Student-Math-Data-Analysis-Workbook

January 6, 2025

Author: Bosley Boka

1 Project Overview

This is the workbook used for analysis on the project. It shows processes used and actions taken, along with code. It is only supplementary to the Student-Math-Data-Analysis-Report provided separately.

1.1 Instructions

Project goal is to generate a cleaned and merged data file in csv format, given 3 ficticious datasets, and provide notes as to what was found in the data, including any missing data.

1.1.1 Provided Files:

1. Assessment file (sample_assessment.csv): Manhattan School District standardized test results - please include Math score.
2. Math product Usage file (sample_usage.csv): Please include both lessons completed and total minutes columns in the merged file. Students in grade levels 5, 6, 7, and 8 have access to the Math product.
3. SIS file (sample_sis.csv): Please include all information listed in this file (all students in grade levels 5, 6, 7, and 8) and all columns.

2 Setup

[1]:
```
# Import modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

# Allows plots to be rendered inside the notebook
%matplotlib inline
```

[2]:
```
# Load files
sis_data = pd.read_csv('./resources/sample_sis.csv')
assessment_data = pd.read_csv('./resources/sample_assessment.csv')
usage_data = pd.read_csv('./resources/sample_usage.csv')
```

[3]:
```
# Create list of dataframes for ease of programming
dataframes = [("SIS Data", sis_data), ("Assessment Data", assessment_data),␣
↪("Usage Data", usage_data)]
```

3 Data Overview

[4]:
```
# View headers
sis_data.head()
```

[4]:

	student_id	grade_level	race_ethnicity	gender	Free/Reduced Price Lunch
0	1254353	7	White	Male	No
1	1254135	5	White	Male	NaN
2	1254423	8	Black/African American	Male	Reduced Price
3	1254598	8	White	Male	Free
4	1254562	6	Black/African American	Male	No

[5]:
```
assessment_data.head()
```

[5]:

	id	student_number	subject	score
0	1254397	54912357	ELA	190
1	1254204	54912164	ELA	567
2	1254785	54912745	ELA	390
3	1254308	54912268	Math	719
4	1254560	54912520	ELA	394

[6]:
```
usage_data.head()
```

[6]:

	student_id	lesson completed	benchmark_1_level	benchmark_2_level	benchmark_3_level	benchmark_4_level	total_minutes
0	1254110	13	level 2	level 1	level 3	level 2	47.808670
1	1254113	14	level 2	level 1	level4	level 1	156.792335
2	1254288	16	level 3	level 1	level 3	level 1	38.135959
3	1254095	15	level 3	level 1	level 1	level 1	18.257427
4	1254250	1	level 2	level 1			

```
4           level 2           level 2      16.479016
```

3.1 Cursory Observations

- Relationship column between files is Student ID (student_id).
  - While the Usage & SIS files have a clearly labeled student_id column, the Assessment file has two possible columns that could be Student ID, either id or student_number.
  - All id-related columns have 7 digits, but the id column in the Assessments file has similar leading numbers as the student_id columns in the other two files.
- Additional header observations:
  - File headers have inconsistent formats- spacing vs underscores, mixed casing, includes slash character.
  - The directions require "lessons completed", whereas the column header is "lesson completed".
- NaN value is present in Free/Reduced Price Lunch.
- Benchmark value format is inconsistent.
- Since Math scores are the focus, I'll need to check & filter the subject column.
- Since Grade Level determines access to the Usage file, I'll need to check and filter the grade_level column.

4 Normalize Headers

[7]:
```
custom_mappings = {
    'lesson_completed': 'lessons_completed',
    'id': 'student_id'
}

for name, df in dataframes:
    df.columns = df.columns.str.lower().str.replace(' ', '_').str.replace('/',␣
 ↪'_').str.replace(r'[^\w]', '')
    if custom_mappings:
        df.rename(columns=custom_mappings, inplace=True)
```

5 Analyze Data in Preparation for Merge

[8]:
```
for name, df in dataframes:
    print(name, "Info:")
    df.info()
    print("\n")
```

```
SIS Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 783 entries, 0 to 782
Data columns (total 5 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
```

```
 0   student_id                783 non-null    int64
 1   grade_level               783 non-null    int64
 2   race_ethnicity            736 non-null    object
 3   gender                    783 non-null    object
 4   free_reduced_price_lunch  669 non-null    object
dtypes: int64(2), object(3)
memory usage: 30.7+ KB


Assessment Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1280 entries, 0 to 1279
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   student_id      1280 non-null   int64
 1   student_number  1280 non-null   int64
 2   subject         1280 non-null   object
 3   score           1280 non-null   int64
dtypes: int64(3), object(1)
memory usage: 40.1+ KB


Usage Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349 entries, 0 to 348
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   student_id         349 non-null    int64
 1   lessons_completed  349 non-null    int64
 2   benchmark_1_level  349 non-null    object
 3   benchmark_2_level  349 non-null    object
 4   benchmark_3_level  349 non-null    object
 5   benchmark_4_level  349 non-null    object
 6   total_minutes      349 non-null    float64
dtypes: float64(1), int64(2), object(4)
memory usage: 19.2+ KB
```

Data Type Consistency Column Data Types are as expected. Student ID's and other int-expected columns are integers. Columns with expected string values are objects. Total minutes are floats.

Null or NaN: I can see there are Null or NaN values in the SIS Data for columns race/ethnicity and free/reduced price lunch, but these are non-essential columns.

5.1 Check ID Columns for Duplicates

[9]:
```
for name, df in dataframes:
    print(f"{name} has duplicates?: {df['student_id'].duplicated().any()}")
```

```
SIS Data has duplicates?: False
Assessment Data has duplicates?: True
Usage Data has duplicates?: False
```

Duplicates as Expected: Assessment Data has student scores for both Math & ELA. SIS & Usage have no duplicates. Assessment Data will need to be handled accordingly.

5.2 Check Numeric Columns for Inconsistencies

[10]:
```
# Check specific columns for values less than 0
for name, df in dataframes:
    print(f"Checking {name}...")
    found = False
    for column in df.select_dtypes(include=['number']).columns:
        if column in df.columns and pd.api.types.is_numeric_dtype(df[column]):
            Q1 = df[column].quantile(0.25)
            Q3 = df[column].quantile(0.75)
            IQR = Q3 - Q1
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR
            if (df[column] < lower_bound).any() | (df[column] > upper_bound).
 ↪any():
                found = True
                print(f"  Column '{column}' in {name} has outliers.")
            if (df[column] < 1).any():
                found = True
                print(f"  Column '{column}' in {name} has less than 1.")
            if (df[column] == 0).any():
                found = True
                print(f"  Column '{column}' in {name} has zeroes.")
            if (df[column] < 0).any():
                found = True
                print(f"  Column '{column}' in {name} has negatives.")

    if (not found):
        print("No inconsistencies found.")
```

```
Checking SIS Data…
No inconsistencies found.
Checking Assessment Data…
No inconsistencies found.
Checking Usage Data…
  Column 'total_minutes' in Usage Data has outliers.
```

```
Column 'total_minutes' in Usage Data has less than 1.
```

[11]:
```
# Usage IDs
usage_data.describe()['total_minutes']
```

[11]:
```
count    349.000000
mean      76.705856
std       56.142303
min        0.167026
25%       32.308333
50%       65.030197
75%      111.739739
max      280.517175
Name: total_minutes, dtype: float64
```

[12]:
```
def get_outliers_and_issues(df):
    result_rows = pd.DataFrame()

    for column in df.select_dtypes(include=['number']).columns:
        col_data = df[column]

        # Calculate IQR for outlier detection
        Q1 = col_data.quantile(0.25)
        Q3 = col_data.quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Identify rows with issues
        issues = df[(col_data < lower_bound) |
                    (col_data > upper_bound) |
                    (col_data == 0) |
                    (col_data < 0) |
                    (col_data < 1)]

        # Add a column to indicate which issue was detected
        issues = issues.copy()  # Avoid SettingWithCopyWarning
        issues['issue_column'] = column  # Column with the detected issue

        result_rows = pd.concat([result_rows, issues])

    # Remove duplicates in case multiple columns flagged the same row
    return result_rows.drop_duplicates()

# Example Usage
outlier_rows = get_outliers_and_issues(usage_data)
outlier_rows
```

[12]:

	student_id	lessons_completed	benchmark_1_level	benchmark_2_level	benchmark_3_level	benchmark_4_level	total_minutes	issue_column
16	1254138	12	level4	level 3	level 1	level 3	255.231657	total_minutes
24	1254393	16	level 2	level 2	level 1	level 1	0.167026	total_minutes
69	1254280	4	level 1	level 2	level 1	level 1	261.391669	total_minutes
154	1254252	9	level 3	level 3	level 2	level4	247.638391	total_minutes
171	1254367	13	level4	level 3	level 2	level 2	280.517175	total_minutes
179	1254255	14	level 2	level 3	level4	level 1	0.865787	total_minutes
261	1254284	23	level 3	level 2	level4	level 3	0.856146	total_minutes

Total Minutes < 1 Data shows 3 students were able to complete between 14-23 lessons in less than 1 minute.

