Admin_attendance_analysis.php: (NOT USED)

```
// Fetch faculty performance data
SELECT f.Faculty_Name,
        AVG(m.Obtained_Marks) AS Average_Marks,
        COUNT(DISTINCT m.Student_ID) AS Students_Taught,
        COUNT(DISTINCT c.Course_ID) AS Courses_Taught
    FROM marks m
    JOIN courses c ON m.Course_ID = c.Course_ID
    JOIN faculties f ON c.course_faculty_id = f.Faculty_ID
    GROUP BY f.Faculty_Name
--To print average marks given by each faculty
```

Admin_attendance_analysis.php:

```
// Fetch attendance data
SELECT c.Course_Name,
        MONTH(a.attendance_date) as Month,
        SUM(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) AS Classes_Attended,
        COUNT(*) AS Total_Classes,
        ROUND(SUM(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS
Attendance_Percentage
    FROM attendance a
    JOIN courses c ON a.Course_ID = c.Course_ID
    GROUP BY c.Course_Name, MONTH(a.attendance_date)
```

calculates the attendance percentage for each course, broken down by month. It counts the number of classes attended and the total number of classes held for each course, then calculates the attendance percentage by dividing the number of classes attended by the total number of classes and multiplying by 100.

Admin_course_performance.php:

```
// Fetch course performance dataSELECT c.Course_Name,
        AVG(m.Obtained_Marks) AS Average_Marks,
        MAX(m.Obtained_Marks) AS Highest_Marks,
        MIN(m.Obtained_Marks) AS Lowest_Marks
    FROM marks m
    JOIN courses c ON m.Course_ID = c.Course_ID
    GROUP BY c.Course_Name
```

Pattern Identification queries:

1. Attendance Patterns by Day of the Week:

```
SELECT DAYOFWEEK(attendance_date) AS Day, AVG(Status = 'Present') AS Average_Attendance
FROM attendance
GROUP BY DAYOFWEEK(attendance_date)
ORDER BY Average
```

2. Average Marks by Semester

```
SELECT e.Semester, AVG(m.Obtained_Marks) AS Average_Marks
FROM marks m
JOIN enrollment_status e ON m.Student_ID = e.Student_ID AND m.Course_ID = e.Course_ID
GROUP BY e.Semester
ORDER BY e.Semester;
```

3. Faculty Activity Patterns

```
SELECT c.course_faculty_id AS Faculty_ID, COUNT(*) AS Number_of_Submissions
FROM marks m
JOIN courses c ON m.Course_ID = c.Course_ID
GROUP BY c.course_faculty_id
ORDER BY Number_of_Submissions DESC;
```

4. Student Average Attendance Patterns

```
SELECT s.Student_Name, AVG(a.Status = 'Present') AS Average_Attendance
FROM attendance a
```

```
JOIN students s ON a.Student_ID = s.Student_ID
GROUP BY s.Student_Name
ORDER BY s.Student_Name;
```

5. Department Popularity

```
SELECT d.department_name, COUNT(s.Student_ID) AS Student_Count
FROM department d
JOIN students s ON d.department_id = s.department_id
GROUP BY d.department_name
ORDER BY Student_Count DESC;
```

6. Average Marks Distribution by Course

```
SELECT c.Course_Name, AVG(m.Obtained_Marks) AS Average_Marks
FROM marks m
JOIN courses c ON m.Course_ID = c.Course_ID
GROUP BY c.Course_Name
ORDER BY c.Course_Name;
```

7. Department Performance

```
SELECT d.department_name, AVG(m.Obtained_Marks) AS Average_Marks
FROM marks m
JOIN students s ON m.Student_ID = s.Student_ID
JOIN department d ON s.department_id = d.department_id
GROUP BY d.department_name
ORDER BY Average_Marks DESC;
```

8. Exam Performance by Type

```
SELECT et.Exam_Type_Name, AVG(m.Obtained_Marks) AS Average_Marks
FROM marks m
JOIN examinations e ON m.Exam_ID = e.Exam_ID
JOIN exam_types et ON e.Exam_Type_ID = et.Exam_Type_ID
GROUP BY et.Exam_Type_Name
ORDER BY Average_Marks DESC;
```

Faculty_provide_marks.php:

```
SELECT s.Student_ID, s.Student_Name
                FROM students s
                JOIN enrollment_status e ON s.Student_ID = e.Student_ID
                WHERE e.Course_ID = ? AND s.department_id = (SELECT department_id FROM courses
WHERE Course_ID = ?)
```

Parent_view_attendance.php:

```
// Fetch attendance data
$sql = "SELECT a.*, c.Course_Name
    FROM attendance a
    JOIN courses c ON a.Course_ID = c.Course_ID
    WHERE a.Student_ID = ?";
```

Generate charts.php:

```
SELECT c.Course_Name, a.Status, COUNT(a.Status) as count
    FROM attendance a
    JOIN courses c ON a.Course_ID = c.Course_ID
    WHERE a.Student_ID = ?
    GROUP BY c.Course_Name, a.Status
```

Generate_report.php:

// Fetch academic data

```
SELECT
    c.Course_Name,
    et.Exam_Type_Name,
    m.Obtained_Marks,
    m.Total_Marks,
    (SELECT MAX(Obtained_Marks) FROM marks WHERE Course_ID = m.Course_ID AND Exam_ID =
m.Exam_ID) AS Highest_Marks
  FROM
    marks m
  JOIN
```

```
    courses c ON m.Course_ID = c.Course_ID
  JOIN
    examinations e ON m.Exam_ID = e.Exam_ID
  JOIN
    exam_types et ON e.Exam_Type_ID = et.Exam_Type_ID
  WHERE
    m.Student_ID = ?";
```

admin_attendance_analysis.php:

```
SELECT
    c.Course_Name,
    COUNT(*) AS Total_Classes,
    SUM(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) AS Classes_Attended,
    ROUND(SUM(CASE WHEN a.Status = 'Present' THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS
Attendance_Percentage
  FROM
    attendance a
  JOIN
    courses c ON a.Course_ID = c.Course_ID
  WHERE
    a.Student_ID = ?
  GROUP BY
    a.Course_ID";
```

Handle Mark submission:

```
"INSERT INTO marks (Exam_ID, Course_ID, Student_ID, Total_Marks, Obtained_Marks)
      VALUES (?, ?, ?, ?, ?) ON DUPLICATE KEY UPDATE Total_Marks = VALUES(Total_Marks),
Obtained_Marks = VALUES(Obtained_Marks)");
```

INSERT INTO marks (Exam_ID, Course_ID, Student_ID, Total_Marks, Obtained_Marks): This specifies the columns into which the data will be inserted.

VALUES (?, ?, ?, ?, ?): This is a placeholder for the values to be inserted. Each "?" represents a parameter that will be bound later.

ON DUPLICATE KEY UPDATE: This clause specifies what to do if there is a duplicate entry (i.e., if there's already a record with the same combination of Exam_ID, Course_ID, and Student_ID).

Total_Marks = VALUES(Total_Marks), Obtained_Marks = VALUES(Obtained_Marks): This sets the values of the Total_Marks and Obtained_Marks columns to the new values provided in case of a duplicate entry.

Manage_attendance.php:

```
// Fetch students enrolled in this course

$sql = "SELECT s.Student_ID, s.Student_Name

    FROM students s

    JOIN enrollment_status e ON s.Student_ID = e.Student_ID

    WHERE e.Course_ID = ? AND s.department_id =

      (SELECT department_id FROM courses WHERE Course_ID = ?)";
```

Record attendance php:

```
// Prepare the SQL query for inserting or updating attendance records

$sql = "INSERT INTO attendance (student_id, course_id, attendance_date, status) VALUES (?, ?, ?, ?)

    ON DUPLICATE KEY UPDATE status = VALUES(status)";
```

Update attendance php:

```
// Prepare the SQL query for inserting or updating attendance records

$sql = "INSERT INTO attendance (student_id, course_id, attendance_date, status)

    VALUES (?, ?, ?, ?)

    ON DUPLICATE KEY UPDATE status = VALUES(status)";
```

Parent_view_statistics.php:

```
// Fetch academic data for charts

$marks_sql = "

  SELECT

    c.Course_Name,

    et.Exam_Type_Name,

    m.Obtained_Marks
```

```
  FROM
    marks m
  JOIN
    courses c ON m.Course_ID = c.Course_ID
  JOIN
    examinations e ON m.Exam_ID = e.Exam_ID
  JOIN
    exam_types et ON e.Exam_Type_ID = et.Exam_Type_ID
  WHERE
    m.Student_ID = ?";
```

Parent_view_marks:

```
// Updated Query
$sql = "
  SELECT
                m.Total_Marks,
    m.Obtained_Marks,
    c.Course_Name,
    et.Exam_Type_Name,
    (SELECT MAX(Obtained_Marks) FROM marks WHERE Course_ID = m.Course_ID AND Exam_ID =
m.Exam_ID) AS Highest_Marks,
    f.Faculty_Name
  FROM
    marks m
  JOIN
    courses c ON m.Course_ID = c.Course_ID
  JOIN
    examinations e ON m.Exam_ID = e.Exam_ID
  JOIN
```

```
    exam_types et ON e.Exam_Type_ID = et.Exam_Type_ID
  JOIN
    faculties f ON c.course_faculty_id = f.Faculty_ID
  WHERE
    m.Student_ID = ?
";
```

Update attendance php:

```
$sql = "SELECT s.Student_ID, s.Student_Name, a.Status
    FROM students s
    JOIN enrollment_status e ON s.Student_ID = e.Student_ID
    LEFT JOIN attendance a ON a.Student_ID = s.Student_ID AND a.course_id = ? AND
a.attendance_date = ?
    WHERE e.Course_ID = ? AND s.department_id =
      (SELECT department_id FROM courses WHERE Course_ID = ?)";
```

Student view marks php:

```
$stmt = $con->prepare("SELECT c.Course_ID, c.Course_Name FROM courses c
                JOIN enrollment_status e ON c.Course_ID = e.Course_ID
                WHERE e.Student_ID = ? AND c.department_id = ?");
```

Save updated attendance php:

```
// Prepare the SQL query for inserting or updating attendance records
$sql = "INSERT INTO attendance (student_id, course_id, attendance_date, status)
    VALUES (?, ?, ?, ?)
    ON DUPLICATE KEY UPDATE status = VALUES(status)";
```

Student view attendance php:

```
$sql = "SELECT a.Status, c.Course_Name, a.attendance_date FROM attendance a
    JOIN courses c ON a.Course_ID = c.Course_ID
    WHERE a.Student_ID = ? AND a.Course_ID = ?";
```

Student view course mark php:

```
// Fetch marks for the student for a specific course
$stmt = $con->prepare("SELECT e.Exam_Type_Name, m.Total_Marks, m.Obtained_Marks
                FROM marks m
```

```
JOIN examinations ex ON m.Exam_ID = ex.Exam_ID
JOIN exam_types e ON ex.Exam_Type_ID = e.Exam_Type_ID
WHERE m.Student_ID = ? AND m.Course_ID = ?");
```

Student view courses:

```
$sql = "SELECT distinct c.* FROM courses c
    JOIN enrollment_status e ON c.Course_ID = e.Course_ID
    WHERE c.department_id = ?";
```

Creation of Users Table:

```
CREATE TABLE users (
user_id INT AUTO_INCREMENT PRIMARY KEY,
username VARCHAR(255) NOT NULL UNIQUE,
password VARCHAR(255) NOT NULL,
role ENUM('admin', 'student', 'parent', 'faculty'),
picture VARCHAR(250)
);
```

Creation of Department Table: -- Create department table

```
CREATE TABLE department (
department_id INT PRIMARY KEY,
department_name VARCHAR(255)
);
```

Creation of Students Table:

```
CREATE TABLE students (
Student_ID INT AUTO_INCREMENT PRIMARY KEY,
Student_Name VARCHAR(255) NOT NULL,
Enrollment_Status VARCHAR(100) NOT NULL,
department VARCHAR(255),
department_id INT,
CONSTRAINT fk_department FOREIGN KEY (department_id) REFERENCES
```

```
department(department_id)
);
```

44

Creation of Parents Table:

```
CREATE TABLE parents (
parent_id INT AUTO_INCREMENT PRIMARY KEY,
parent_name VARCHAR(255) NOT NULL,
student_id INT,
email VARCHAR(255),
phone VARCHAR(255),
CONSTRAINT parents_ibfk_1 FOREIGN KEY (student_id) REFERENCES
students(Student_ID)
);
```

Creation of Notices Table:

```
CREATE TABLE notices (
notice_id INT AUTO_INCREMENT PRIMARY KEY,
notice_date DATE,
notice_details TEXT
);
```

Creation of Courses Table:

```
CREATE TABLE courses (
Course_ID INT AUTO_INCREMENT PRIMARY KEY,
Course_Name VARCHAR(255) NOT NULL,
Course_Details TEXT,
Department VARCHAR(255),
department_id INT,
course_faculty_id INT,
```

45

```
CONSTRAINT courses_ibfk_1 FOREIGN KEY (department_id) REFERENCES
```

department(department_id),

CONSTRAINT fk_course_department FOREIGN KEY (course_faculty_id)

REFERENCES faculties(Faculty_ID)

);

Creation of Enrollment_Status Table:

CREATE TABLE enrollment_status (

Enrollment_ID INT AUTO_INCREMENT PRIMARY KEY,

Student_ID INT,

Course_ID INT,

Semester VARCHAR(255),

CONSTRAINT enrollment_status_ibfk_1

REFERENCES students(Student_ID),

CONSTRAINT

enrollment_status_ibfk_2

REFERENCES courses(Course_ID)

);

Creation of Faculties Table:

CREATE TABLE faculties (

FOREIGN

FOREIGN

Faculty_ID INT AUTO_INCREMENT PRIMARY KEY,

Faculty_Name VARCHAR(255) NOT NULL,

Department VARCHAR(255),

Email_ID VARCHAR(255),

Phone_Number VARCHAR(255)

);

KEY

KEY

(Student_ID)

(Course_ID)

46

Creation of Examinations Table:

```
CREATE TABLE examinations (
Exam_ID INT AUTO_INCREMENT PRIMARY KEY,
Semester VARCHAR(255),
Course_ID INT,
Schedule DATETIME,
Exam_Type_ID INT,
CONSTRAINT examinations_ibfk_1 FOREIGN KEY (Course_ID) REFERENCES
courses(Course_ID),
CONSTRAINT examinations_ibfk_2
REFERENCES exam_types(Exam_Type_ID)
);
```

Creation of Attendance Table:

```
CREATE TABLE attendance (
FOREIGN
KEY
Attendance_ID INT AUTO_INCREMENT PRIMARY KEY,
Student_ID INT,
Course_ID INT,
Timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
Status ENUM('Present', 'Absent'),
attendance_date DATE,
(Exam_Type_ID)
CONSTRAINT attendance_ibfk_1 FOREIGN KEY (Student_ID) REFERENCES
students(Student_ID),
CONSTRAINT attendance_ibfk_2 FOREIGN KEY (Course_ID) REFERENCES
courses(Course_ID)
```

);

47

Creation of Exam_Types table:

```
CREATE TABLE exam_types (
Exam_Type_ID INT AUTO_INCREMENT PRIMARY KEY,
Exam_Type_Name VARCHAR(255) NOT NULL
);
```

Creation of Marks Table:

```
Marks_ID INT AUTO_INCREMENT PRIMARY KEY,
Exam_ID INT,
Course_ID INT,
Total_Marks INT,
Obtained_Marks INT,
Student_ID INT,
CONSTRAINT fk_exam_id FOREIGN KEY (Exam_ID) REFERENCES
examinations(Exam_ID),
CONSTRAINT marks_ibfk_2 FOREIGN KEY (Course_ID) REFERENCES
courses(Course_ID),
CONSTRAINT marks_ibfk_3 FOREIGN KEY (Student_ID) REFERENCES
students(Student_ID)
);
```