Digital Egypt Pioneers

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

Project Report

AI – Data Science

Student Performance Analysis System

NO.	Team Members
1	عمر رضا الشاعر
2	سما السيد احمد
3	فيبي اميل رشدي
4	محمد حسام الدين مصطفى
5	عمار محمود عويدات

Table of Contents

1. Executive Summary

This project involves the end-to-end development of a data analysis pipeline designed to evaluate student academic performance. The process begins with cleaning and preprocessing raw academic records using Python, followed by normalizing the data into a relational SQLite database. By utilizing complex SQL queries and Python visualization libraries (Matplotlib/Pandas), the system generates actionable insights regarding subject difficulty, student attendance patterns, and overall grade distribution. This tool aims to assist educators and administrators in making data-driven decisions to support student success.

2. Introduction

Managing and analyzing student data is critical for educational institutions to identify performance gaps and attendance issues. The objective of this project was to transform raw, unstructured CSV data into a structured database and a visual dashboard.

Key Objectives:

- Clean and standardize raw student datasets.
- Design a normalized relational database schema.
- Execute SQL queries to extract key performance metrics.
- Create visual representations of data to facilitate pattern recognition.

Note on Data Source:
Please note that the dataset utilized in this project was **synthetically generated** for proof-of-concept purposes. While the data structure simulates real-world academic records, the specific values and resulting insights are illustrative and intended to demonstrate the functionality of the analysis pipeline rather than actual student performance.

3. Phase I: Data Preprocessing (ETL)

Before analysis could begin, the raw dataset required significant cleaning and enrichment. A Python script using the Pandas library was developed to handle this Extract-Transform-Load (ETL) process.

3.1 Cleaning Logic

The script performs the following operations on the raw CSV file:

- **Handling Missing Data:** Rows with missing values (except for the 'Score' column) are removed. Missing scores are filled with a default value of 0.
- **Attendance Logic:** A business rule is applied where any student marked as "absent" automatically receives a Score of 0.
- **De-duplication:** Duplicate records are removed based on unique combinations of Student ID, Level, and Subject to ensure data integrity.
- **Date Formatting:** The 'Date' column is converted into a standard datetime format.

3.2 Grading Algorithm

A custom function, `assign grade(score)`, was implemented to categorize numerical scores into letter grades based on the following scale:

- **A:** ≥ 89
- **B:** 76 – 88
- **C:** 67 – 75
- **D:** 60 – 66
- **F:** < 60

	A	B	C	D	E	F	G	H	I	J	K
1	ID	Name	Level	Subject	SubjID	Date	Attendanc	Score	Performance_Grade		exam id
2	S0001	Kurt Dway	Junior	English	ENG101	1/18/2023	Excused	81	B		1
3	S0001	Kurt Dway	Junior	Biology	BIO101	12/29/2023	Excused	55	F		2
4	S0001	Kurt Dway	Junior	Physics	PHYS101	7/13/2023	Present	68	C		3
5	S0001	Kurt Dway	Junior	Chemistry	CHEM101	11/14/2023	Late	72	C		4
6	S0001	Kurt Dway	Junior	Chemistry	CHEM101	1/15/2023	Present	68	C		5
7	S0001	Kurt Dway	Junior	Chemistry	CHEM101	7/19/2023	Excused	60	D		6
8	S0002	Teresa Me	Senior	Chemistry	CHEM101	10/9/2023	Present	74.9558	C		7
9	S0002	Teresa Me	Senior	Chemistry	CHEM101	3/29/2023	Present	90	A		8
10	S0002	Teresa Me	Senior	Biology	BIO101	5/31/2023	Absent	98	A		9
11	S0002	Teresa Me	Senior	Chemistry	CHEM101	10/31/2023	Excused	90	A		10
12	S0002	Teresa Me	Senior	Chemistry	CHEM101	12/1/2023	Absent	94	A		11
13	S0002	Teresa Me	Senior	English	ENG101	5/23/2023	Excused	68	C		12
14	S0002	Teresa Me	Senior	English	ENG101	11/8/2023	Late	91	A		13
15	S0002	Teresa Me	Senior	Chemistry	CHEM101	2/22/2023	Excused	85	B		14
16	S0002	Teresa Me	Senior	English	ENG101	10/21/2023	Excused	60	D		15
17	S0003	Corey Mac	Senior	Chemistry	CHEM101	11/24/2023	Present	68	C		16
18	S0003	Corey Mac	Senior	English	ENG101	9/12/2023	Absent	58	F		17
19	S0003	Corey Mac	Senior	Chemistry	CHEM101	4/2/2023	Excused	95	A		18
20	S0003	Corey Mac	Senior	English	ENG101	2/10/2023	Late	62	D		19
21	S0003	Corey Mac	Senior	Physics	PHYS101	12/18/2023	Late	50	F		20
22	S0004	Amanda D	Sophomor	Chemistry	CHEM101	6/24/2023	Absent	88	B		21
23	S0004	Amanda D	Sophomor	Physics	PHYS101	10/20/2023	Absent	61	D		22
24	S0004	Amanda D	Sophomor	Biology	BIO101	3/28/2023	Late	82	B		23
25	S0004	Amanda D	Sophomor	Chemistry	CHEM101	12/5/2023	Excused	82	B		24
26	S0004	Amanda D	Sophomor	Biology	BIO101	12/21/2023	Absent	76	B		25