[13]:
```
# Check SIS Grade Levels are as expected (5-8)
print(sis_data['grade_level'].unique())
```

```
[7 5 8 6]
```

[14]:
```
# Check SIS Race/Ethnicity values
print(sis_data['race_ethnicity'].unique())
```

```
['White' 'Black/African American' 'Two or More Races' nan 'Asian'
 'Native Havaiian/Other Pacific Islander' 'American Indian/Alaska Native'
 'Hispanic/Latino']
```

[15]:
```
# Check SIS Gender values
print(sis_data['gender'].unique())
```

```
['Male' 'Female']
```

[16]:
```
# Check SIS Free/Reduced Price Lunch values
print(sis_data['free_reduced_price_lunch'].unique())
```

```
['No' nan 'Reduced Price' 'Free']
```

[17]:
```
# Check Assessment Subjects are as expected (Math or ELA)
print(assessment_data['subject'].unique())
```

```
['ELA' 'Math']
```

5.3 Handle SIS Data

[19]:
```
# Check all SIS columns for NaN or blank values
null_counts = sis_data.isnull().sum()
blank_counts = sis_data.apply(lambda col: col.astype(str).str.strip().eq('').
 ↪sum())

# Combine into a summary DataFrame
issue_summary = pd.DataFrame({
    'Null Count': null_counts,
    'Blank Count': blank_counts
})
print(issue_summary)
```

```
                        Null Count  Blank Count
student_id                       0            0
grade_level                      0            0
race_ethnicity                  47            0
gender                           0            0
free_reduced_price_lunch       114            0
```

[20]:
```
# Get the IDs where NaN in SIS Race/Ethnicity column
ids_with_race_nan = sis_data[sis_data['race_ethnicity'].isna()]['student_id'].
 ↪tolist()
print(ids_with_race_nan)

# Filter rows with NaN in Race/Ethnicity column
race_ethnicity_nans = sis_data[sis_data['race_ethnicity'].isna()]
race_ethnicity_nans.head(5)
```

```
[1254825, 1254771, 1254180, 1254104, 1254079, 1254332, 1254503, 1254475,
1254810, 1254574, 1254378, 1254328, 1254327, 1254815, 1254364, 1254350, 1254745,
1254755, 1254756, 1254299, 1254545, 1254643, 1254488, 1254089, 1254778, 1254537,
1254291, 1254555, 1254616, 1254305, 1254214, 1254245, 1254387, 1254762, 1254663,
1254461, 1254689, 1254377, 1254449, 1254735, 1254518, 1254094, 1254249, 1254549,
1254493, 1254192, 1254708]
```

[20]:

	student_id	grade_level	race_ethnicity	gender	free_reduced_price_lunch
33	1254825	7	NaN	Male	NaN
35	1254771	5	NaN	Male	No
48	1254180	8	NaN	Male	NaN
61	1254104	8	NaN	Male	No
65	1254079	5	NaN	Female	No

[21]:
```
# Get the IDs where NaN in SIS Free/Reduced Lunch column
ids_with_lunch_nan = sis_data[sis_data['free_reduced_price_lunch'].
 ↪isna()]['student_id'].tolist()
print(ids_with_lunch_nan)
```

```
# Filter rows with NaN in SIS Free/Reduced Lunch column
free_reduced_price_lunch_nans = sis_data[sis_data['free_reduced_price_lunch'].
 ↪isna()]
free_reduced_price_lunch_nans.head(5)
```

```
[1254135, 1254128, 1254588, 1254529, 1254825, 1254767, 1254180, 1254112,
1254308, 1254819, 1254292, 1254516, 1254799, 1254577, 1254489, 1254508, 1254258,
1254212, 1254166, 1254839, 1254614, 1254743, 1254436, 1254260, 1254617, 1254666,
1254441, 1254400, 1254583, 1254818, 1254628, 1254697, 1254726, 1254760, 1254201,
1254700, 1254167, 1254533, 1254654, 1254746, 1254363, 1254442, 1254824, 1254123,
1254541, 1254576, 1254478, 1254840, 1254426, 1254627, 1254664, 1254597, 1254343,
1254846, 1254324, 1254147, 1254145, 1254256, 1254836, 1254089, 1254276, 1254837,
1254551, 1254572, 1254605, 1254222, 1254103, 1254404, 1254233, 1254136, 1254672,
1254205, 1254099, 1254295, 1254710, 1254163, 1254655, 1254293, 1254220, 1254121,
1254111, 1254307, 1254543, 1254582, 1254278, 1254625, 1254741, 1254229, 1254369,
1254482, 1254134, 1254550, 1254368, 1254348, 1254590, 1254440, 1254641, 1254548,
1254281, 1254325, 1254411, 1254479, 1254785, 1254669, 1254091, 1254843, 1254702,
1254119, 1254829, 1254523, 1254624, 1254708, 1254283, 1254826]
```

[21]:

	student_id	grade_level	race_ethnicity	gender	free_reduced_price_lunch
1	1254135	5	White	Male	NaN
17	1254128	8	White	Male	NaN
19	1254588	7	Two or More Races	Male	NaN
25	1254529	7	Black/African American	Female	NaN
33	1254825	7	NaN	Male	NaN

5.4 Handle Assessment Data

[22]:
```
# Split Assessment Data into Math vs ELA
assessment_ela = assessment_data[assessment_data['subject'] == 'ELA']
assessment_math = assessment_data[assessment_data['subject'] == 'Math']

assessment_math.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 640 entries, 3 to 1279
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   student_id      640 non-null    int64
 1   student_number  640 non-null    int64
 2   subject         640 non-null    object
 3   score           640 non-null    int64
dtypes: int64(3), object(1)
memory usage: 25.0+ KB
```

[23]:
```
# Create list of new dataframes for ease of programming
math_dataframes = [("SIS Data", sis_data), ("Assessment Math Data",␣
 ↪assessment_math), ("Usage Data", usage_data)]
```

[24]:
```
for name, df in math_dataframes:
    print(f"{name} has duplicates?: {df['student_id'].duplicated().any()}")
```

```
SIS Data has duplicates?: False
Assessment Math Data has duplicates?: False
Usage Data has duplicates?: False
```

No Duplicate IDs Found No duplicates found after removing ELA from Assessments

6 Comparing Differences and Intersections

[26]:
```
# Comparing Differences between data
assessment_math_ids = np.array(assessment_math['student_id'])
assessment_ela_ids = np.array(assessment_ela['student_id'])
usage_ids = np.array(usage_data['student_id'])
sis_ids = np.array(sis_data['student_id'])

print("IDs in Assessment but not in SIS:", len(np.
 ↪setdiff1d(assessment_math_ids, sis_ids)))
print("IDs in Usage but not in SIS:", len(np.setdiff1d(usage_ids, sis_ids)))
print("IDs in SIS but not in Assessment:", len(np.setdiff1d(sis_ids,␣
 ↪assessment_math_ids)))
print("IDs in SIS but not in Usage:", len(np.setdiff1d(sis_ids, usage_ids)))
print("IDs in Assessment but not in Usage:", len(np.
 ↪setdiff1d(assessment_math_ids, usage_ids)))
print("IDs in Usage but not in Assessment:", len(np.setdiff1d(usage_ids,␣
 ↪assessment_math_ids)))
print("\n")
sis_assess_intersect_ids = np.intersect1d(sis_ids, assessment_math_ids)
all_intersecting_ids = np.intersect1d(sis_assess_intersect_ids, usage_ids)
print("IDs in both SIS and Assessment:", len(sis_assess_intersect_ids))
print("IDs in ALL files:", len(all_intersecting_ids))
```

```
IDs in Assessment but not in SIS: 0
IDs in Usage but not in SIS: 8
IDs in SIS but not in Assessment: 143
```

```
IDs in SIS but not in Usage: 442
IDs in Assessment but not in Usage: 409
IDs in Usage but not in Assessment: 118


IDs in both SIS and Assessment: 640
IDs in ALL files: 231
```

[30]:
```
# List Missing Data (ID's in Usage but not in SIS)
missing_usage_ids = np.setdiff1d(usage_ids, sis_ids)
print("IDs in Usage but not in SIS:", missing_usage_ids)

missing_usage_data = usage_data[usage_data['student_id'].
 ↪isin(missing_usage_ids)]

missing_usage_data
```

```
IDs in Usage but not in SIS: [1254065 1254066 1254067 1254068 1254069 1254070
1254071 1254072]
```

[30]:

	student_id	lessons_completed	benchmark_1_level	benchmark_2_level	benchmark_3_level	benchmark_4_level	total_minutes
40	1254070	7	level4	level 3	level 1	level4	134.195032
46	1254069	23	level4	level 2	level 3	level 1	59.247430
77	1254065	5	level4	level 3	level 1	level4	124.325568
144	1254072	24	level 1	level 3	level 1	level 3	129.205541
200	1254068	18	level 3	level 3	level4	level 1	5.879212
221	1254066	16	level 1	level 1	level4	level 3	12.492581
263	1254067	4	level 2	level4	level 2	level 2	63.419238
309	1254071	14	level 2	level 3	level4	level 2	228.975415

Missing Data: These rows are in the Usage file, but not in the SIS file.

[31]:
```
# Check for assessment data in missing Usage ID's
print("Common IDs between missing usage data & assessment data:", len(np.
 ↪intersect1d(missing_usage_ids, assessment_math_ids)))
```

```
Common IDs between missing usage data & assessment data: 0
```

[28]:
```
# Check for unique ELA ID's
print("IDs in Assessment ELA but not in Assessment Math:", len(np.
 ↪setdiff1d(assessment_ela_ids, assessment_math_ids)))
```

```
IDs in Assessment ELA but not in Assessment Math: 0
```

7 Test Data

[32]:
```
sis_intersecting_data = sis_data[sis_data['student_id'].
 ↪isin(all_intersecting_ids)]
print(sis_intersecting_data.head(5)["student_id"].to_list())
sis_intersecting_data.head(5)
```

```
[1254353, 1254213, 1254275, 1254296, 1254322]
```

[32]:

	student_id	grade_level	race_ethnicity	gender	free_reduced_price_lunch
0	1254353	7	White	Male	No
5	1254213	8	White	Male	No
7	1254275	7	White	Male	Reduced Price
10	1254296	6	White	Male	Reduced Price
11	1254322	6	Black/African American	Male	No

[33]:
```
# Test Data IDs
test_IDs = [1254353, 1254213, 1254275, 1254296, 1254322]
assess_intersecting_data = assessment_math[assessment_math['student_id'].
 ↪isin(test_IDs)]
assess_intersecting_data
```

[33]:

	student_id	student_number	subject	score
474	1254275	54912235	Math	601
653	1254353	54912313	Math	289
694	1254296	54912256	Math	255
958	1254322	54912282	Math	154
974	1254213	54912173	Math	518

[34]:
```
usage_intersecting_data = usage_data[usage_data['student_id'].isin(test_IDs)]
usage_intersecting_data
```

[34]:

	student_id	lessons_completed	benchmark_1_level	benchmark_2_level
107	1254353	24	level 2	level 2

```
148     1254213                 21            level4          level 3
257     1254275                 16           level 2          level 3
266     1254322                 20           level 2          level 3
320     1254296                 21            level4           level4

    benchmark_3_level benchmark_4_level  total_minutes
107           level 2           level 2      21.040558
148            level4           level 1      11.374870
257            level4           level 3     133.622386
266            level4            level4      14.605244
320           level 2           level 3     128.115228
```

8 Merge & Clean Data

- Use SIS data as base, into which other data is merged

[35]:
```
# Merge SIS and Usage
merged_df = sis_data.merge(usage_data, on='student_id', how='left')

# Merge the result with Assessment
merged_df = merged_df.merge(assessment_math, on='student_id', how='left')

# Rename score column for clarity
merged_df.rename(columns={'score': 'math_score'}, inplace=True)

# Convert columns back to int64 Data Type
merged_df['lessons_completed'] = merged_df['lessons_completed'].astype('Int64')
merged_df['student_number'] = merged_df['student_number'].astype('Int64')
merged_df['math_score'] = merged_df['math_score'].astype('Int64')

# Display the merged dataset
merged_df.head(10)
```

