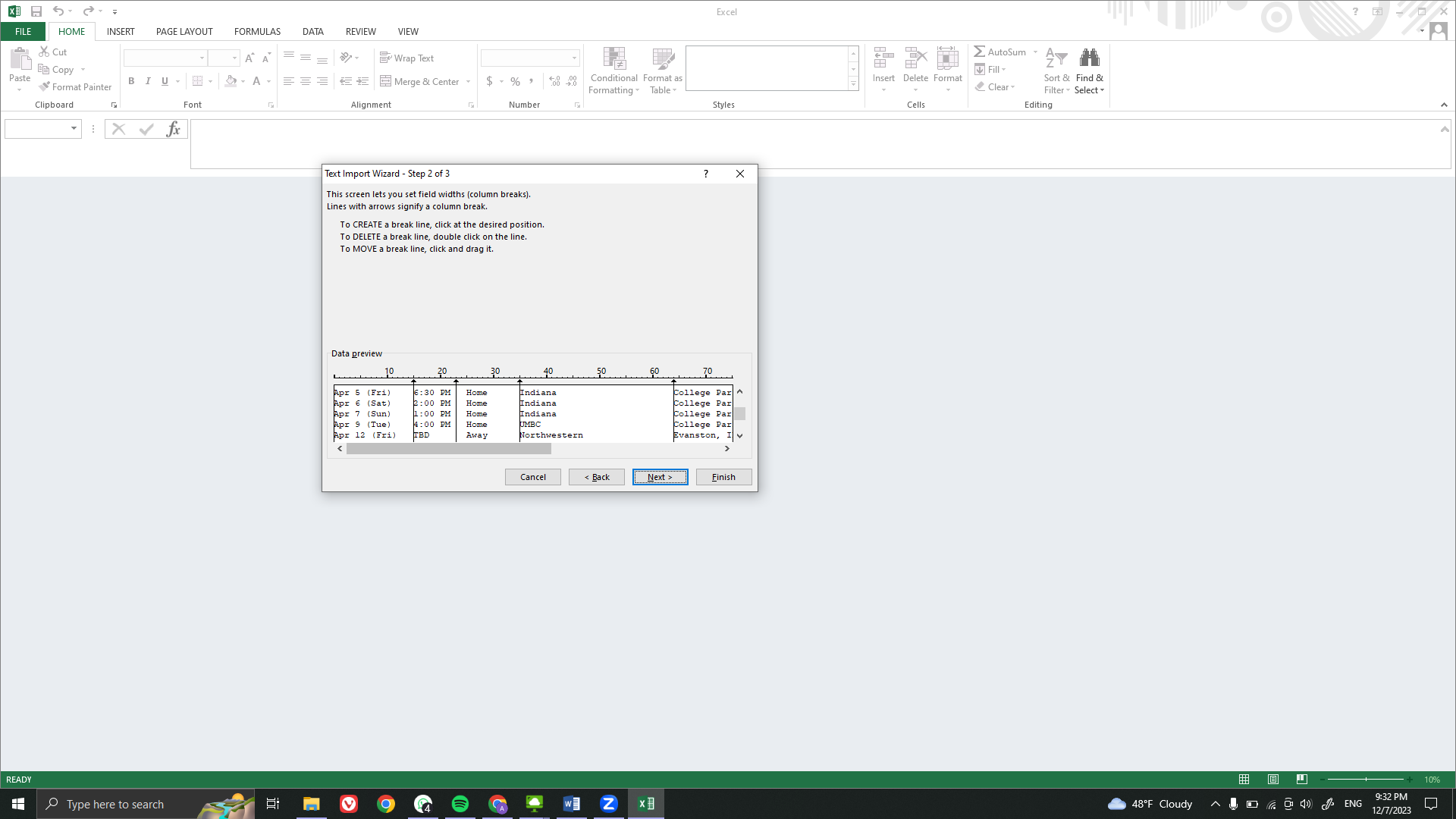
This readme file outlines every procedure our team performed for our project. Screenshots are included for ease of testing and visual documentation.

**COLLECTING DATA FROM UMTERPS TO EXCEL**

* Viewed data from UMTerps using the “Text-Only” format and downloaded a .txt file

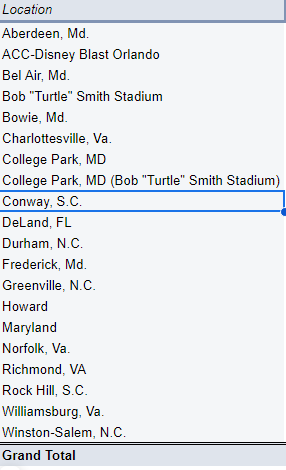




* Directly imported the text file into Excel, specifying the column divisions in the data:  
  

**CLEANING DATA WITHIN EXCEL**

* Used pivot tables in Excel for each column to check for duplicate, missing data, and erroneous data formats:

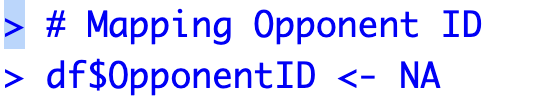


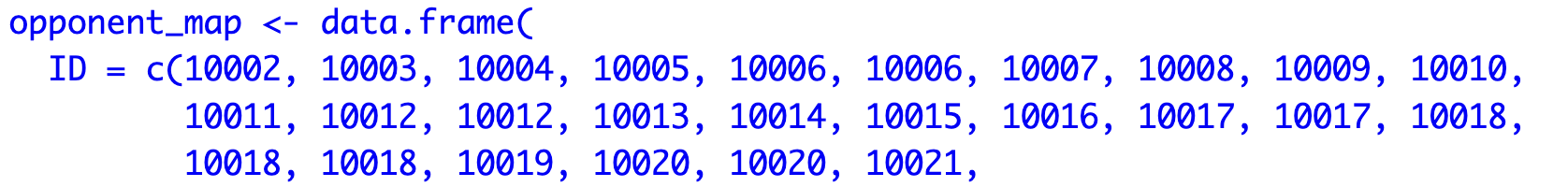
* Standardized any duplicates in the pivot tables and deleted any rows with missing data.
* For erroneous data formats, the format was changed. For example, “Noon” was an entry for time and this was changed to the standard format presented in the data.

