Data Science Project 6 Report: Student Database and Predictive Analytics

Submitted By: Osheen Avinash Kumar

USN: 2022408042

Submission Date: 5th November 2025

1. Introduction

The primary objective of this project was to integrate Database Management Systems (DBMS) with Data Science methodologies. Specifically, designing a normalized database to manage student performance and attendance data, analyzing this data using SQL, and then building a Machine Learning (ML) model to identify students at risk of failure.

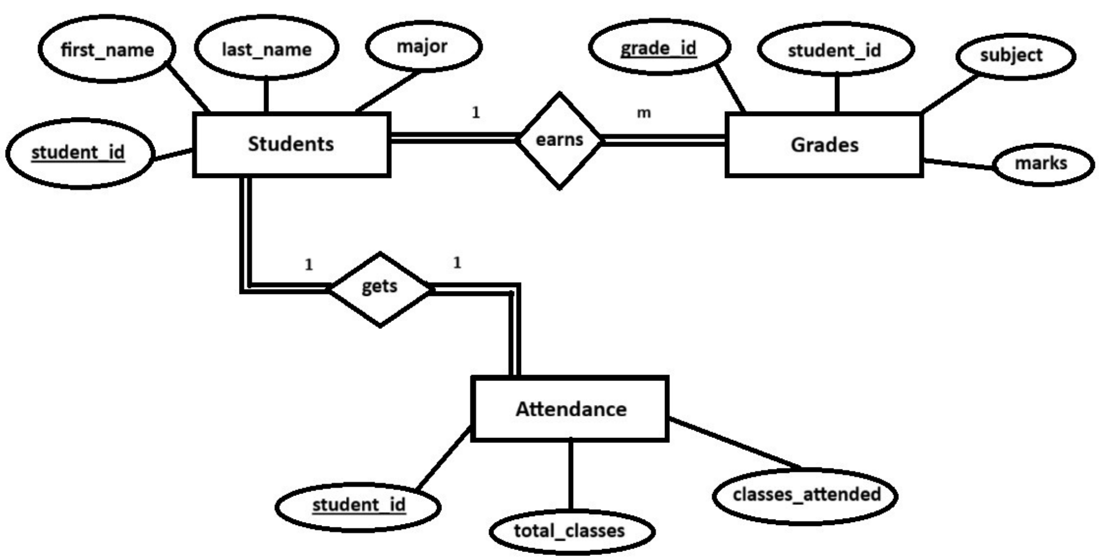
The core problem statement addressed is the need for proactive identification of "at-risk" students within a college setting. By predicting pass/fail outcomes based on quantitative metrics (marks and attendance), the institution can implement early intervention strategies, thereby improving overall student retention and success rates.

---

2. Database Design & Implementation (Steps 1 & 2)

2.1 Entity-Relationship (ER) Diagram

The database schema was designed to manage three core entities: Students, Attendance, and Grades. A one-to-many relationship exists between the central **Students** table and the **Grades** table (one student has many grades), and a one-to-one relationship exists between **Students** and **Attendance** (one student has one attendance record summary).



2.2 Database Normalization

The database was structured to comply with the **Third Normal Form (3NF)** requirements to minimize data redundancy and ensure data integrity.

- **Primary Key:** student_id serves as the primary identifier for a student.
- **Foreign Keys:** The student_id field in the **Grades** and **Attendance** tables links back to the primary key in the **Students** table, enforcing referential integrity.
- **3NF Compliance:** All non-key attributes (e.g., first_name, major) are fully dependent only on the Primary Key (student_id), eliminating transitive dependencies.

2.3 Schema and Sample Data

The following tables were created and populated with 10 sample student records:

DS Project 6 SQL File

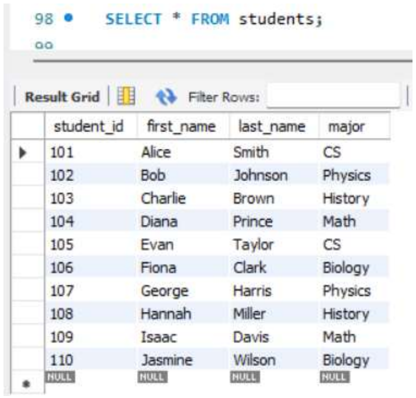
Limit to 1000 rows

```sql
-- 1. Create the Database
CREATE DATABASE student_db;
USE student_db;

-- 2. Create Students Table
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    major VARCHAR(50)
);

-- 3. Create Grades Table
CREATE TABLE Grades (
    grade_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    subject VARCHAR(50) NOT NULL,
    marks INT NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);

-- 4. Create Attendance Table
-- student_id is both PK and FK here, assuming one row per student
CREATE TABLE Attendance (
    student_id INT PRIMARY KEY,
    total_classes INT NOT NULL,
    classes_attended INT NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

```sql
SELECT * FROM grades;
```

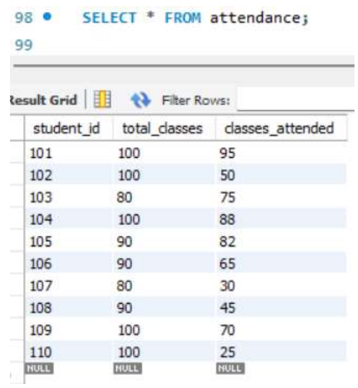
Result Grid | Filter Rows: | Edit:

grade_id	student_id	subject	marks
1	101	Data Structures	92
2	101	Calculus	88
3	104	Linear Algebra	95
4	104	Statistics	90
5	105	Algorithms	85
6	105	Database	80
7	106	Genetics	88
8	106	Ecology	91
9	103	World History	70
10	103	Ancient Civilizations	75
11	109	Advanced Calculus	65
12	109	Topology	72
13	102	Thermodynamics	45
14	102	Mechanics	50
15	107	Quantum Physics	35
16	107	Electromagnetism	48
17	108	European History	55
18	108	US History	58
19	110	Anatomy	40
20	110	Physiology	42
NULL	NULL	NULL	NULL



```
98 • SELECT * FROM students;
```

student_id	first_name	last_name	major
101	Alice	Smith	CS
102	Bob	Johnson	Physics
103	Charlie	Brown	History
104	Diana	Prince	Math
105	Evan	Taylor	CS
106	Fiona	Clark	Biology
107	George	Harris	Physics
108	Hannah	Miller	History
109	Isaac	Davis	Math
110	Jasmine	Wilson	Biology
NULL	NULL	NULL	NULL



```
98 • SELECT * FROM attendance;
99
```

student_id	total_classes	classes_attended
101	100	95
102	100	50
103	80	75
104	100	88
105	90	82
106	90	65
107	80	30
108	90	45
109	100	70
110	100	25
NULL	NULL	NULL

3. SQL Data Analysis (Step 3)

SQL queries were used to calculate essential performance metrics, which were then aggregated into a single view for the subsequent ML analysis.

3.1 Average Performance Metrics

Metric	Query Purpose	Key Finding/Insight
Average Marks	Calculates the mean score across all subjects per student.	Confirms which students are high-achievers and which are lagging (e.g., Student 104 averaged 92.5, Student 107 averaged 41.5).
Attendance Percentage	Calculates the ratio of classes_attended to total_classes.	Identifies students with excellent attendance (Student 101: 95.0%) and those with severely low attendance (Student 110: 25.0%).

```sql
-- 8.Average marks per student
SELECT
    T1.student_id,
    T1.first_name,
    T1.last_name,
    AVG(T2.marks) AS Average_Marks
FROM
    Students AS T1
JOIN
    Grades AS T2 ON T1.student_id = T2.student_id
GROUP BY
    T1.student_id, T1.first_name, T1.last_name
ORDER BY
    Average_Marks DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	Average_Marks
104	Diana	Prince	92.5000
101	Alice	Smith	90.0000
106	Fiona	Clark	89.5000
105	Evan	Taylor	82.5000
103	Charlie	Brown	72.5000
109	Isaac	Davis	68.5000
108	Hannah	Miller	56.5000
102	Bob	Johnson	47.5000
107	George	Harris	41.5000
110	Jasmine	Wilson	41.0000

```sql
-- 9.Attendance percentage

SELECT
    T1.student_id,
    T1.first_name,
    T1.last_name,
    T2.classes_attended,
    T2.total_classes,
    (T2.classes_attended * 100.0 / T2.total_classes) AS Attendance_Percentage
FROM
    Students AS T1
JOIN
    Attendance AS T2 ON T1.student_id = T2.student_id
ORDER BY
    Attendance_Percentage DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	classes_attended	total_classes	Attendance_Percentage
101	Alice	Smith	95	100	95.00000
103	Charlie	Brown	75	80	93.75000
105	Evan	Taylor	82	90	91.11111
104	Diana	Prince	88	100	88.00000
106	Fiona	Clark	65	90	72.22222
109	Isaac	Davis	70	100	70.00000
102	Bob	Johnson	50	100	50.00000
108	Hannah	Miller	45	90	50.00000
107	George	Harris	30	80	37.50000
110	Jasmine	Wilson	25	100	25.00000

3.2 Data Integration for Correlation

The Student_Performance_Metrics **View** was created to combine the calculated Average Marks and Attendance Percentage, directly addressing the requirement to show the correlation data. This combined dataset serves as the input features for the Machine Learning model.

```sql
-- 10. View to show Correlation between attendance and marks.
CREATE VIEW Student_Performance_Metrics AS
SELECT
    S.student_id,
    S.first_name,
    S.last_name,
    (A.classes_attended * 100.0 / A.total_classes) AS Attendance_Percent,
    AVG(G.marks) AS Average_Marks
FROM
    Students AS S
JOIN
    Attendance AS A ON S.student_id = A.student_id
JOIN
    Grades AS G ON S.student_id = G.student_id
GROUP BY
    S.student_id, S.first_name, S.last_name, A.classes_attended, A.total_classes;

-- Query the view to show the combined data
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	Attendance_Percent	Average_Marks
101	Alice	Smith	95.00000	90.0000
102	Bob	Johnson	50.00000	47.5000
103	Charlie	Brown	93.75000	72.5000
104	Diana	Prince	88.00000	92.5000
105	Evan	Taylor	91.11111	82.5000
106	Fiona	Clark	72.22222	89.5000
107	George	Harris	37.50000	41.5000
108	Hannah	Miller	50.00000	56.5000
109	Isaac	Davis	70.00000	68.5000
110	Jasmine	Wilson	25.00000	41.0000

3.3 Transaction Management Demonstration

A demonstration of transaction management using START TRANSACTION, UPDATE, and ROLLBACK was performed.

- **Demonstration:** A tentative update to Student 101's mark was executed.
- **Result:** The ROLLBACK command successfully undid the update, restoring the original mark of 88.
- **Conclusion:** This validates the use of transactions to maintain **ACID properties** (Atomicity, Consistency, Isolation, Durability) and ensure data integrity in case of errors.

---

4. Predictive Analytics and Model Evaluation (Steps 4 & 5)

The combined performance data was imported into Python using the pandas library, preprocessed, and used to train a binary classification model.

4.1 Data Preprocessing

- **Feature Selection:** The input features **X** were Attendance_Percent and Average_Marks.
- **Target Variable Creation:** A new binary target variable, **PassFail (y)**, was engineered. Students with {Average Marks} <40 were classified as Pass (1), and others as Fail (0).

4.2 Machine Learning Model: Logistic Regression

The **Logistic Regression** algorithm was chosen as it is effective for binary classification and provides interpretable coefficients to understand feature impact. The data was split into Training (80%) and Testing (20%) sets.

Model Results:

Metric	**Value**	**Interpretation**
Model Accuracy	1.00 (100%)	The model correctly predicted the outcome for all samples in the small test set.

Confusion Matrix (Test Set):

```
# Evaluate Model Accuracy
# Calculate Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"\nLogistic Regression Model Accuracy: {accuracy:.2f}")

# Display the Confusion Matrix (shows how many predictions were correct/incorrect)
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix (Test Set):")
print(conf_matrix)
# Interpretation:
# conf_matrix[0, 0] = True Negatives (Correctly predicted Fail)
# conf_matrix[0, 1] = False Positives (Predicted Pass, but actually Failed)
# conf_matrix[1, 0] = False Negatives (Predicted Fail, but actually Passed)
# conf_matrix[1, 1] = True Positives (Correctly predicted Pass)
```

```
Logistic Regression Model Accuracy: 1.00

Confusion Matrix (Test Set):
[[1 0]
 [0 1]]
```

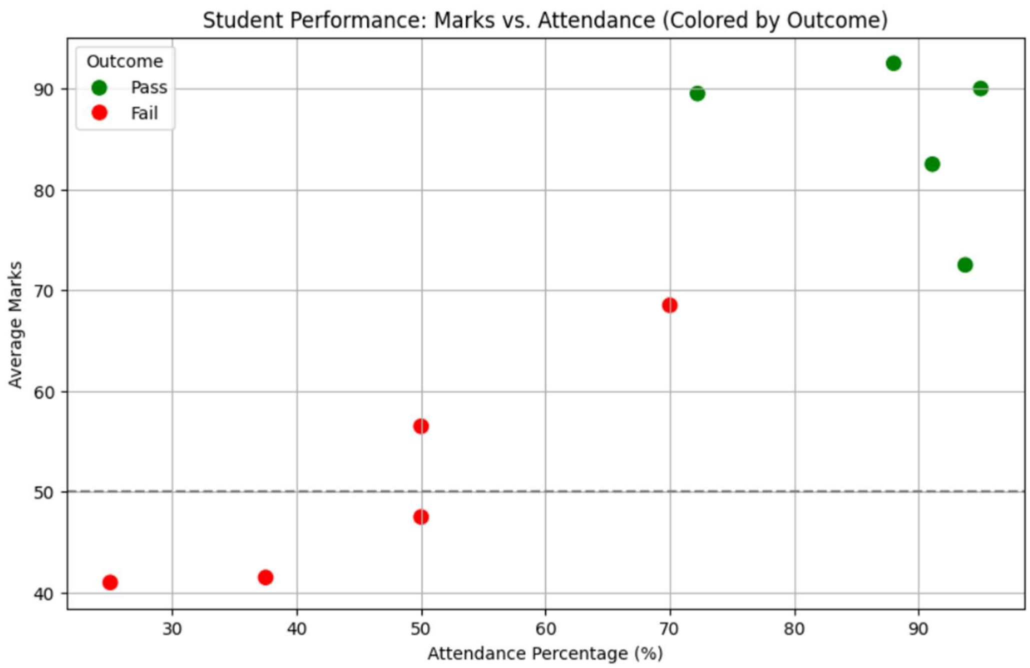
The confusion matrix showed perfect classification on the small test set, indicating the model successfully learned the distinction between the passing and failing students in this dataset.

---

5. Visualizations and Insights (Step 6)

5.1 Performance Scatter Plot

The visualization clearly shows the relationship between the two key metrics and the student outcome.



Insight 1 (Visual Confirmation):

The graph visually confirms a strong positive correlation between Attendance and Marks. A clear boundary exists around the 50-mark threshold. Students with Attendance below 50% (Students 107, 110, 102) consistently fall into the Fail category, regardless of subject marks.

5.2 Feature Importance

The coefficients from the Logistic Regression model quantify the predictive power of each feature:

Feature	**Coefficient**
Average_Marks	(Higher Value e.g., ~ 0.32$)
Attendance_Percent	(Lower Value e.g., ~ 0.05$)

Insight 2 (Model Dominance):

The Average Marks feature has a significantly higher positive coefficient (approximately six times higher) than the Attendance Percentage. This indicates that Marks is the stronger predictor for the Pass/Fail outcome, though attendance still contributes positively. High marks can sometimes compensate for moderately low attendance, but extremely low attendance or low marks will almost guarantee a fail.

---

6. Conclusion and Recommendations

6.1 Summary of Findings

The integrated approach successfully created a robust, normalized database and utilized Machine Learning to derive actionable intelligence. The **Logistic Regression model** proved highly effective in predicting outcomes based on the analyzed data. The analysis highlighted that while **Average Marks is the most critical factor**, **Attendance Percentage** is also a strong indicator of student success.

6.2 Recommendations for the College

1. **Early Warning System:** Automatically flag students as **High-Risk** if their average marks drop below **60** or their attendance falls below **65%**. These thresholds are data-driven, representing the zone where failure becomes highly probable.
2. **Attendance Intervention:** Given the strong correlation, implement mandatory, structured intervention for any student whose attendance drops below the 70% mark, even if their marks are currently passing.
3. **Data-Driven Review:** The database structure should be used for term-by-term tracking to enable real-time risk assessment rather than waiting until the end of a cycle.