The final cleaned dataset is exported as studentsdataset_cleaned1.csv for database insertion.

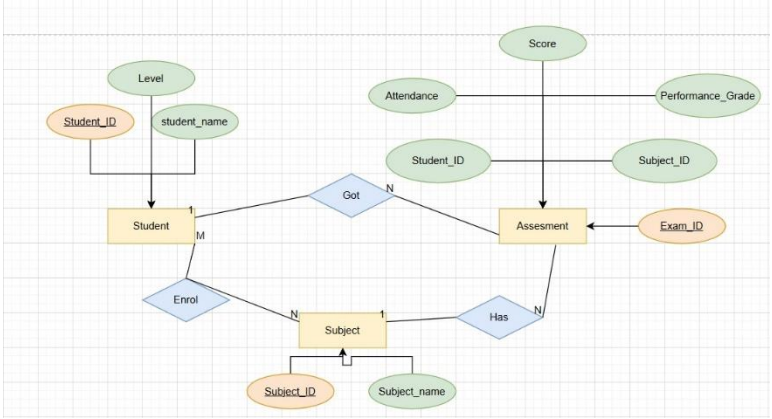
```python
import pandas as pd
df =pd.read_csv("/content/drive/MyDrive/students_expanded_40000.csv")
df=df.sort_values("ID")
def assign_grade(score):
    if score < 60:
        return "F"
    elif 60 <= score < 67:
        return "D"
    elif 67 <= score < 76:
        return "C"
    elif 76 <= score < 89:
        return "B"
    else:
        return "A"
df = df.dropna(subset=[col for col in df.columns if col != 'Score'])
df['Score'].fillna(0, inplace=True)
df.loc[df['Attendance'].str.strip().str.lower() == 'absent', 'Score'] = 0
df = df.drop_duplicates(subset=['ID','Level','Subject'], keep='first')
df['Date'] = pd.to_datetime(df['Date'])
df['Performance_Grade'] = df['Score'].apply(assign_grade)
display(df)
df.to_csv("studentsdataset_cleaned1.csv", index=False)
```

4. Phase II:

Database Design & Implementation

To ensure efficient querying, the cleaned data was normalized and migrated into an SQLite database (`students.db`). The schema consists of three related tables.

4.1 Database Schema (ER Diagram)



1. **Table: Student**
   - Stores student demographics.
   - Columns: Student (PK), Student name, Level.
2. **Table: Subject**
   - Stores course information.
   - Columns: Subject (PK), Subject name.