[35]:
```
   student_id  grade_level          race_ethnicity  gender  \
0     1254353            7                   White    Male
1     1254135            5                   White    Male
2     1254423            8  Black/African American    Male
3     1254598            8                   White    Male
4     1254562            6  Black/African American    Male
5     1254213            8                   White    Male
6     1254122            5  Black/African American  Female
7     1254275            7                   White    Male
8     1254600            6  Black/African American  Female
9     1254828            5                   White  Female

  free_reduced_price_lunch  lessons_completed benchmark_1_level  \
0                       No                 24           level 2
```

```
1                      NaN                  1           level 2
2            Reduced Price               <NA>               NaN
3                     Free               <NA>               NaN
4                       No               <NA>               NaN
5                       No                 21            level4
6            Reduced Price                  8           level 2
7            Reduced Price                 16           level 2
8                       No               <NA>               NaN
9                       No               <NA>               NaN

  benchmark_2_level benchmark_3_level benchmark_4_level  total_minutes  \
0           level 2           level 2           level 2      21.040558
1           level 2           level 3           level 3      53.124390
2               NaN               NaN               NaN            NaN
3               NaN               NaN               NaN            NaN
4               NaN               NaN               NaN            NaN
5           level 3            level4           level 1      11.374870
6           level 3            level4            level4      82.582013
7           level 3            level4           level 3     133.622386
8               NaN               NaN               NaN            NaN
9               NaN               NaN               NaN            NaN

   student_number subject  math_score
0        54912313    Math         289
1            <NA>     NaN        <NA>
2        54912383    Math         614
3        54912558    Math         578
4        54912522    Math         395
5        54912173    Math         518
6            <NA>     NaN        <NA>
7        54912235    Math         601
8        54912560    Math         705
9            <NA>     NaN        <NA>
```

9 Confirm Integrity Post-Merge

- Using Test Data

[36]:
```
merged_df[merged_df['student_id'].isin(test_IDs)]
```

[36]:
```
    student_id  grade_level          race_ethnicity gender  \
0      1254353            7                   White   Male
5      1254213            8                   White   Male
7      1254275            7                   White   Male
10     1254296            6                   White   Male
11     1254322            6  Black/African American   Male
```

```
   free_reduced_price_lunch  lessons_completed benchmark_1_level  \
0                        No                 24           level 2
5                        No                 21            level4
7             Reduced Price                 16           level 2
10            Reduced Price                 21            level4
11                       No                 20           level 2

   benchmark_2_level benchmark_3_level benchmark_4_level  total_minutes  \
0            level 2           level 2           level 2      21.040558
5            level 3            level4           level 1      11.374870
7            level 3            level4           level 3     133.622386
10            level4           level 2           level 3     128.115228
11           level 3            level4            level4      14.605244

    student_number subject  math_score
0         54912313    Math         289
5         54912173    Math         518
7         54912235    Math         601
10        54912256    Math         255
11        54912282    Math         154
```

10 Clean & Create CSV

[37]:
```
final_csv = merged_df[['student_id', 'grade_level', 'race_ethnicity', 'gender',␣
↪'free_reduced_price_lunch', 'lessons_completed', 'total_minutes',␣
↪'math_score']]
```

[34]:
```
final_csv.to_csv('./resources/final_cleaned_data.csv', index=False)
```

[38]:
```
final_csv.head(10)
```

[38]:
```
   student_id  grade_level          race_ethnicity  gender  \
0     1254353            7                   White    Male
1     1254135            5                   White    Male
2     1254423            8  Black/African American    Male
3     1254598            8                   White    Male
4     1254562            6  Black/African American    Male
5     1254213            8                   White    Male
6     1254122            5  Black/African American  Female
7     1254275            7                   White    Male
8     1254600            6  Black/African American  Female
9     1254828            5                   White  Female

  free_reduced_price_lunch  lessons_completed  total_minutes  math_score
0                       No                 24      21.040558         289
1                      NaN                  1      53.124390        <NA>
```

```
2            Reduced Price               <NA>            NaN         614
3                     Free               <NA>            NaN         578
4                       No               <NA>            NaN         395
5                       No                 21      11.374870         518
6            Reduced Price                  8      82.582013        <NA>
7            Reduced Price                 16     133.622386         601
8                       No               <NA>            NaN         705
9                       No               <NA>            NaN        <NA>
```

[39]:
```
# Confirm test values
final_csv[final_csv['student_id'].isin(test_IDs)]
```

[39]:
```
    student_id  grade_level          race_ethnicity gender  \
0      1254353            7                   White   Male
5      1254213            8                   White   Male
7      1254275            7                   White   Male
10     1254296            6                   White   Male
11     1254322            6  Black/African American   Male

   free_reduced_price_lunch  lessons_completed  total_minutes  math_score
0                        No                 24      21.040558         289
5                        No                 21      11.374870         518
7             Reduced Price                 16     133.622386         601
10            Reduced Price                 21     128.115228         255
11                       No                 20      14.605244         154
```

11 Analyze Missing Data

[41]:
```
final_csv.info()
```

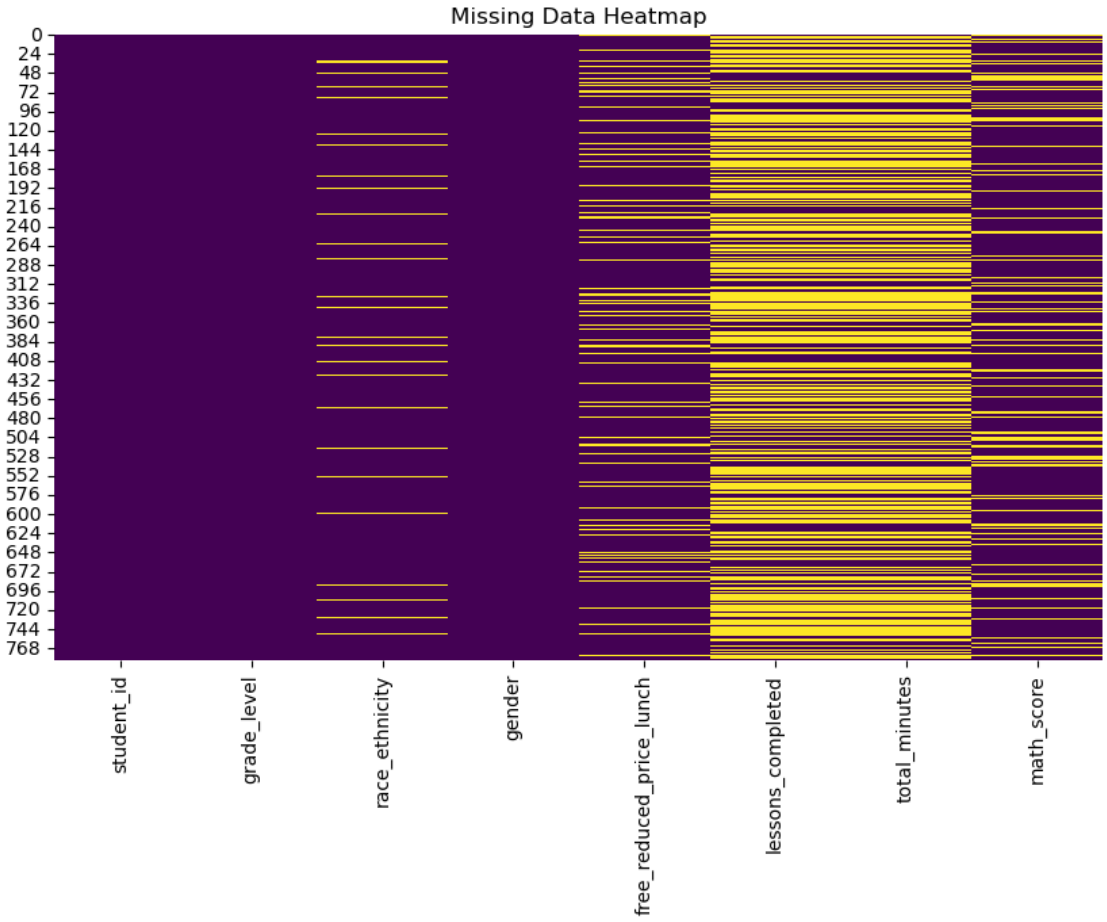
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 783 entries, 0 to 782
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   student_id               783 non-null    int64
 1   grade_level              783 non-null    int64
 2   race_ethnicity           736 non-null    object
 3   gender                   783 non-null    object
 4   free_reduced_price_lunch  669 non-null    object
 5   lessons_completed        341 non-null    Int64
 6   total_minutes            341 non-null    float64
 7   math_score               640 non-null    Int64
dtypes: Int64(2), float64(1), int64(2), object(3)
memory usage: 50.6+ KB
```

[42]:
```
null_summary = final_csv.isnull().sum().reset_index()
null_summary.columns = ['Column', 'Null Count']
null_summary['Null Percentage'] = (null_summary['Null Count'] / len(final_csv))␣
↪* 100
null_summary
```

[42]:

	Column	Null Count	Null Percentage
0	student_id	0	0.000000
1	grade_level	0	0.000000
2	race_ethnicity	47	6.002554
3	gender	0	0.000000
4	free_reduced_price_lunch	114	14.559387
5	lessons_completed	442	56.449553
6	total_minutes	442	56.449553
7	math_score	143	18.263091

[43]:
```
plt.figure(figsize=(10, 6))
sb.heatmap(final_csv.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Data Heatmap")
plt.show()
```



[44]:
```
# Row count with all non-null values
all_non_null_count = final_csv.dropna(how='any')
all_non_null_count.shape[0]
```

[44]: 187

[45]:
```
# Row count with non-nulls in math columns
specific_columns = ['lessons_completed', 'total_minutes', 'math_score']
non_null_specific = final_csv[specific_columns].dropna(how='any')
non_null_specific.shape[0]
```

[45]: 231

[46]:
```
# Row count with null values from both merged files
all_null_specific = final_csv[specific_columns].isnull().all(axis=1)
all_null_specific.sum()
```

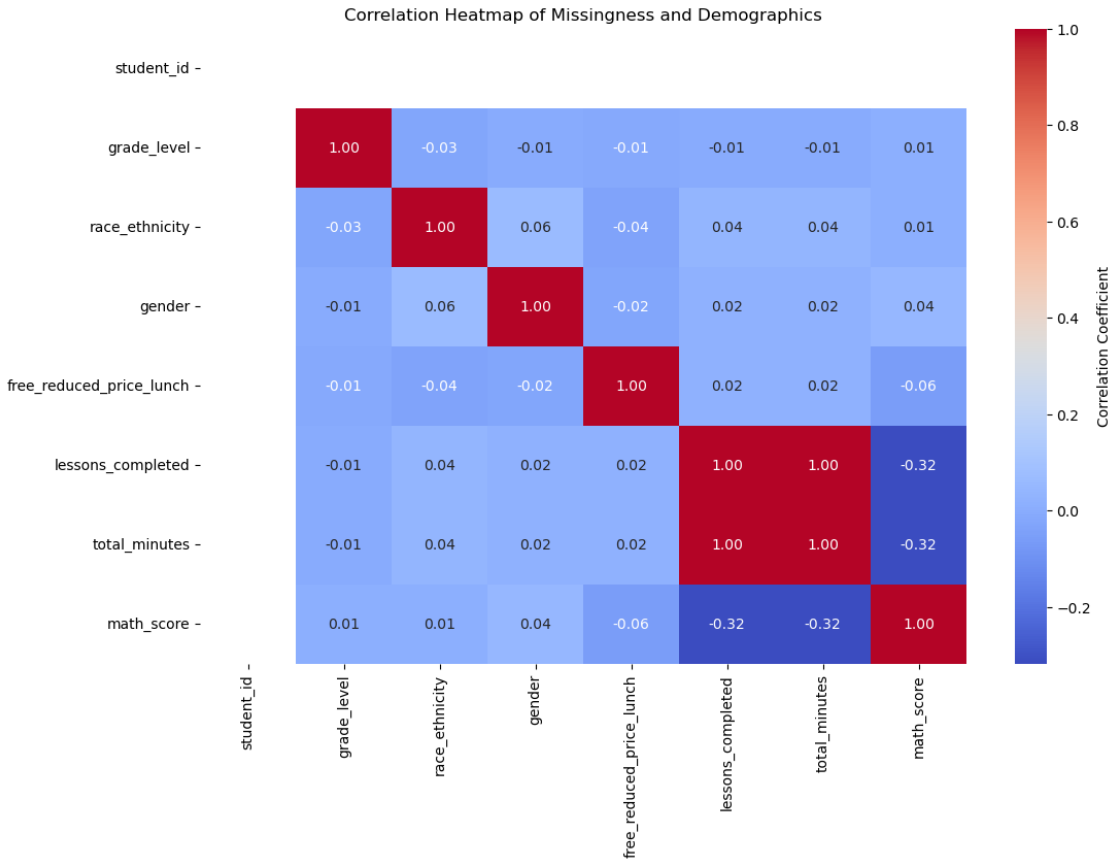
[46]: 33

11.1 Missing Data Compared to Demographics

[47]:
```
# Setup
demographics = ['grade_level', 'race_ethnicity', 'gender',␣
 ↪'free_reduced_price_lunch']
missingness_columns = ['lessons_completed', 'math_score']
```

[48]:
```
overview_df = final_csv.copy()

