Data Analysis Assessment 2025

Candidate: Hulya Alpagu

Email: hulyalpagu@gmail.com

Phone: + 1 (862) 2977302

Project Overview

This assessment is based on four interconnected data tables from a simulated digital learning platform. The objective is to evaluate the effectiveness of the learning product by cleaning and analyzing the data, uncovering actionable insights using Python, and communicating those insights through impactful visualizations created in Power BI. The final outcome should guide data-driven decisions for product improvement.

Data Summary

Here's a quick overview of the four main data tables:

- **student_table**
  - Contains student performance metrics.
  - 1000 students with columns like student_id, final_grade, total_time_spent, total_resources_completed, mastery_level, badges_earned, dropout_flag
- **asset_table**
  - Logs student activity with learning assets.
  - 80,503 resource interactions records, including student_id, timestamp, resource_type, resource_id
- **event_table**
  - Records actions taken by students on resources.
  - 80,503 event records, including student_id, resource_id, action_type, time_spent_seconds
- **performance_table**
  - Contains quiz/test performance data.
  - 11,540 quiz/test results, including student_id, resource_type, resource_id, score, difficulty

Data Cleaning, Preparation, and Export in Python for Power BI Visualization

After reviewing the project objectives and exploring the structure of each table, the next step was to clean and prepare the data for merging. This included checking for missing

values, removing duplicates, validating data types, and optimizing the dataset for analysis.

1. **Missing Values:**
   No missing values found in any of the four tables.

```
In [28]:
#Check missing data
print("Missing values:")
print(student_df.isnull().sum())
print(asset_df.isnull().sum())
print(event_df.isnull().sum())
print(performance_df.isnull().sum())
```

```
Missing values:
student_id                    0
final_grade                   0
total_time_spent              0
total_resources_completed     0
mastery_level                 0
badges_earned                 0
dropout_flag                  0
dtype: int64
student_id        0
timestamp         0
resource_type     0
resource_id       0
dtype: int64
student_id             0
resource_id            0
action_type            0
time_spent_seconds     0
dtype: int64
student_id        0
resource_type     0
resource_id       0
score             0
difficulty        0
dtype: int64
```

2. **Duplicates Found:**
   After verifying the structure of each table, I checked for duplicate rows to ensure data quality. The results were as follows:

```
In [29]:
#Check Dublicate
print(f"Student Table Duplicates: {student_df.duplicated().sum()}")
print(f"Asset Table Duplicates: {asset_df.duplicated().sum()}")
print(f"Event Table Duplicates: {event_df.duplicated().sum()}")
print(f"Performance Table Duplicates: {performance_df.duplicated().sum()}")
```

```
Student Table Duplicates: 0
Asset Table Duplicates: 1
Event Table Duplicates: 46
Performance Table Duplicates: 1
```

   All identified duplicates were removed using drop_duplicates() in Python to

maintain data accuracy and prevent skewed results in the analysis.

```
In [31]:
# Remove duplicates
asset_df = asset_df.drop_duplicates()
event_df = event_df.drop_duplicates()
performance_df = performance_df.drop_duplicates()
```

3. **Data Types**

   All data types were correct. No conversions were needed.

```
In [54]:
print("Student Table Types:\n", student_df.dtypes)
print("\nAsset Table Types:\n", asset_df.dtypes)
print("\nEvent Table Types:\n", event_df.dtypes)
print("\nPerformance Table Types:\n", performance_df.dtypes)
```

```
Student Table Types:
 student_id                    int64
final_grade                   float64
total_time_spent                int64
total_resources_completed       int64
mastery_level                  object
badges_earned                   int64
dropout_flag                    int64
dtype: object

Asset Table Types:
 student_id                int64
timestamp         datetime64[ns]
resource_type             object
resource_id               object
dtype: object

Event Table Types:
 student_id              int64
resource_id            object
action_type            object
time_spent_seconds      int64
dtype: object

Performance Table Types:
 student_id         int64
resource_type     object
resource_id       object
score              int64
difficulty         int64
dtype: object
```

4. **Feature Reduction**

   I observed that both the asset_table and performance_table contain a column named resource_type. However, in the performance_table, this column only contains a single value: 'quiz', while the asset_table includes all resource types

such as 'reading', 'survey', 'video', 'game', 'quiz', 'forum', and 'assignment'.

In [40]:
```
# Show unique values in 'resource_type' column for both asset_table and performance_table
unique_asset_types = asset_df['resource_type'].unique()
print("resource_type column of asset_df:", unique_asset_types)

unique_performance_types = performance_df['resource_type'].unique()
print("resource_type column of performance_df:", unique_performance_types)
```

```
resource_type column of asset_df: ['reading' 'survey' 'video' 'game' 'quiz' 'forum' 'assignment']
resource_type column of performance_df: ['quiz']
```

Since the resource_type column in the performance_table is redundant and does not provide additional information beyond what is already available in the asset_table, I decided to remove it to avoid duplication and keep the dataset clean.

In [24]:
```
# Drop 'resource_type' column from performance_df
performance_df.drop(columns=['resource_type'], inplace=True)
print(performance_df.columns)  # Confirm the column is removed
```

```
Index(['student_id', 'resource_id', 'score', 'difficulty'], dtype='object')
```

5. **Normalization Values**

I converted the numeric values in the dropout_flag column of the student_table into descriptive labels:

- 0 → **"Active Student"**
- 1 → **"Dropped Out Student"**

This transformation makes the data easier to understand for both technical and non-technical stakeholders and is especially helpful when creating visualizations, reports, or dashboards.

```
In [87]: # Show unique values in 'dropout_flag' column from student_table
unique_asset_types = student_df['dropout_flag'].unique()
print("resource_type column of asset_df:", unique_asset_types)

resource_type column of asset_df: [0 1]
```

```
In [88]: # Convert dropout_flag to descriptive labels in student_df
student_df['dropout_flag'] = student_df['dropout_flag'].map({
    0: 'Active Student',
    1: 'Dropped Out Student'
})

# Show unique values after converting dropout_flag in student_table
unique_asset_types = student_df['dropout_flag'].unique()
print("resource_type column of asset_df:", unique_asset_types)

resource_type column of asset_df: ['Active Student' 'Dropped Out Student']
```

Merging the Tables

After completing the data cleaning process, the dataset was ready for integration. The goal was to combine all four tables into a single comprehensive dataset suitable for analysis and Power BI visualization.
Before merging, here's a summary of the cleaned table shapes:

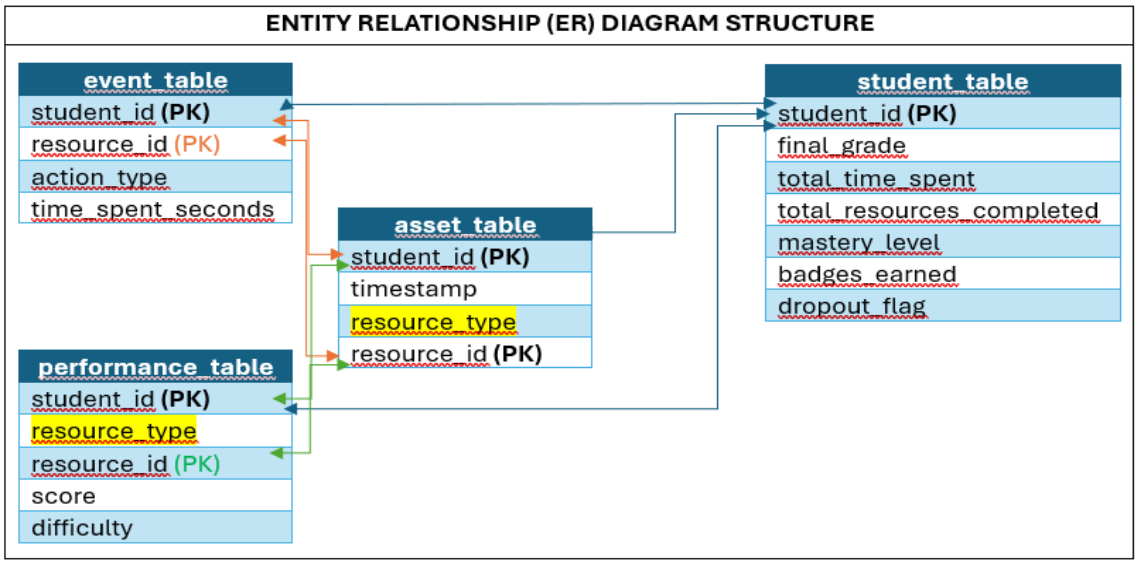
```
In [60]: print("Cleaned DataFrame shapes:")
print("Student Table:",student_df.shape)
print("Asset Table Types:",asset_df.shape)
print("Event Table Types:",event_df.shape)
print("Performance Table:",performance_df.shape)

Cleaned DataFrame shapes:
Student Table: (1000, 7)
Asset Table Types: (80502, 4)
Event Table Types: (80457, 4)
Performance Table: (11538, 5)
```

1.Merging Strategy Based on ER Diagram

To build a unified dataset, the following key relationships—illustrated in the ER diagram—were used:

- event_table was merged with asset_table on both student_id and resource_id
- The result was merged with performance_table on student_id and resource_id
- Finally, the combined dataset was merged with student_table on student_id

This approach preserved the granularity of each interaction while linking it to performance and student-level metrics.



In [90]:

```python
#MERGE THE TABLES
# Step 1: Merge event_table with asset_table on student_id and resource_id
event_asset_df = pd.merge(event_df, asset_df, on=['student_id', 'resource_id'], how='left')

# Step 2: Merge with performance_table on student_id and resource_id
event_asset_perf_df = pd.merge(event_asset_df, performance_df, on=['student_id', 'resource_id'], how='left')

# Step 3: Merge with student_table on student_id
final_df = pd.merge(event_asset_perf_df, student_df, on='student_id', how='left')

# Show the final dataframe shape and preview
print("\nFinal dataset shape:", final_df.shape)
print("\nDataset Info")
final_df.info()
```

```
Final dataset shape: (85297, 14)

Dataset Info
<class 'pandas.core.frame.DataFrame'>
Int64Index: 85297 entries, 0 to 85296
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   student_id                 85297 non-null  int64
 1   resource_id                85297 non-null  object
 2   action_type                85297 non-null  object
 3   time_spent_seconds         85297 non-null  int64
 4   timestamp                  85297 non-null  datetime64[ns]
 5   resource_type              85297 non-null  object
 6   score                      13176 non-null  float64
 7   difficulty                 13176 non-null  float64
 8   final_grade                85297 non-null  float64
 9   total_time_spent           85297 non-null  int64
 10  total_resources_completed  85297 non-null  int64
 11  mastery_level              85297 non-null  object
 12  badges_earned              85297 non-null  int64
 13  dropout_flag               85297 non-null  object
dtypes: datetime64[ns](1), float64(3), int64(5), object(5)
memory usage: 9.8+ MB
```

The dataset is fully cleaned, merged, and ready for analysis and visualization in Power BI.

Summary Analysis

The descriptive statistics above include metrics such as mean, standard deviation, minimum, maximum, and quartiles. These values help explain the distribution and variability of key variables such as time spent, quiz scores, final grades, and badges earned, offering a clearer view of overall student performance and engagement.

In [142]:
```
# 1. Summary statistics of all numeric columns
summary_stats = final_df.describe()
print("Summary Statistics:\n", summary_stats)
```

```
Summary Statistics:
          student_id  time_spent_seconds         score     difficulty  \
count  85297.000000        85297.000000  13176.000000  13176.000000
mean     511.666331          274.558015     50.431466      2.994460
std      292.617853          298.834275     29.153510      1.400314
min        1.000000            5.000000      0.000000      1.000000
25%      259.000000           59.000000     25.000000      2.000000
50%      512.000000          177.000000     51.000000      3.000000
75%      789.000000          387.000000     76.000000      4.000000
max     1000.000000         3553.000000    100.000000      5.000000

        final_grade  total_time_spent  total_resources_completed  \
count  85297.000000      85297.000000               85297.000000
mean      71.001415      24753.199456                 108.513910
std       20.319274       6077.158150                 192.026428
min       11.630000      11753.000000                  22.000000
25%       58.830000      20215.000000                  38.000000
50%       72.970000      24343.000000                  46.000000
75%       87.240000      28698.000000                  54.000000
max       99.970000      39286.000000                 757.000000

       badges_earned
count   85297.000000
mean        4.990058
std         3.213492
min         0.000000
25%         2.000000
50%         5.000000
75%         8.000000
max        10.000000
```

Exporting the Dataset to Power BI

After preparing the final dataset, I exported it to a CSV file using Python to enable visualization in Power BI. This step ensures the data is ready for interactive reporting and dashboard creation.

In [94]:
```
# Export to CSV for Power BI
final_df.to_csv("final_cleaned_dataset.csv", index=False)
```

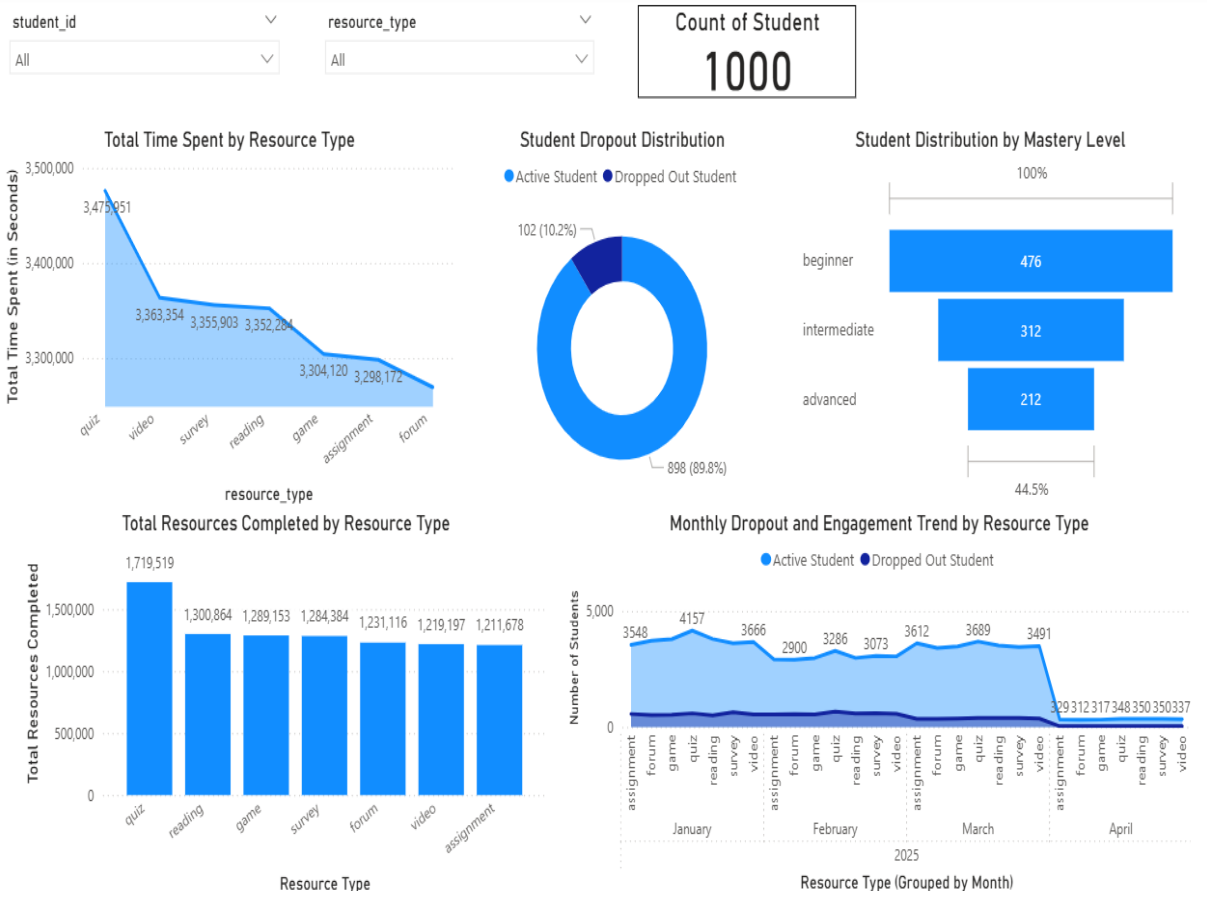
The exported file can now be imported into Power BI for visual exploration of student performance, engagement, and quiz outcomes

Data Visualization

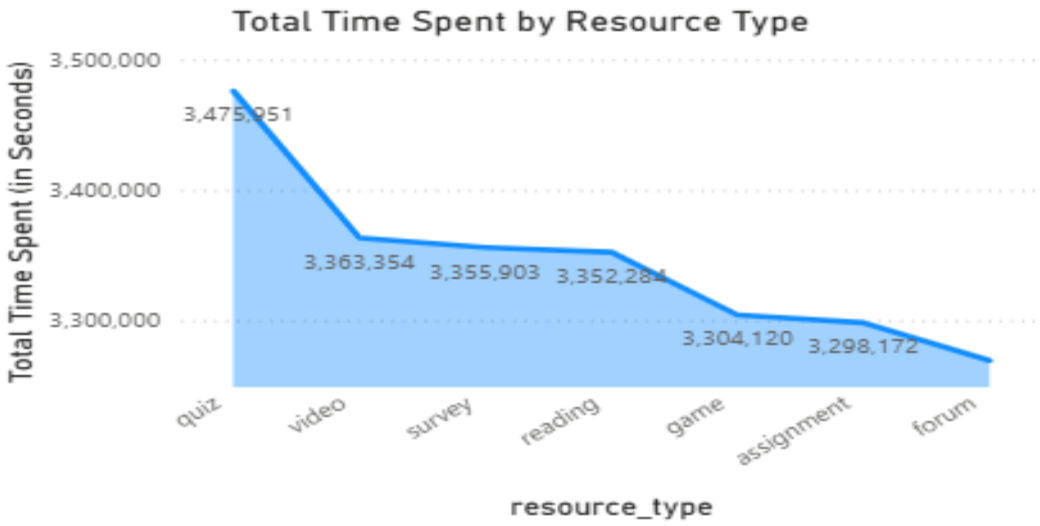
I created an interactive Power BI dashboard with three pages: the 'General Overview' page summarizes overall engagement, the 'Final Grade Analysis' page explores student grade patterns, and the 'Quiz Score Analysis' page focuses on quiz performance. Each section highlights a different aspect of student behavior and learning outcomes

1. General Overview

This page provides a high-level summary of platform activity and student engagement. It highlights how learners interact with various resource types and shows engagement patterns, mastery levels, and dropout trends.

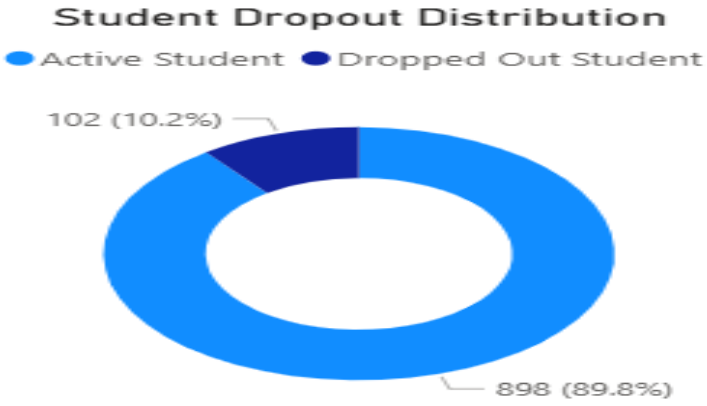


- **Total Time Spent by Resource Type**



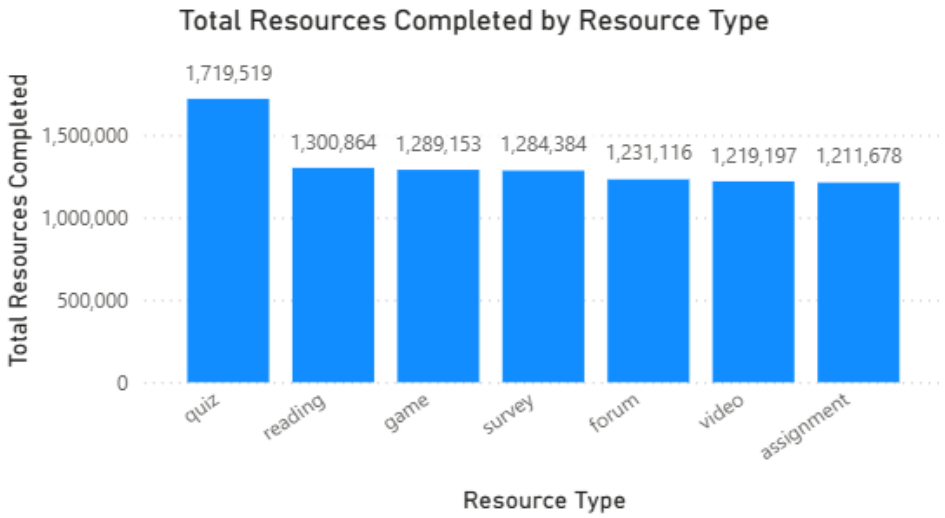
This area chart shows the total time students spent on each resource type. Quizzes had the highest time spent, followed by videos, surveys, and readings. Forum and assignments were the lowest, suggesting students spent the most time on interactive or evaluative content.

- **Student Dropout Distribution**



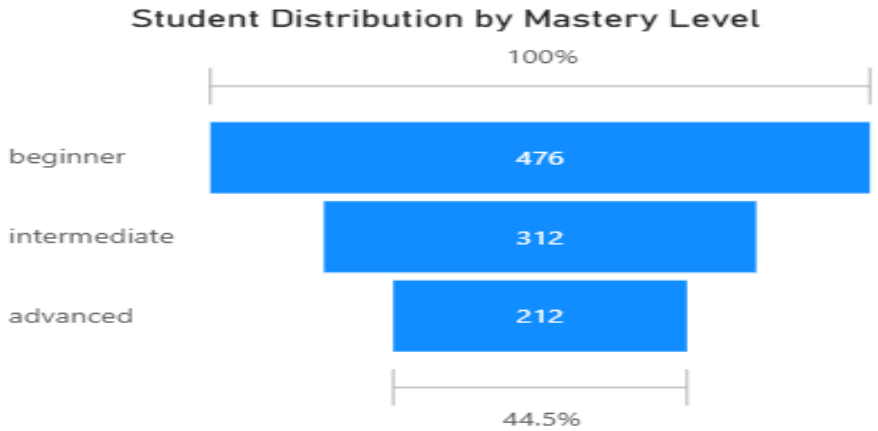
This chart shows the distribution of student dropout status. About **89.8%** of students remained active, while **10.2%** dropped out. This indicates a relatively high retention rate on the platform.

- **Total Resources Completed by Resource Type**



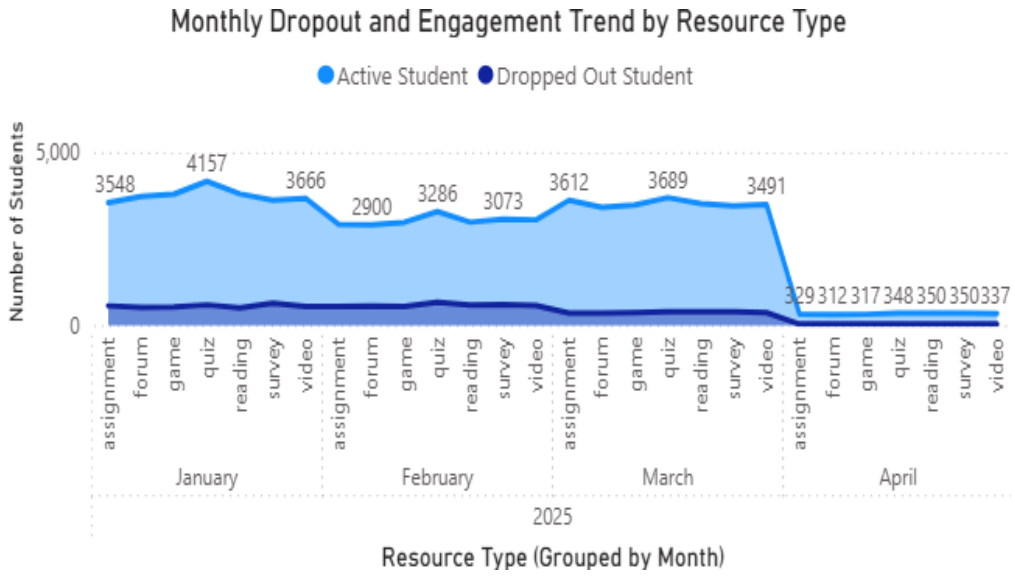
This bar chart shows the total number of completed resources by type. Quizzes had the highest completion count (over 1.7 million), followed by reading, game, and survey resources. Assignments and videos had the lowest totals, suggesting students were most engaged with evaluative and content-heavy materials like quizzes and readings.

- **Student Distribution by Mastery Level**



This chart displays the number of students at each mastery level. Most students are at the beginner level (476), followed by intermediate (312), and advanced (212). This suggests that a majority of learners are still early in their learning progression.

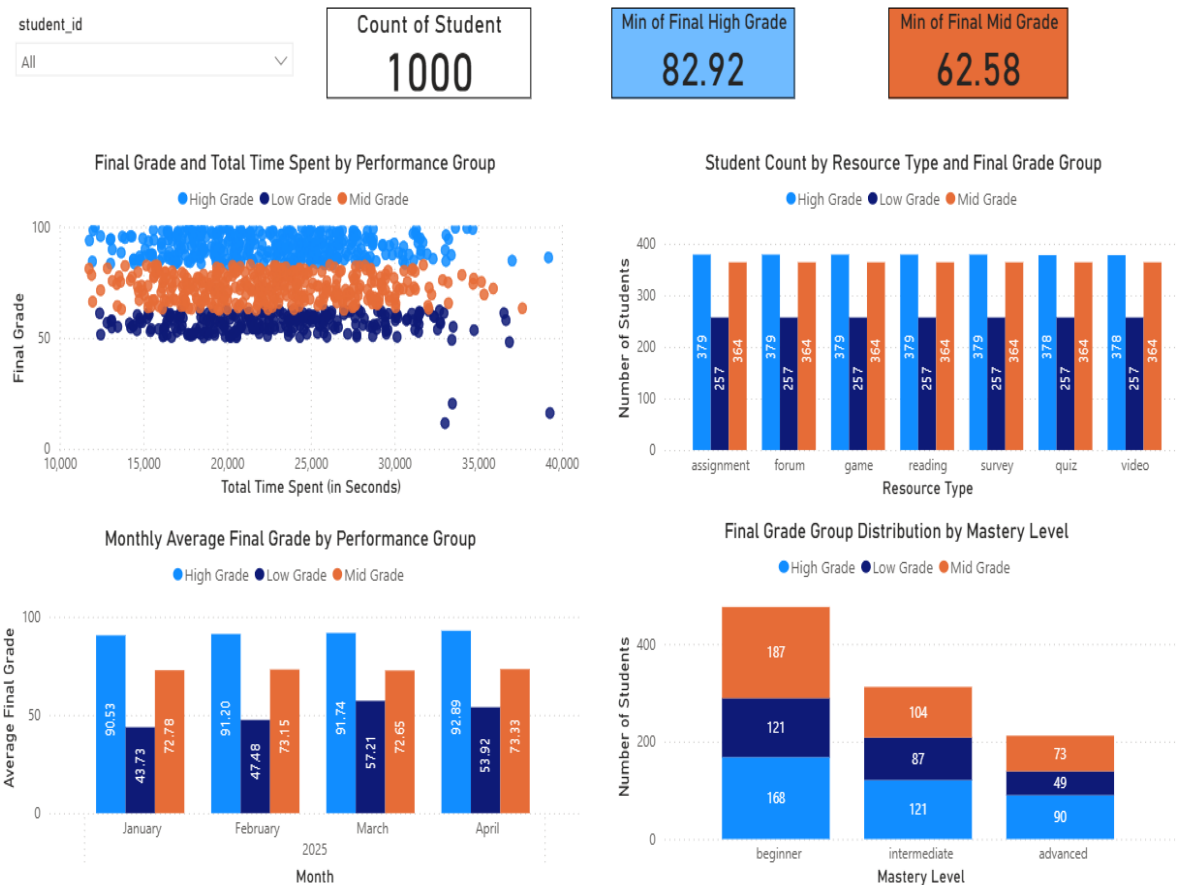
- **Monthly Dropout and Engagement Trend by Resource Type**



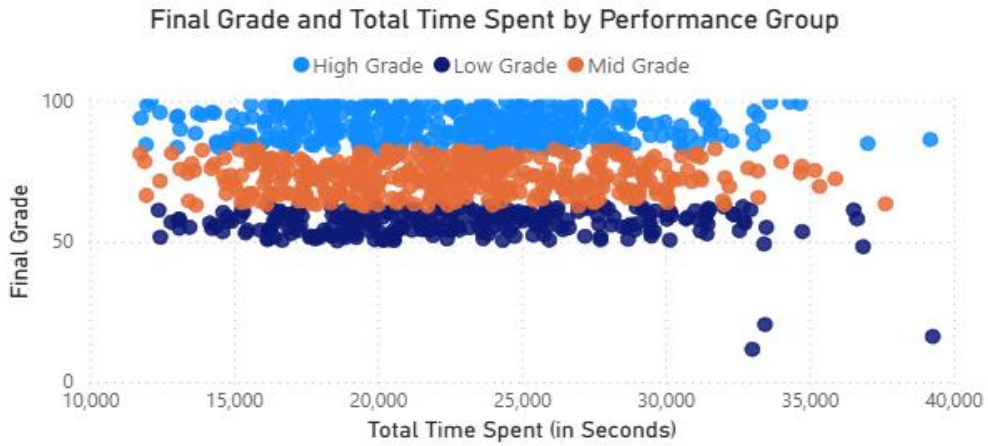
This area chart shows the monthly count of active vs. dropped-out students across different resource types. The number of active students remains consistently high, while dropouts are consistently low, suggesting stable engagement. A noticeable drop in April may indicate incomplete or missing data for that month.

2. Final Grade Analysis

The goal of this page is to evaluate how engagement affects final academic performance and identify patterns among high, mid, and low-grade groups.

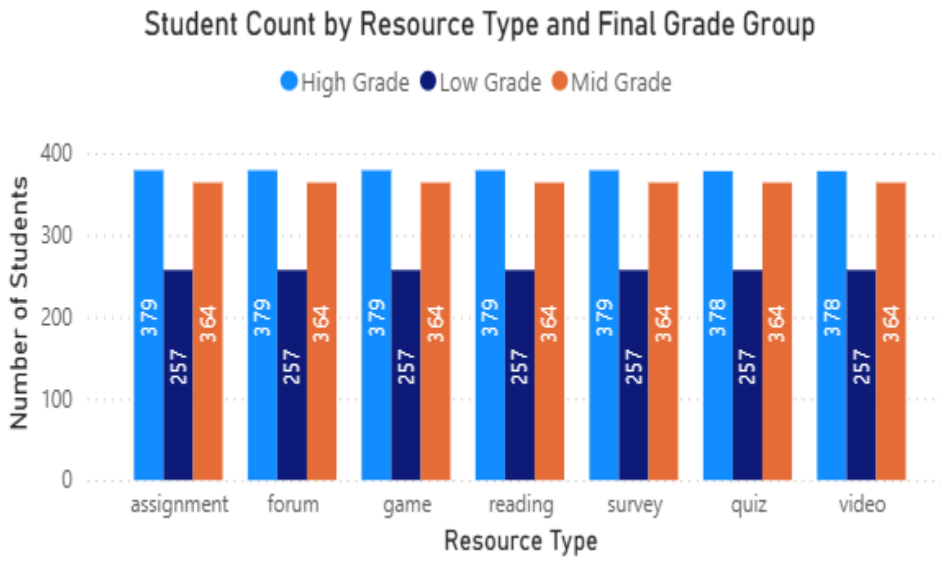


- **Final Grade and Total Time Spent by Performance Group**



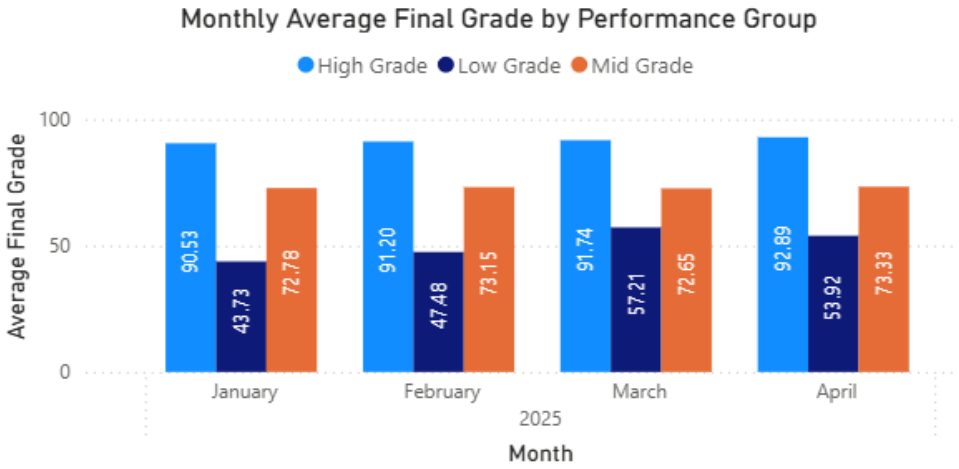
This scatter plot illustrates the relationship between total time spent and final grades. High-performing students are concentrated at the top, while low-performing students stay in the lower range regardless of time invested. This suggests that time spent helps but does not guarantee high performance, and other factors may also play a role.

- **Student Count by Resource Type and Final Grade Group**



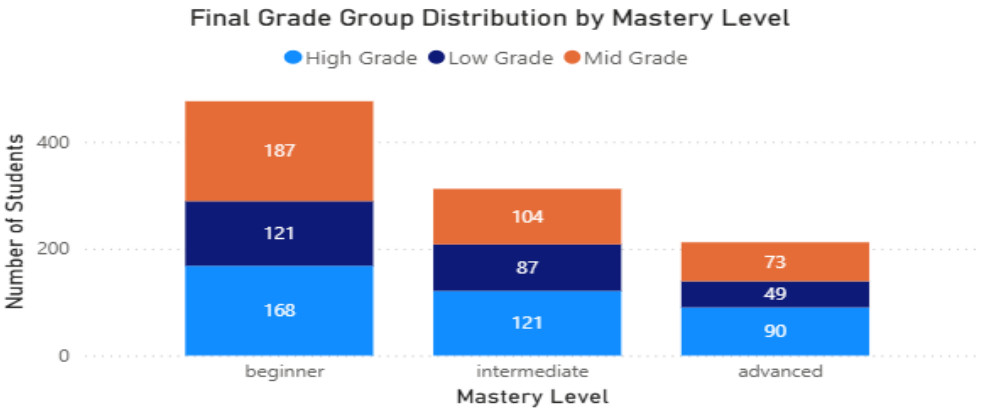
This chart compares the number of students by resource type and their final grade performance group. Across all resource types, most students fall into the high-grade category, while low-grade counts remain consistent. This suggests strong academic performance regardless of resource type.

- **Monthly Average Final Grade by Performance Group**



This chart shows the monthly average final grades by performance group. Students in the high-grade group consistently scored above 90, while mid-grade students averaged 73 across all months. Low-grade scores increased slightly over time, suggesting potential improvement or better engagement.

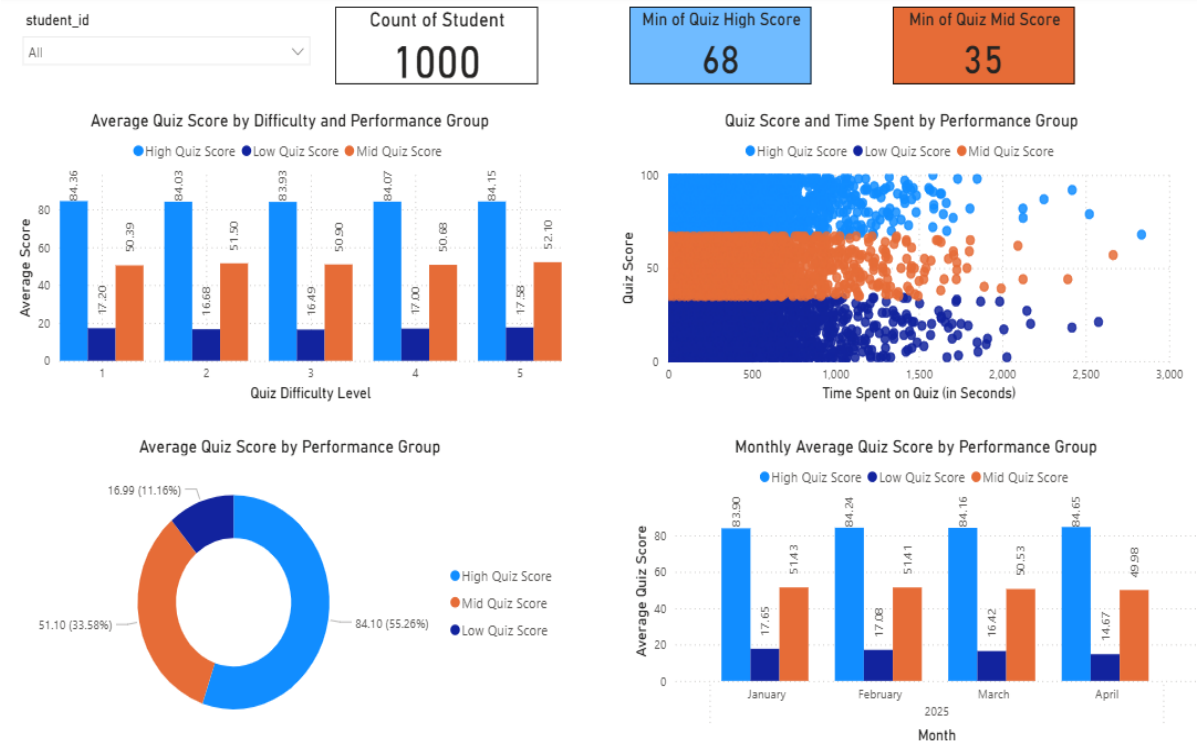
- **Student Performance by Mastery Level and Final Grade Group**



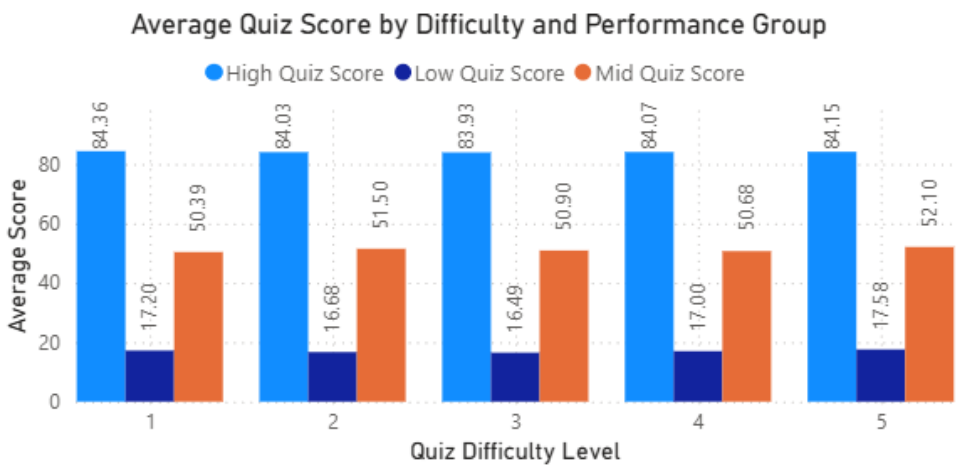
This chart shows how final grade performance is distributed across mastery levels. While all groups contain students of varying performance, advanced students have a higher proportion of high grades, while beginners show a more balanced spread across all grade levels.

3. Quiz Score Analysis

This page explores student quiz performance by analyzing score distribution across difficulty levels, time spent on quizzes, and monthly trends. It highlights how different performance groups (high, mid, low) respond to varying quiz conditions and helps identify areas where students may need additional support.

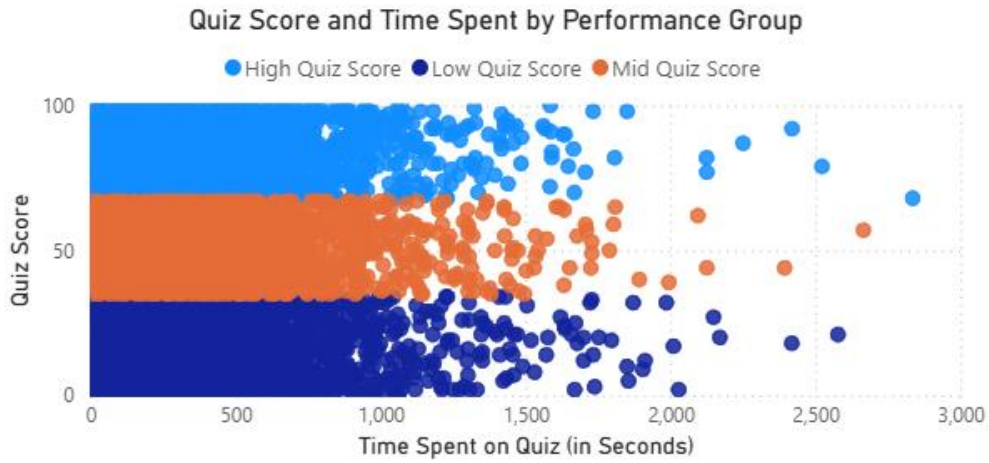


- **Average Quiz Score by Difficulty and Performance Group**



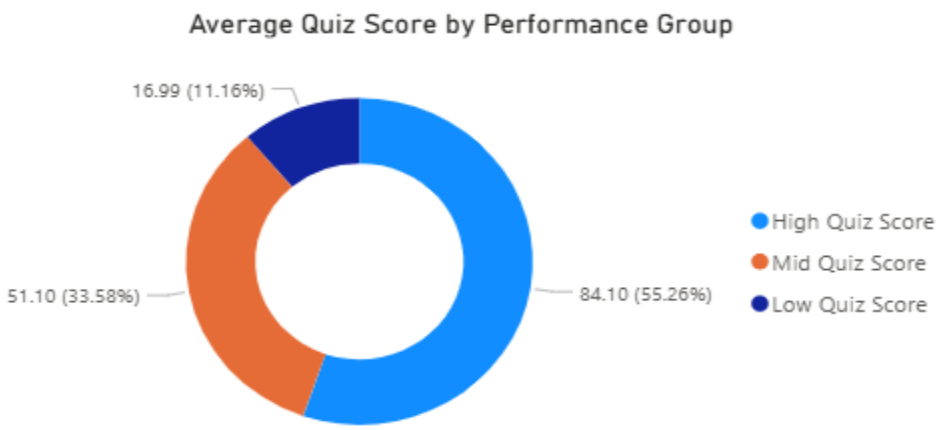
This chart shows the average quiz scores across different difficulty levels, grouped by performance. The scores remain consistent across all difficulty levels, with high performers averaging 84, mid performers around 50–52, and low performers around 16–18. This suggests that quiz difficulty had minimal impact on performance distribution.

- **Quiz Score and Time Spent by Performance Group**



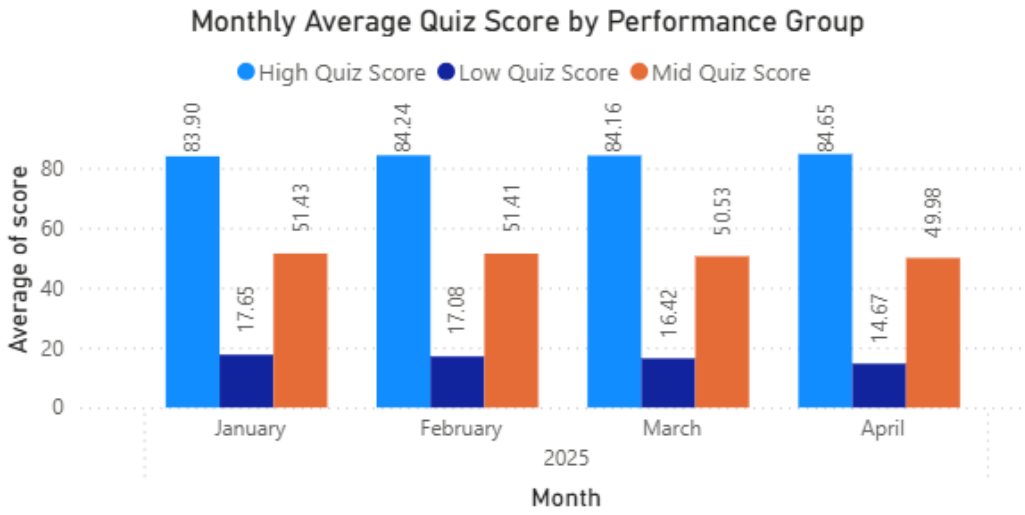
This scatter plot shows the relationship between time spent on quizzes and the score achieved, grouped by quiz performance level. Students with high scores tend to cluster in the lower to mid time ranges, suggesting efficiency. Low scorers appear across all time ranges, indicating that spending more time doesn't necessarily lead to better scores.

- **Average Quiz Score by Performance Group**



This donut chart shows average quiz scores by performance group. Most students are in the high-score group, averaging 84.10. Mid performers average 51.10, and low performers average 16.99. The distribution highlights strong overall performance with a small portion needing support.

- **Monthly Average Quiz Score by Performance Group**



This chart shows the monthly average quiz scores for each performance group. High scorers consistently averaged above 83, while mid scorers remained around 50. Low scorers showed a slight decline over time, from 17.65 in January to 14.67 in April, indicating a growing performance gap.

Conclusion

This project involved end-to-end data preparation, analysis, and visualization using Python and Power BI. By cleaning, merging, and exploring the dataset, I was able to uncover meaningful patterns in student engagement, academic performance, and quiz behavior. The interactive dashboard effectively highlights areas of strength—such as high overall quiz performance and student retention—as well as areas for improvement,

such as performance gaps among lower-scoring groups. These insights can support data-driven decisions to enhance the learning platform’s effectiveness and student outcomes.