-

3. **Table: Assessment**
   - The central transaction table linking students to subjects and scores.
   - Columns: Exam_ID (PK), Subject (FK), Student (FK), Attendance, Score, Performance_Grade.

```sql
CREATE TABLE Student(
Student_ID TEXT(20) PRIMARY KEY ,
Student_name TEXT(50),
Level TEXT(20)
);
SELECT * FROM Student
```

Student_ID	Student_name	Level
S0001	Kurt Dwayne Taylor	Junior
S0002	Teresa Melissa Brewer	Senior
S0003	Corey Madison Chase	Senior
S0004	Amanda Donald Phillips	Sophomore
S0005	Nathan Justin Schultz	Freshman
S0006	Jonathon Bradley Freeman	Senior
S0007	Kristin Paul Liu	Junior
S0008	Danielle Mary Stewart	Freshman
S0009	Paul Jermaine Velez	Freshman
S0010	Mark Tony Burton	Junior

```sql
CREATE TABLE Subject(
Subject_ID TEXT(20) PRIMARY KEY ,
Subject_name TEXT(25)
);
SELECT * FROM Subject
```

Subject_ID	Subject_name
ENG101	English
BIO101	Biology
PHYS101	Physics
CHEM101	Chemistry
MATH101	Math

```sql
CREATE TABLE Assesment(
Exam_ID INTEGER PRIMARY KEY AUTOINCREMENT,
Subject_ID TEXT(20),
Student_ID TEXT(20),
Attendance TEXT(20),
Score FLOAT,
Performance_Grade CHAR,
FOREIGN KEY (subject_id) REFERENCES Subject(subject_id),
FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
)
```

Exam_ID	Subject_ID	Student_ID	Attendance	Score	Performance_Gr
1	ENG101	S0001	Excused	81	B
2	BIO101	S0001	Excused	55	F
3	PHYS101	S0001	Present	68	C
4	CHEM101	S0001	Late	72	C
5	CHEM101	S0002	Present	74.95579658	C
6	BIO101	S0002	Absent	0	F
7	ENG101	S0002	Excused	68	C
8	CHEM101	S0003	Present	68	C
9	ENG101	S0003	Absent	0	F
10	PHYS101	S0003	Late	50	F
11	CHEM101	S0004	Absent	0	F
12	PHYS101	S0004	Absent	0	F
13	BIO101	S0004	Late	82	B
14	BIO101	S0005	Excused	84	B
15	MATH101	S0005	Absent	0	F

5. Phase III: SQL Data Analysis

With the database established, SQL queries were written to extract specific analytical insights.

5.1 Top Performers Analysis

- **Goal:** Identify the highest-performing student for each subject.
- **Logic:** Used a subquery to calculate the maximum score (`MAX`) per subject and matched it to the student record.
- **Result:** A list of students achieving the top scores, sorted alphabetically by subject.

```sql
SELECT
    s.Subject_name,
    st.Student_name,
    a1.Score,
    a1.Performance_Grade
FROM Assessment a1
JOIN Student st ON a1.Student_ID = st.Student_ID
JOIN Subject s ON a1.Subject_ID = s.Subject_ID
WHERE a1.Score = (
    SELECT MAX(a2.Score)
    FROM Assessment a2
    WHERE a2.Subject_ID = a1.Subject_ID
)
ORDER BY s.Subject_name;
```

Subject_name	Student_name	Score	Performance_Grade
Biology	Aaron David Moore	100	A
Biology	Erik Lisa King	100	A
Biology	Holly Natalie Coleman	100	A
Biology	Kelsey Shannon Stone	100	A
Biology	James Leah Gregory	100	A
Biology	Megan Alex Little	100	A

5.2 Attendance Statistics

- **Goal:** Assess student engagement levels per subject.
- **Logic:** Calculated the Present Count vs. Absent Count. An Attendance Rate percentage was derived (Present / Total) and rounded to two decimal places.
- Result**:** Subjects are ranked by attendance rate, highlighting courses with potential engagement issues.

```sql
SELECT
    s.Subject_name,
    SUM(a.Attendance IN ('Present', 'Late')) AS Present_Count,
    SUM(a.Attendance IN ('Absent', 'Excused')) AS Absent_Count,
    ROUND(100.0 * SUM(a.Attendance IN ('Present', 'Late')) / COUNT(*), 2) AS Attendance_Rate
FROM Assessment a
JOIN Subject s ON a.Subject_ID = s.Subject_ID
GROUP BY s.Subject_name
ORDER BY Attendance_Rate DESC;
```

Subject_name	Present_Count	Absent_Count	Attendance_Rate
Biology	655	613	51.66
Math	640	616	50.96
English	630	624	50.24
Chemistry	592	650	47.67
Physics	589	651	47.5

5.3 Subject Performance Averages

- **Goal:** Determine which subjects are statistically the most difficult.
- **Logic:** Aggregated scores by subject to compute the mean (`AVG`) score.
- **Result:** Data sorted by average score in ascending order, allowing administrators to easily identify "weak" subjects.

Run | Top performers by subject | Attendance Query | AvgPerSub

```sql
SELECT s.subject_name ,
round(avg(a.score),2) AS AveragePerSubject
FROM Assessment AS a
JOIN Subject s ON a.Subject_ID = s.Subject_ID
GROUP BY subject_name
ORDER BY AveragePerSubject
```

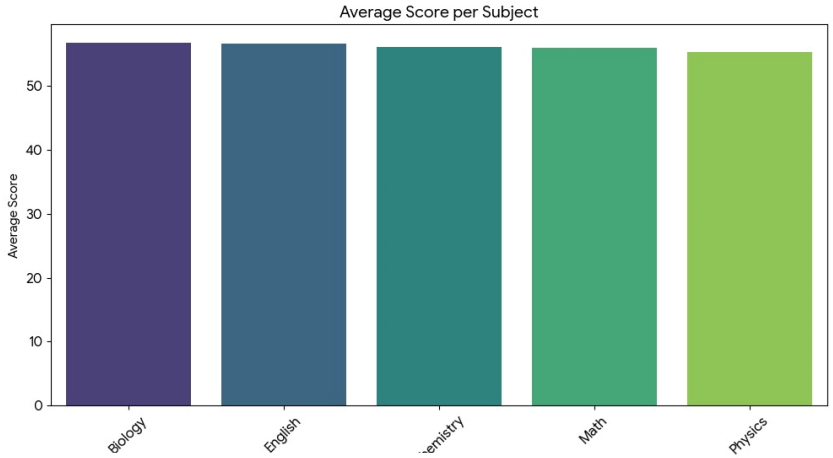
Subject_name	AveragePerSubject
Physics	55.34
Math	55.96
Chemistry	56.08
English	56.59
Biology	56.78

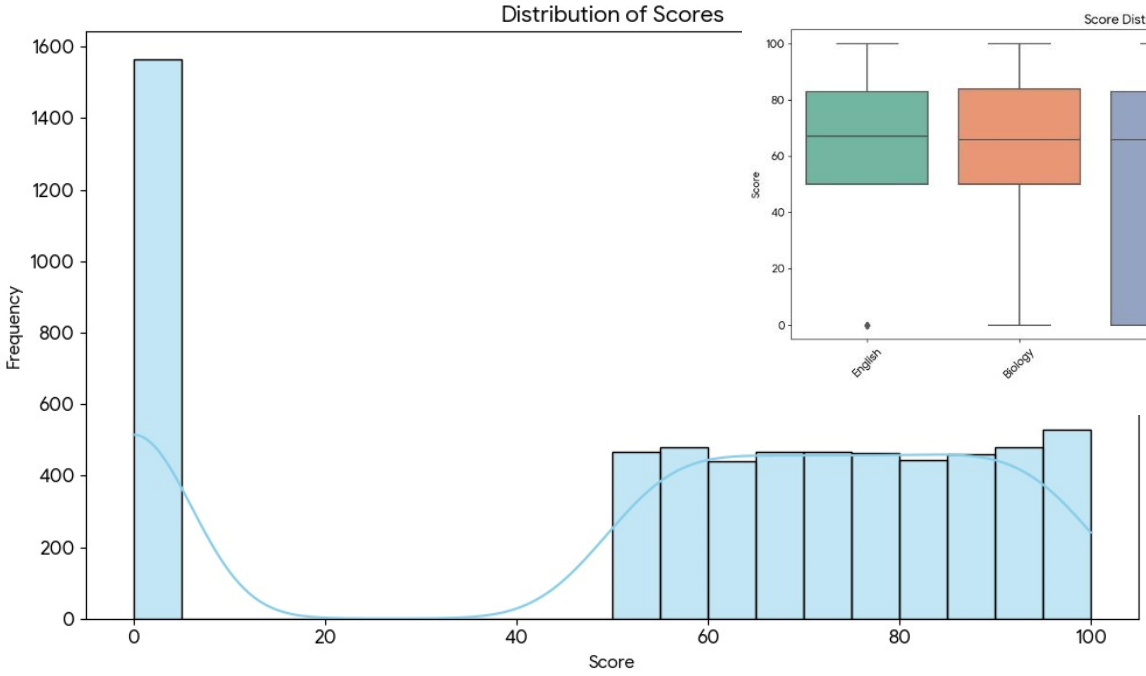
6. Phase IV: Visual Analytics & Python Integration

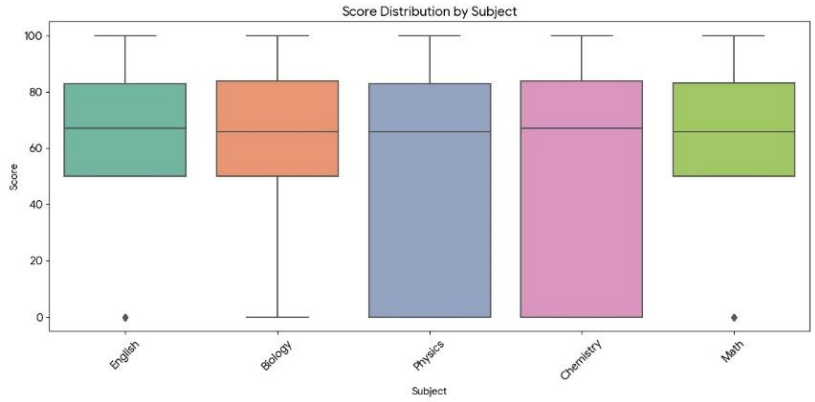
The final phase involved connecting Python to the SQLite database to visualize the SQL results. The tables were merged into a single Pandas Data Frame to facilitate graphing.

6.1 Visualizations Generated

- **Average Score per Subject (Bar Chart):** Visualizes difficulty levels across different courses.
- **Score Distribution (Histogram):** Shows the spread of grades across the student population (identifying skewness or bell curves).

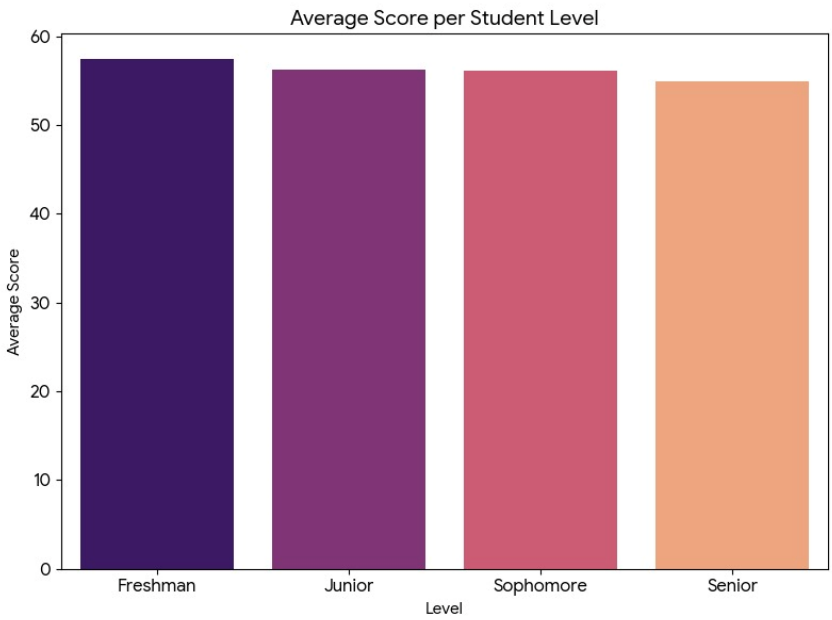


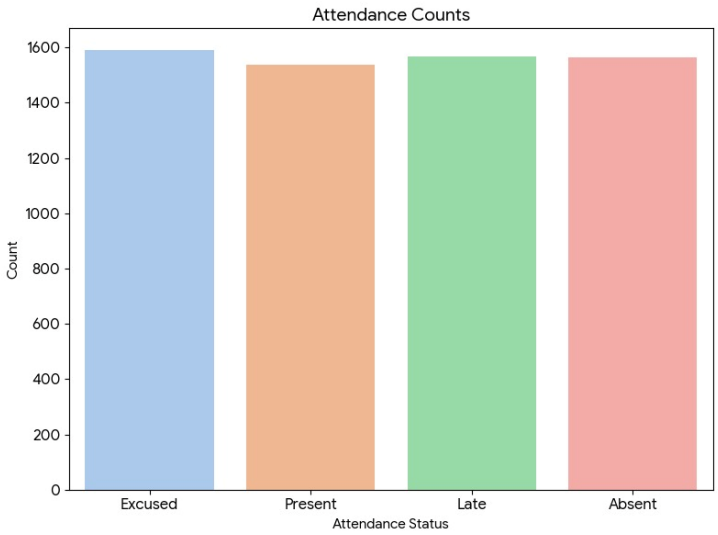




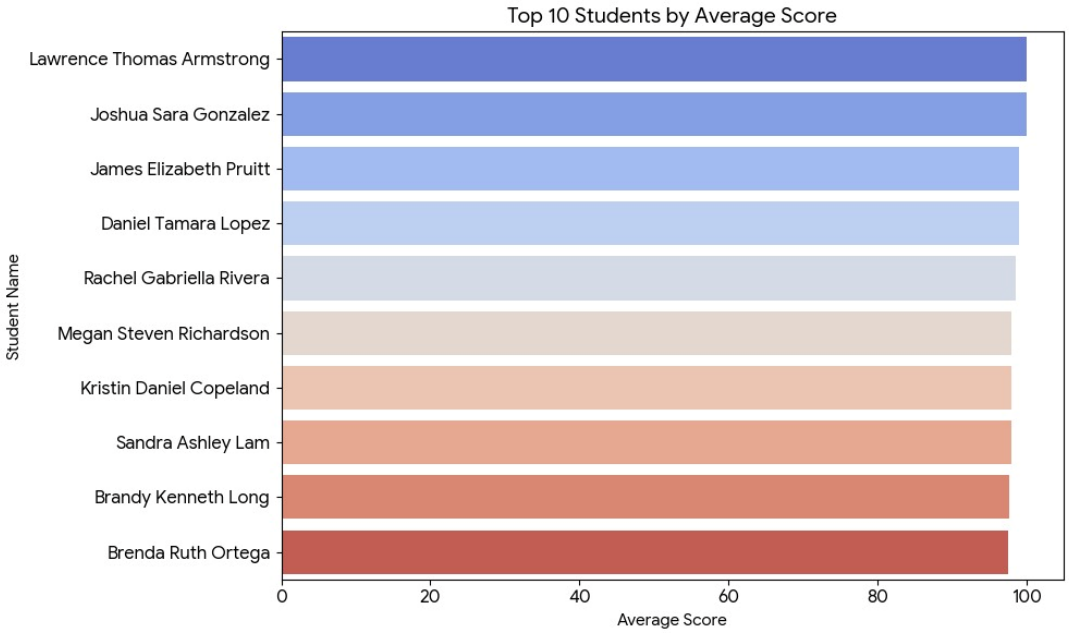
- **Attendance Status Count (Bar Chart):** Displays the gross volume of Present vs. Absent vs. Late statuses.

- **Average Score by Student Level (Bar Chart):** Compares performance between Freshman, Sophomore, Junior, and Senior levels.





- **Top 10 Students (Horizontal Bar Chart):** A ranking of the highest-performing students based on their average scores.



The code for the previous charts:

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Connect to the database
conn = sqlite3.connect('students.db')

# 2. Load the data tables
df_student = pd.read_sql_query("SELECT * FROM Student;", conn)
df_subject = pd.read_sql_query("SELECT * FROM Subject;", conn)
df_assess = pd.read_sql_query("SELECT * FROM Assessment;", conn)

# 3. Merge data into a single DataFrame
df = df_assess.merge(df_student, on='Student_ID', how='left')
df = df.merge(df_subject, on='Subject_ID', how='left')

# 4. Clean the data
# Ensure scores are numeric, handling any potential errors
df['Score'] = pd.to_numeric(df['Score'], errors='coerce')


# --- Visualization 3: Attendance Counts ---
plt.figure(figsize=(8, 6))
sns.countplot(x='Attendance', data=df, palette='pastel')
plt.title('Attendance Counts')
plt.xlabel('Attendance Status')
plt.ylabel('Count')
plt.tight_layout()
plt.savefig('attendance_counts.png')
plt.close()

# --- Visualization 4: Average Score per Level ---
if 'Level' in df.columns:
    avg_level = df.groupby('Level', as_index=False)
['Score'].mean().sort_values('Score', ascending=False)
    plt.figure(figsize=(8, 6))
    sns.barplot(x='Level', y='Score', data=avg_level, palette='magma')
    plt.title('Average Score per Student Level')
    plt.xlabel('Level')
    plt.ylabel('Average Score')
    plt.tight_layout()
    plt.savefig('avg_score_per_level.png')
    plt.close()
```

```
# --- Visualization 1: Average Score per Subject ---
avg_sub = df.groupby('Subject_name', as_index=False)
['Score'].mean().sort_values('Score', ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x='Subject_name', y='Score', data=avg_sub, palette='viridis')
plt.title('Average Score per Subject')
plt.xlabel('Subject')
plt.ylabel('Average Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('avg_score_per_subject.png')
plt.close()

# --- Visualization 2: Score Distribution ---
plt.figure(figsize=(10, 6))
sns.histplot(df['Score'].dropna(), bins=20, kde=True, color='skyblue')
plt.title('Distribution of Scores')
plt.xlabel('Score')
plt.ylabel('Frequency')
plt.tight_layout()
plt.savefig('score_distribution.png')
plt.close()


# --- Visualization 5: Top 10 Students by Average Score ---
top_students = df.groupby('Student_name', as_index=False)
['Score'].mean().sort_values('Score', ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x='Score', y='Student_name', data=top_students, palette='coolwarm')
plt.title('Top 10 Students by Average Score')
plt.xlabel('Average Score')
plt.ylabel('Student Name')
plt.tight_layout()
plt.savefig('top_10_students.png')
plt.close()

# --- Visualization 6: Score Distribution by Subject (Boxplot) ---
plt.figure(figsize=(12, 6))
sns.boxplot(x='Subject_name', y='Score', data=df, palette='Set2')
plt.title('Score Distribution by Subject')
plt.xlabel('Subject')
plt.ylabel('Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('boxplot_scores_by_subject.png')
plt.close()

print("Visualizations generated successfully.")
```

7. Key Findings & Conclusion

7.1 System Capabilities Demonstrated

Based on the visual and query analysis performed on the synthetic dataset, the system successfully demonstrated the following analytical capabilities:

- **Subject-Level Analysis:** The pipeline can identify performance disparities across subjects, highlighting courses with consistently high scores versus those with wider grade distributions.
- **Cohort Comparison:** The system effectively segments students by academic level, enabling trend analysis across different student populations.
- **Attendance Correlation:** The tool successfully tracks attendance patterns and can identify potential relationships between attendance and academic performance.

7.2 Conclusion

This project successfully established a **proof-of-concept pipeline** for converting raw academic data into actionable visual intelligence. By integrating Python for data cleaning and visualization with SQL for structured querying, the system architecture demonstrates:

1. **Technical Feasibility:** The complete ETL-to-visualization workflow functions correctly and can scale to real-world datasets.
2. **Analytical Framework:** The queries and visualizations provide a template for the types of insights that could support educators in identifying at-risk students and curriculum weaknesses.
3. **Reproducibility:** The modular design allows this system to be deployed with actual institutional data with minimal modification.

Future Implementation: When deployed with genuine student records, this system would enable teachers to monitor student progress in real-time, allow administrators to pinpoint underperforming courses, and provide stakeholders with evidence-based recommendations for academic interventions.