# Generate missingness indicators
missingness_df = overview_df.isnull().astype(int)

# Add demographic indicators
for col in demographics:
    missingness_df[col] = overview_df[col].astype('category').cat.codes

# Calculate correlations
correlation_matrix = missingness_df.corr()

# Plot heatmap
plt.figure(figsize=(12, 8))
sb.heatmap(
    correlation_matrix,
    annot=True,
    cmap="coolwarm",
```

```
    fmt=".2f",
    cbar_kws={'label': 'Correlation Coefficient'}
)
plt.title("Correlation Heatmap of Missingness and Demographics")
plt.show()
```



[49]:
```
# Check correlation between demographics and nulls.
binary_nulls_df = final_csv.copy()
for col in missingness_columns:
    binary_nulls_df[f'{col}_isnull'] = binary_nulls_df[col].isnull().astype(int)
binary_nulls_df.head()
```

[49]:
```
   student_id  grade_level          race_ethnicity gender  \
0     1254353            7                   White   Male
1     1254135            5                   White   Male
2     1254423            8  Black/African American   Male
3     1254598            8                   White   Male
4     1254562            6  Black/African American   Male

  free_reduced_price_lunch  lessons_completed  total_minutes  math_score  \
```

```
0                       No                  24      21.040558         289
1                      NaN                   1      53.124390        <NA>
2            Reduced Price                <NA>             NaN         614
3                     Free                <NA>             NaN         578
4                       No                <NA>             NaN         395

   lessons_completed_isnull  math_score_isnull
0                         0                  0
1                         0                  1
2                         1                  0
3                         1                  0
4                         1                  0
```

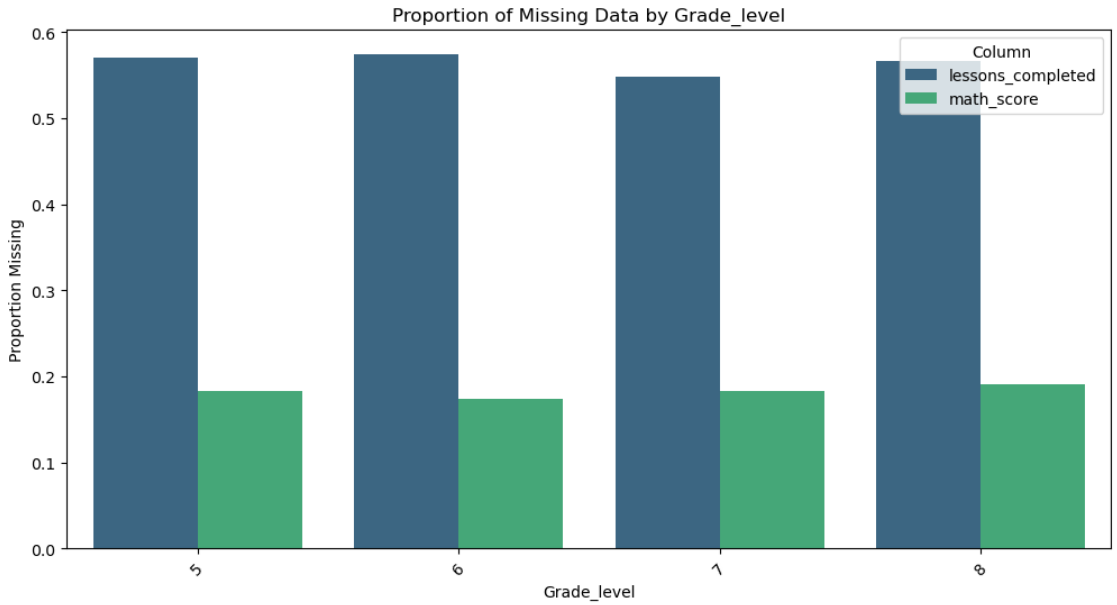
[50]:
```
for col in demographics:
    for missing_col in missingness_columns:
        crosstab = pd.crosstab(binary_nulls_df[col],␣
↪binary_nulls_df[f'{missing_col}_isnull'])
        print(f"Correlation between {col} and missingness in {missing_col}:")
        print(crosstab)
```

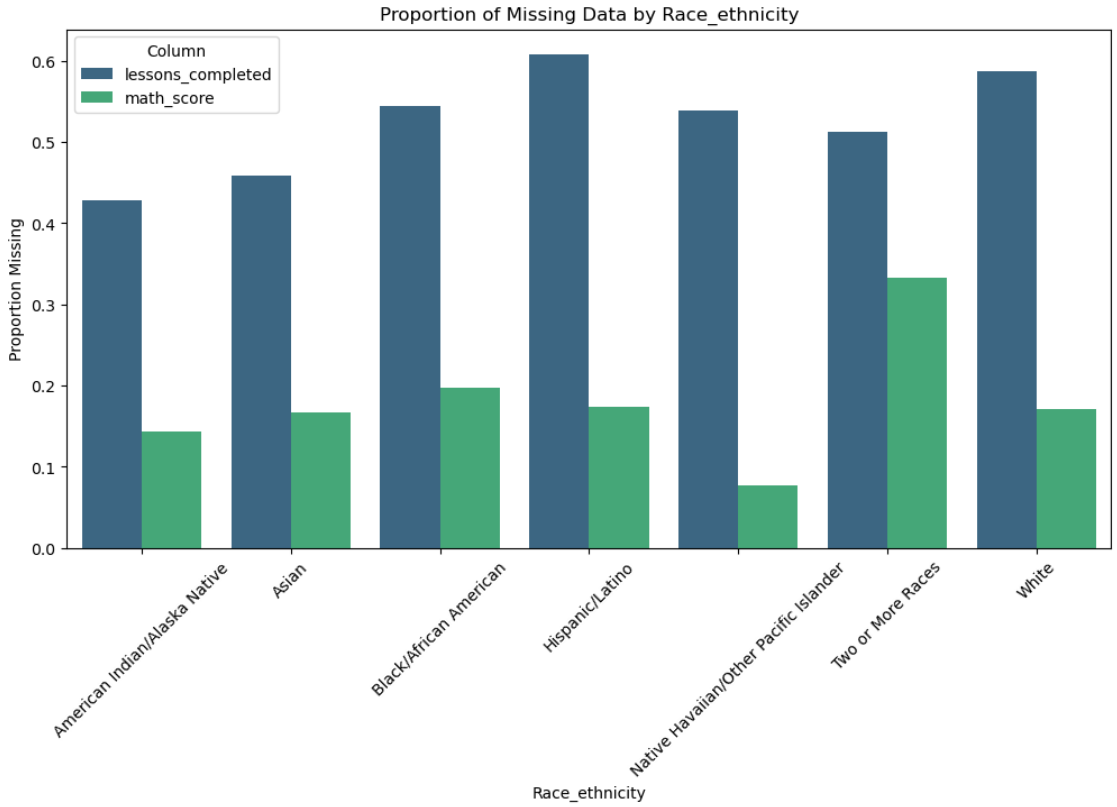
```
Correlation between grade_level and missingness in lessons_completed:
lessons_completed_isnull   0    1
grade_level
5                         80  106
6                         81  109
7                         89  108
8                         91  119
Correlation between grade_level and missingness in math_score:
math_score_isnull    0   1
grade_level
5                  152  34
6                  157  33
7                  161  36
8                  170  40
Correlation between race_ethnicity and missingness in lessons_completed:
lessons_completed_isnull                  0    1
race_ethnicity
American Indian/Alaska Native             4    3
Asian                                    13   11
Black/African American                  106  127
Hispanic/Latino                           9   14
Native Havaiian/Other Pacific Islander    6    7
Two or More Races                        19   20
White                                   164  233
Correlation between race_ethnicity and missingness in math_score:
math_score_isnull                         0   1
race_ethnicity
```