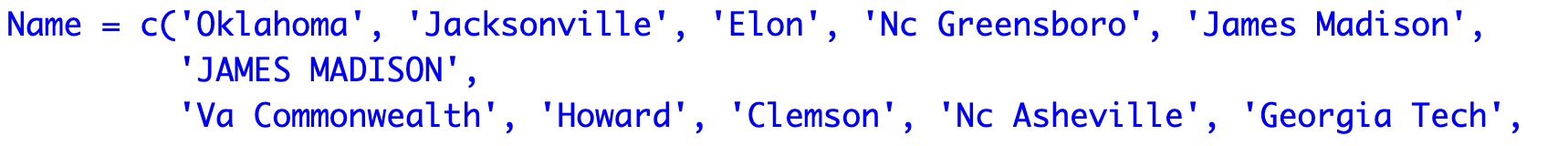
**CREATING INSERTION STATEMENTS FOR TEAM, OPPONENT, CONFERENCE, YEAR AND LOCATION TABLES**

* We created the insert statements for these tables manually.
* For the team table, teamId was typed using a 5 character length ID 10001. The team name, “Maryland”, was keyed in for this field manually.
* For the opponent table, opntId was typed using a 5 character length ID starting from 10002, limited to IDs starting with ‘1’. The opponent names were keyed in for this field manually.
* For the conference table, confId was typed using a 5 character length ID starting with the digit 2. The conference names were keyed in for this field manually.
* For the year table, year was entered using the INT data type for all 25 years starting from 1999 to 2023.
* For the location table, locnId was typed using a 5 character length ID starting with the digit 3. The location names were keyed in for this field manually.

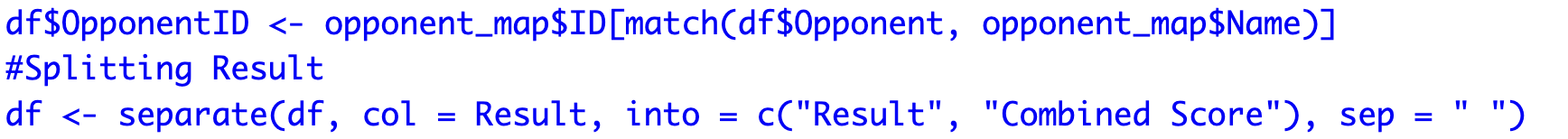
**ADDING EXTRA COLUMNS TO THE DATA**

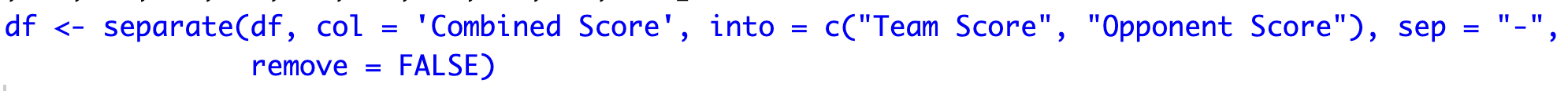
* Creating the insertion data for the play and perform tables is a massive challenge: there are too many entries for us to input manually. We decided to accomplish this task via a python script
* We had to first create data we can work with in python. As such, we had to add opponent ID, location ID, and conference ID columns to our data, convert the dates within our data into SQL format, as well as split the scored attribute into Team Score and Opponent Score
* Adding corresponding opponent IDs to each row of our data was accomplished using a mapping approach in R, and the new score attributes were created using the separate() function of dplyr. The R script is attached as part of our submission as ‘Main Data Cleaning.R’
* Below is a snippet of how the R script works for mapping (with screenshots):
* Creating the Opponent ID column:  
   
* Creating the mapping dataframe



****

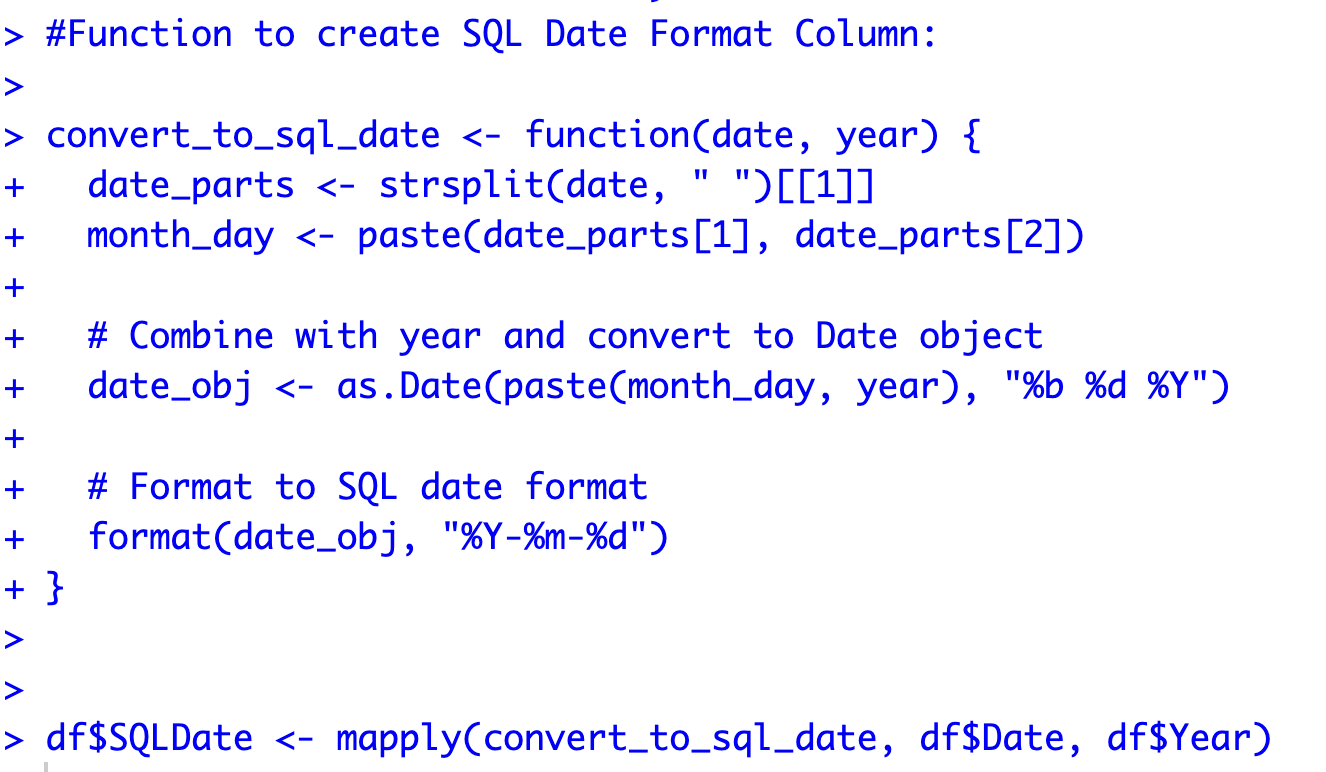
* Performing the mapping operation:  
  
* Creating new score columns:





* Writing and applying a SQL format conversion function for the date:

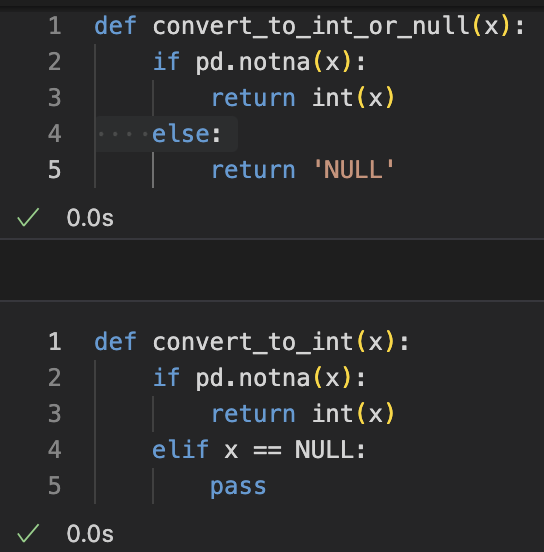
SHOWN BELOW



**CREATING INSERTION STATEMENTS FOR PLAY AND PERFORM**

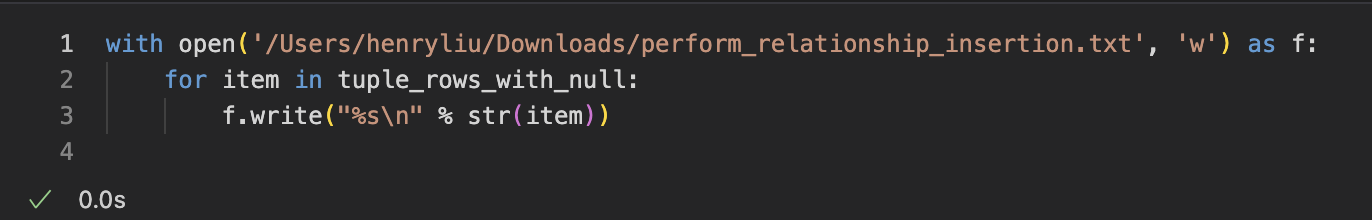
Now, we have the required data to create the insertion statements for play and perform. This was done in pandas, and the jupyter notebook is included as part of the submission:

* We first have to create functions to convert the ID columns, which are currently strings, to integers for a successful insertion. At the same time, NA values have to be changed to NULL

