Title: "Sunshine High School Information System"
Majo Area: Education System
Minor Area: Student, Teacher information, Attendance, Courses

Description of the Database:

The Sunshine High School Information System database is meticulously structured to encompass various scopes within the educational institution, aiming to streamline administrative processes and enhance the overall learning experience. Below is a detailed description of the key components and scopes within the database:

1. **Student Information Management**

Students Table

The cornerstone of the system, the "Students" table, captures comprehensive student information. It includes attributes such as student ID, first and last names, date of birth, gender, address, and parent contact information. This scope allows for precise identification and communication with each student.

Classes Table

The "Classes" table is designed to manage different classes within the school, each identified by a unique class ID. It includes details such as the class ID and associated fee amounts. This scope organizes the academic landscape by providing specific information about each class.

Fees Table

The "Fees" table intricately links students, classes, and financial transactions. It includes the fee ID, student ID, class ID, fee month, and fee amount. This scope ensures transparent financial records, tracking each student's financial obligations in association with their enrolled classes.

2. Academic Management

Courses Table

The "Courses" table facilitates the management of various academic courses offered by the school. It includes a unique course ID and the course name. This scope ensures efficient organization and tracking of the diverse courses available to students.

Teachers Table

The "Teachers" table captures details about the faculty, including a unique teacher ID, first and last names. This scope enables the association of teachers with the courses they lead, contributing to the academic growth of students.

Enrollments Table

The "Enrollments" table serves as a pivotal scope in managing student academic pursuits. It records the enrollment ID, student ID, and course ID, creating a seamless link between students and the courses they are enrolled in.

3. Financial Management

Monthly_Salary_Teachers Table

The "Monthly_Salary_Teachers" table focuses on the financial aspect of the teaching staff. It includes the salary ID, teacher ID, salary month, and salary amount. This scope ensures timely and transparent disbursement of salaries to teachers.

4. Attendance and Engagement Tracking

Attendance Table

The "Attendance" table contributes to accountability and engagement by maintaining attendance records. It includes the attendance_id, student ID, course ID, date, and attendance status. This scope serves as a valuable tool for assessing student participation and engagement.

Overall System Integration

The interconnectivity of these tables creates a cohesive system where student information, academic details, financial transactions, and attendance records are seamlessly integrated. This design empowers administrators with a holistic view of the educational landscape, facilitating strategic decision-making and contributing to the success of Sunshine High School as a beacon of education.

Story (Detailed Description of the Database):

Once upon a time in the bustling city, Sunshine High School stood proudly as an educational beacon, nurturing young minds and shaping the future. The heart of this institution was its comprehensive student information system, meticulously designed to streamline administrative processes and enhance the overall learning experience.

The central pillar of the system was the "Students" table, where each student was uniquely identified by a student ID. Their personal details, including first and last names, date of birth, gender, address, and parent contact information, were stored with utmost care. To further organize the academic landscape, students were assigned to various classes, denoted by the "class_id" foreign key referencing the "Classes" table.

Speaking of classes, the "Classes" table not only held information about different classes but also recorded the corresponding fee amounts. The diligent administrators ensured that the financial aspects were meticulously tracked in the "Fees" table, detailing the fee id, student id, class id, fee month, and the corresponding fee amount.

The academic journey of students extended beyond just classes; it encompassed a variety of courses, each managed through the "Courses" table. A vibrant faculty, as seen in the "Teachers" table, led these courses, contributing their expertise to the students' intellectual growth.

Enrollment, a pivotal process in any school, was elegantly handled by the "Enrollments" table. This table maintained a record of students enrolled in specific courses, creating a seamless link between students and their academic pursuits.

Recognizing the hard work and dedication of the teaching staff, the school implemented the "Monthly_Salary_Teachers" table. This table chronicled the monthly disbursement of salaries to teachers, emphasizing transparency and efficiency in financial transactions.

In the spirit of accountability and engagement, the school also implemented the "Student_Rolls" table. Here, attendance records were diligently maintained, tracking student presence in various courses on specific dates. This not only facilitated accurate record-keeping but also served as a valuable tool for assessing student participation and engagement.

As the semesters unfolded, the school's administrators found themselves equipped with a robust system that not only managed the nuts and bolts of student and teacher information but also facilitated strategic decision-making. Sunshine High School, with its state-of-the-art information system, continued to be a beacon of education, nurturing young minds and fostering a vibrant learning community.

Expected query outputs from the project::

1. Retrieve the Names of All Students in a Specific Class:

In the bustling corridors of Sunshine High School, gather the names of students enrolled in the class "Math 101."

2. Find the Total Number of Students in Each Class:

Explore the diverse classes at Sunshine High School and discover the total number of students in each class.

3. List Courses Taught by a Specific Teacher:

Delve into the expertise of Teacher Alice Johnson at Sunshine High School by listing the courses she passionately teaches.

4. Get the Monthly Salary of a Specific Teacher:

Uncover the financial transparency at Sunshine High School by retrieving the monthly salary details for Teacher Alice Johnson.

5. Find Students with Outstanding Fees:

In the realm of academic pursuits at Sunshine High School, identify students with outstanding fees, ensuring financial harmony.

6. List Teachers and the Courses They Teach:

Navigate through the educational tapestry of Sunshine High School, unveiling the dedicated teachers and the diverse courses they impart.

7. Calculate Average Monthly Salary for All Teachers:

Dive into the fiscal landscape of Sunshine High School and compute the average monthly salary for the esteemed teaching faculty.

8. Find Students with Perfect Attendance:

Embark on a journey through the attendance records at Sunshine High School, identifying students with a perfect record of presence.

9. Get the Total Fees Collected for a Specific Class:

Peer into the financial ledger of the class "Math 101" at Sunshine High School, revealing the total fees collected.

9. Total Number of Students:

How many students are exist in the school

10. List of Students with Their Corresponding Fees (Including Those Who Haven't Paid)

List of students who paid their monthly fee also who can’t.

11.List of Students who didn’t pay FEE

Only those students who didn’t pay their Study fee.

12.Details of Courses and Enrolled Students (Including Courses Without Enrollees)

Details of all courses.

13. Top 5 Highest Paid Teachers

Table Schema:

Students = **(student_id, first_name, last_name, dob, gender, address, parent_contact, class_id)**

Teachers = **(teacher_id, first_name, last_name, dob, gender, address, contact_number, salary)**

Classes = **(class_id, class_name, fee_amount)**

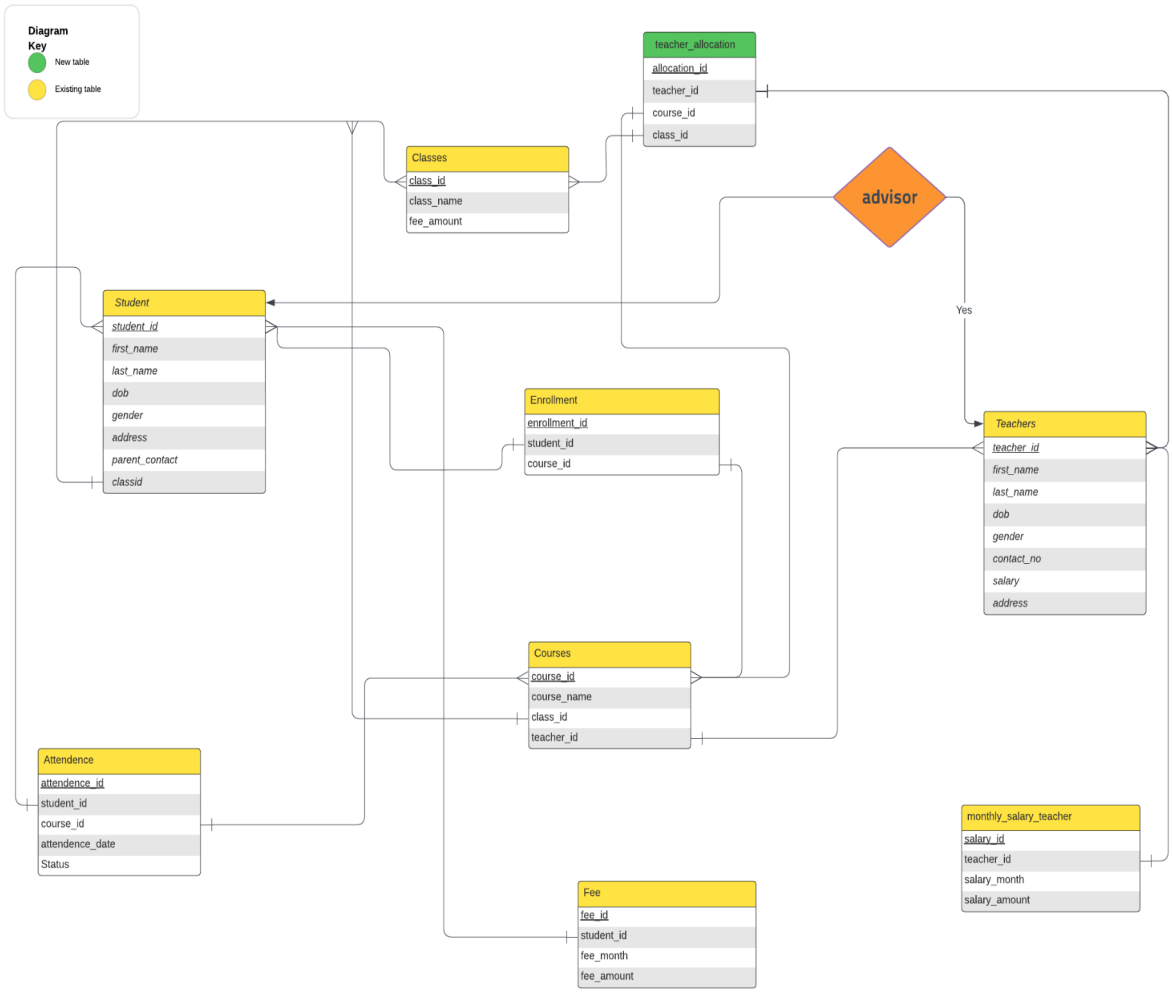
Courses = **(course_id, course_name, class_id, teacher_id)**

Enrollments = **(enrollment_id, student_id, course_id)**

Fees = **(fee_id, student_id, class_id, fee_month, fee_amount)**

Monthly_Salary_Teachers = **(salary_id, teacher_id, salary_month, salary_amount)**

Attendance= **(attendance_id, student_id, course_id, attendance_date, attendance_status)**



Functional Dependencies:

Functional dependencies are relationships between attributes in a database table. Identifying them is crucial for normalization and maintaining data integrity. Below is a list

of functional dependencies for the Sunshine High School Information System project based on the table structures:

Students Table

1. **student_id -> {first_name, last_name, dob, gender, address, parent_contact, class_id}**
   - The student ID uniquely determines all other attributes in the Students table.

Teachers Table

2. **teacher_id -> {first_name, last_name, dob, gender, address, contact_number, salary}**
   - The teacher ID uniquely determines all other attributes in the Teachers table.

Classes Table

3. **class_id -> {class_name, fee_amount}**
   - The class ID uniquely determines the class name and fee amount.

Courses Table

4. **course_id -> {course_name, class_id, teacher_id}**
   - The course ID uniquely determines the course name, associated class, and teacher.

Monthly_Salary_Teachers Table

5. **salary_id -> {teacher_id, salary_month, salary_amount}**
   - The salary ID uniquely determines the teacher, salary month, and salary amount.

Enrollments Table

6. **enrollment_id -> {student_id, course_id}**
   - The enrollment ID uniquely determines the student and course.

Fees Table

7. **fee_id -> {student_id, class_id, fee_month, fee_amount}**
   - The fee ID uniquely determines the student, associated class, fee month, and fee amount.

Attendance Table

8. **attendance_id -> {student_id, course_id, attendance_date, status}**
   - The attendance ID uniquely determines the student, course, attendance date, and status.

These functional dependencies are crucial for understanding how the attributes in each table relate to each other. They help guide database design and normalization processes to ensure data integrity and reduce redundancy. If there are additional attributes or relationships, you may need to adjust these dependencies accordingly.

Tables:

Students :

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	student_id	NUMBER	Primary key	Unique Value
2	first_name	VARCHAR2(50)		
3	last_name	VARCHAR2(50)		
4	dob			
5	gender	VARCHAR2(50)		
6	address	VARCHAR2(50)		
7	parent_contact	VARCHAR2(50)		Atomic Value
8	class_id	NUMBER	Foreign key	From Classes table
The Students table is the repository for essential student information at Sunshine High School. It stores details such as student names, date of birth, gender, address, parent contact information, and the class to which they are assigned. This information is crucial for administrative processes, class assignments, and overall student management.				

Teachers:

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	teacher_id	NUMBER	Primary key	Unique
2	first_name	VARCHAR2(50)		
3	last_name	VARCHAR2(50)		

4	dob	DATE		
5	gender	VARCHAR2(50)		
6	address	VARCHAR2(50)		
7	contact_number	VARCHAR2(50)		Atomic Value
8	salary	NUMBER	Foreign key	From Salary table
The Teachers table holds comprehensive information about the teaching staff at Sunshine High School. It includes details such as teacher names, date of birth, gender, address, contact numbers, and salary. This table facilitates the management of teacher-related information, including salary disbursement and overall staff administration.				

Classes :

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	class_id	NUMBER	Primary key	Unique
2	class_name	VARCHAR2(50)		
3	fee_amount	NUMBER		
The Classes table serves as a reference for different classes offered at Sunshine High School. It includes information about class names and associated fee amounts. This table helps in organizing and categorizing various courses, providing clarity on the financial aspects associated with each class.				

Courses :

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	course_id	NUMBER	Primary key	Unique
2	course_name	VARCHAR2(50)		
3	class_id	NUMBER	FK	From classes table

4	teacher_id	NUMBER	FK	From teacher table
The Courses table manages information about the academic courses offered at Sunshine High School. It includes details such as course names, the class to which the course belongs, and the teacher responsible for teaching the course. This table plays a vital role in structuring the academic curriculum and linking students and teachers to specific courses.				

Enrollments:

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	enrollment_id	NUMBER	PK	
2	student_id	NUMBER	FK	From students table
3	course_id	NUMBER	FK	From courses table
The Enrollments table maintains records of students enrolled in specific courses. It establishes a connection between students and their academic pursuits, allowing efficient tracking of student enrollment data for each course.				

Fees :

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	fee_id	NUMBER	PK	
2	student_id	NUMBER	FK	From students table
3	class_id	NUMBER	FK	From courses table
4	fee_month	Data		
5	fee_amount	NUMBER		

The Fees table is dedicated to tracking the financial transactions related to student fees at Sunshine High School. It includes information about fee id, student id, class id, fee month, and the corresponding fee amount. This table aids in financial management and ensures transparency in fee-related processes.

Monthly_Salary_Teachers :

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	salary_id	NUMBER	PK	
2	tracher_id	NUMBER	FK	From students table
3	salary_month	Date	FK	From courses table
4	salary_amount	NUMBER		

The Monthly_Salary_Teachers table documents the monthly disbursement of salaries to the teaching staff. It includes details such as salary id, teacher id, salary month, and the corresponding salary amount. This table supports financial transparency and efficient salary management for teachers.

Attendance:

S/N	Attribute	Data Type	Constraint(If any)	Comments
1	attendence_id	NUMBER	PK	
2	student_id	NUMBER	FK	From students table
3	course_id	NUMBER	FK	From courses table
4	Attendance_date	Date		
5	status	VARCHAR(20)		yes/no

The Monthly_Salary_Teachers table documents the monthly disbursement of salaries to the teaching staff. It includes details such as salary id, teacher

id, salary month, and the corresponding salary amount. This table supports financial transparency and efficient salary management for teachers.

Table-level Expected Constraints:

1. Students Table:
   - Primary Key Constraint: Ensures each student has a unique identifier (student_id).
   - Foreign Key Constraint (class_id): Links each student to a specific class.
2. Teachers Table:
   - Primary Key Constraint: Ensures each teacher has a unique identifier (teacher_id).
3. Classes Table:
   - Primary Key Constraint: Ensures each class has a unique identifier (class_id).
4. Courses Table:
   - Primary Key Constraint: Ensures each course has a unique identifier (course_id).
   - Foreign Key Constraints (class_id, teacher_id): Links each course to a specific class and teacher.
5. Enrollments Table:
   - Primary Key Constraint: Ensures each enrollment has a unique identifier (enrollment_id).
   - Foreign Key Constraints (student_id, course_id): Links each enrollment to a specific student and course.
6. Fees Table:
   - Primary Key Constraint: Ensures each fee transaction has a unique identifier (fee_id).
   - Foreign Key Constraints (student_id, class_id): Links each fee transaction to a specific student and class.
7. Monthly_Salary_Teachers Table:
   - Primary Key Constraint: Ensures each salary transaction has a unique identifier (salary_id).
   - Foreign Key Constraint (teacher_id): Links each salary transaction to a specific teacher.
8. Attendance Table:
   - Primary Key Constraint: Ensures each attendance record has a unique identifier (attendance_id).
   - Foreign Key Constraints (student_id, course_id): Links each attendance record to a specific student and course.

Project-level Expected Constraints:

1. Data Integrity:
   - Ensure data integrity through appropriate foreign key relationships, preventing orphan records.
   - Enforce constraints to maintain accurate and meaningful data across all tables.

2. Security:
   - Implement role-based access control to restrict access to sensitive information.
   - Encrypt sensitive data to protect student and teacher information.

3. Scalability:
   - Design the database to handle potential growth in the number of students, teachers, and courses.
   - Optimize queries and indexes for efficient performance as the dataset expands.

4. Auditability:
   - Implement logging mechanisms to track changes to critical data, providing an audit trail for accountability.

5. Consistency:
   - Enforce consistent naming conventions and coding standards to enhance readability and maintainability.

6. Backup and Recovery:
   - Establish regular backup procedures to prevent data loss and ensure quick recovery in case of system failures.

7. Normalization:
   - Design the database in at least 3rd Normal Form (3NF) to minimize redundancy and maintain data consistency.

8. Documentation:
   - Maintain comprehensive documentation for the database schema, constraints, and any stored procedures, facilitating ease of understanding and future development.

These constraints collectively contribute to the reliability, security, and efficiency of the Sunshine High School Information System database.

SQLs to implement the project with example outputs:

Create Table SQL:

```
-- Create the Students table
    CREATE TABLE Students (
      student_id NUMBER PRIMARY KEY,
      first_name VARCHAR2(50),
      last_name VARCHAR2(50),
      dob DATE,
      gender VARCHAR2(10),
      address VARCHAR2(100),
      parent_contact VARCHAR2(20),
      class_id NUMBER,
      FOREIGN KEY (class_id) REFERENCES Classes(class_id)
    );

-- Create the Teachers table
    CREATE TABLE Teachers (
      teacher_id NUMBER PRIMARY KEY,
      first_name VARCHAR2(50),
      last_name VARCHAR2(50),
      dob DATE,
      gender VARCHAR2(10),
      address VARCHAR2(100),
      contact_number VARCHAR2(20),
      salary NUMBER
    );

-- Create the Classes table
    CREATE TABLE Classes (
      class_id NUMBER PRIMARY KEY,
      class_name VARCHAR2(50),
      fee_amount NUMBER
    );

-- Create the Courses table
    CREATE TABLE Courses (
```

```
    course_id NUMBER PRIMARY KEY,
    course_name VARCHAR2(100),
    class_id NUMBER,
    teacher_id NUMBER,
    FOREIGN KEY (class_id) REFERENCES Classes(class_id),
    FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);

-- Create the Enrollments table
CREATE TABLE Enrollments (
    enrollment_id NUMBER PRIMARY KEY,
    student_id NUMBER,
    course_id NUMBER,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);

-- Create the Fees table
CREATE TABLE Fees (
    fee_id NUMBER PRIMARY KEY,
    student_id NUMBER,
    class_id NUMBER,
    fee_month DATE,
    fee_amount NUMBER,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (class_id) REFERENCES Classes(class_id)
);

-- Create the Monthly Salary table for Teachers
CREATE TABLE Monthly_Salary_Teachers (
    salary_id NUMBER PRIMARY KEY,
    teacher_id NUMBER,
    salary_month DATE,
    salary_amount NUMBER,
    FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);

-- Create the Student Rolls table
CREATE TABLE Attendance (
    attendance_id NUMBER PRIMARY KEY,
    student_id NUMBER,
    course_id NUMBER,
    attendance_date DATE,
    status VARCHAR2(20),
    CONSTRAINT fk_attendance_students
        FOREIGN KEY (student_id) REFERENCES Students(student_id),
```

```
    CONSTRAINT fk_attendance_courses
      FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

SQL for Insertion in Table :

Teacher:

-- Inserting the first record

```
INSERT INTO Teachers (teacher_id, first_name, last_name, dob, gender, address,
contact_number, salary)
VALUES (101, 'Alice', 'Johnson', TO_DATE('1980-05-20', 'YYYY-MM-DD'), 'Female',
'111 Maple Ave', '555-1234', 6000);
```

-- Inserting the second record

```
INSERT INTO Teachers (teacher_id, first_name, last_name, dob, gender, address,
contact_number, salary)
VALUES (102, 'Robert', 'Smith', TO_DATE('1975-10-15', 'YYYY-MM-DD'), 'Male', '222
Oak Ave', '555-5678', 5500);
```

-- Inserting the third record

```
INSERT INTO Teachers (teacher_id, first_name, last_name, dob, gender, address,
contact_number, salary)
VALUES (103, 'Olivia', 'Davis', TO_DATE('1982-03-08', 'YYYY-MM-DD'), 'Female', '333
Pine Ave', '555-9876', 5200);
```

-- Inserting the fourth record

```
INSERT INTO Teachers (teacher_id, first_name, last_name, dob, gender, address,
contact_number, salary)
VALUES (104, 'William', 'Brown', TO_DATE('1978-08-25', 'YYYY-MM-DD'), 'Male', '444
Elm Ave', '555-4321', 5800);
```

-- Inserting the fifth record

```
INSERT INTO Teachers (teacher_id, first_name, last_name, dob, gender, address,
contact_number, salary)
VALUES (105, 'Emma', 'Miller', TO_DATE('1985-01-30', 'YYYY-MM-DD'), 'Female', '555
Birch Ave', '555-8765', 5600);
```

Classes:

-- Inserting the first record

```
INSERT INTO Classes (class_id, class_name, fee_amount)
VALUES (101, 'Math 101', 200);
```

-- Inserting the second record

```
INSERT INTO Classes (class_id, class_name, fee_amount)
VALUES (102, 'History 202', 250);
```

-- Inserting the third record

```
INSERT INTO Classes (class_id, class_name, fee_amount)
VALUES (103, 'Science Lab', 300);
```

-- Inserting the fourth record

```
INSERT INTO Classes (class_id, class_name, fee_amount)
VALUES (104, 'English 301', 220);
```

-- Inserting the fifth record

```
INSERT INTO Classes (class_id, class_name, fee_amount)
VALUES (105, 'Art 102', 180);
```

Courses:

-- Inserting the first record

```
INSERT INTO Courses (course_id, course_name, class_id, teacher_id)
VALUES (1, 'Mathematics', 101, 101);
```

-- Inserting the second record

```
INSERT INTO Courses (course_id, course_name, class_id, teacher_id)
VALUES (2, 'History of Civilization', 102, 102);
```

-- Inserting the third record

```
INSERT INTO Courses (course_id, course_name, class_id, teacher_id)
VALUES (3, 'Biology Lab', 103, 103);
```

-- Inserting the fourth record

```
INSERT INTO Courses (course_id, course_name, class_id, teacher_id)
VALUES (4, 'English Literature', 104, 104);
```

-- Inserting the fifth record

```
INSERT INTO Courses (course_id, course_name, class_id, teacher_id)
VALUES (5, 'Art Appreciation', 105, 105);
```

Monthly_Salary_Teachers Table:

-- Inserting the first record

```
INSERT INTO Monthly_Salary_Teachers (salary_id, teacher_id, salary_month,
salary_amount)
VALUES (1, 101, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 6000);
```

-- Inserting the second record

```
INSERT INTO Monthly_Salary_Teachers (salary_id, teacher_id, salary_month,
salary_amount)
VALUES (2, 102, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 5500);

-- Inserting the third record
INSERT INTO Monthly_Salary_Teachers (salary_id, teacher_id, salary_month,
salary_amount)
VALUES (3, 103, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 5200);

-- Inserting the fourth record
INSERT INTO Monthly_Salary_Teachers (salary_id, teacher_id, salary_month,
salary_amount)
VALUES (4, 104, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 5800);

-- Inserting the fifth record
INSERT INTO Monthly_Salary_Teachers (salary_id, teacher_id, salary_month,
salary_amount)
VALUES (5, 105, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 5600);
```

Enrollments:

```
-- Inserting the first record
INSERT INTO Enrollments (enrollment_id, student_id, course_id)
VALUES (1, 1, 1);

-- Inserting the second record
INSERT INTO Enrollments (enrollment_id, student_id, course_id)
VALUES (2, 2, 2);

-- Inserting the third record
INSERT INTO Enrollments (enrollment_id, student_id, course_id)
VALUES (3, 3, 3);

-- Inserting the fourth record
INSERT INTO Enrollments (enrollment_id, student_id, course_id)
VALUES (4, 4, 4);

-- Inserting the fifth record
INSERT INTO Enrollments (enrollment_id, student_id, course_id)
VALUES (5, 5, 5);
```

Fees:

```
-- Inserting the first record
INSERT INTO Fees (fee_id, student_id, class_id, fee_month, fee_amount)
VALUES (1, 1, 101, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 200);
```

-- Inserting the second record

```
INSERT INTO Fees (fee_id, student_id, class_id, fee_month, fee_amount)
VALUES (2, 2, 102, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 250);
```

-- Inserting the third record

```
INSERT INTO Fees (fee_id, student_id, class_id, fee_month, fee_amount)
VALUES (3, 3, 103, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 300);
```

-- Inserting the fourth record

```
INSERT INTO Fees (fee_id, student_id, class_id, fee_month, fee_amount)
VALUES (4, 4, 104, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 220);
```

-- Inserting the fifth record

```
INSERT INTO Fees (fee_id, student_id, class_id, fee_month, fee_amount)
VALUES (5, 5, 105, TO_DATE('2023-02-01', 'YYYY-MM-DD'), 180);
```

Attendance:

-- Inserting the first record

```
INSERT INTO Attendance (attendance_id, student_id, course_id, attendance_date,
status)
VALUES (1, 1, 1, TO_DATE('2023-02-05', 'YYYY-MM-DD'), 'Present');
```

-- Inserting the second record

```
INSERT INTO Attendance (attendance_id, student_id, course_id, attendance_date,
status)
VALUES (2, 2, 2, TO_DATE('2023-02-05', 'YYYY-MM-DD'), 'Present');
```

-- Inserting the third record

```
INSERT INTO Attendance (attendance_id, student_id, course_id, attendance_date,
status)
VALUES (3, 3, 3, TO_DATE('2023-02-05', 'YYYY-MM-DD'), 'Absent');
```

-- Inserting the fourth record

```
INSERT INTO Attendance (attendance_id, student_id, course_id, attendance_date,
status)
VALUES (4, 4, 4, TO_DATE('2023-02-05', 'YYYY-MM-DD'), 'Present');
```

-- Inserting the fifth record

```
INSERT INTO Attendance (attendance_id, student_id, course_id, attendance_date,
status)
VALUES (5, 5, 5, TO_DATE('2023-02-05', 'YYYY-MM-DD'), 'Absent');
```

Students:

-- Inserting the first record

```
INSERT INTO Students (student_id, first_name, last_name, dob, gender, address,
parent_contact, class_id)
VALUES (1, 'John', 'Doe', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'Male', '123 Main
St', '123-456-7890', 101);
```

-- Inserting the second record

```
INSERT INTO Students (student_id, first_name, last_name, dob, gender, address,
parent_contact, class_id)
VALUES (2, 'Jane', 'Smith', TO_DATE('1999-05-15', 'YYYY-MM-DD'), 'Female', '456
Oak St', '987-654-3210', 102);
```

-- Inserting the third record

```
INSERT INTO Students (student_id, first_name, last_name, dob, gender, address,
parent_contact, class_id)
VALUES (3, 'Alex', 'Johnson', TO_DATE('2001-03-20', 'YYYY-MM-DD'), 'Male', '789
Pine St', '555-123-4567', 103);
```

-- Inserting the fourth record

```
INSERT INTO Students (student_id, first_name, last_name, dob, gender, address,
parent_contact, class_id)
VALUES (4, 'Emily', 'Wilson', TO_DATE('2002-07-10', 'YYYY-MM-DD'), 'Female', '987
Elm St', '333-999-8888', 101);
```

-- Inserting the fifth record

```
INSERT INTO Students (student_id, first_name, last_name, dob, gender, address,
parent_contact, class_id)
VALUES (5, 'Daniel', 'Miller', TO_DATE('2000-12-05', 'YYYY-MM-DD'), 'Male', '543
Birch St', '777-222-1111', 102);
```

Expected Query:

Example Query 1: Retrieve the Names of All Students in a Specific Class

```
SELECT first_name, last_name
FROM Students
WHERE class_id = 101;
```

Output:

	FIRST_NAME	LAST_NAME
1	John	Doe
2	Emily	Wilson

Example Query 2: Find the Total Number of Students in Each Class

(Aggregate Function)

```
SELECT class_id, COUNT(student_id) AS total_students
FROM Students
GROUP BY class_id;
```

output:

	CLASS_ID	TOTAL_STUDENTS
1	101	2
2	103	1
3	102	2

Example Query 3: List Courses Taught by a Specific Teacher

```
SELECT c.course_name
FROM Courses c
WHERE c.teacher_id = 101;
```

Output:

	COURSE_NAME
1	Mathematics

Example Query 4: Get the Monthly Salary of a Specific Teacher

```
SELECT salary_month, salary_amount
FROM Monthly_Salary_Teachers
WHERE teacher_id = 101;
```

Output:

	SALARY_MONTH	SALARY_AMOUNT
1	01-FEB-23	6000

Example Query 5: Find Students with Outstanding Fees

(Outer Join)

```
SELECT s.first_name, s.last_name, f.fee_month, f.fee_amount
FROM Students s
```

```
LEFT JOIN Fees f ON s.student_id = f.student_id
WHERE f.fee_amount > 0 OR f.fee_amount IS NULL;
```

Output:

	FIRST_NAME	LAST_NAME	FEE_MONTH	FEE_AMOUNT
1	John	Doe	01-FEB-23	200
2	Jane	Smith	01-FEB-23	250
3	Alex	Johnson	01-FEB-23	300
4	Emily	Wilson	01-FEB-23	220
5	Daniel	Miller	01-FEB-23	180

Example Query 6: List Teachers and the Courses They Teach

(Outer Join)

```
SELECT t.first_name, t.last_name, c.course_name
FROM Teachers t
JOIN Courses c ON t.teacher_id = c.teacher_id;
```

Output:

	FIRST_NAME	LAST_NAME	COURSE_NAME
1	Alice	Johnson	Mathematics
2	Robert	Smith	History of Civilization
3	Olivia	Davis	Biology Lab
4	William	Brown	English Literature
5	Emma	Miller	Art Appreciation

Example Query 7: Calculate Average Monthly Salary for All Teachers

(Aggregate Function)

```
SELECT AVG(salary_amount) AS avg_salary
FROM Monthly_Salary_Teachers;
```

Output:

	AVG_SALARY
1	5620

Example Query 8: Find Students with Perfect Attendance

```
SELECT
   Attendance.attendance_id,
```

```
    Attendance.attendance_date,
    Attendance.status
FROM
    Attendance
JOIN
    Students ON Attendance.student_id = Students.student_id
WHERE
    Students.student_id = 1;
```

Output:

	ATTENDANCE_ID	ATTENDANCE_DATE	STATUS
1	1	05-FEB-23	Present

Example Query 9: Get the Total Fees Collected for a Specific Class

(Aggregate Function)

```
SELECT class_id, SUM(fee_amount) AS total_fees_collected
FROM Fees
WHERE class_id = 101
GROUP BY class_id;
```

Output:

	CLASS_ID	TOTAL_FEES_COLLECTED
1	101	200

Example Query 9: We want to retrieve all students along with their enrollment information and attendance information if available.

(Outer Join)

```
SELECT
    Students.student_id,
    Students.first_name,
    Students.last_name,
    Enrollments.enrollment_id,
    Enrollments.course_id,
    Attendance.attendance_id,
    Attendance.attendance_date,
    Attendance.status
FROM
    Students
LEFT JOIN
    Enrollments ON Students.student_id = Enrollments.student_id
```

```
LEFT JOIN
   Attendance ON Students.student_id = Attendance.student_id
WHERE
   Students.student_id = 2;
```

Output:

	STUDENT_ID	FIRST_NAME	LAST_NAME	ENROLLMENT_ID	COURSE_ID	ATTENDANCE_ID	ATTENDANCE_DATE	STATUS
1	2	Jane	Smith	2	2	2	05-FEB-23	Present

Let's analyze the normalization forms for your database tables:

1. Courses Table:

- No apparent issues. Each column seems to represent an atomic value.
- Assuming that `class_id` and `teacher_id` are the primary key, this table satisfies the requirements of 2NF and 3NF.

2. Teachers Table:

- No apparent issues. Each column seems to represent an atomic value.
- Assuming that `teacher_id` is the primary key, this table satisfies the requirements of 2NF and 3NF.

3. Classes Table:

- No apparent issues. Each column seems to represent an atomic value.
- Assuming that `class_id` is the primary key, this table satisfies the requirements of 2NF and 3NF.

4. Enrollments Table:

- Assuming that `enrollment_id` is the primary key.
- The table is likely in 1NF as each column seems to represent an atomic value.
- To achieve 2NF and 3NF, you need to ensure that non-prime attributes are fully functionally dependent on the primary key.

5. Fees Table:

- Assuming that `fee_id` is the primary key.
- The table is likely in 1NF as each column seems to represent an atomic value.

- To achieve 2NF and 3NF, you need to ensure that non-prime attributes are fully functionally dependent on the primary key.

6. Monthly_Salary_Teachers Table:
- Assuming that `salary_id` is the primary key.
- The table is likely in 1NF as each column seems to represent an atomic value.
- To achieve 2NF and 3NF, you need to ensure that non-prime attributes are fully functionally dependent on the primary key.

7. Student_Rolls Table:
- Assuming that `roll_id` is the primary key.
- The table is likely in 1NF as each column seems to represent an atomic value.
- To achieve 2NF and 3NF, you need to ensure that non-prime attributes are fully functionally dependent on the primary key.

8. Students Table:
- Assuming that `student_id` is the primary key.
- The table is likely in 1NF as each column seems to represent an atomic value.
- To achieve 2NF and 3NF, you need to ensure that non-prime attributes are fully functionally dependent on the primary key.

9. **Teacher_Allocation Table:**
- Assuming that `allocation_id` is the primary key.
- The table is likely in 1NF as each column seems to represent an atomic value.
- To achieve 2NF and 3NF, you need to ensure that non-prime attributes are fully functionally dependent on the primary key.

Future Works:

The current database design which I implement lays a solid foundation for Sunshine High School's information system, but there are opportunities for future enhancements and expansions. Some potential areas for improvement and future works include:

1. Enhanced Security Measures: Implement advanced security measures to safeguard sensitive student and teacher information. This may involve role-based access control, encryption of certain fields, and regular security audits.

2. Integration with Online Learning Platforms: In the ever-evolving landscape of education, consider integrating the database with online learning platforms to facilitate remote learning, track virtual attendance, and manage digital assessments.

3. Advanced Reporting and Analytics: Develop a robust reporting system that provides in-depth insights into student performance, teacher effectiveness, and overall school metrics. This can assist administrators in making data-driven decisions.

4. Mobile Application Development: Create a mobile application for students, parents, and teachers to access relevant information, such as grades, attendance, and upcoming events, fostering better communication and engagement.

5. Alumni Management: Extend the database to include an alumni management system, keeping track of former students, their accomplishments, and potentially fostering a network for mentorship or collaboration.

Conclusions:

The completion of the database project marks a significant milestone in enhancing the efficiency and organization of Sunshine High School's administrative processes. The meticulously designed schema and relationships between tables offer a comprehensive solution for managing student and teacher data, courses, fees, and attendance.

The database aligns with the school's commitment to transparency, accountability, and data-driven decision-making. It provides a solid foundation for future scalability and integration with emerging technologies. As Sunshine High School continues its journey as an educational beacon, this information system will contribute to fostering a vibrant learning community and nurturing the next generation of leaders.