```
American Indian/Alaska Native             6   1
Asian                                    20   4
Black/African American                  187  46
Hispanic/Latino                          19   4
Native Havaiian/Other Pacific Islander   12   1
Two or More Races                        26  13
White                                   329  68
Correlation between gender and missingness in lessons_completed:
lessons_completed_isnull    0    1
gender
Female                    177  220
Male                      164  222
Correlation between gender and missingness in math_score:
math_score_isnull    0   1
gender
Female             331  66
Male               309  77
Correlation between free_reduced_price_lunch and missingness in
lessons_completed:
lessons_completed_isnull    0    1
free_reduced_price_lunch
Free                         83   98
No                          161  200
Reduced Price                49   78
Correlation between free_reduced_price_lunch and missingness in math_score:
math_score_isnull           0   1
free_reduced_price_lunch
Free                      148  33
No                        304  57
Reduced Price             103  24
```

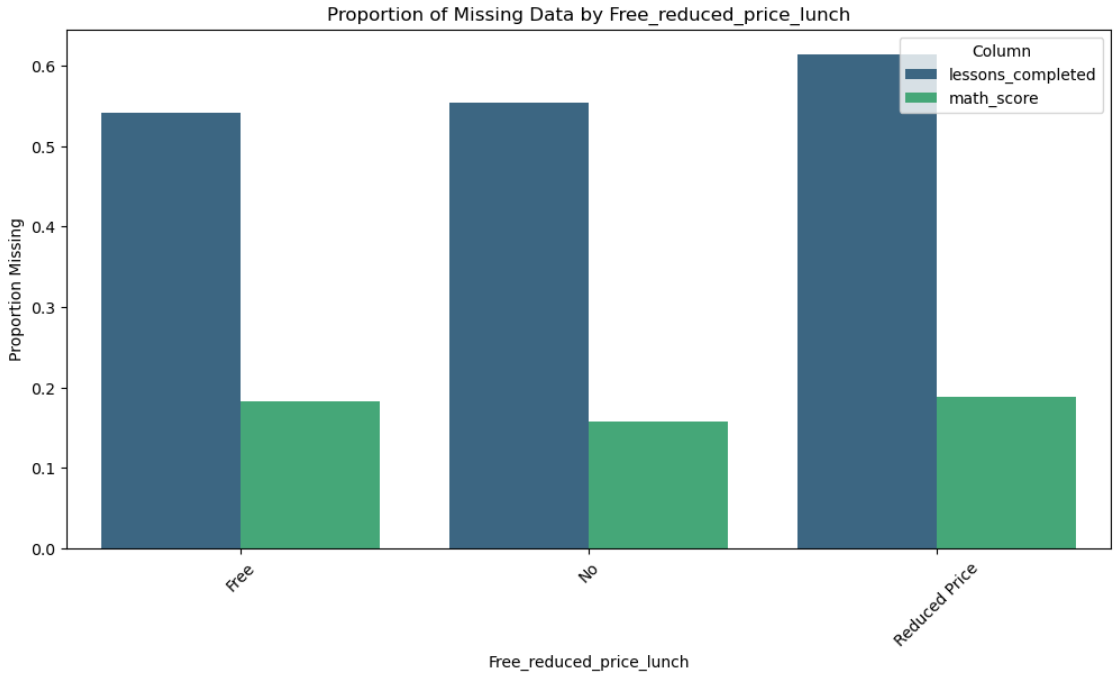
[51]:
```
for demo_col in demographics:
    # Calculate proportions for missing values
    proportions = (
        final_csv.groupby(demo_col)[missingness_columns]
        .apply(lambda x: x.isnull().mean())
        .reset_index()
        .melt(id_vars=demo_col, var_name='Column', value_name='Proportion␣
↪Missing')
    )

    # Plot the proportions
    plt.figure(figsize=(12, 6))
    sb.barplot(
        x=demo_col,
        y='Proportion Missing',
        hue='Column',
```

```
    data=proportions,
    palette="viridis"
)
plt.title(f"Proportion of Missing Data by {demo_col.capitalize()}")
plt.xlabel(demo_col.capitalize())
plt.ylabel("Proportion Missing")
plt.xticks(rotation=45)
plt.legend(title="Column")
plt.show()
```




Proportion of Missing Data by Race_ethnicity
Column
lessons_completed
math_score
Proportion Missing
0.0
0.1
0.2
0.3
0.4
0.5
0.6
American Indian/Alaska Native
Asian
Black/African American
Hispanic/Latino
Native Hawaiian/Other Pacific Islander
Two or More Races
White
Race_ethnicity


Proportion of Missing Data by Gender
Column
lessons_completed
math_score
Proportion Missing
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Female
Male
Gender



12 Math Data Analysis

[55]:
```
final_csv.describe()
```

[55]:

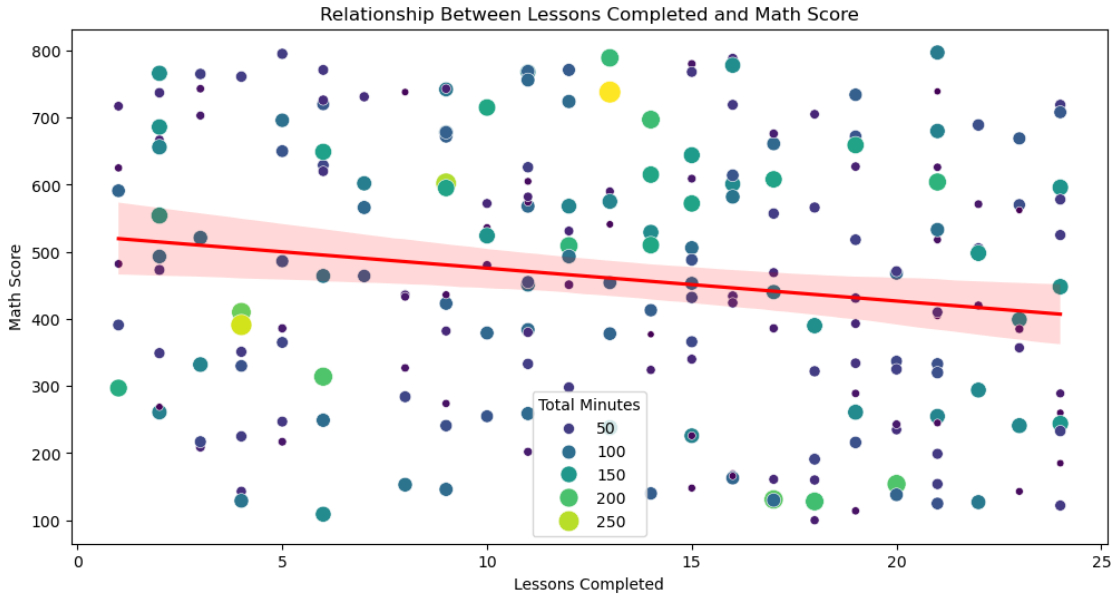
	student_id	grade_level	lessons_completed	total_minutes	math_score
count	7.830000e+02	783.000000	341.0	341.000000	640.0
mean	1.254464e+06	6.550447	12.86217	76.283296	457.909375
std	2.261769e+02	1.122746	6.781142	55.722120	201.147579
min	1.254073e+06	5.000000	1.0	0.167026	100.0
25%	1.254268e+06	6.000000	8.0	32.308333	277.0
50%	1.254464e+06	7.000000	13.0	65.030197	458.5
75%	1.254660e+06	8.000000	19.0	108.942186	632.5
max	1.254855e+06	8.000000	24.0	280.517175	798.0

[54]:
```
math_data = final_csv.copy()

# Fill NaN in lessons_completed and total_minutes with 0
math_data['lessons_completed'].fillna(0);
math_data['total_minutes'].fillna(0);

# Fill NaN in score with -1 to indicate no assessment taken
math_data['math_score'].fillna(-1);
```

[63]:
```
# Correlations
for col in ['lessons_completed', 'total_minutes']:
    print(f"Correlation with Score for {col}: {final_csv[col].
 ↪corr(final_csv['math_score'])}")
```

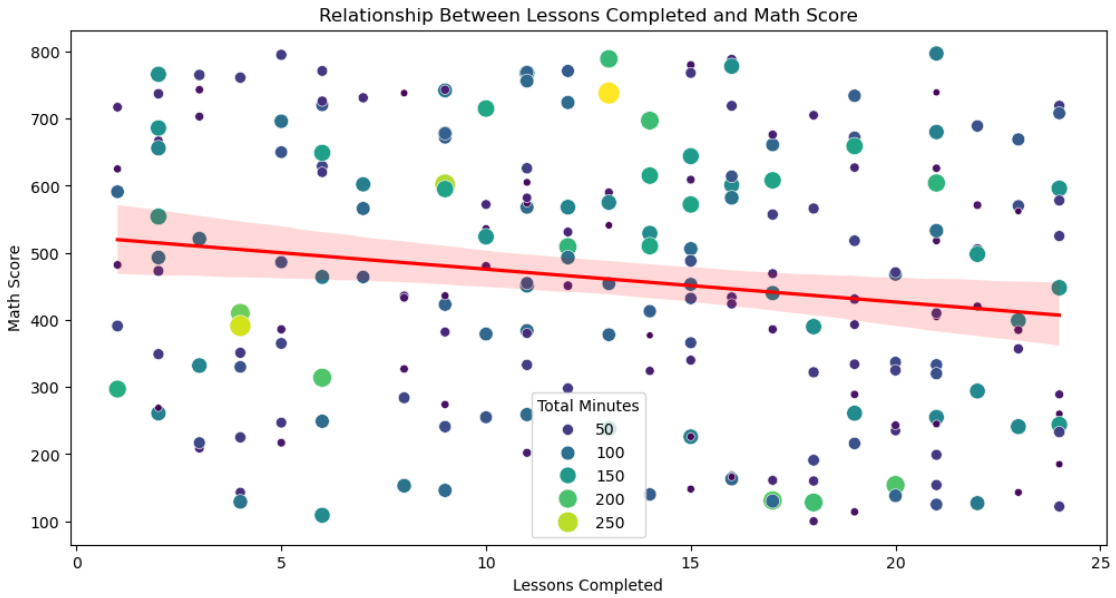
```
Correlation with Score for lessons_completed: -0.16664187473992423
Correlation with Score for total_minutes: 0.1072420936688609
```

[59]:
```
# Filter out rows where math_score = -1 (imputed value for missing scores)
filtered_df = final_csv[final_csv['math_score'] != -1]

# Scatter plot for lessons_completed vs math_score, colored by total_minutes
plt.figure(figsize=(12, 6))
sb.scatterplot(
    x='lessons_completed',
    y='math_score',
    hue='total_minutes',
    size='total_minutes',
    sizes=(20, 200),
    palette='viridis',
    data=filtered_df
)
sb.regplot(
    x='lessons_completed',
    y='math_score',
    scatter=False,
    data=filtered_df,
    color='red',
    line_kws={'label': 'Regression Line'}
)
plt.title("Relationship Between Lessons Completed and Math Score")
plt.xlabel("Lessons Completed")
plt.ylabel("Math Score")
plt.legend(title="Total Minutes")
plt.show()
```

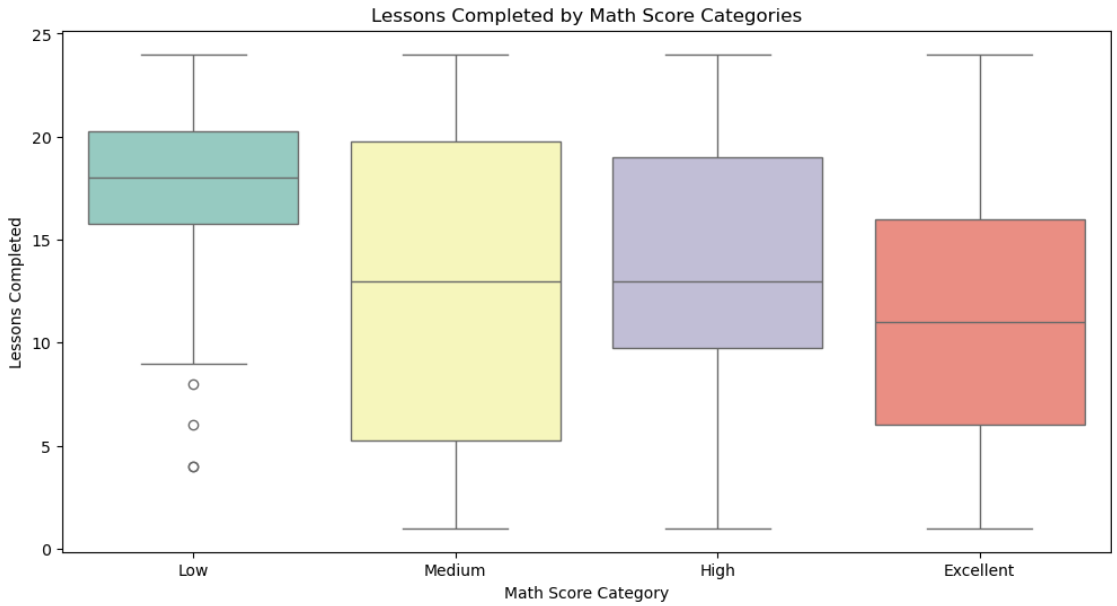


[57]:
```
# Same as above but using math_data with fillna columns with 0 for usage and -1␣
 ↪for math_score column

# Filter out rows where math_score = -1 (imputed value for missing scores)
filtered_df_filled = math_data[math_data['math_score'] != -1]

# Scatter plot for lessons_completed vs math_score, colored by total_minutes
plt.figure(figsize=(12, 6))
sb.scatterplot(
    x='lessons_completed',
    y='math_score',
    hue='total_minutes',
    size='total_minutes',
    sizes=(20, 200),
    palette='viridis',
    data=filtered_df_filled
)
sb.regplot(
    x='lessons_completed',
    y='math_score',
    scatter=False,
    data=filtered_df_filled,
    color='red',
    line_kws={'label': 'Regression Line'}
)
plt.title("Relationship Between Lessons Completed and Math Score")
plt.xlabel("Lessons Completed")
```

```
plt.ylabel("Math Score")
plt.legend(title="Total Minutes")
plt.show()
```

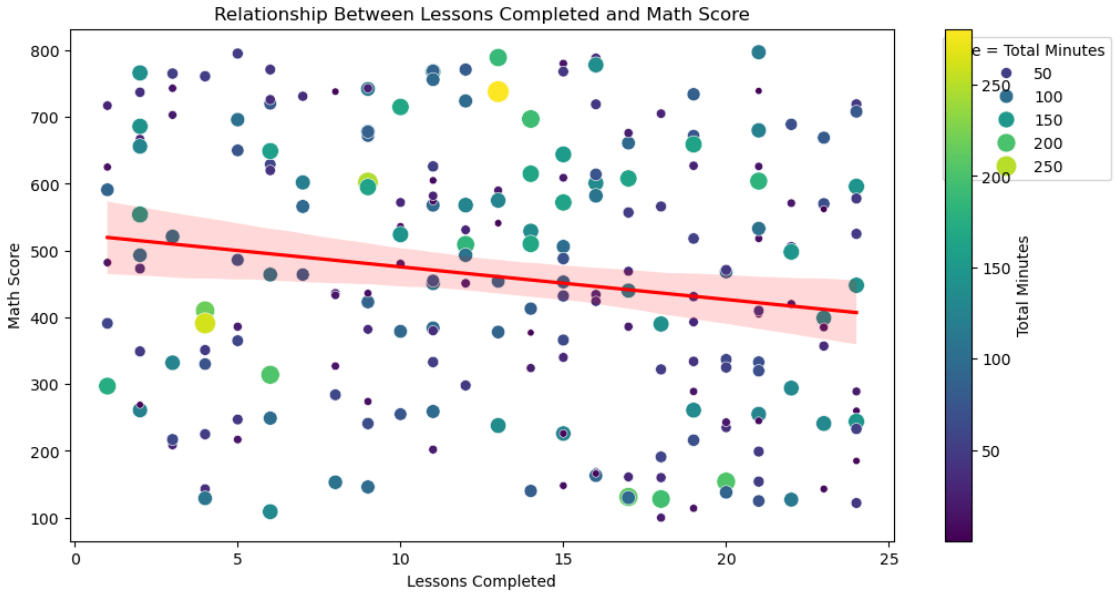


[93]:
```
box_df = filtered_df.copy()
# Create categories for math_score
box_df['score_category'] = pd.cut(
    box_df['math_score'],
    bins=[-1, 200, 400, 600, 800],
    labels=['Low', 'Medium', 'High', 'Excellent']
)

# Box plot for lessons_completed by math score categories
plt.figure(figsize=(12, 6))
sb.boxplot(
    x='score_category',
    y='lessons_completed',
    hue='score_category',
    dodge=False,
    data=box_df,
    palette='Set3'
)
plt.title("Lessons Completed by Math Score Categories")
plt.xlabel("Math Score Category")
plt.ylabel("Lessons Completed")
plt.show()
```



[61]:
```
from matplotlib.colors import Normalize
import matplotlib.colorbar as mcb

# Define a normalization for the hue to reflect the true range
hue_norm = Normalize(vmin=filtered_df_filled['total_minutes'].min(),
                     vmax=filtered_df_filled['total_minutes'].max())

# Create the scatter plot
plt.figure(figsize=(12, 6))
scatter = sb.scatterplot(
    x='lessons_completed',
    y='math_score',
    hue='total_minutes',
    size='total_minutes',
    sizes=(20, 200),
    palette='viridis',
    hue_norm=hue_norm,
    data=filtered_df_filled
)

# Add regression line
sb.regplot(
    x='lessons_completed',
    y='math_score',
    scatter=False,
    data=filtered_df_filled,
    color='red',
```

```
    line_kws={'label': 'Regression Line'}
)

# Add a continuous color bar
sm = plt.cm.ScalarMappable(cmap='viridis', norm=hue_norm)
sm.set_array([])
cbar = plt.colorbar(sm, ax=plt.gca())
cbar.set_label("Total Minutes")

plt.title("Relationship Between Lessons Completed and Math Score")
plt.xlabel("Lessons Completed")
plt.ylabel("Math Score")
plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1), title="Size = Total␣
 ↪Minutes")
plt.show()
```



[90]:
```
filtered_df.head()
```

[90]:

	student_id	grade_level	race_ethnicity	gender	free_reduced_price_lunch	lessons_completed	total_minutes	math_score
0	1254353	7	White	Male	No	24	21.040558	289
2	1254423	8	Black/African American	Male				
3	1254598	8	White	Male				
4	1254562	6	Black/African American	Male				
5	1254213	8	White	Male				

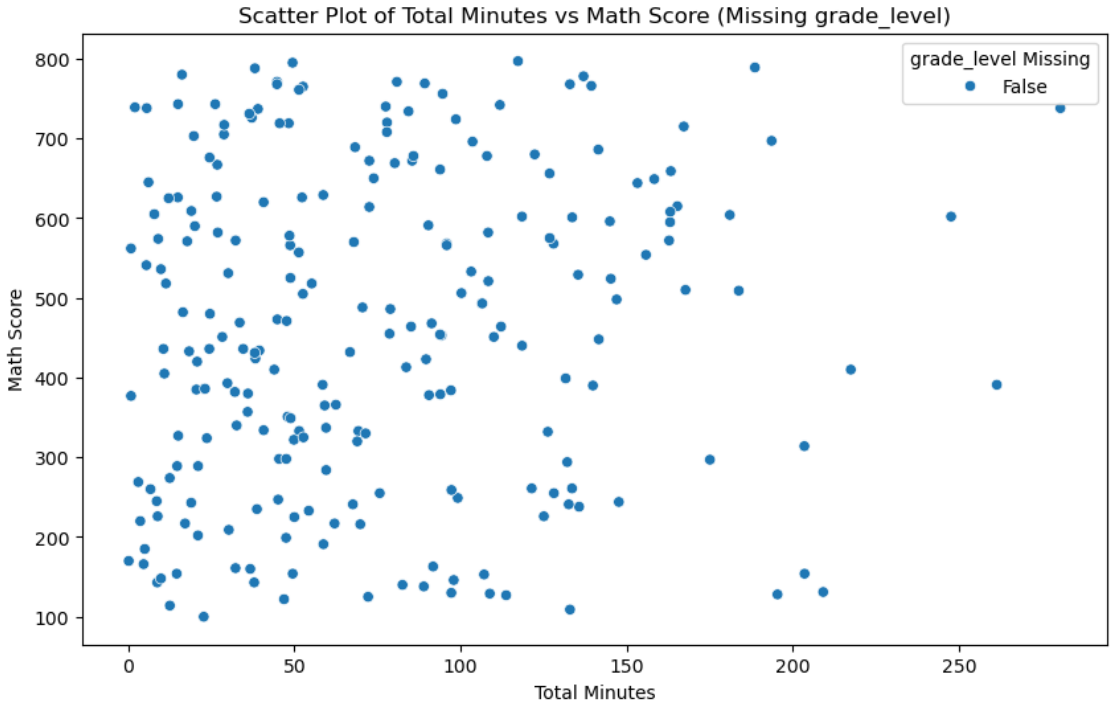
```
2            Reduced Price                <NA>            NaN         614
3                     Free                <NA>            NaN         578
4                       No                <NA>            NaN         395
5                       No                  21      11.374870         518