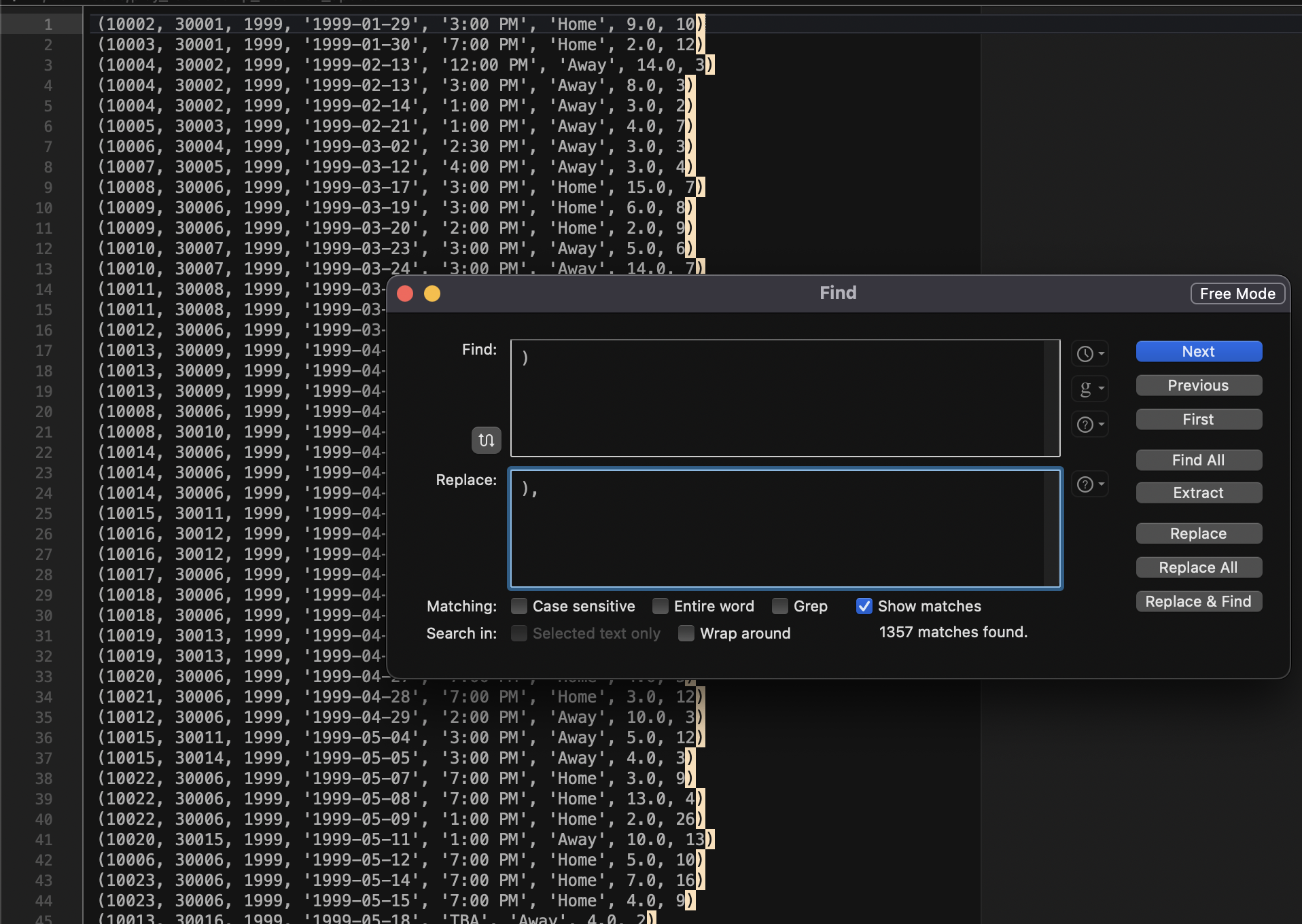


Next, we applied the conversion function to the ID columns and used a lambda function to create our rows of tuples required for the insertion statement:



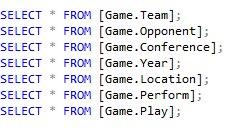
We then wrote the assigned output into a text file for insertion:  


Unfortunately, the text file lacks a comma at the end of each tuple row, this was implemented using bbedit for a quick fix:

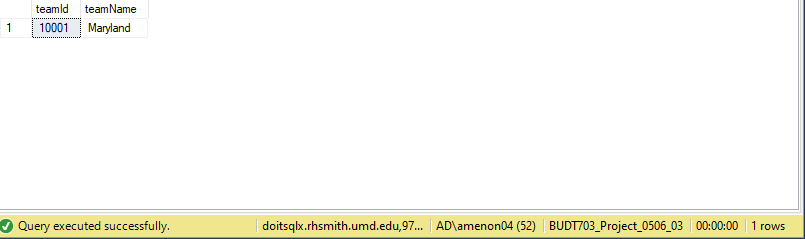


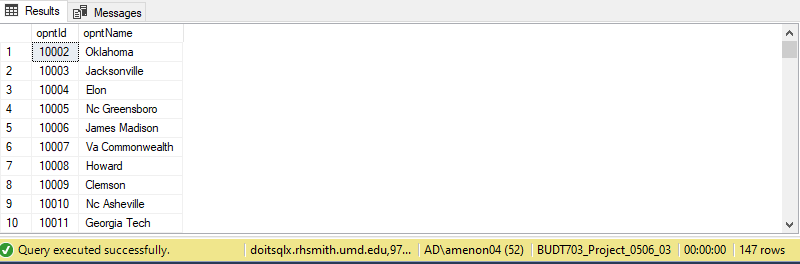
**TESTING THE PROJECT SQL TABLES**

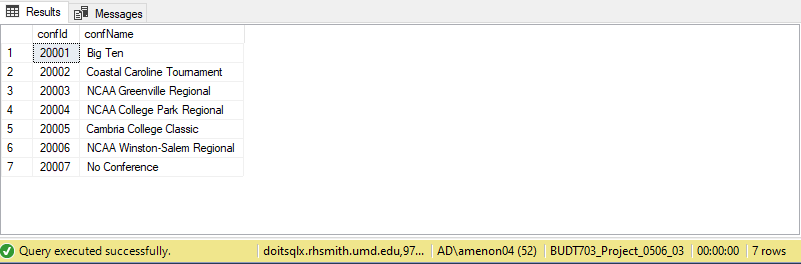
* We can test the individual tables generated in the database using the following SQL SELECT statements:

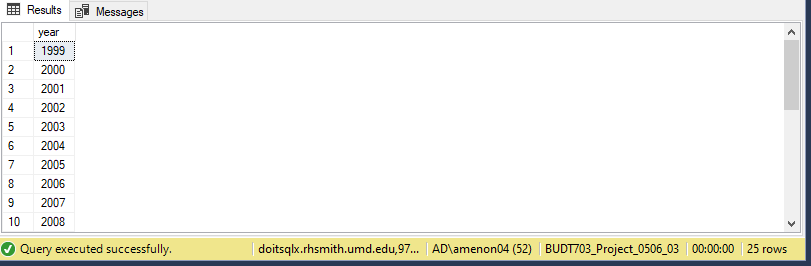


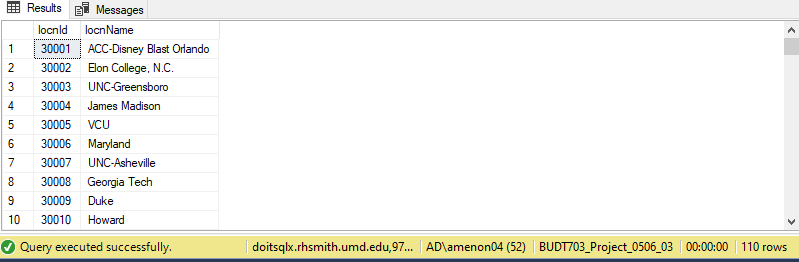
* The outputs for the SELECT statements are as follows, respectively:

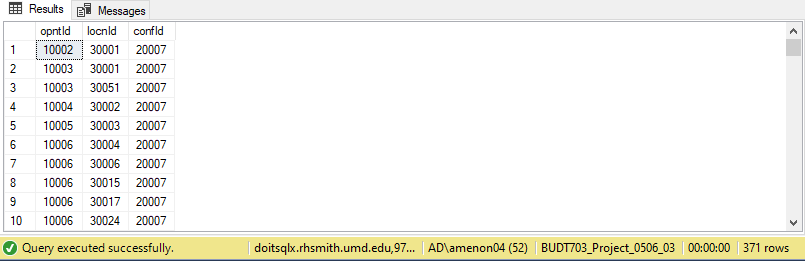


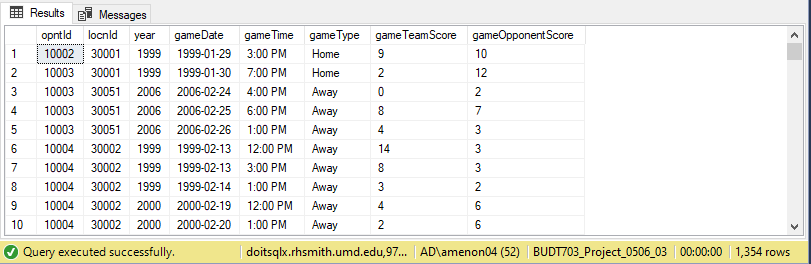


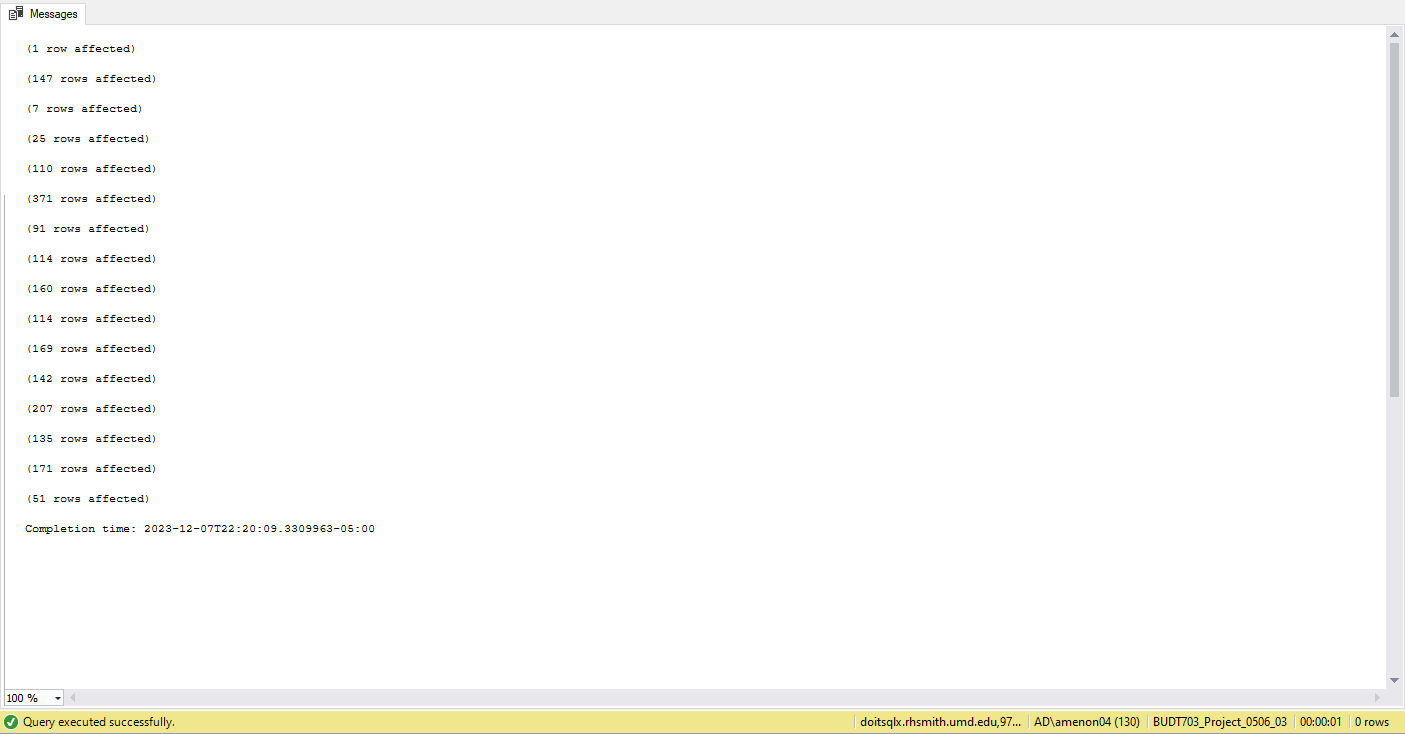






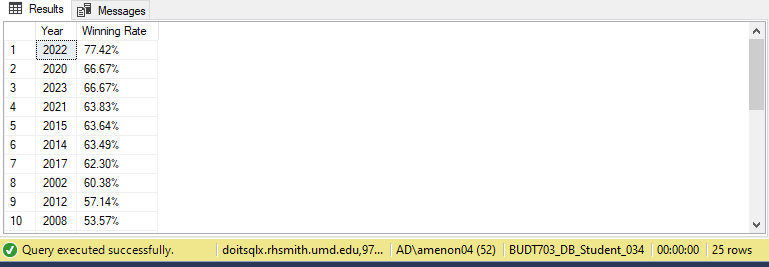




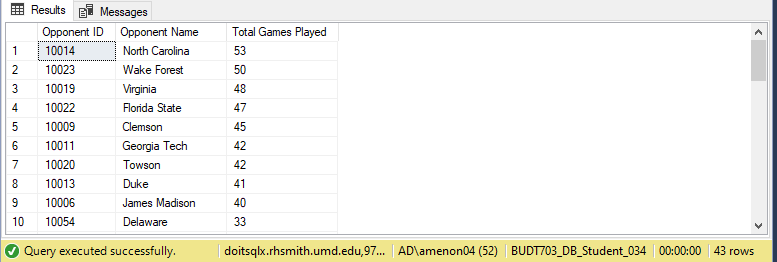
* The output for the DDL is as follows, which can be obtained by running the Project\_0506\_03\_purpose.sql file without the DML code:  
  

**TESTING THE PROJECT SQL DML QUERIES**

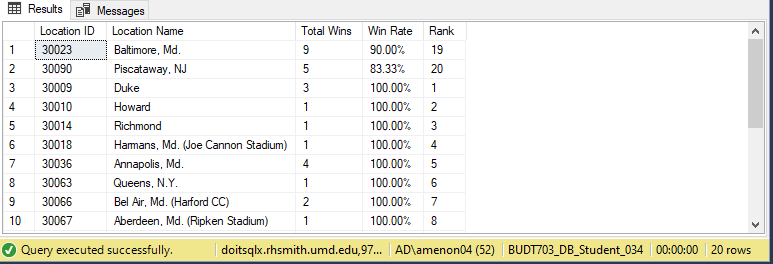
* For the DML queries, the outputs are as follows:
* For business transaction warehouse question 1:



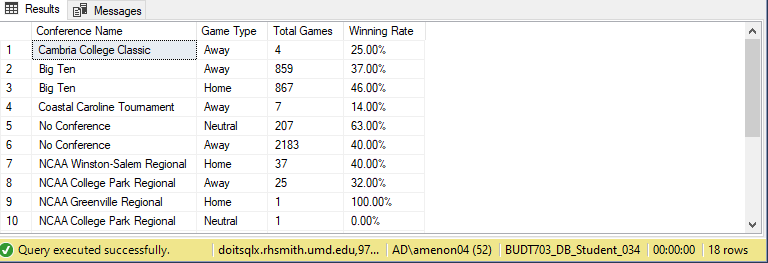
* For business transaction warehouse question 2:



* For business transaction warehouse question 3:

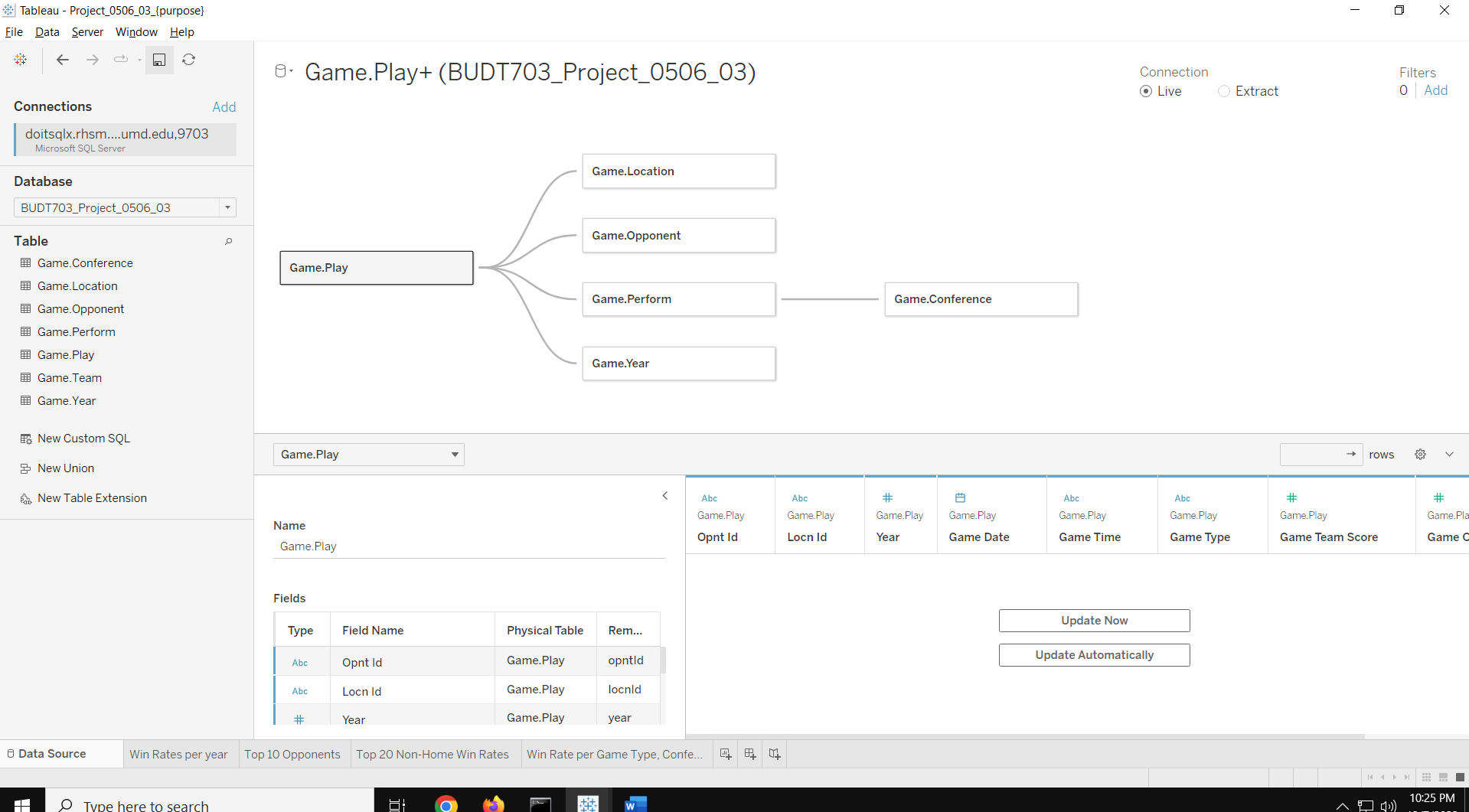


* For business transaction warehouse question 4:

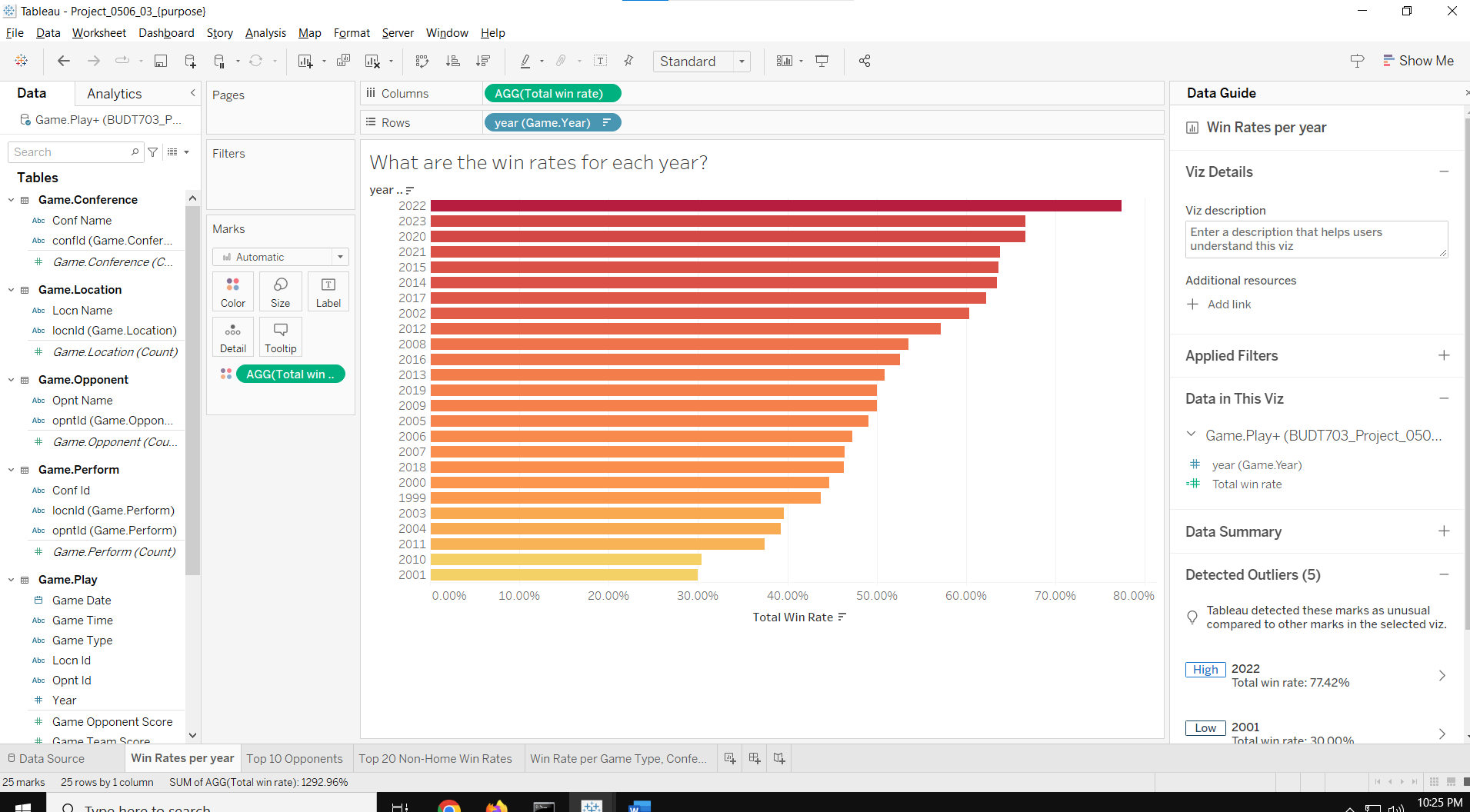


**TESTING THE TABLEAU OUTPUTS:**

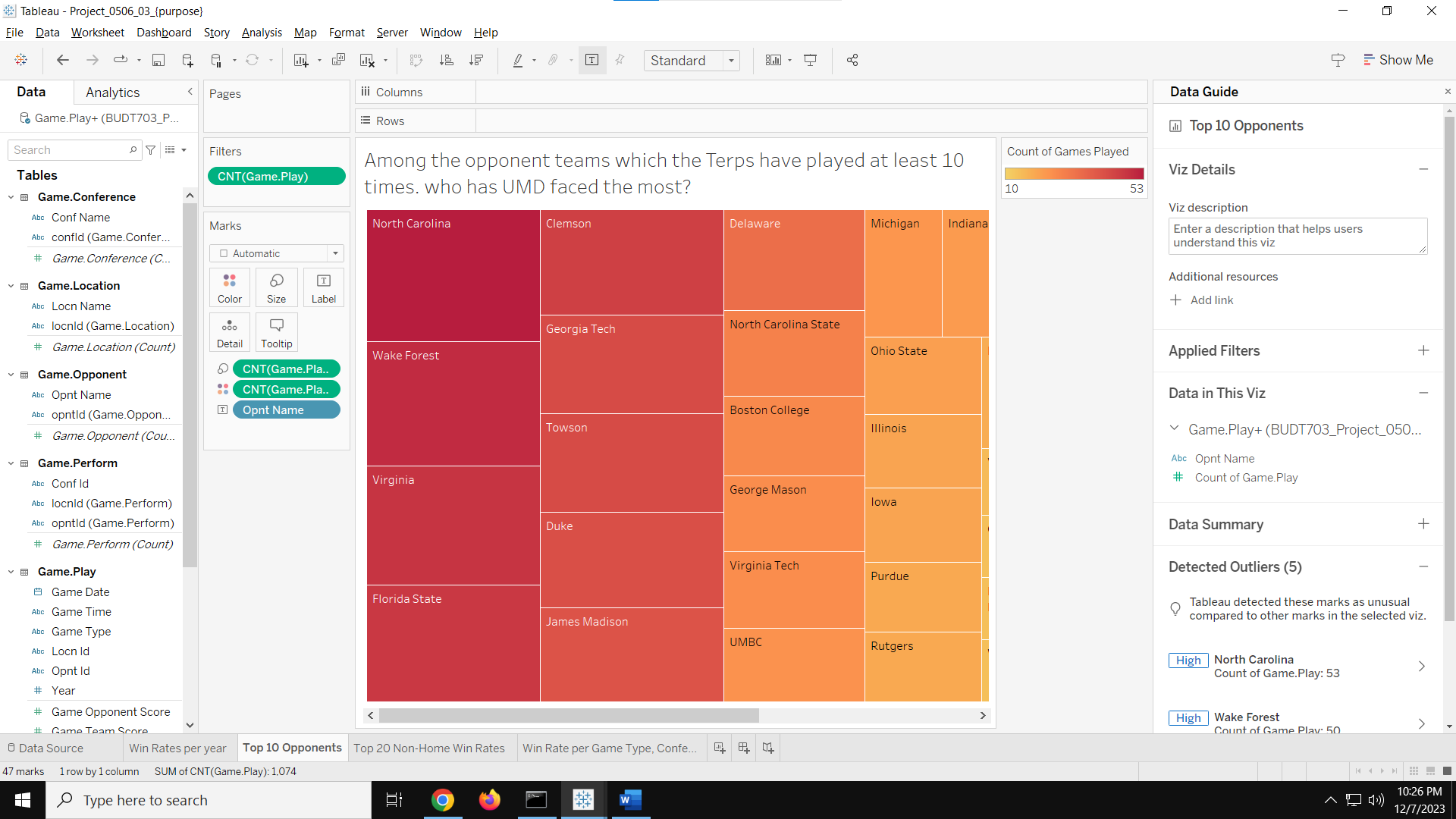
* For the tableau visualizations, the data source is as follows, using the data from the project database BUDT703\_Project\_0506\_03:



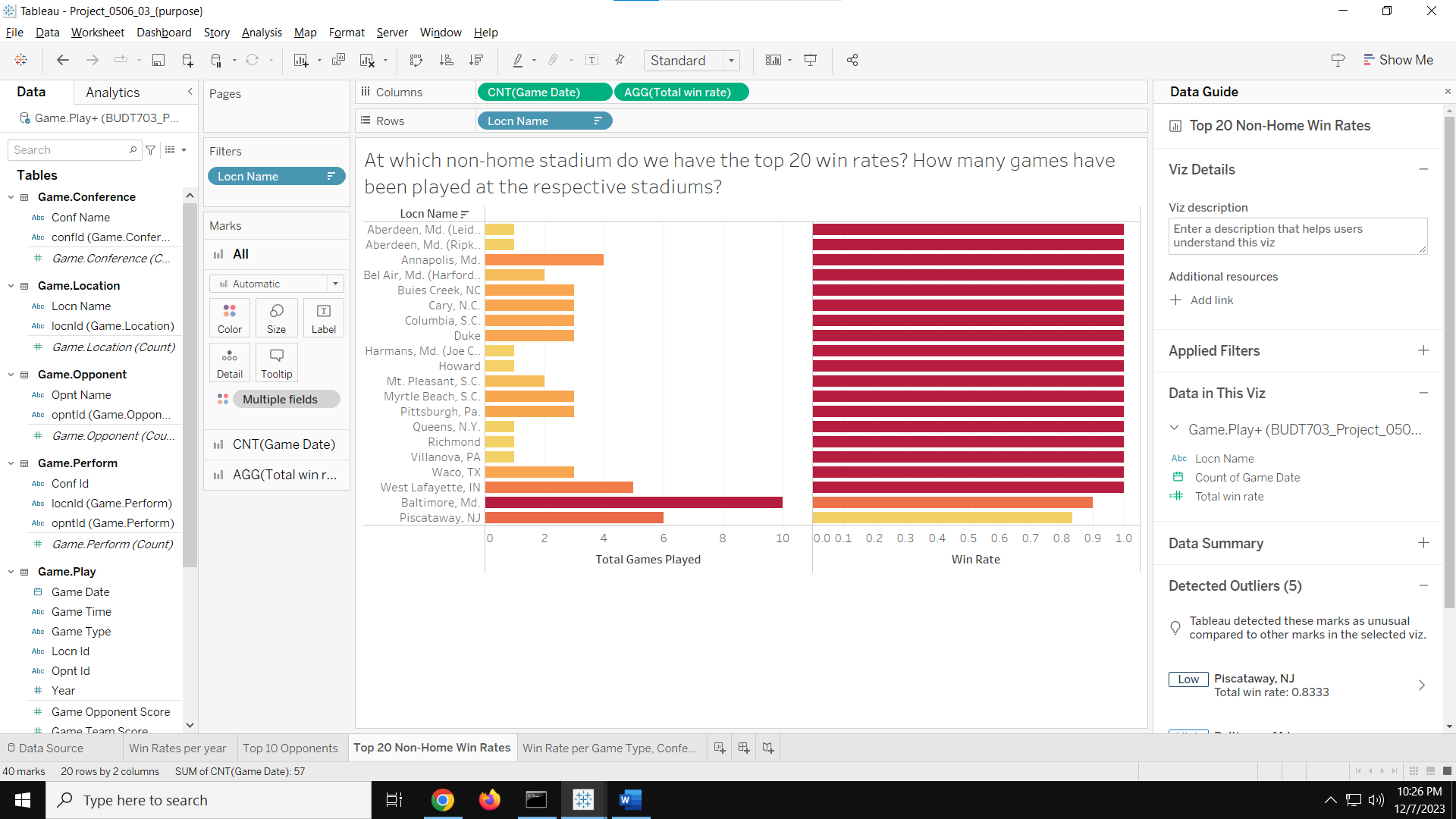
* For business transaction warehouse question 1, the tableau output is as follows:



* For business transaction warehouse question 2, the tableau output is as follows:



* For business transaction warehouse question 3, the tableau output is as follows:



* For business transaction warehouse question 4, the tableau output is as follows:

