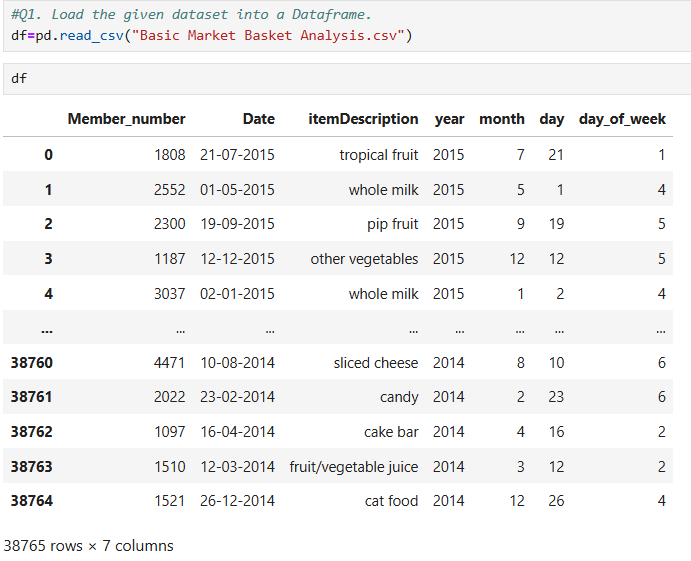
**MiniProject**

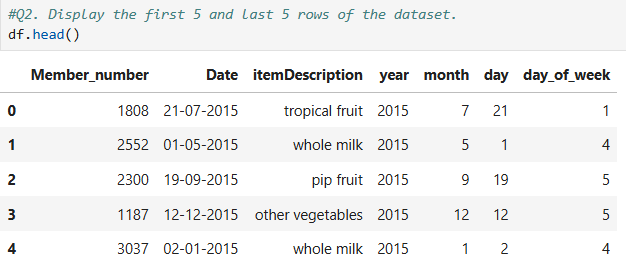
**(Basic Market Basket Analysis – By Divya Karotra TYBCA-A)**

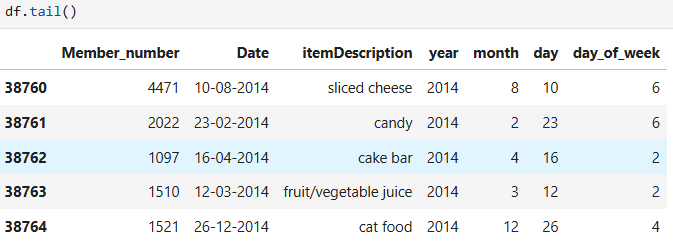
1. **Data Exploration**

**Q1. Load the given dataset into a Dataframe.**

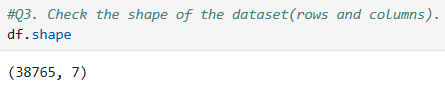
****

**Q2. Display the first 5 and last 5 rows of the dataset.**

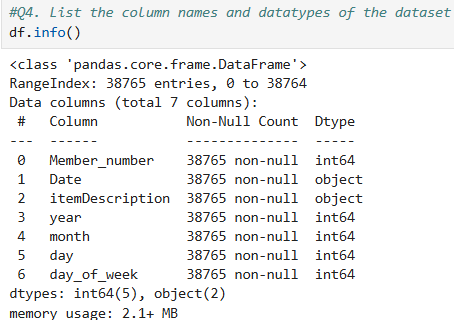
****

****

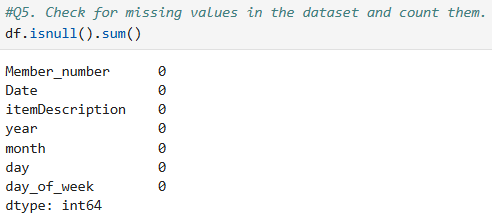
**Q3. Check the shape of the dataset(rows and columns).**

****

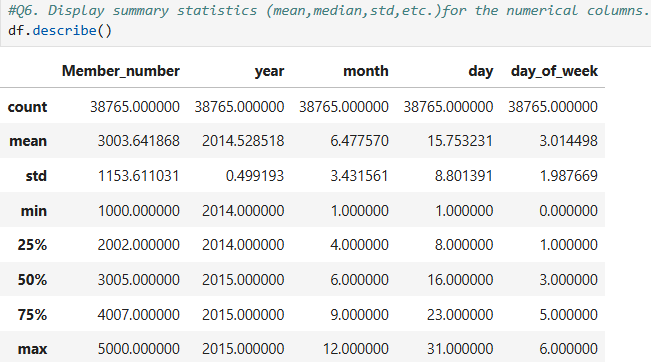
**Q4. List the column names and datatypes of the dataset**

****

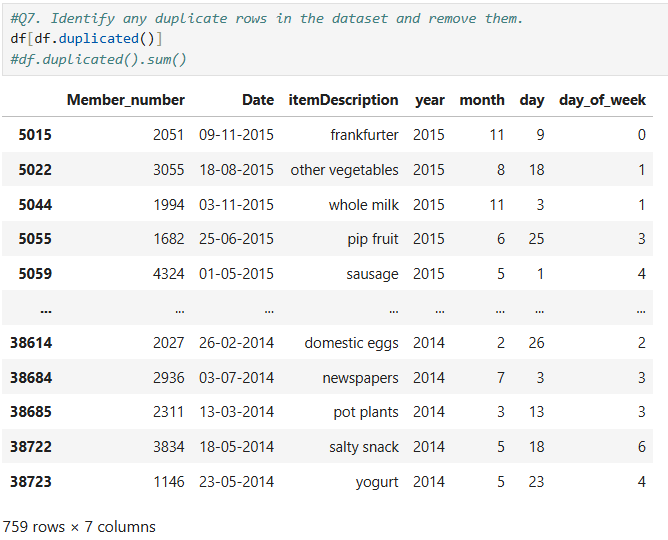
**Q5. Check for missing values in the dataset and count them.**

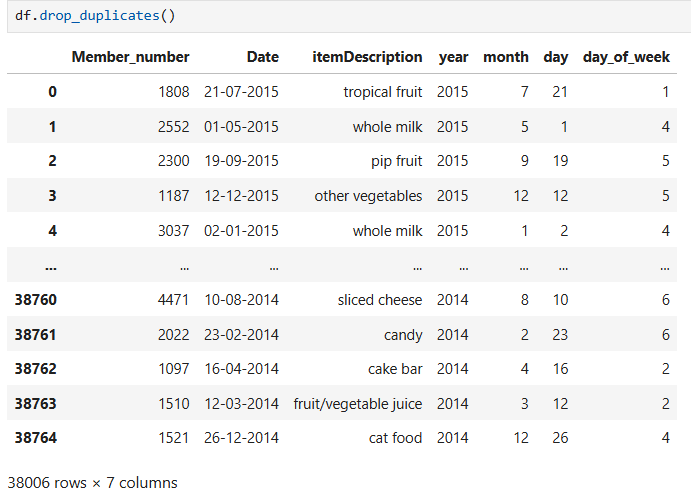
****

**Q6. Display summary statistics (mean,median,std,etc.)for the numerical columns.**

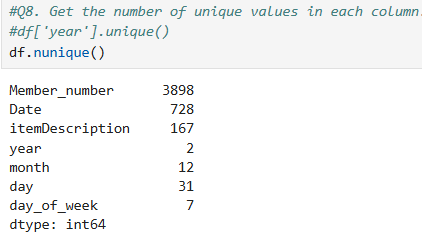
****

**Q7. Identify any duplicate rows in the dataset and remove them.**

****

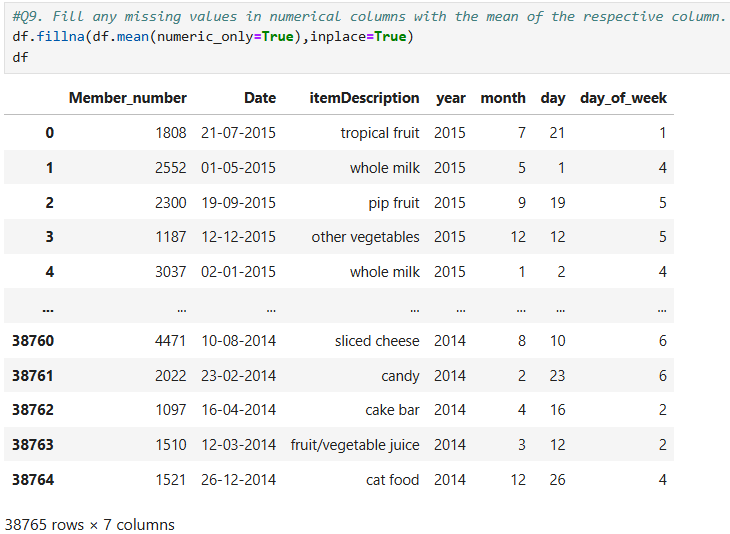
****

**Q8. Get the number of unique values in each column.**

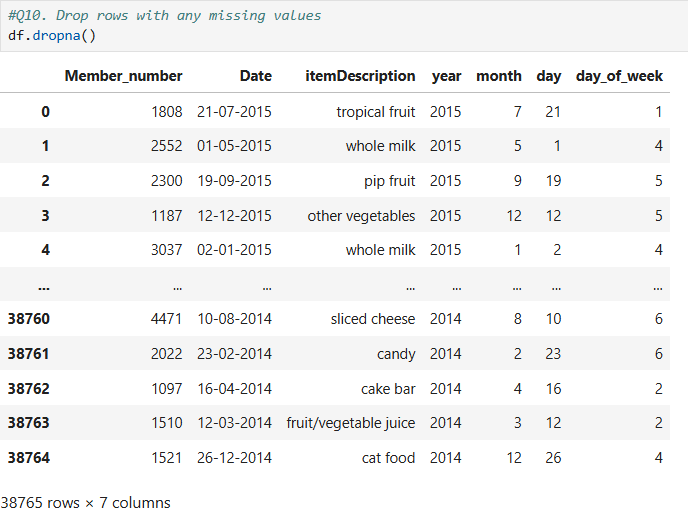
****

1. **Data Cleaning**

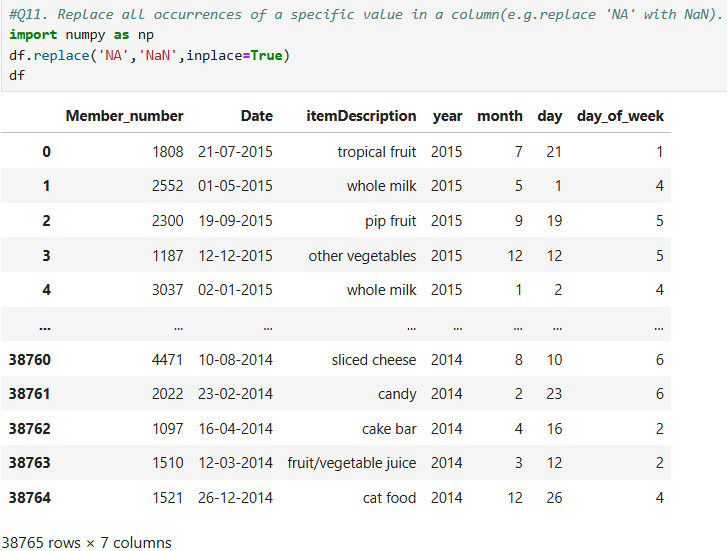
**Q9. Fill any missing values in numerical columns with the mean of the respective column.**

****

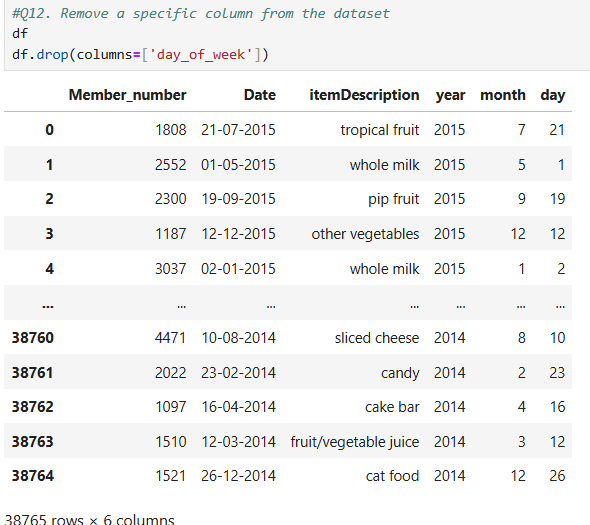
**Q10. Drop rows with any missing values**

****

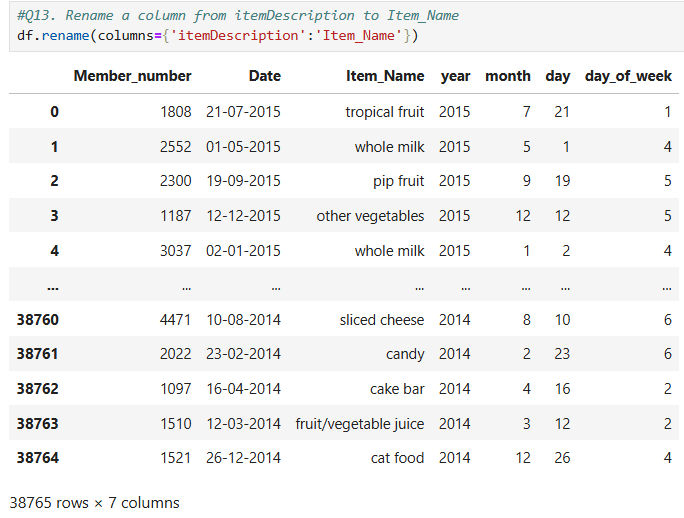
**Q11. Replace all occurrences of a specific value in a column(e.g.replace 'NA' with NaN).**

****

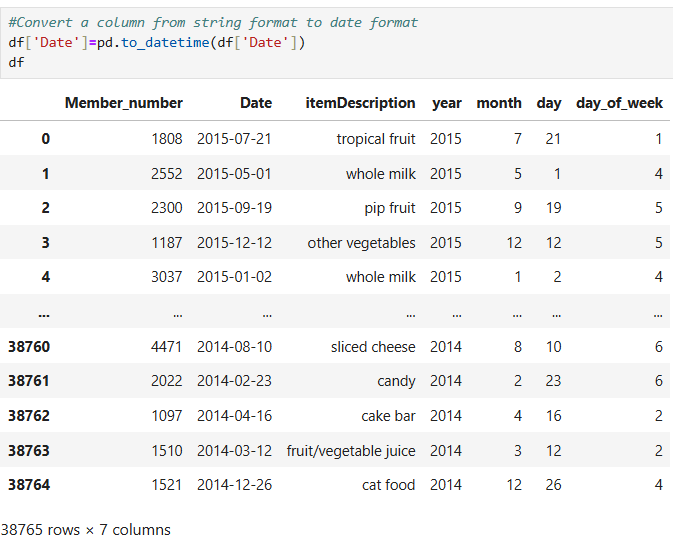
**Q12. Remove a specific column from the dataset**

****

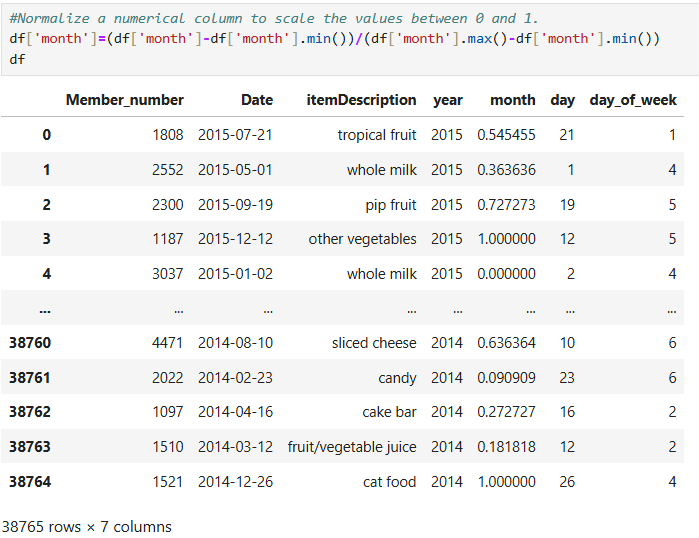
**Q13. Rename a column from itemDescription to Item\_Name**

****

**Q14. Convert a column from string format to date format**

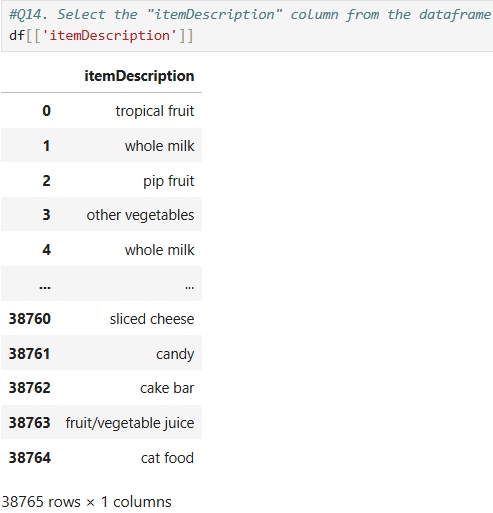
****

**Q15. Normalize a numerical column to scale the values between 0 and 1.**

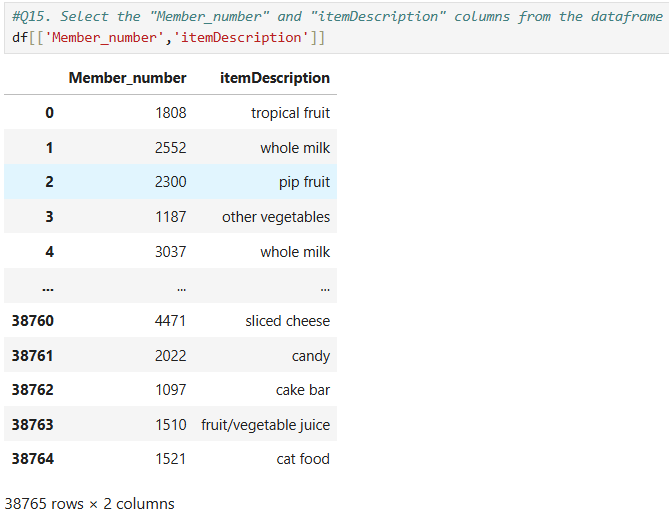
****

1. **Data Selection and Filtering**

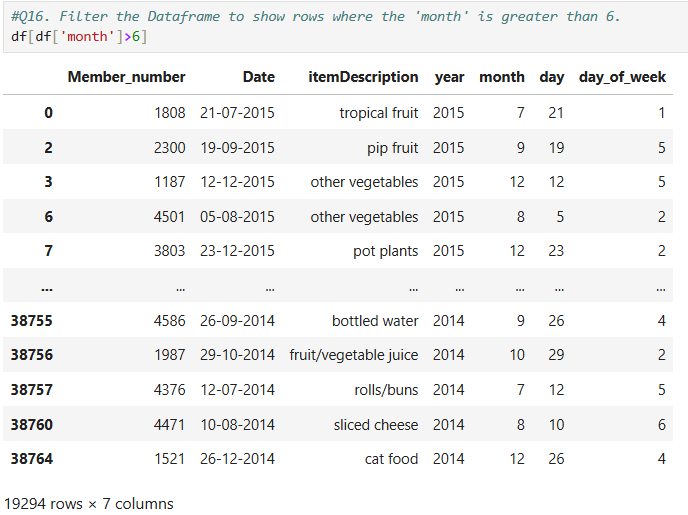
**Q16. Select the "itemDescription" column from the dataframe**

****

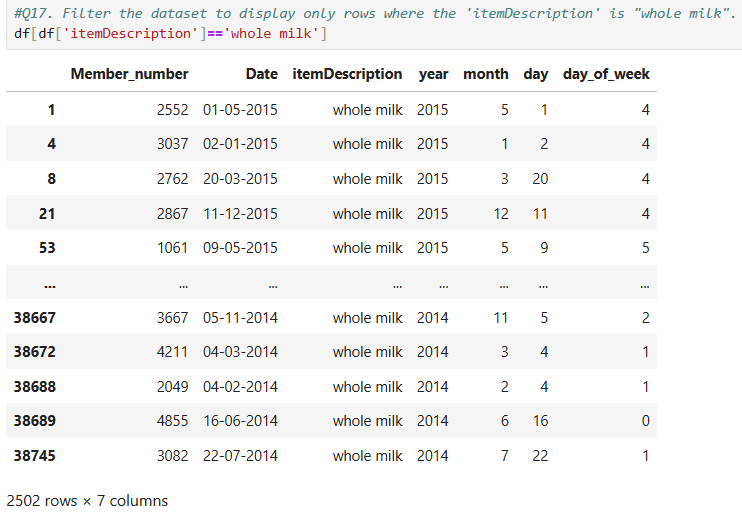
**Q17. Select the "Member\_number" and "itemDescription" columns from the dataframe**

****

**Q18. Filter the Dataframe to show rows where the 'month' is greater than 6.**

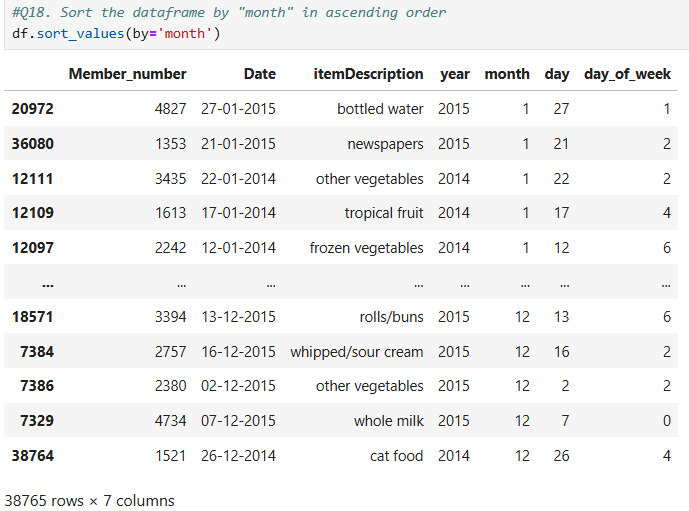
****

**Q19. Filter the dataset to display only rows where the 'itemDescription' is "whole milk".**

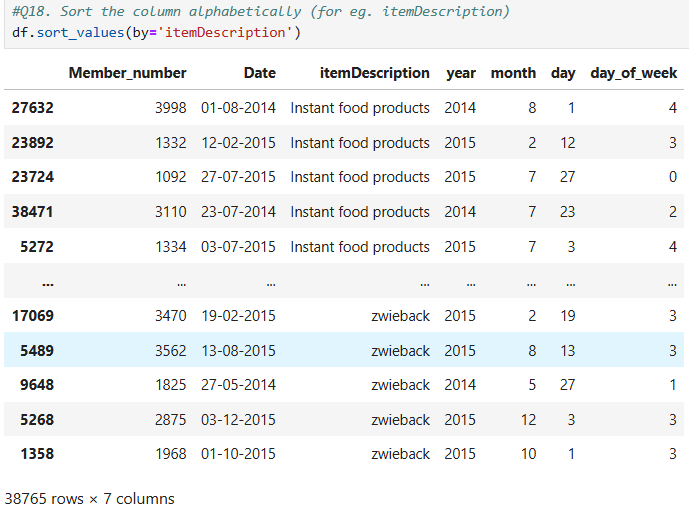
****

1. **Sorting and Ordering**

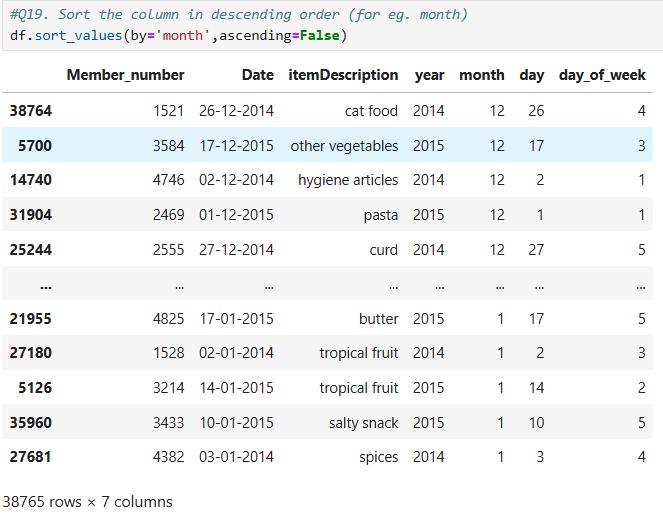
**Q20. Sort the dataframe by "month" in ascending order**

****

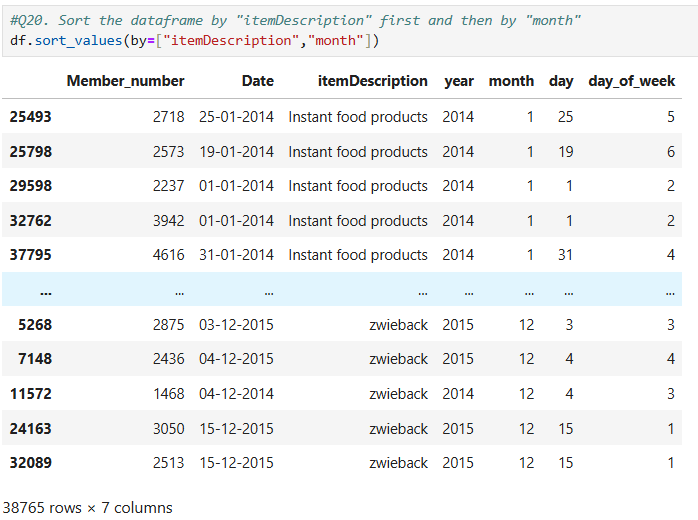
**Q21. Sort the column alphabetically (for eg. itemDescription)**

****

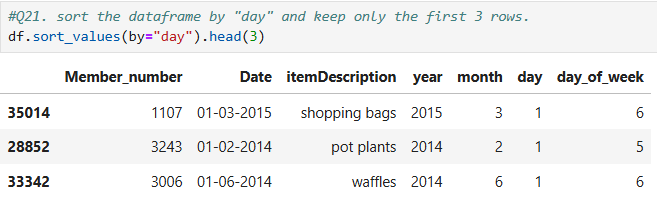
**Q22. Sort the column in descending order (for eg. month)**

****

**Q23. Sort the dataframe by "itemDescription" first and then by "month"**

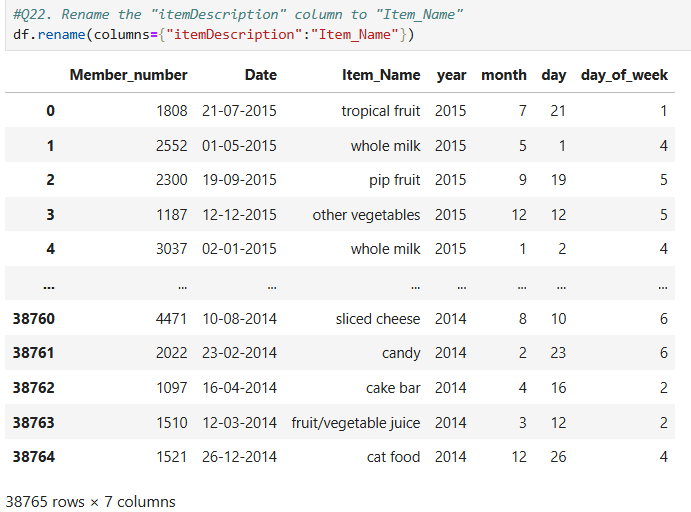
****

**Q24. Sort the dataframe by "day" and keep only the first 3 rows.**

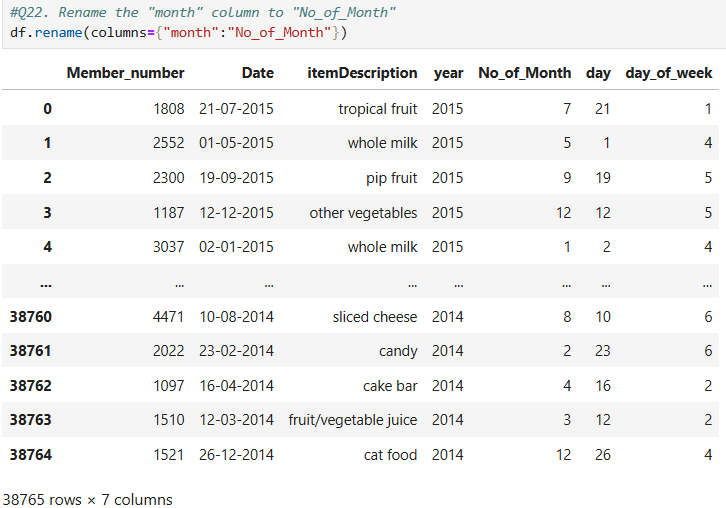
****

1. **Renaming Columns**

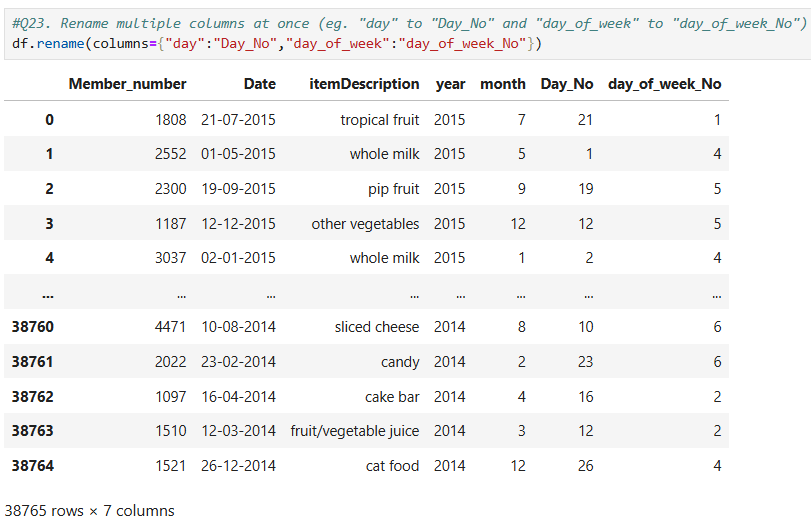
**Q25. Rename the "itemDescription" column to "Item\_Name"**

****

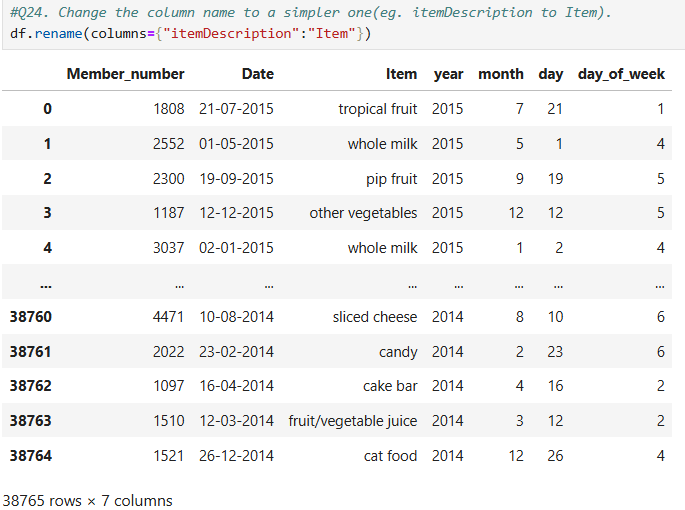
**Q26. Rename the "month" column to "No\_of\_Month"**

****

**Q27. Rename multiple columns at once (eg. "day" to "Day\_No" and "day\_of\_week" to "day\_of\_week\_No")**

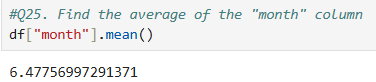
****

**Q28. Change the column name to a simpler one(eg. itemDescription to Item).**

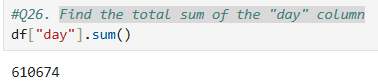
****

1. **Data Aggregation**

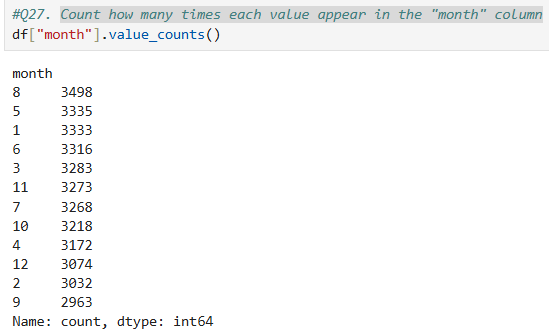
**Q29. Find the average of the "month" column**

****

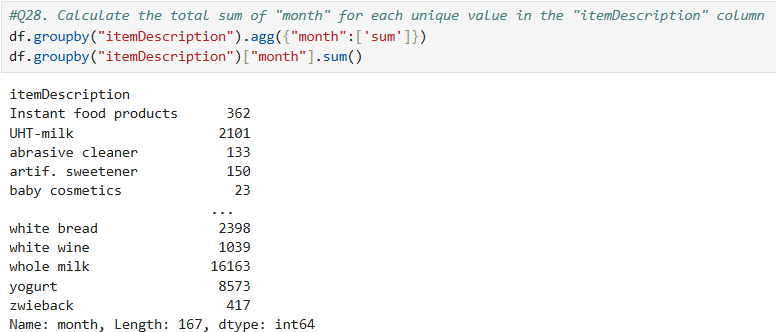
**Q30. Find the total sum of the "day" column**

****

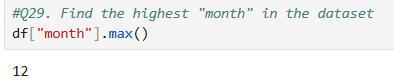
**Q31. Count how many times each value appear in the "month" column**

****

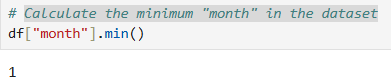
**Q32. Calculate the total sum of "month" for each unique value in the "itemDescription" column**

****

**Q33. Find the highest "month" in the dataset**

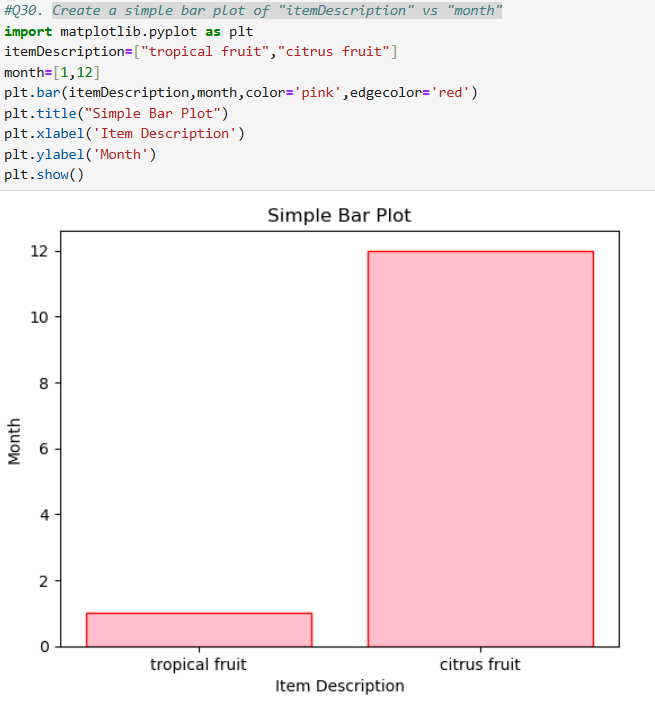
****

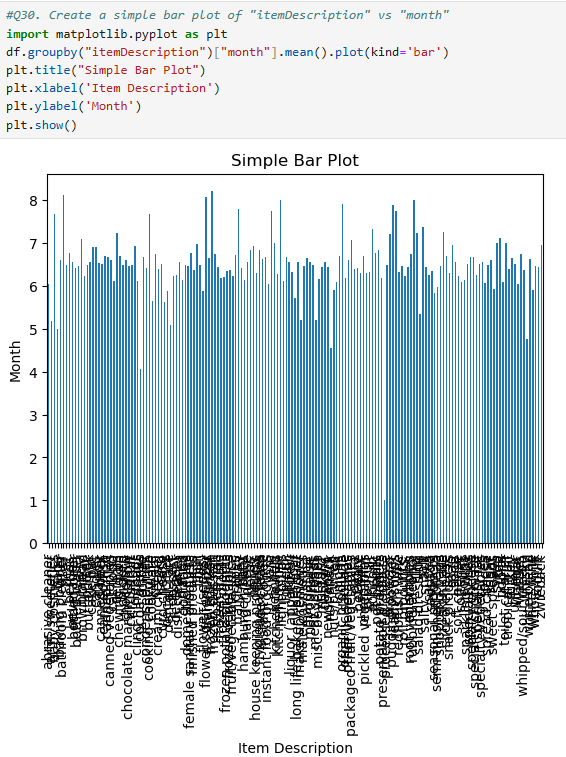
**Q34. Calculate the minimum "month" in the dataset**

****

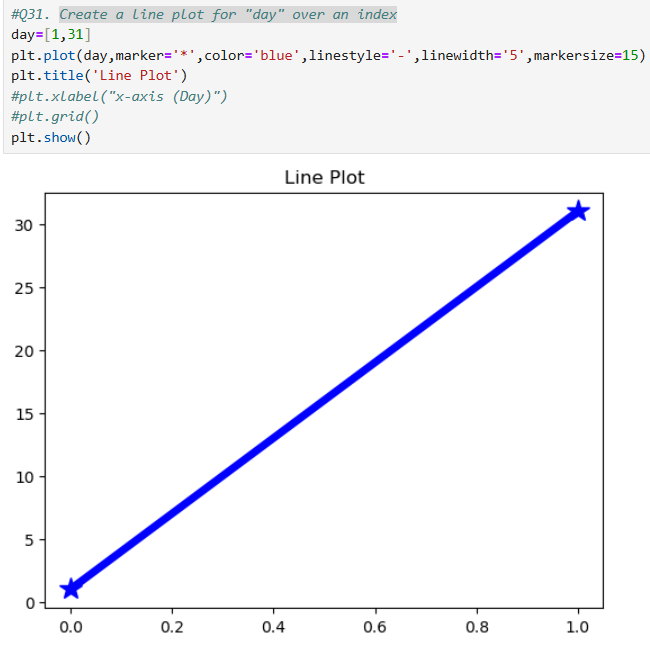
1. **Simple Data Visualization**

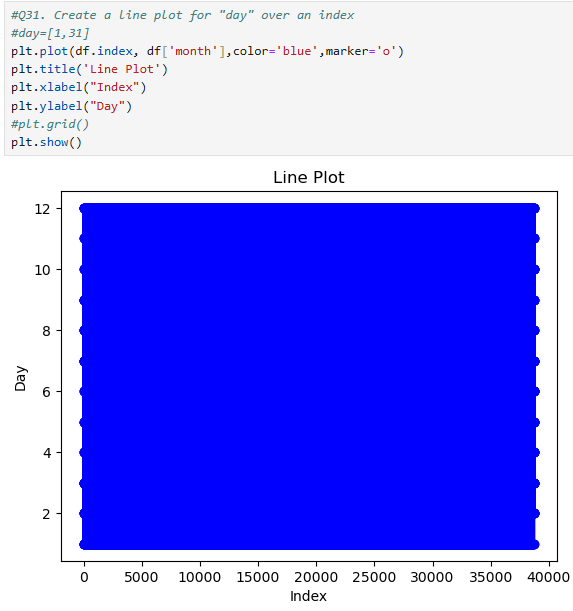
**Q35. Create a simple bar plot of "itemDescription" vs "month"**

****

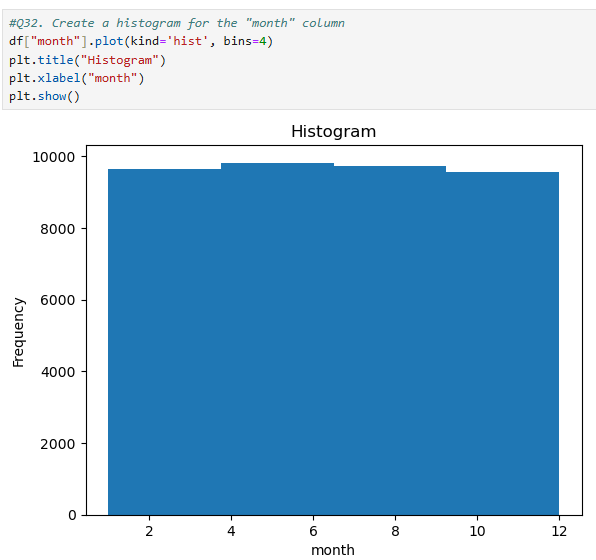
****

**Q36. Create a line plot for "day" over an index**

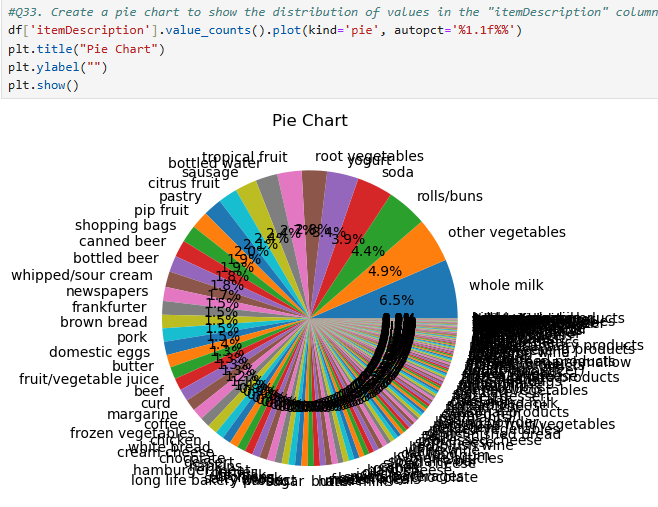
****

****

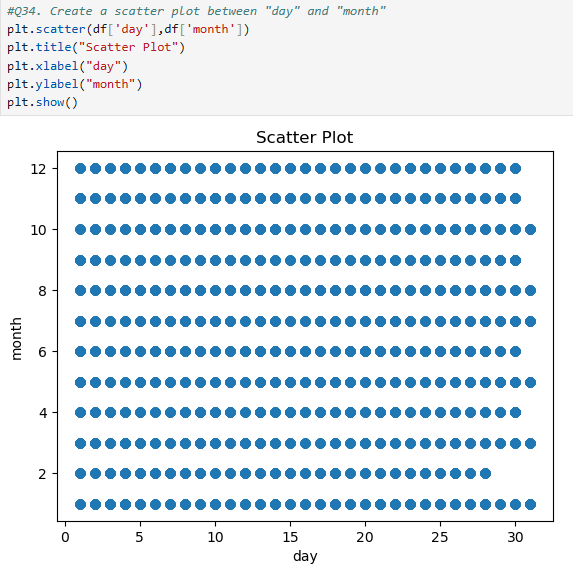
**Q37. Create a histogram for the "month" column**

****

**Q38. Create a pie chart to show the distribution of values in the "itemDescription" column**

****

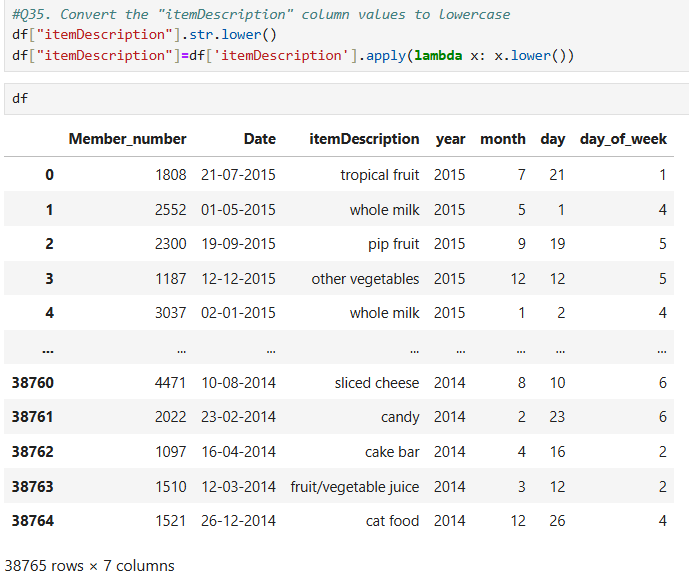
**Q39. Create a scatter plot between "day" and "month"**

****

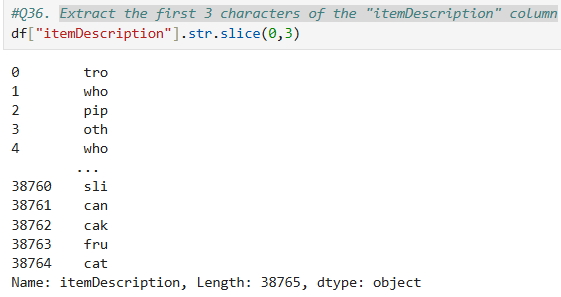
**Q40. Display a box plot for the "month" column**

1. **Basic String Manipulation**

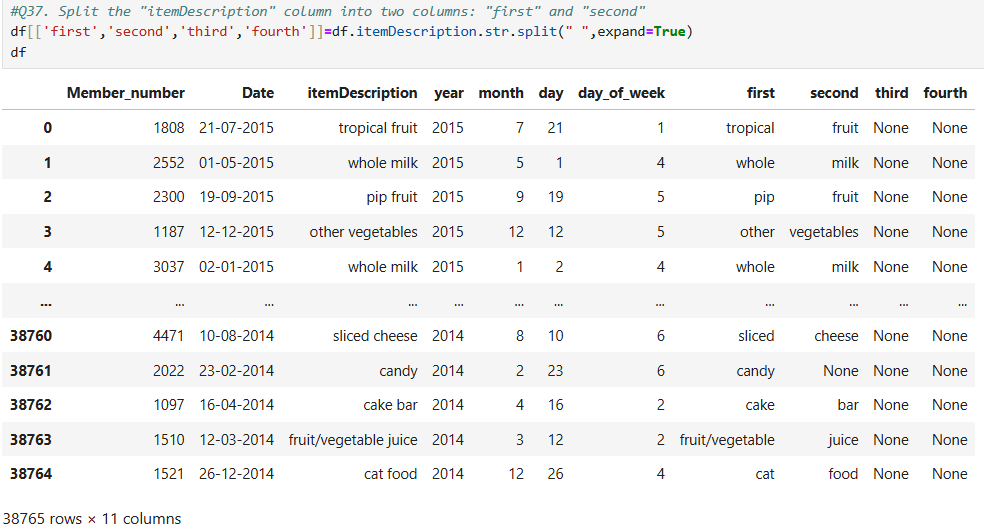
**Q41. Convert the "itemDescription" column values to lowercase**

****

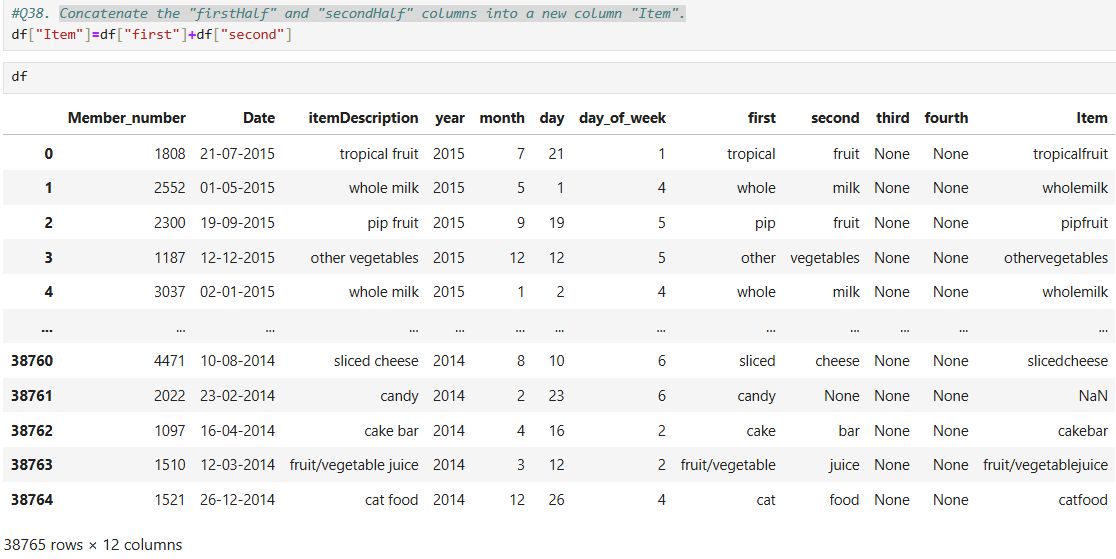
**Q42. Extract the first 3 characters of the "itemDescription" column**

****

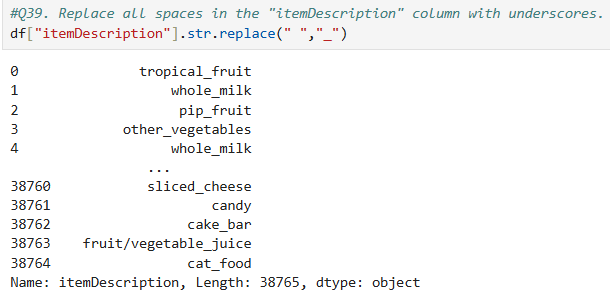
**Q43. Split the "itemDescription" column into two columns: "first" and "second"**

****

**Q44. Concatenate the "firstHalf" and "secondHalf" columns into a new column "Item".**

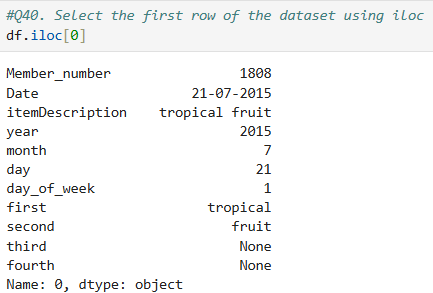
****

**Q45. Replace all spaces in the "itemDescription" column with underscores.**

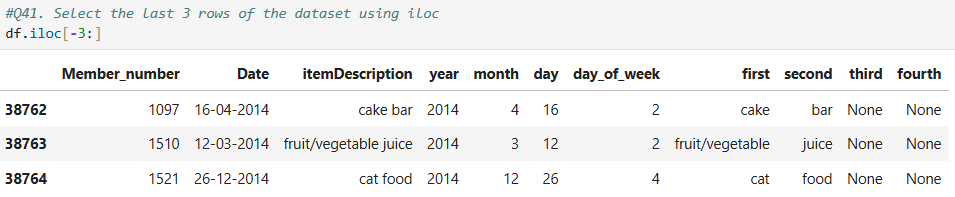
****

1. **iloc – Integer-location based indexing**

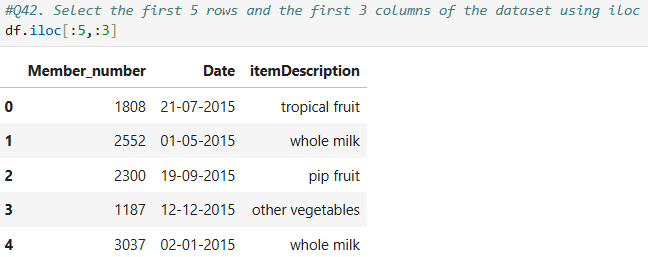
**Q46. Select the first row of the dataset using iloc**

****

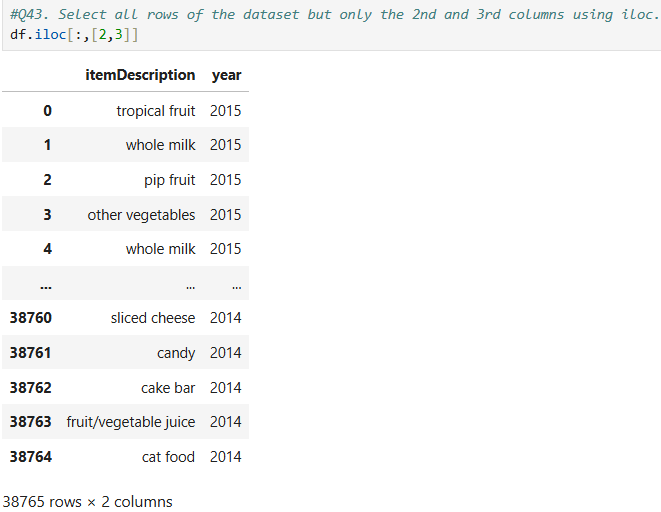
**Q47. Select the last 3 rows of the dataset using iloc**

****

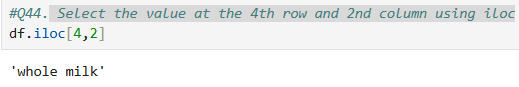
**Q48. Select the first 5 rows and the first 3 columns of the dataset using iloc**

****

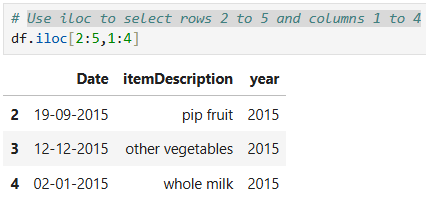
**Q49. Select all rows of the dataset but only the 2nd and 3rd columns using iloc.**

****

**Q50. Select the value at the 4th row and 2nd column using iloc**

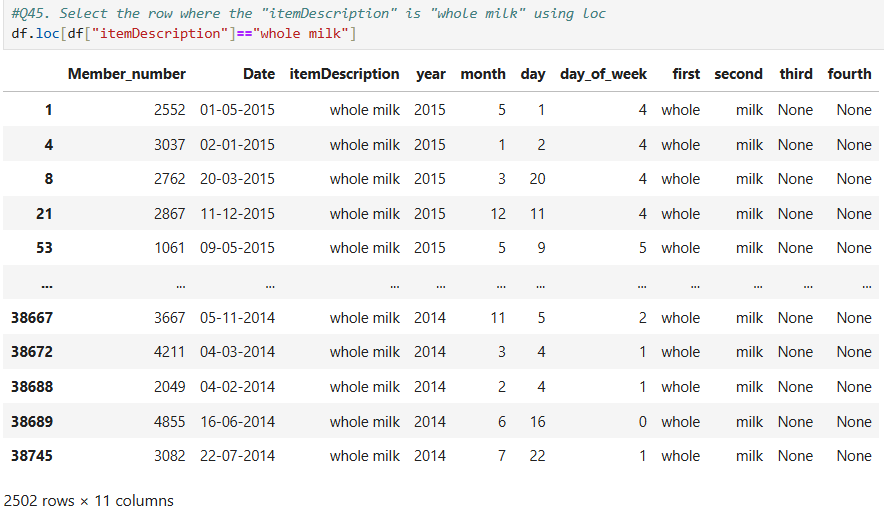
****

**Q51. Use iloc to select rows 2 to 5 and columns 1 to 4**

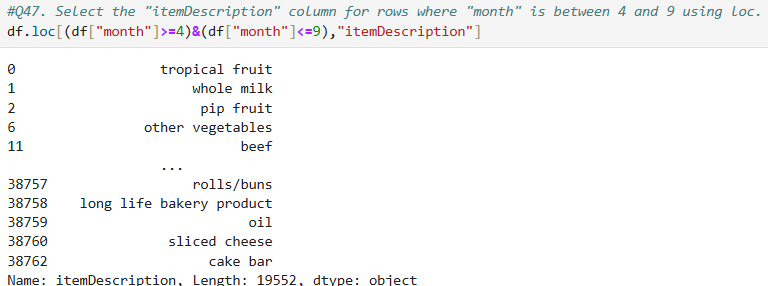
****

1. **loc – Label – based indexing**

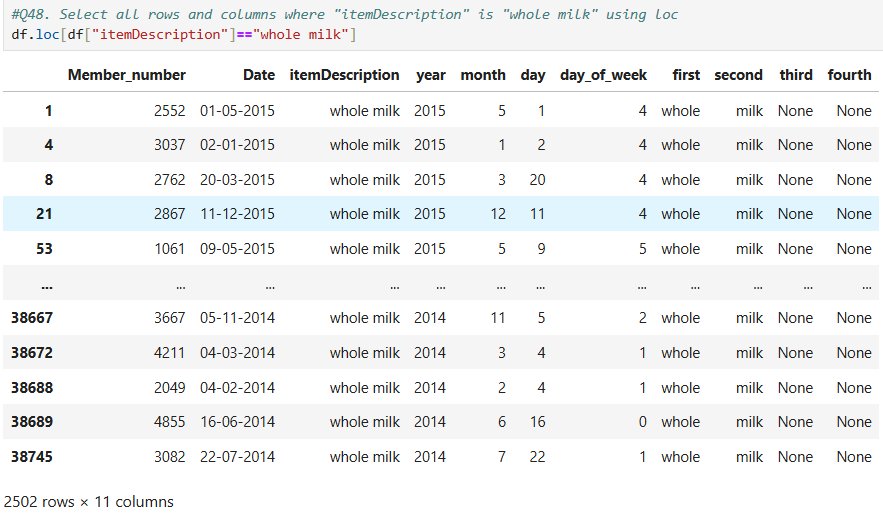
**Q52. Select the row where the "itemDescription" is "whole milk" using loc**

****

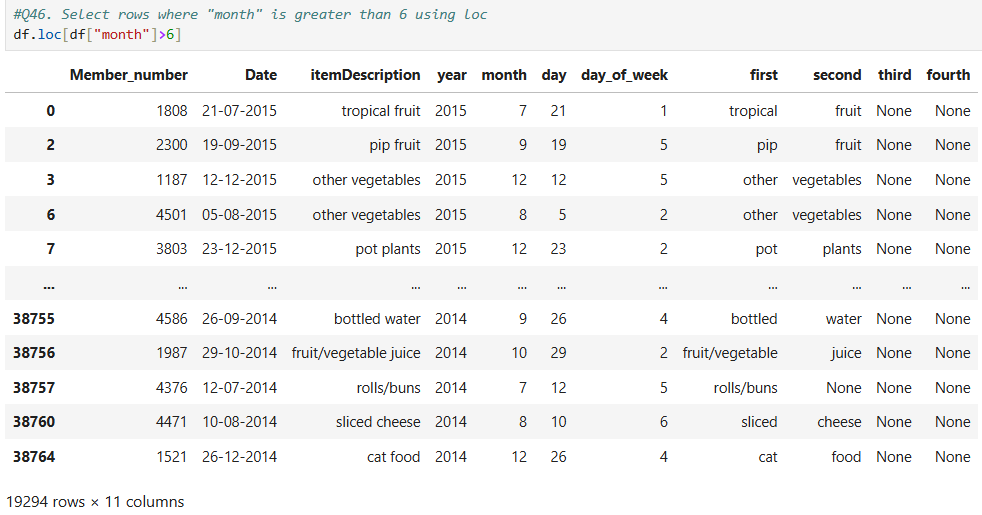
**Q53. Select the "itemDescription" column for rows where "month" is between 4 and 9 using loc.**

****

**Q54.** **Select all rows and columns where "itemDescription" is "whole milk" using loc**

****

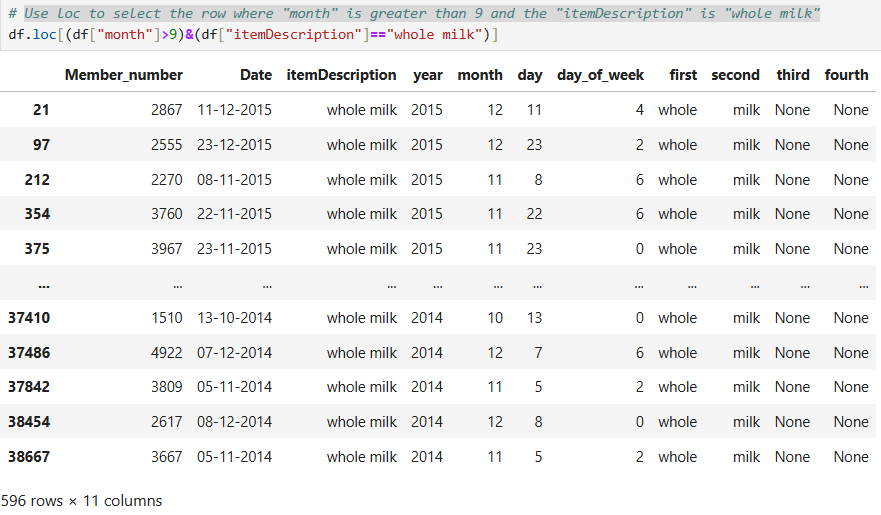
**Q55. Select rows where "month" is greater than 6 using loc**

****

**Q56. Select multiple columns("itemDescription","month") for a specific row with loc.**

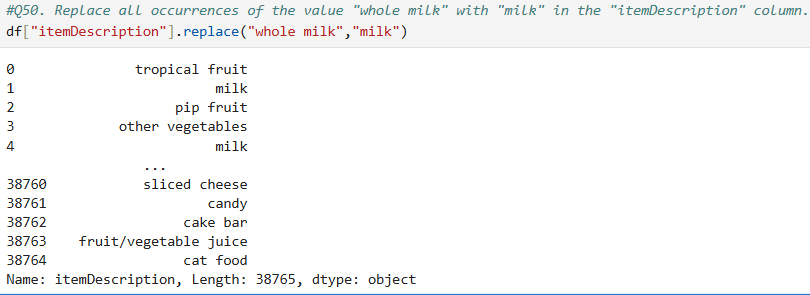
****

**Q57. Use loc to select the row where "month" is greater than 9 and the "itemDescription" is "whole milk"**

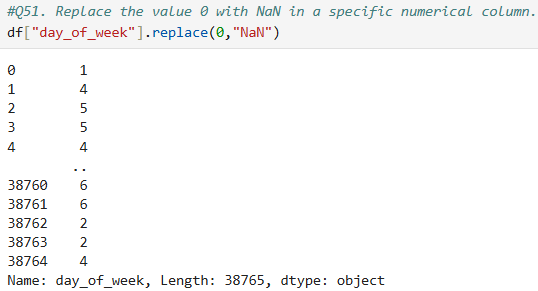
****

1. **Replace - Replacing values in a DataFrame**

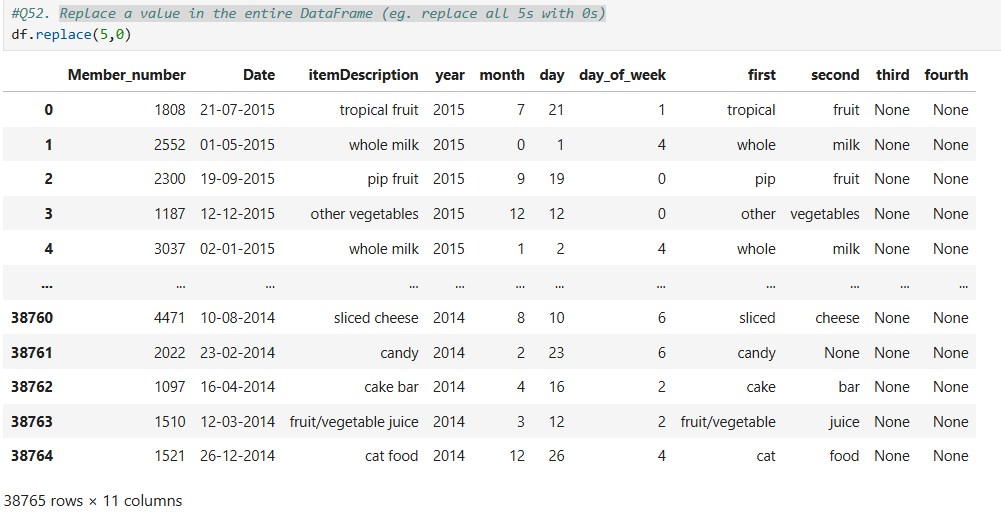
**Q58. Replace all occurrences of the value "whole milk" with "milk" in the "itemDescription" column.**

****

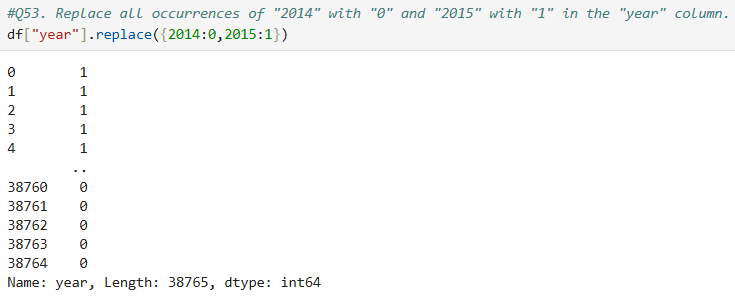
**Q59. Replace the value 0 with NaN in a specific numerical column.**

****

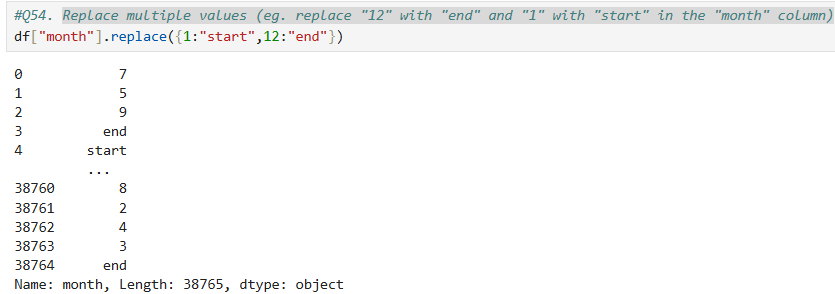
**Q60. Replace a value in the entire DataFrame (eg. replace all 5s with 0s)**

****

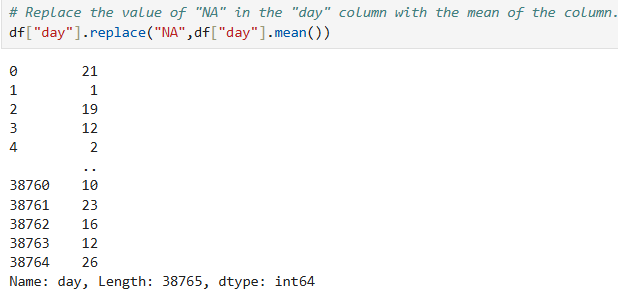
**Q61. Replace all occurrences of "2014" with "0" and "2015" with "1" in the "year" column.**

****

**Q62. Replace multiple values (eg. replace "12" with "end" and "1" with "start" in the "month" column)**

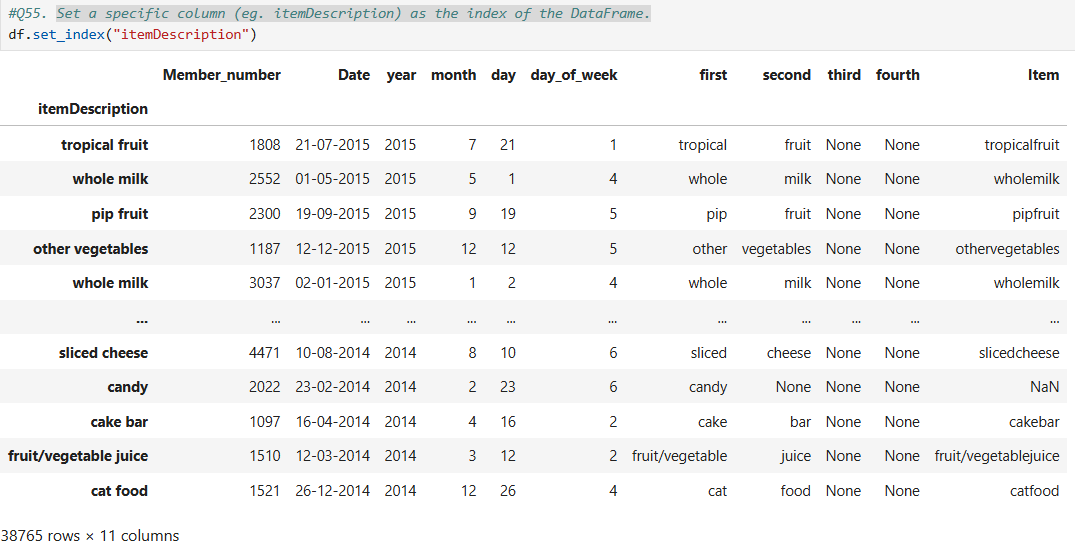
****

**Q63. Replace the value of "NA" in the "day" column with the mean of the column.**

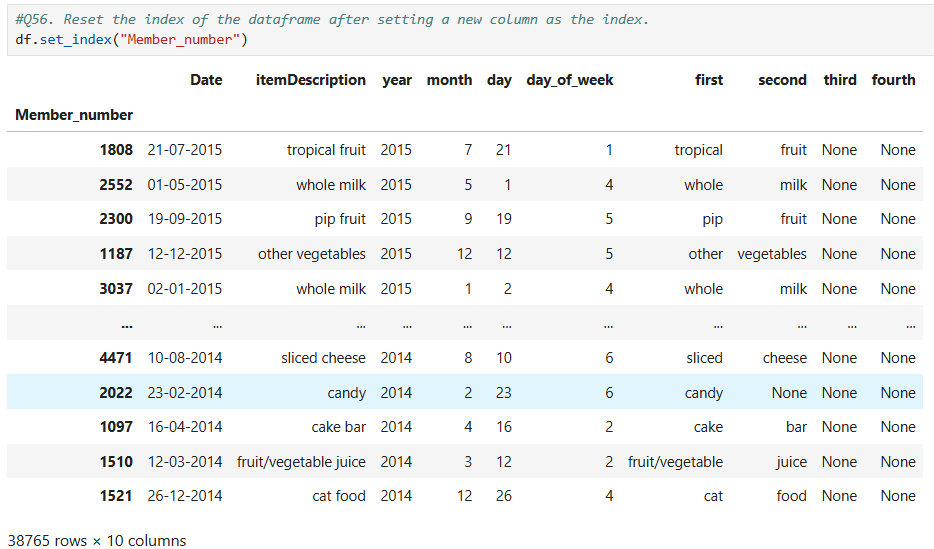
****

1. **Index - Indexing and resetting index**

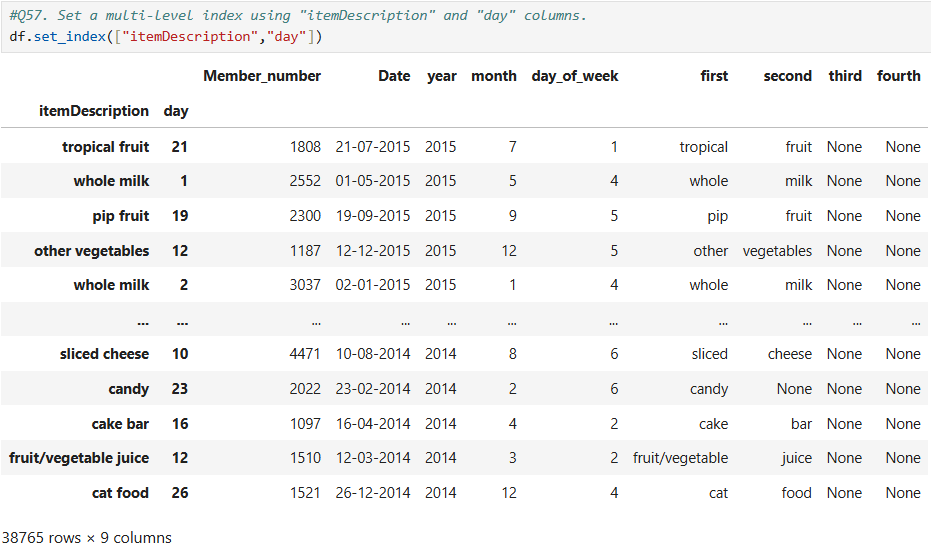
**Q64. Set a specific column (eg. itemDescription) as the index of the DataFrame.**

****

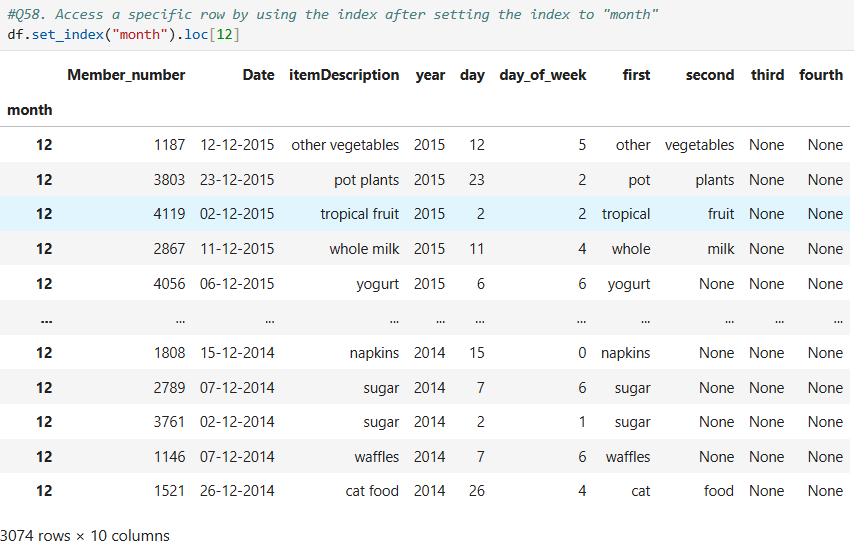
**Q65. Reset the index of the dataframe after setting a new column as the index.**

****

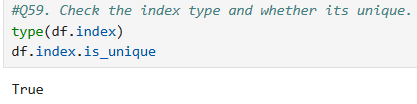
**Q66. Set a multi-level index using "itemDescription" and "day" columns.**

****

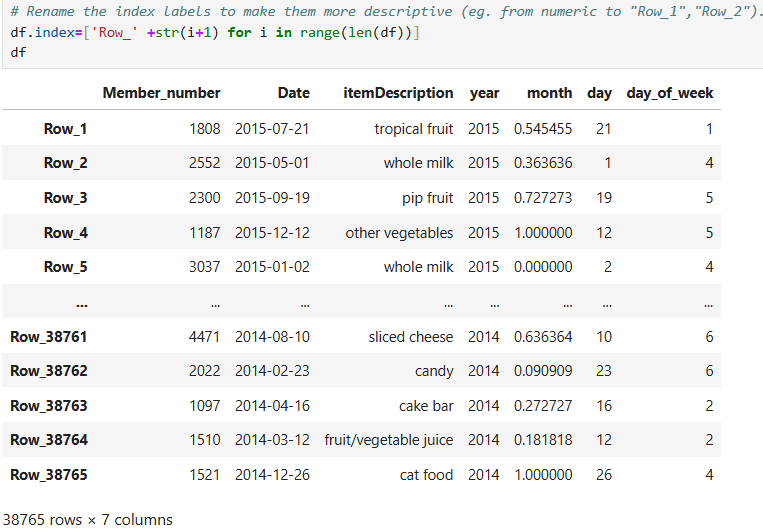
**Q67. Access a specific row by using the index after setting the index to "month"**

****

**Q68. Check the index type and whether its unique.**

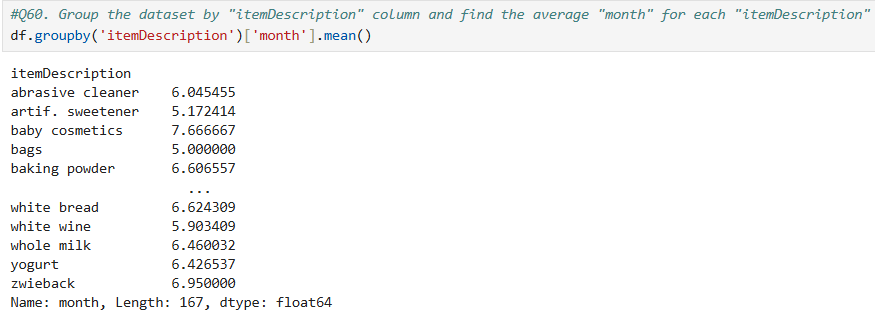
****

**Q69. Rename the index labels to make them more descriptive (eg. from numeric to "Row\_1","Row\_2").**

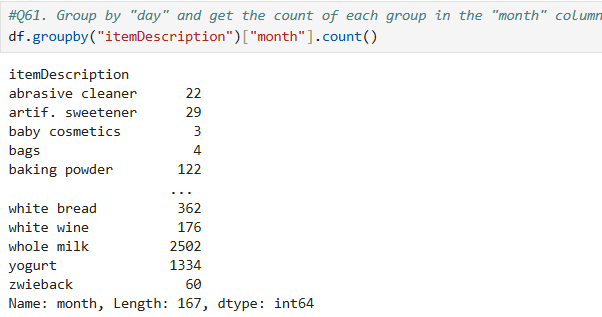
****

1. **groupby – grouping data**

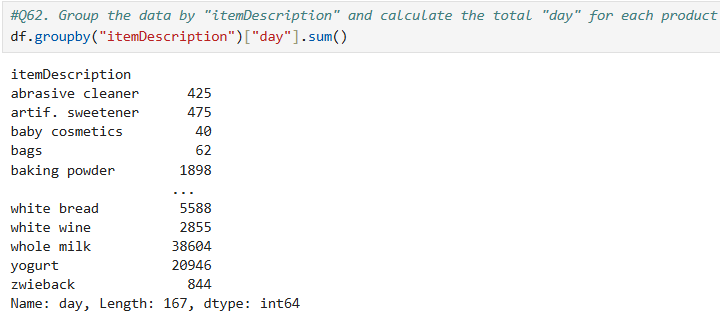
**Q70. Group the dataset by "itemDescription" column and find the average "month" for each "itemDescription"**

****

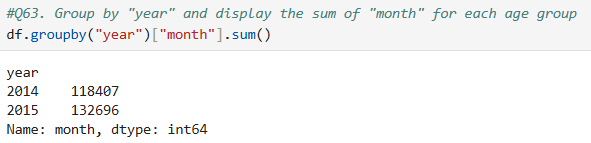
**Q71. Group by "day" and get the count of each group in the "month" column.**

****

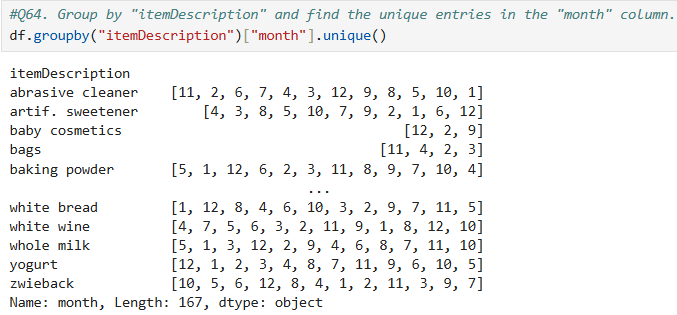
**Q72. Group the data by "itemDescription" and calculate the total "day" for each product**

****

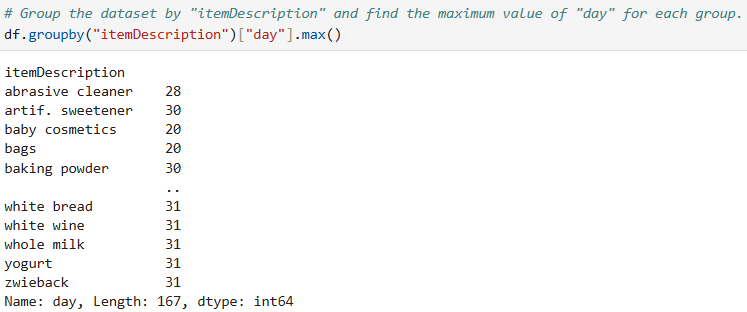
**Q73. Group by "year" and display the sum of "month" for each age group**

****

**Q74. Group by "itemDescription" and find the unique entries in the "month" column.**

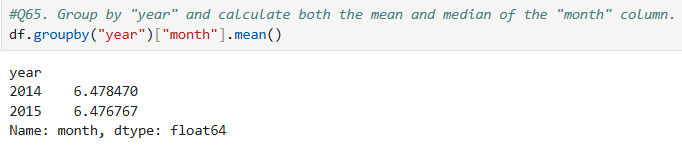
****

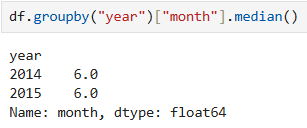
**Q75. Group the dataset by "itemDescription" and find the maximum value of "day" for each group**

****

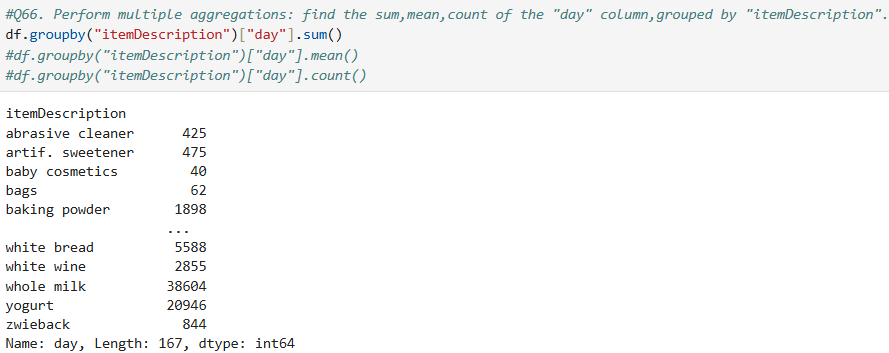
1. **Aggregation - Performing multiple aggregations**

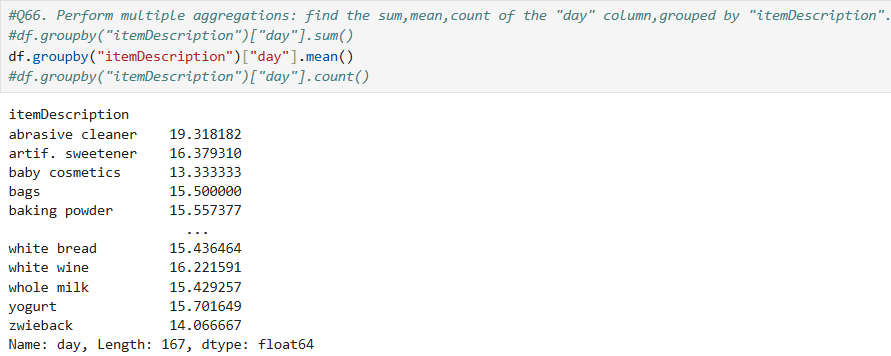
**Q76. Group by "year" and calculate both the mean and median of the "month" column.**

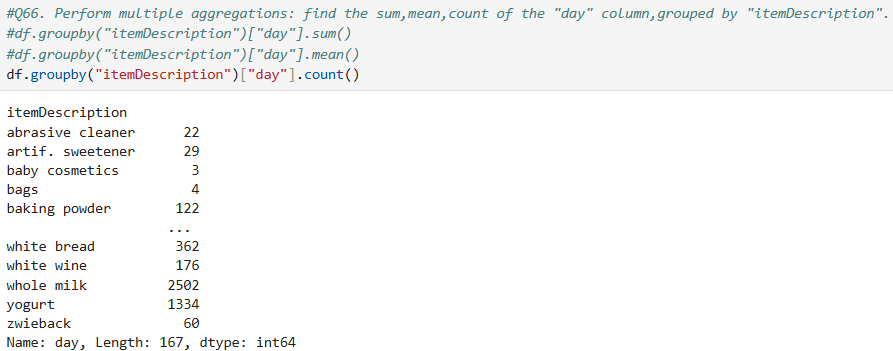
****

****

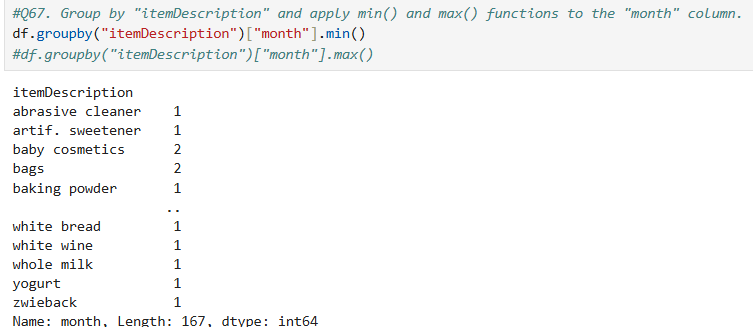
**Q77. Perform multiple aggregations: find the sum,mean,count of the "day" column,grouped by "itemDescription".**

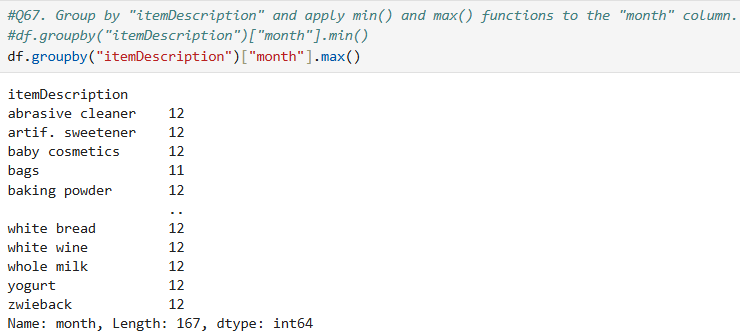
****

****

****

**Q78. Group by "itemDescription" and apply min() and max() functions to the "month" column.**

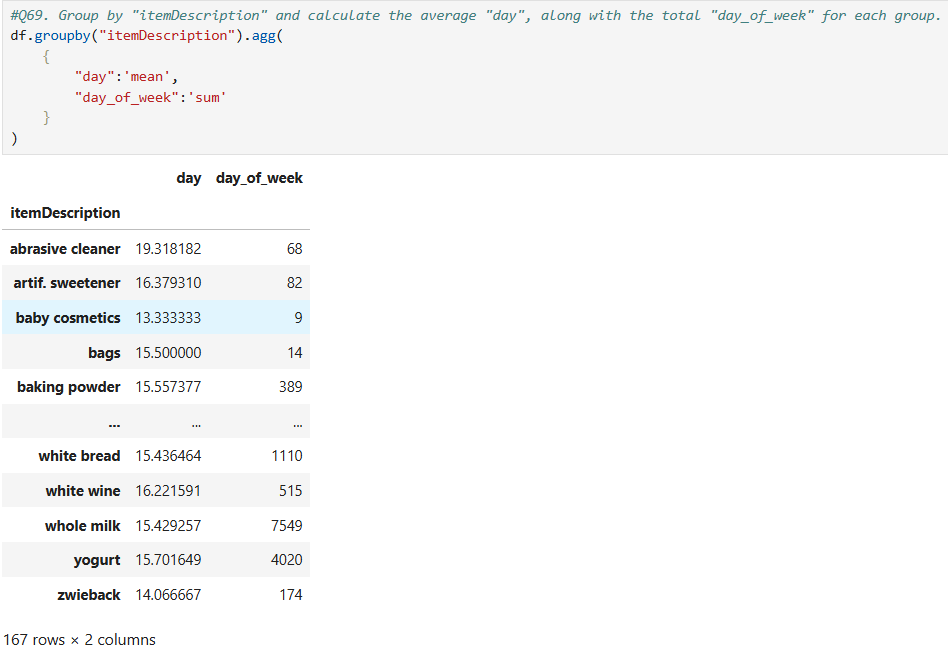
****

****

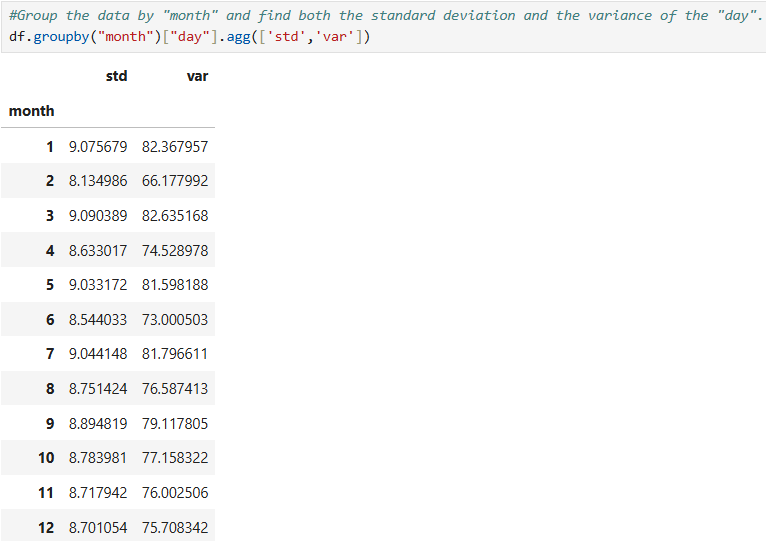
**Q79. Group by "itemDescription" column and apply a custom aggregation function(find the range of "month" within each itemDescription)**

****

**Q80. Group by "itemDescription" and calculate the average "day", along with the total "day\_of\_week" for each group.**

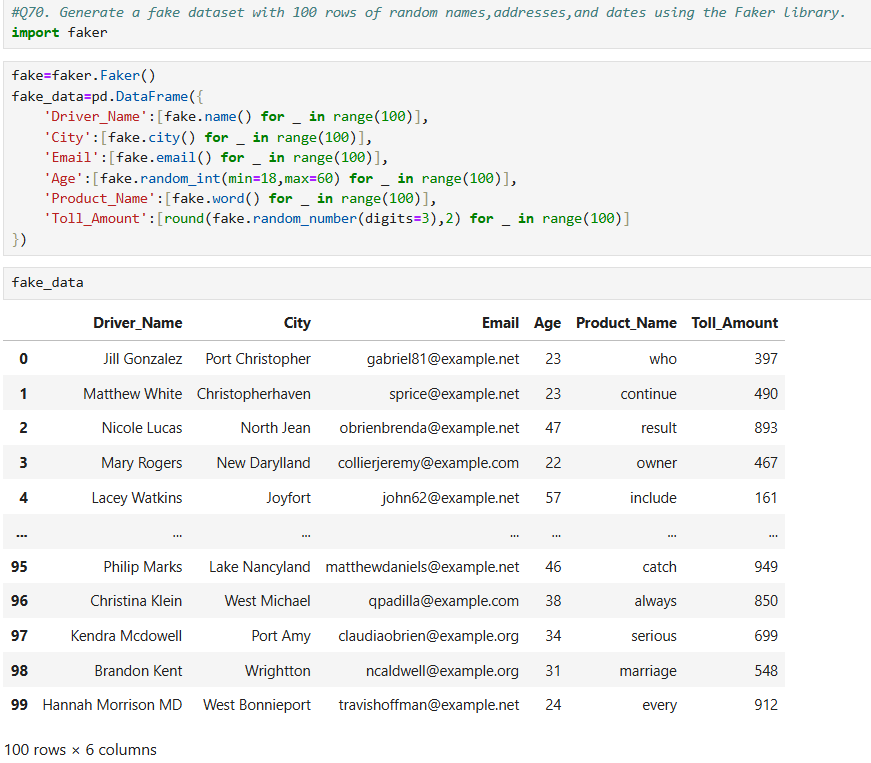
****

**Q81. Group the data by "month" and find both the standard deviation and the variance of the "day".**

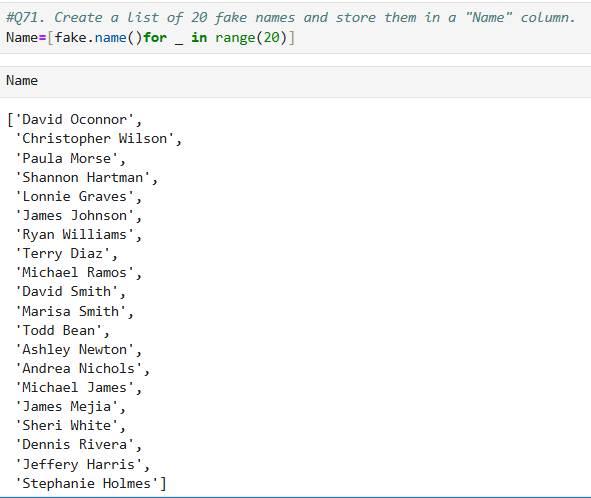
****

1. **Faker – Generating Fake Data for Testing**

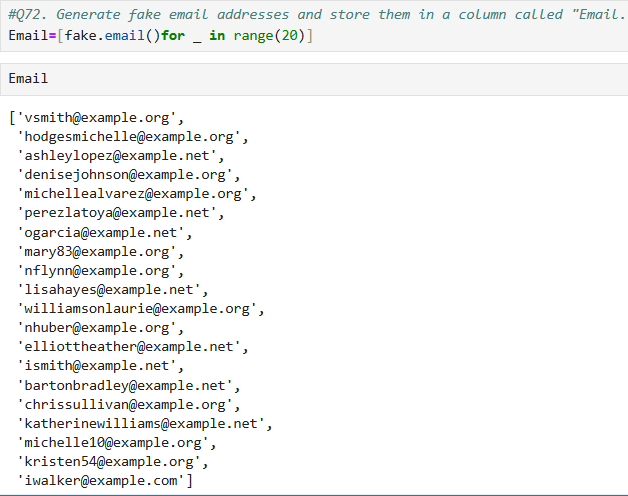
**Q82. Generate a fake dataset with 100 rows of random names,addresses,and dates using the Faker library.**

****

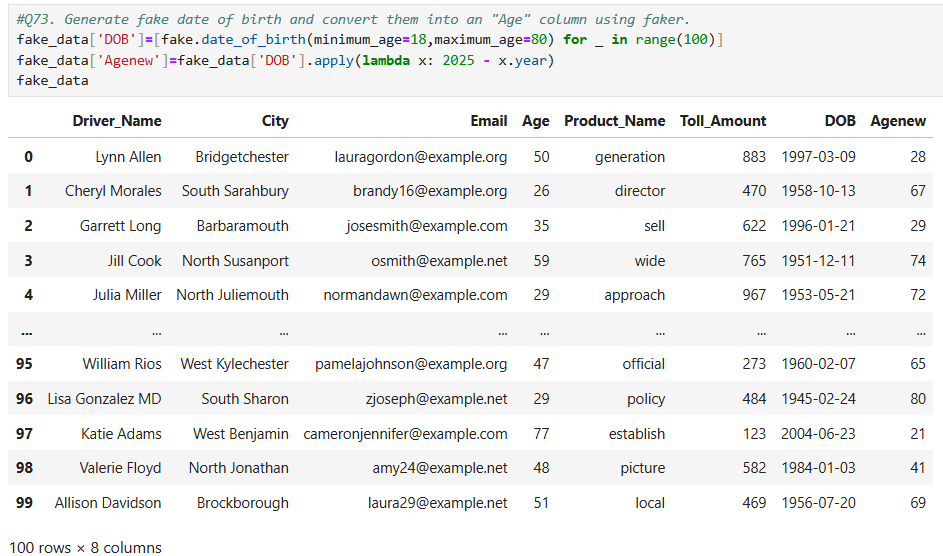
**Q83. Create a list of 20 fake names and store them in a "Name" column.**

****

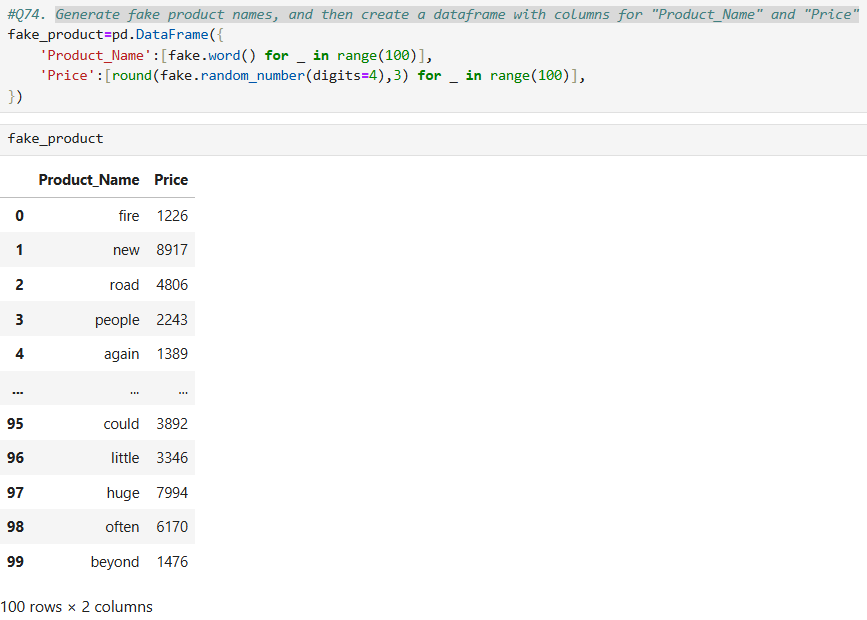
**Q84. Generate fake email addresses and store them in a column called "Email.**

****

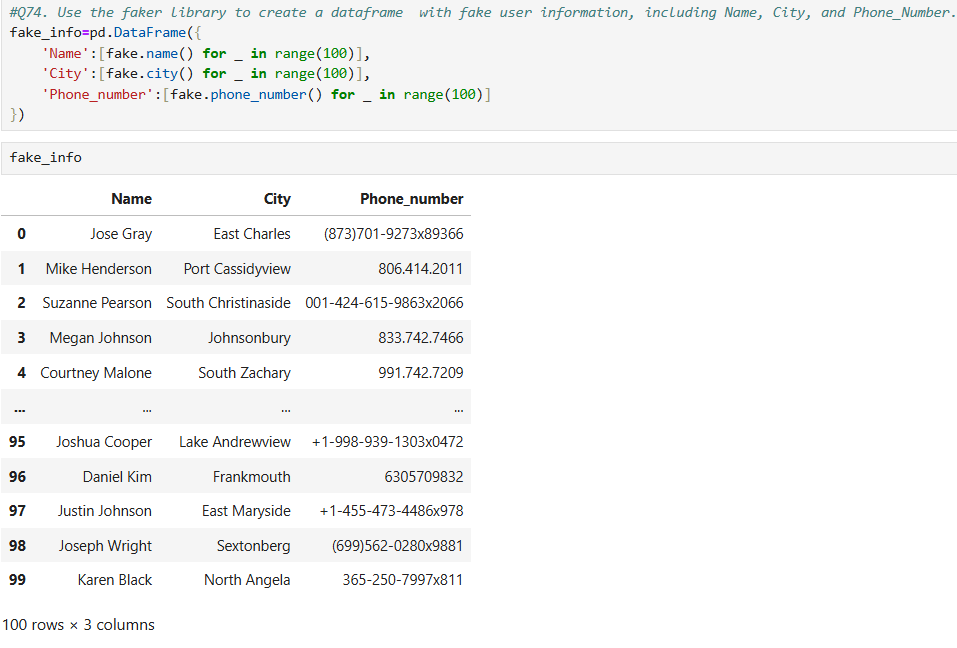
**Q85. Generate fake date of birth and convert them into an "Age" column using faker.**

****

**Q86. Generate fake product names, and then create a dataframe with columns for "Product\_Name" and "Price".**

****

**Q87. Use the faker library to create a dataframe with fake user information, including Name, City, and Phone\_Number.**

****

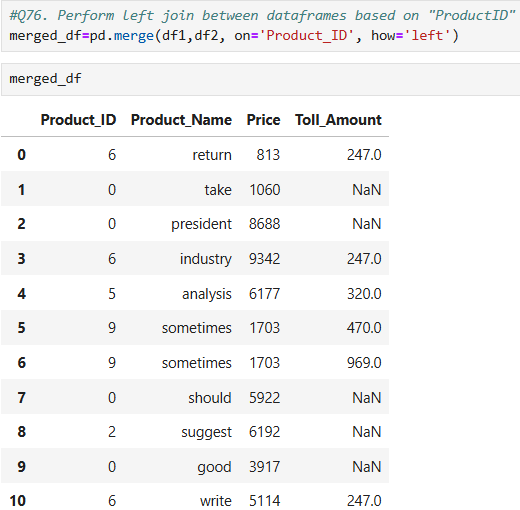
1. **merge – Merging DataFrames**

**Q88. Merge two dataframes on common column Product\_ID**

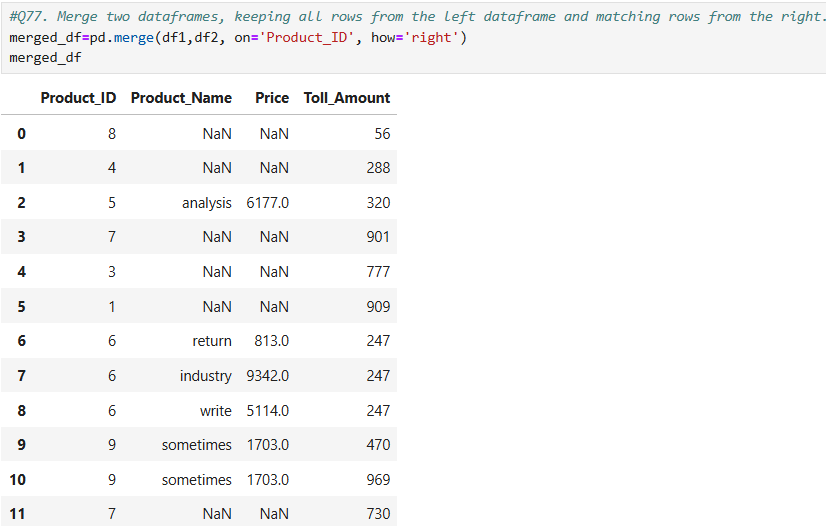
****

****

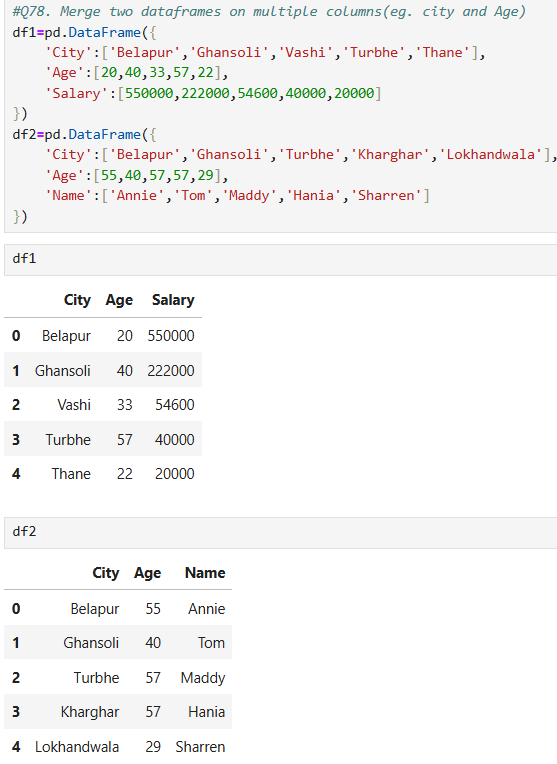
**Q89. Perform left join between dataframes based on "ProductID"**

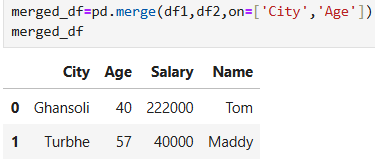
****

**Q90. Merge two dataframes, keeping all rows from the left dataframe and matching rows from the right.**

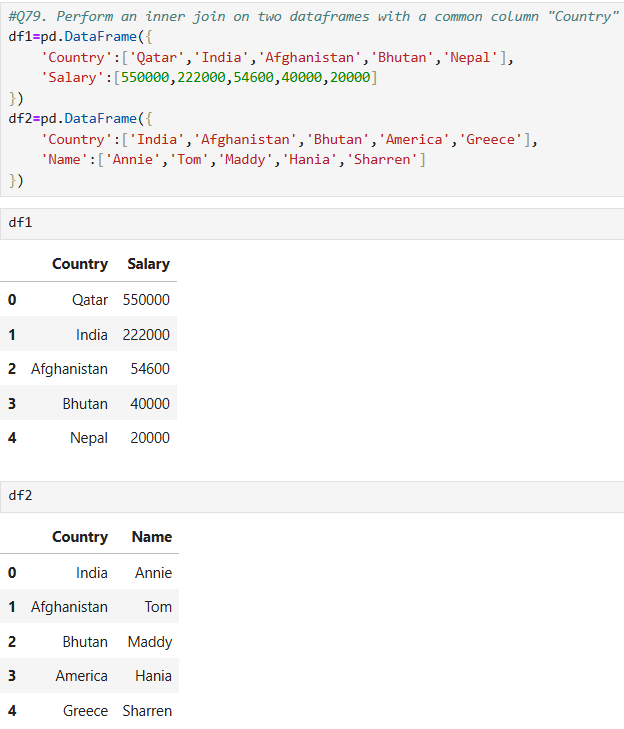
****

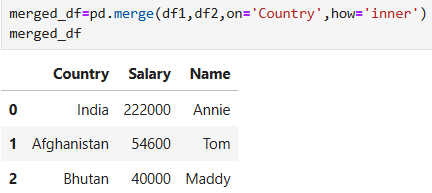
**Q91. Merge two dataframes on multiple columns(eg. city and Age)**

****

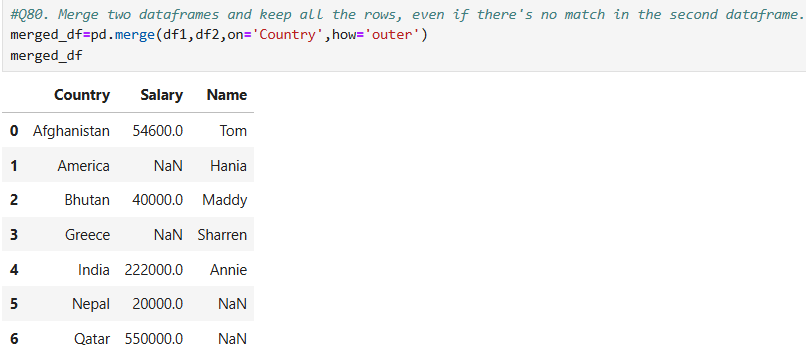
****

**Q92. Perform an inner join on two dataframes with a common column "Country"**

****

****

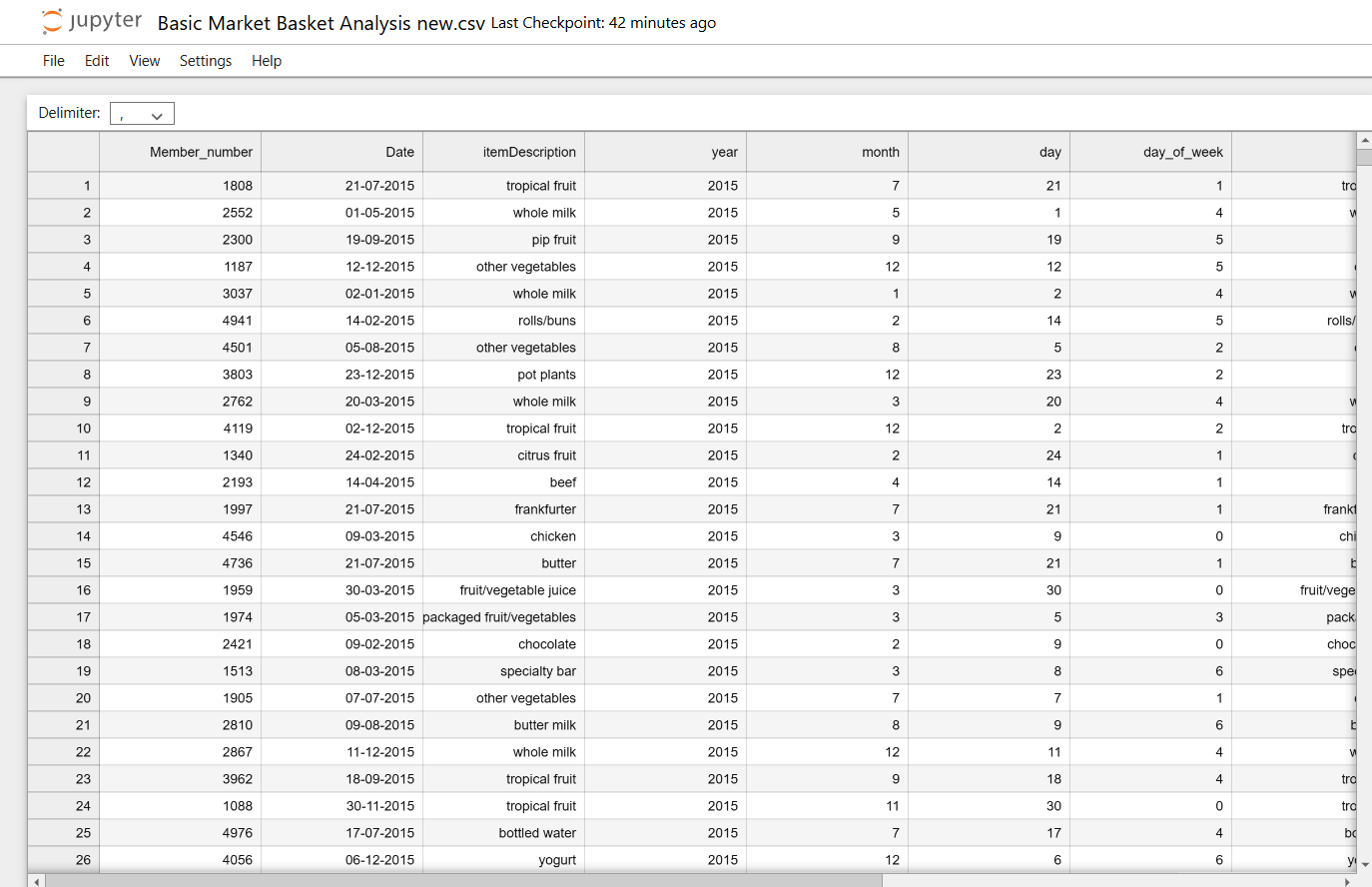
**Q93. Merge two dataframes and keep all the rows, even if there's no match in the second dataframe.**

****

1. **Saving and Exporting Data**

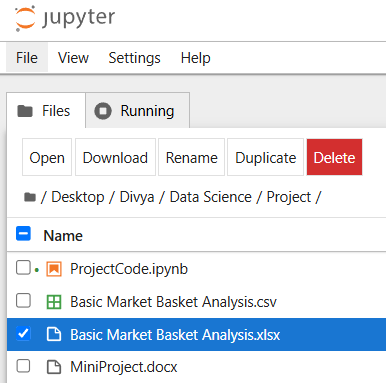
**Q94. Save the dataframe to a new csv file.**

****

****

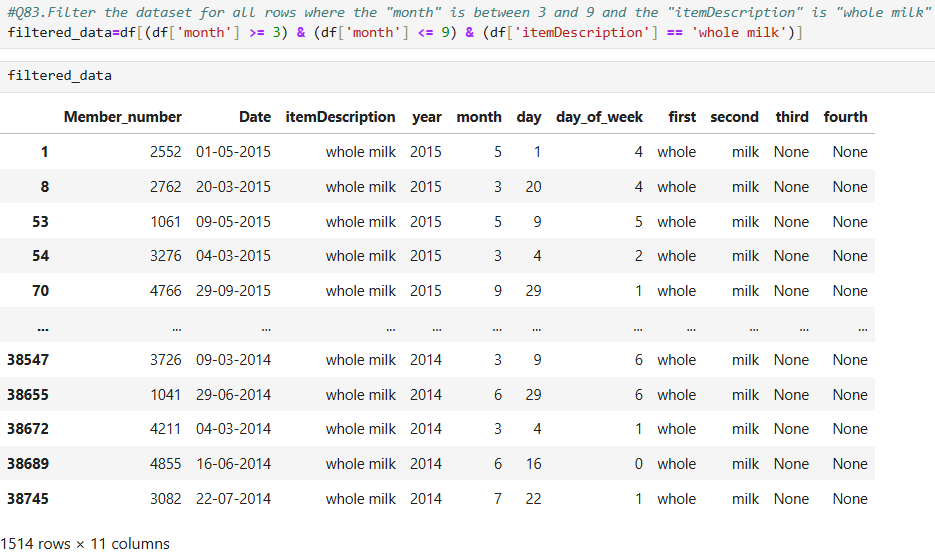
**Q95. Export the dataframe to an excel file.**

****

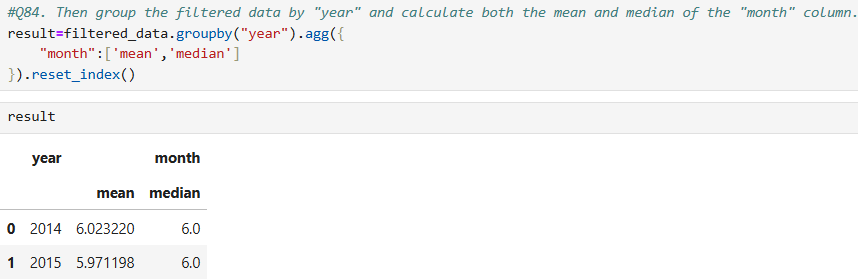
****

1. **Complex Filtering and Aggregation**

**Q96. Filter the dataset for all rows where the "month" is between 3 and 9 and the "itemDescription" is "whole milk"**

****

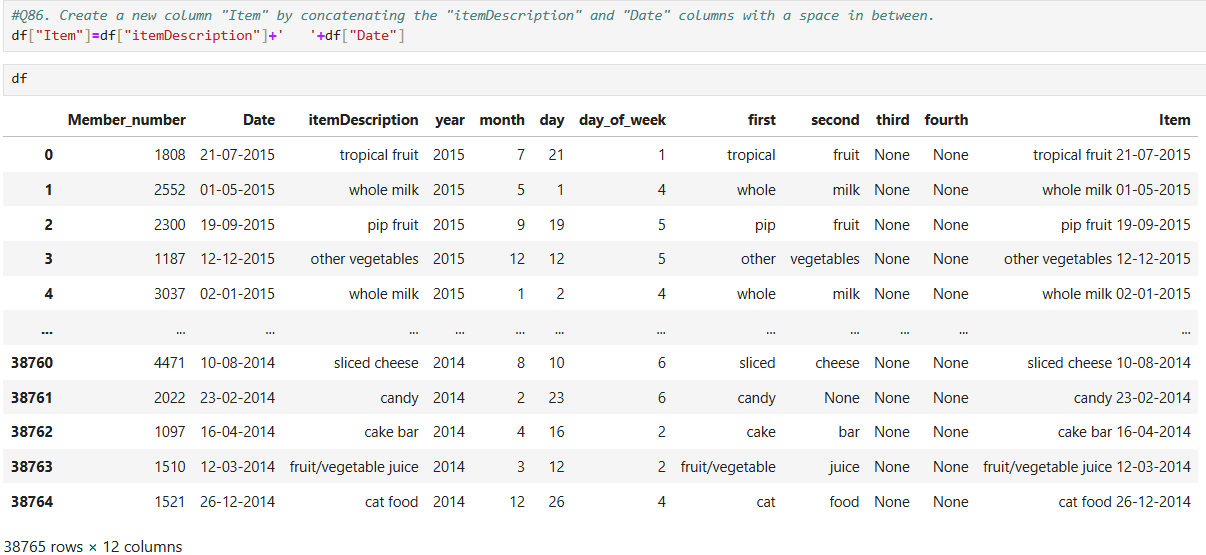
**Q97. Then group the filtered data by "year" and calculate both the mean and median of the "month" column.**

****

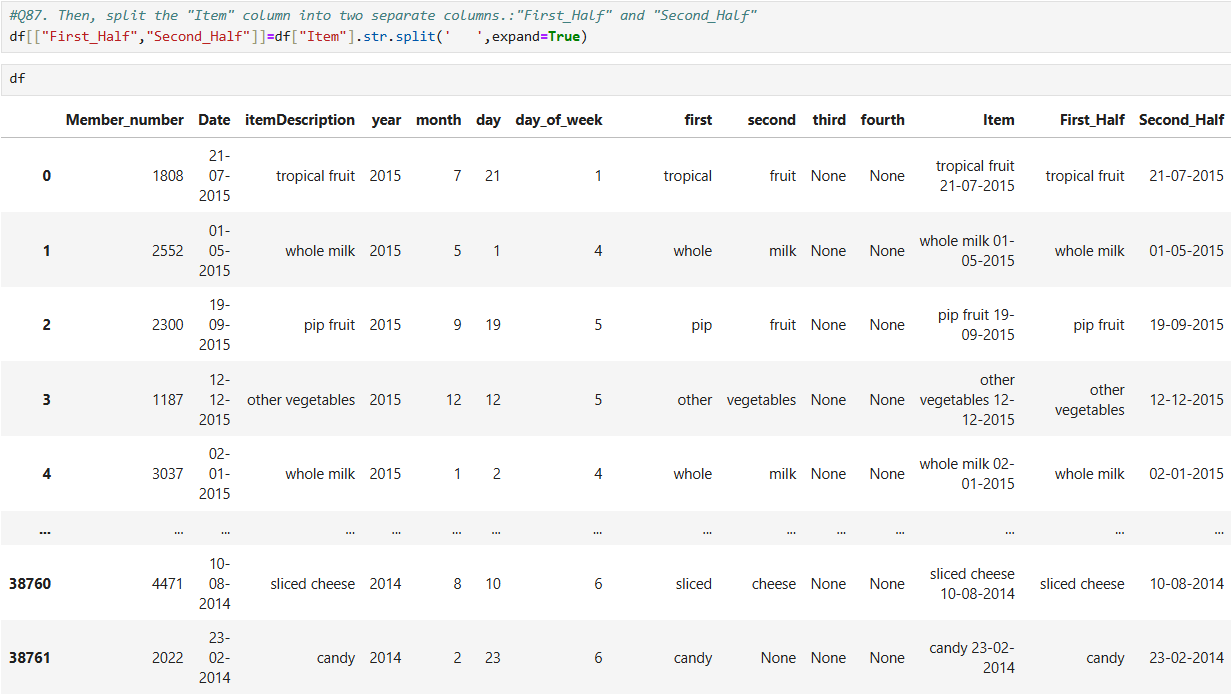
**Q98. After that, sort the result in descending order by the "month" column**

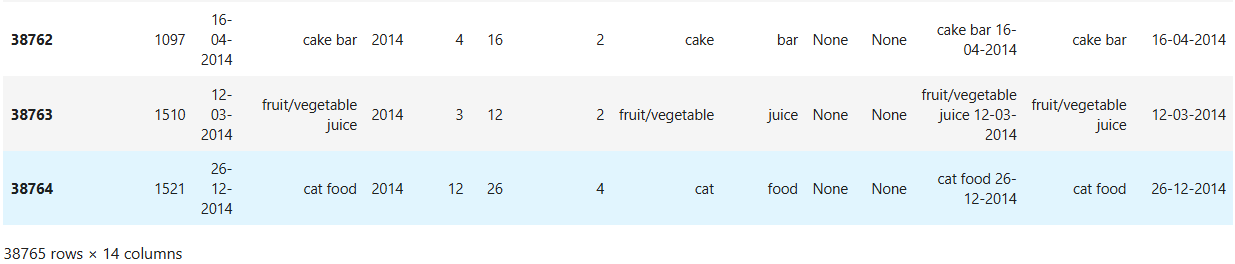
1. **Advanced String Manipulation and Grouping**

**Q99. Create a new column "Item" by concatenating the "itemDescription" and "Date" columns with a space in between.**

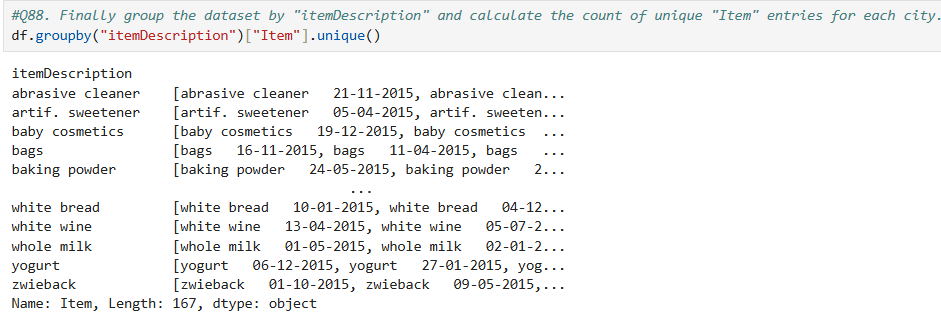
****

**Q100. Then, split the "Item" column into two separate columns.:"First\_Half" and "Second\_Half".**

****

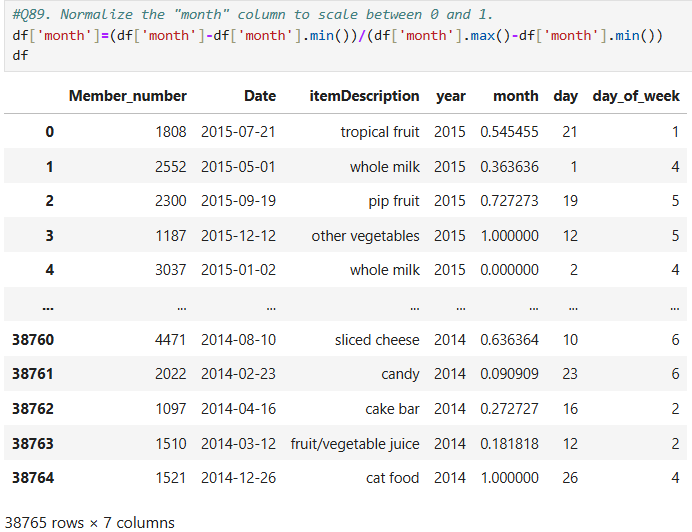
****

**Q101. Finally group the dataset by "itemDescription" and calculate the count of unique "Item" entries for each city.**

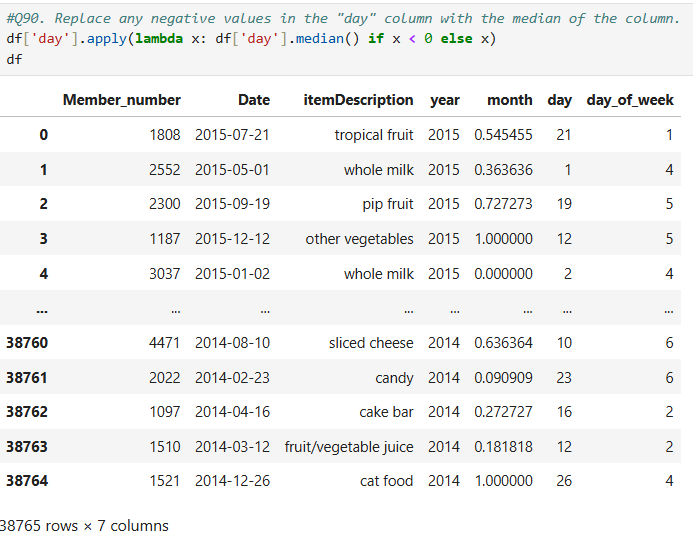
****

1. **Multiple Data Transformation and Merging**

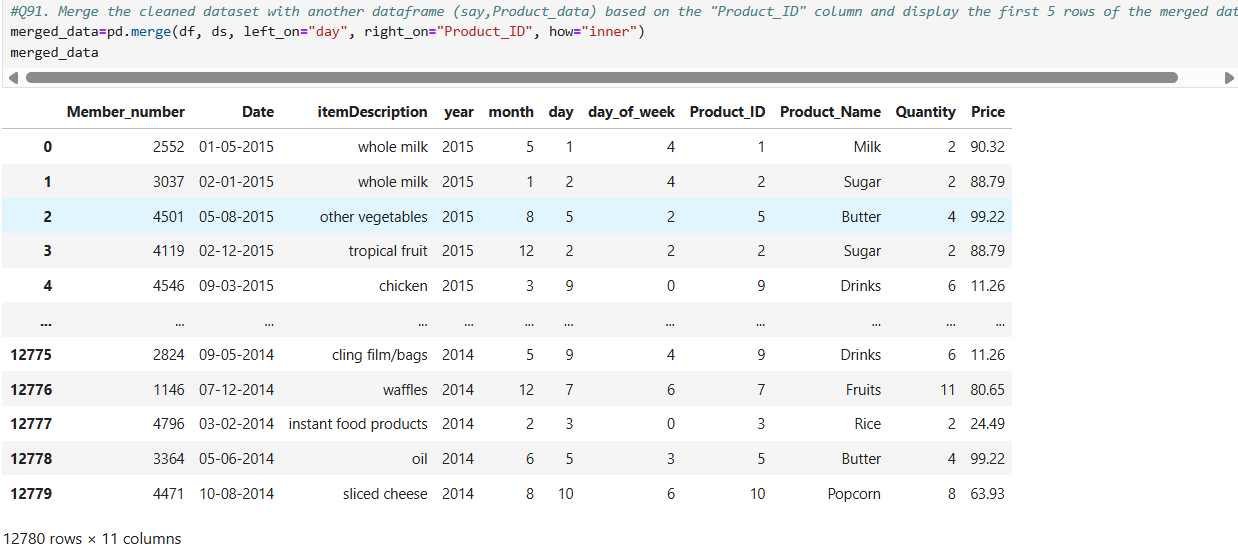
**Q102. Normalize the "month" column to scale between 0 and 1.**

****

**Q103. Replace any negative values in the "day" column with the median of the column.**

****

**Q104. Merge the cleaned dataset with another dataframe (say,Product\_data) based on the "Product\_ID" column and display the first 5 rows of the merged dataset.**

****

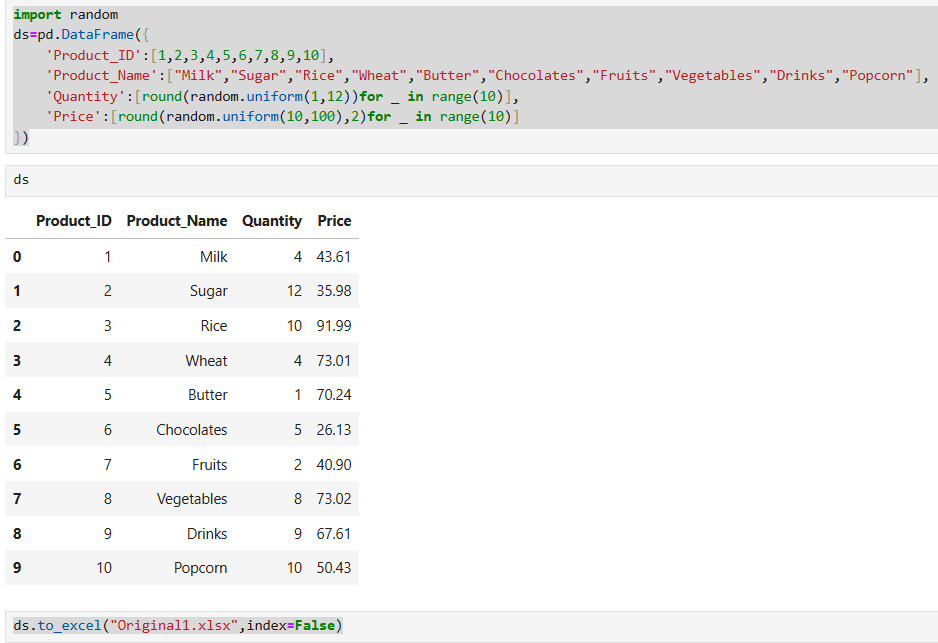
**Q105. Create a DataFrame with 'Product ID', 'Product Name', 'Quantity', and 'Price'. Add 10**

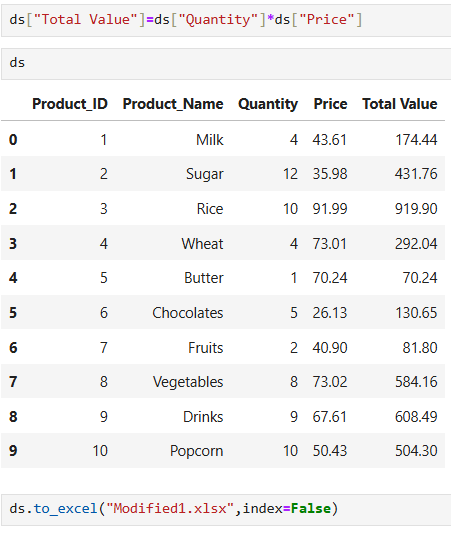
**rows of data for different products. Perform an operation to calculate the 'Total Value'**

**(Quantity \* Price) and add it as a new column. Export the modified DataFrame to an**

**Excel file and compare the original and modified versions. What changes can you**

**observe in the data?**

****

****

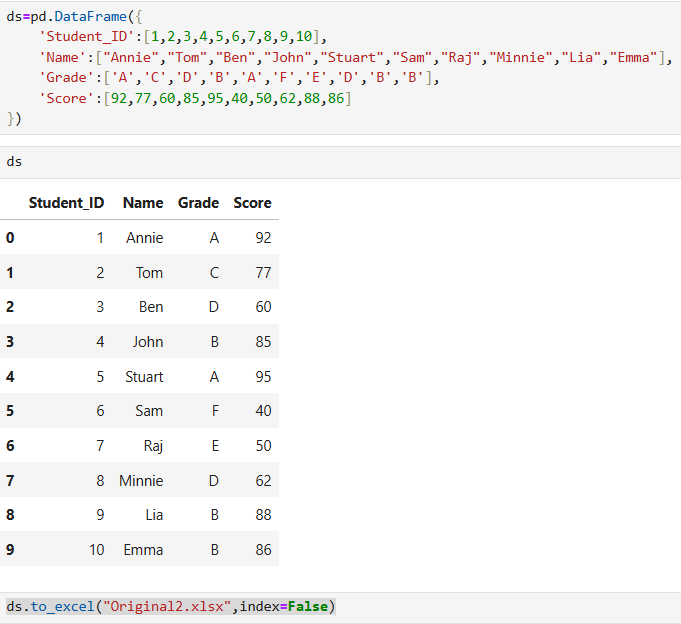
**Q106. Create a DataFrame with 'Student ID', 'Name', 'Grade', and 'Score'. Add 10 rows with**

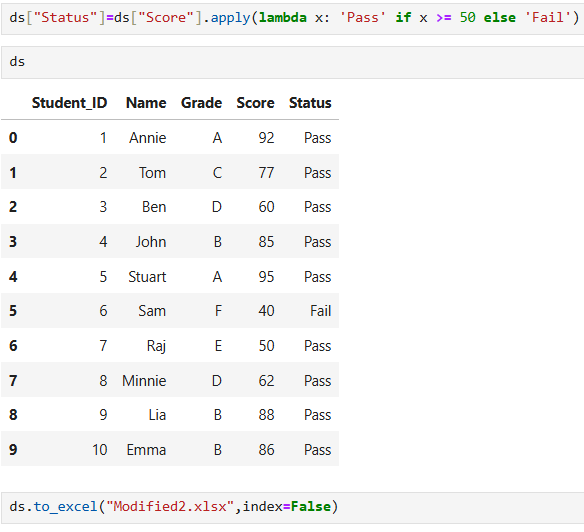
**data. Perform an operation to assign a 'Pass/Fail' status based on whether the score is**

**above 50 (Pass) or below 50 (Fail). Export the modified DataFrame to an Excel file and**

**show the comparison between the initial file and the updated one. What new columns or**

**changes can you see?**

****

****

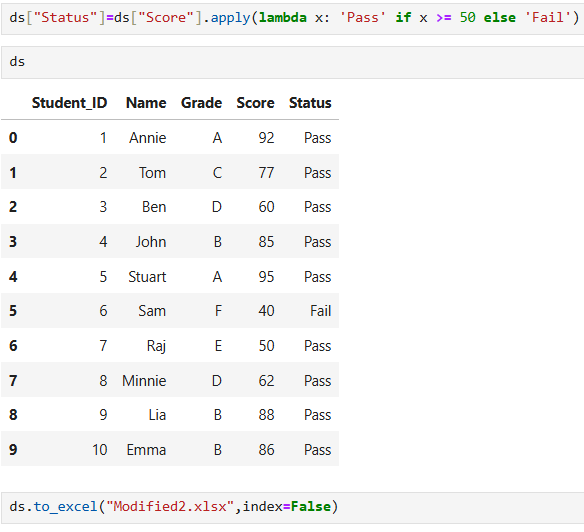
**Q107. Create a CSV file and an Excel file with columns 'Student ID', 'Name', 'Grade', and**

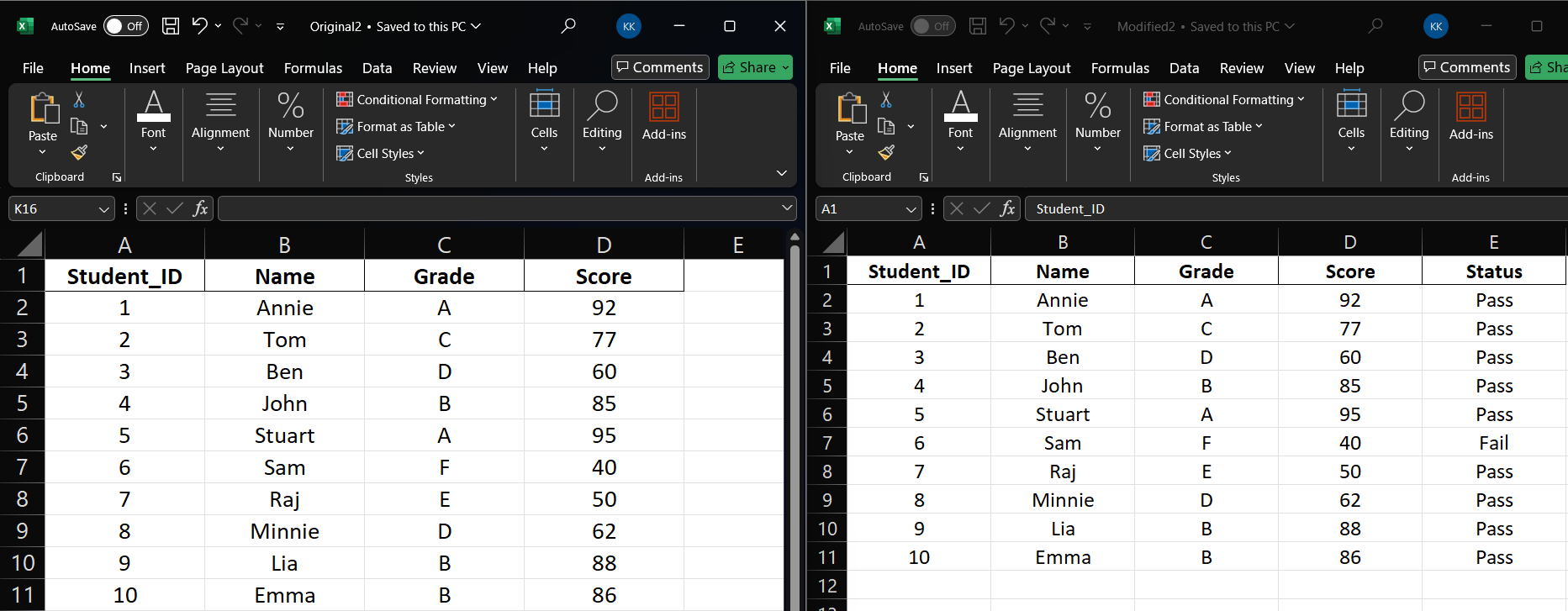
**'Score'. Add 5 rows of student data to each file. Upload the files to Jupyter Notebook,**

**then perform an operation to assign a 'Pass/Fail' status based on whether the score is**

**above 50. Export the modified CSV and Excel files and compare them with the originals.**

**Explain the differences and the new columns added**

****

****

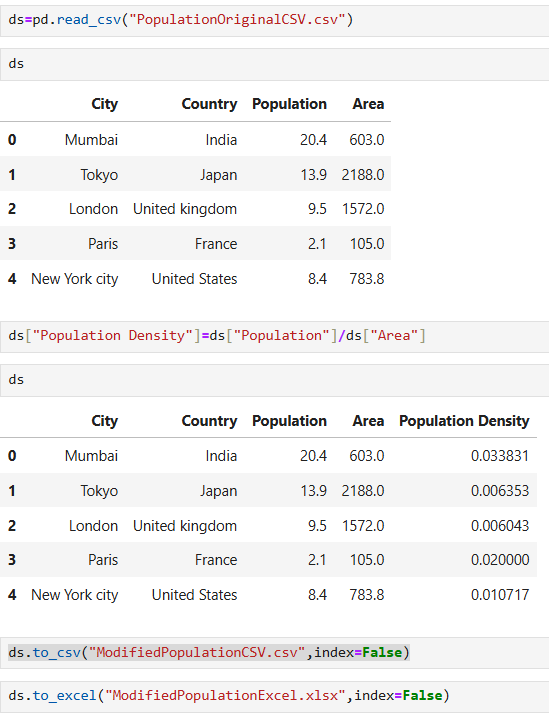
**Q108. Create a CSV file and an Excel file with 'City', 'Country', 'Population', and 'Area'. Add 5**

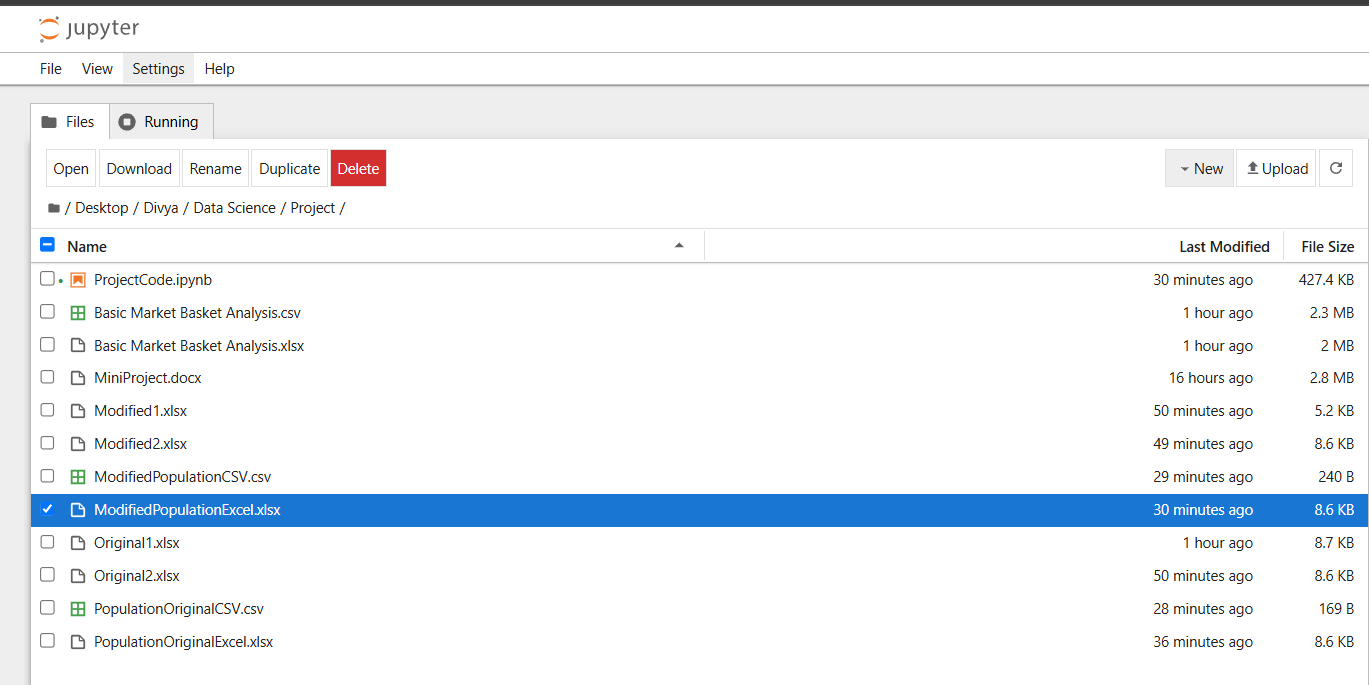
**rows of city data to both files. Upload the files to Jupyter Notebook, then perform an**

**operation to calculate the 'Population Density' (Population / Area) and add it as a new**

**column. Export both files after modification and compare the original and updated files.**

**What new information can you observe in the 'Population Density' column?**

****

****