   lessons_completed_isnull  total_minutes_isnull  math_score_isnull  \
0                         0                     0                  0
2                         1                     1                  0
3                         1                     1                  0
4                         1                     1                  0
5                         0                     0                  0

  score_category
0            NaN
2            NaN
3            NaN
4            NaN
5            NaN
```

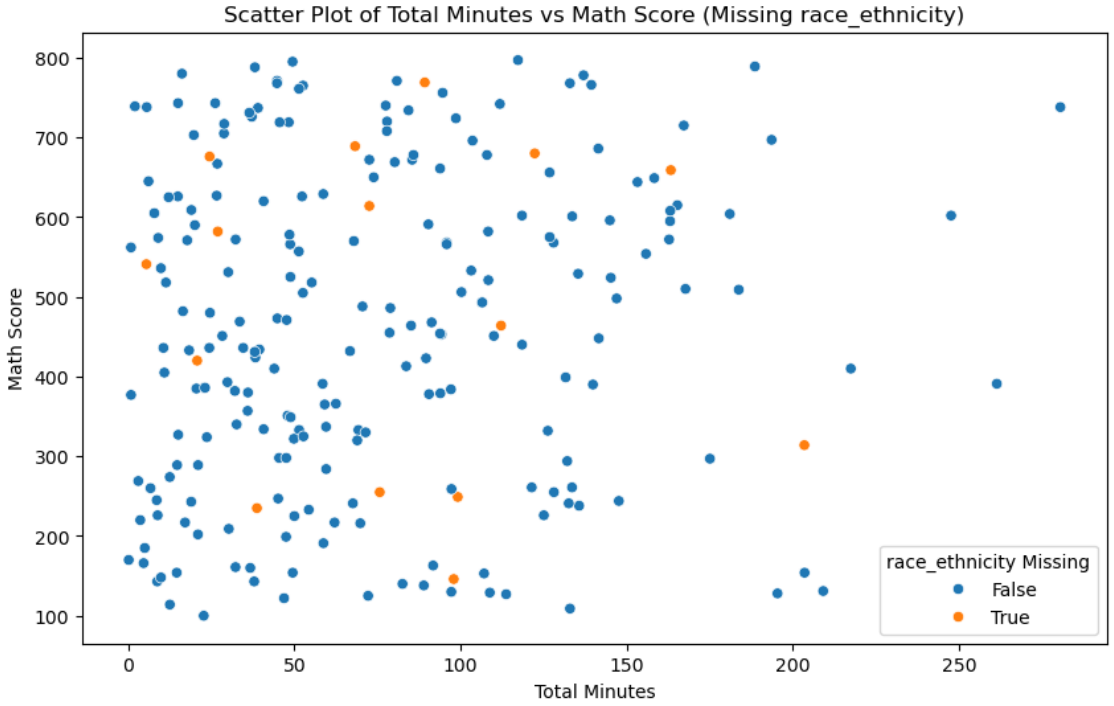
13 End of Workbook

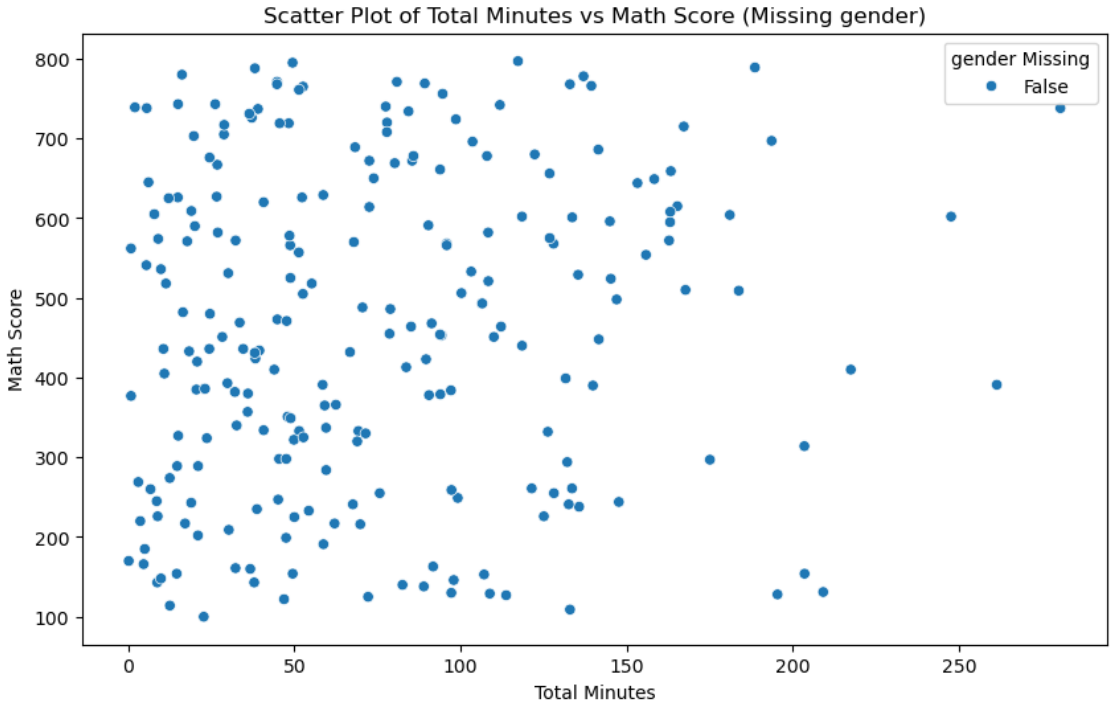
The below portion of this workbook is a drafted work and scratch work. It does not relate to the associated report. I'm leaving it as is for personal reference or possible further study in the future. Please disregard.

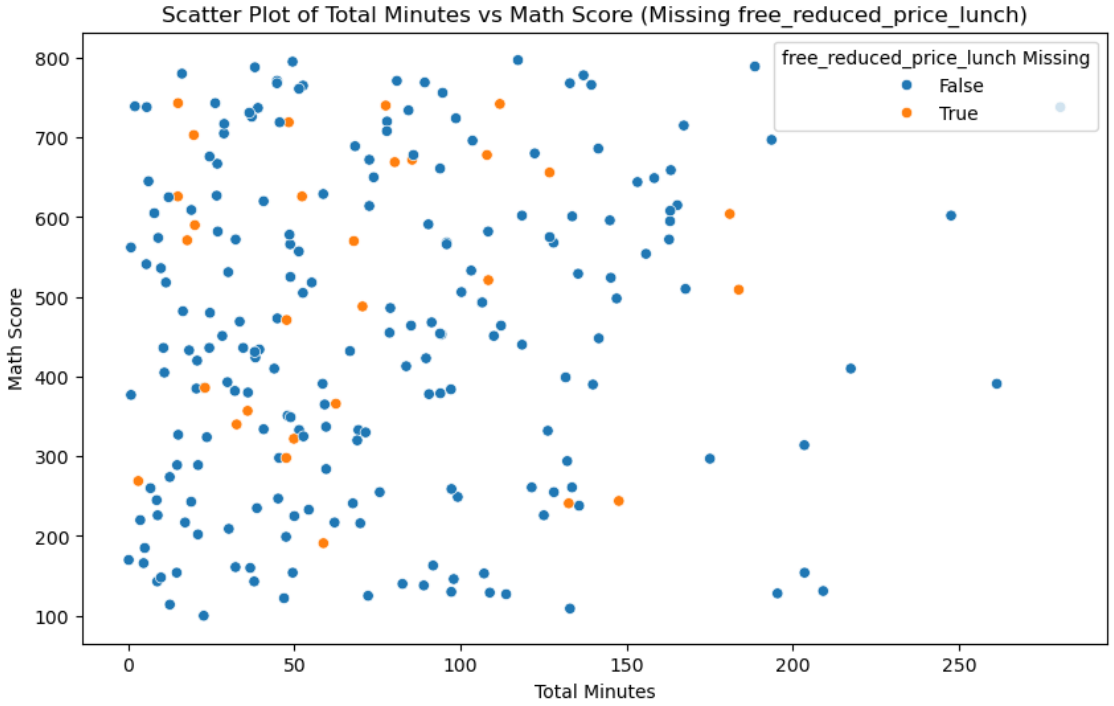
---

[72]:
```
for col in ['grade_level', 'race_ethnicity', 'gender',␣
 ↪'free_reduced_price_lunch']:
    plt.figure(figsize=(10, 6))
    sb.scatterplot(
        x='total_minutes',
        y='math_score',
        hue=final_csv[col].isnull(),
        data=final_csv
    )
    plt.title(f"Scatter Plot of Total Minutes vs Math Score (Missing {col})")
    plt.xlabel("Total Minutes")
    plt.ylabel("Math Score")
    plt.legend(title=f"{col} Missing")
    plt.show()
```


Scatter Plot of Total Minutes vs Math Score (Missing grade_level)
grade_level Missing
False
Math Score
Total Minutes
800
700
600
500
400
300
200
100
0
50
100
150
200
250


Scatter Plot of Total Minutes vs Math Score (Missing race_ethnicity)
race_ethnicity Missing
False
True
Math Score
Total Minutes
800
700
600
500
400
300
200
100
0
50
100
150
200
250


Scatter Plot of Total Minutes vs Math Score (Missing gender)
gender Missing
False
Math Score
Total Minutes
800
700
600
500
400
300
200
100
0
50
100
150
200
250


Scatter Plot of Total Minutes vs Math Score (Missing free_reduced_price_lunch)
free_reduced_price_lunch Missing
False
True
Math Score
Total Minutes
800
700
600
500
400
300
200
100
0
50
100
150
200
250

14 Benchmark Data

14.1 Handle Usage Data

[32]:
```
# Put Benchmark columns into a list
benchmark_columns = ['benchmark_1_level', 'benchmark_2_level',␣
 ↪'benchmark_3_level', 'benchmark_4_level']

# Check Usage Benchmarks
for col in benchmark_columns:
    print(f"Unique values in {col}: {usage_data[col].unique()}")
```

```
Unique values in benchmark_1_level: ['level 2' 'level 3' 'level 1' 'level4']
Unique values in benchmark_2_level: ['level 1' 'level 2' 'level 3' 'level4']
Unique values in benchmark_3_level: ['level 3' 'level4' 'level 1' 'level 2']
Unique values in benchmark_4_level: ['level 2' 'level 1' 'level4' 'level 3']
```

Messy Data in Benchmark Columns: All level values have a space between text and number except for "level4" in each column.

[33]:
```
# Replace "level4" with "level 4" in the specified columns
for column in benchmark_columns:
    usage_data[column] = usage_data[column].replace('level4', 'level 4')

usage_data.head(5)
```

[33]:

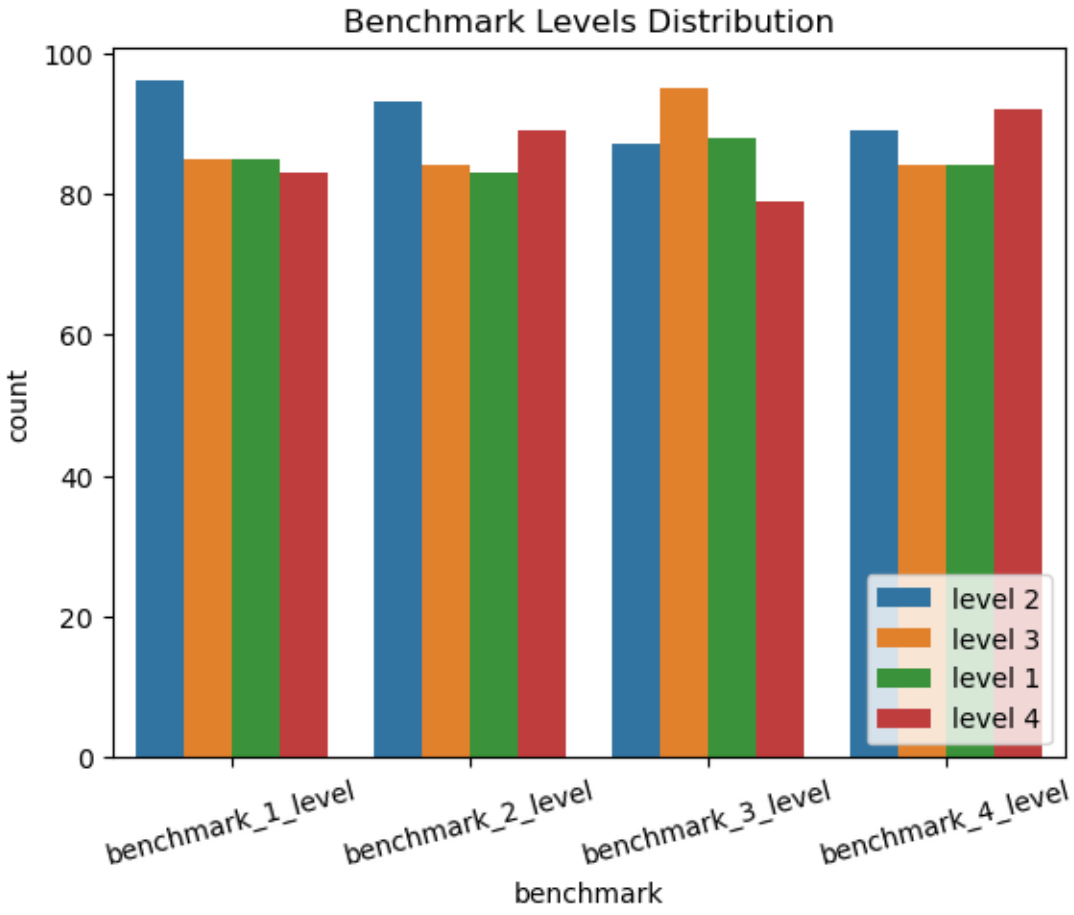
	student_id	lessons_completed	benchmark_1_level	benchmark_2_level	benchmark_3_level	benchmark_4_level	total_minutes
0	1254110	13	level 2	level 1	level 3	level 2	47.808670
1	1254113	14	level 2	level 1	level 4	level 1	156.792335
2	1254288	16	level 3	level 1	level 3	level 1	38.135959
3	1254095	15	level 3	level 1	level 1	level 1	18.257427
4	1254250	1	level 2	level 1	level 2	level 2	16.479016

[17]:
```
# Benchmark level distribution
melted = usage_data.melt(id_vars=['student_id'],
                         value_vars=['benchmark_1_level', 'benchmark_2_level',␣
 ↪'benchmark_3_level', 'benchmark_4_level'],
                         var_name='benchmark',
                         value_name='level')
melted.head(10)
```

[17]:

	student_id	benchmark	level
0	1254110	benchmark_1_level	level 2
1	1254113	benchmark_1_level	level 2
2	1254288	benchmark_1_level	level 3
3	1254095	benchmark_1_level	level 3
4	1254250	benchmark_1_level	level 2
5	1254177	benchmark_1_level	level 1
6	1254277	benchmark_1_level	level 3
7	1254078	benchmark_1_level	level 2
8	1254165	benchmark_1_level	level 2
9	1254381	benchmark_1_level	level 3

[18]:
```
sb.countplot(data=melted, x='benchmark', hue='level')
plt.title("Benchmark Levels Distribution")
plt.xticks(rotation=15)
plt.legend(loc = 4)
plt.show()
```



[48]:
```
for col in ['benchmark_1_level', 'benchmark_2_level', 'benchmark_3_level',␣
 ↪'benchmark_4_level']:
    print(usage_data[col].value_counts())
```

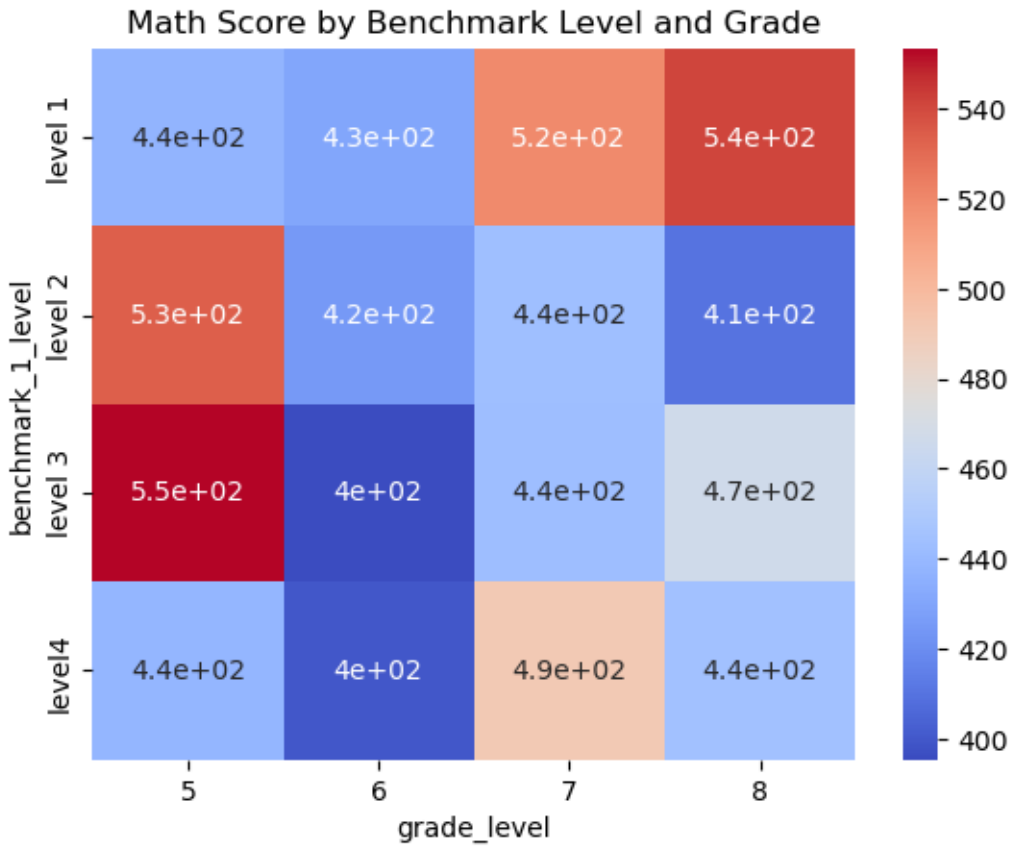
```
benchmark_1_level
level 2    96
level 3    85
level 1    85
level4     83
Name: count, dtype: int64
benchmark_2_level
level 2    93
level4     89
level 3    84
level 1    83
Name: count, dtype: int64
benchmark_3_level
level 3    95
level 1    88
level 2    87
level4     79
Name: count, dtype: int64
benchmark_4_level
level4     92
level 2    89
level 1    84
level 3    84
Name: count, dtype: int64
```

15 Scratch Pad

[51]:
```
for col in ['lessons_completed']:
    print(f"Correlation with total_minutes for {col}: {usage_data[col].
 ↪corr(usage_data['total_minutes'])}")
```

```
Correlation with total_minutes for lessons_completed: -0.04444975594260643
```

[54]:
```
heatmap_data = merged_df.groupby(['benchmark_1_level', 'grade_level'])['score'].
 ↪mean().unstack()
sb.heatmap(heatmap_data, annot=True, cmap='coolwarm')
plt.title("Math Score by Benchmark Level and Grade")
plt.show()
```


Math Score by Benchmark Level and Grade
benchmark_1_level
level 1
level 2
level 3
level4
4.4e+02
4.3e+02
5.2e+02
5.4e+02
5.3e+02
4.2e+02
4.4e+02
4.1e+02
5.5e+02
4e+02
4.4e+02
4.7e+02
4.4e+02
4e+02
4.9e+02
4.4e+02
5
6
7
8
grade_level
540
520
500
480
460
440
420
400

[ ]: