Project Report

This Project is about creating a database for Student Management System contains “students” table and “courses” table. Information on enrolled students at the university is contained in the “students” table, while details about the different courses that are offered are contained in the “courses” table.

Now let’s go through this project step-by-step.

1: Database Setup and Sample Data

First, **create a database** that will contain the tables with the data.

Query: `CREATE DATABASE miniproject_DB;`

Explanation: Here, to create a database named as “miniproject_DB” in SQL , “CREATE DATABASE” statement is used. To insert the tables in this database we write the query “USE miniproject_DB”

1.1 - Create Tables, as per the requirement there should be two tables in the database as

1.1.a – Students table

Query:
```
CREATE TABLE students (
        student_id INT PRIMARY KEY,
        student_name VARCHAR (30),
        date_of_birth DATE,
        email VARCHAR (50),
        major VARCHAR (30)
    );
```

Explanation: In this query “CREATE TABLE” statement is used to create a table named as “students” includes columns for student_id, student_name, date_of_birth, email, major. And to ensure that each value must be unique, the student_id is set as primary key.

1.1.b – Courses table

Query:
```
CREATE TABLE courses (
        Course_id INT PRIMARY KEY,
        course_name VARCHAR (40),
        credit_hours INT,
        instructor VARCHAR (30)
    );
```

Explanation: In this query “CREATE TABLE” statement is used to create a table named as “courses” includes columns for course_id, course_name, credit_hours, instructor. And to ensure that each value must be unique, the course_id is set as primary key.

1.2- Now Inserting the data into our tables

1.2.a - In students table:

Query:
```
INSERT INTO students
    VALUES (101, "Eren", "1997-08-07", "erenYeager01@gmail.com", "Anatomy"),
           (102, "Levi", "1995-08-07", "captainLevi02@gmail.com", "Surveying"),
           (103, "Mikasa", "1998-01-02", "ackerman03@gmail.com", "Surveying"),
           (104, "Armin", "1997-0201", "armin123@gmail.com", "Management&Planning");
```

Explanation: In this query "INSERT" statement paired with "INTO" is used to insert the sample data into the "students" table as per the required columns in the table.

1.2.b - In courses table:

Query:
```
INSERT INTO courses
    VALUES (101, "Geology", 3, "Prof.Rathi"),
           (102, "Engineering Graphics", 2, "Prof.Ashish"),
           (103, "Fluid Mechanics", 3, "Prof.Ashish"),
           (104, "Mineral Exploration", 4,"Prof.Trihan");
```

Explanation: In this query "INSERT" statement paired with "INTO" is used to insert the sample data into the "courses" table as per the required columns in the table.

2: SQL Queries and Data Retrieval

2.1- Retrieve Information

2.1.a - Retrieve the list of all students.

Query:
```
SELECT * FROM students;
```

Expected Output:

student_id	student_name	date_of_birth	email	major
101	Eren	1997-08-07	erenYeager01@gmail.com	Anatomy
102	Levi	1995-08-07	captainLevi02@gmail.com	Surveying
103	Mikasa	1998-01-02	ackerman03@gmail.com	Surveying
104	Armin	1997-02-01	armin116@gmail.com	Management&Planning

Explanation: "SELECT" statement is used to retrieve the list of all students from the "students" table.

2.1.b - Retrieve the list of all courses.

Query:
```
SELECT * FROM courses;
```

Expected Output:

Course_id	Course_name	Credit_hours	instructor
101	Geology	3	Prof.Rathi
102	Engineering Graphics	2	Prof.Ashish
103	Fluid Mechanics	3	Prof.Ashish
104	Mineral Exploration	4	Prof.Trihan

Explanation: "SELECT" statement is used to retrieve the list of all students from the "courses" table.

2.1.c - Retrieve the list of students majoring in a specific major

Query:
```
SELECT * FROM students
    WHERE major = "Surveying";
```

Expected Output:

student_id	student_name	date_of_birth	email	major
102	Levi	1995-08-07	captainLevi02@gmail.com	Surveying
103	Mikasa	1998-01-02	ackerman03@gmail.com	Surveying

Explanation: The WHERE clause in SQL used to specify conditions to filter and select specific rows from a table. It retrieve only that data that matches your criteria, and make your queries more precise and relevant. Here the query retrieves all rows from the the "students" table where the major is equal to "Surveying".

2.1.d - Retrieve the list of students who were born before a specific date.

Query:
```
SELECT * FROM students
    WHERE date_of_birth < "1998-01-01";
```

Expected Output:

student_id	student_name	date_of_birth	email	major
101	Eren	1997-08-07	erenYeager01@gmail.com	Anatomy
102	Levi	1995-08-07	captainLevi02@gmail.com	Surveying
104	Armin	1997-02-01	armin116@gmail.com	Management&Planning

Explanation: The WHERE clause in this query is used to filter students according to their birthdates. It provides a list of students who were born before the date "1998-01-01" by retrieving all rows from the "Students" table where the "date_of_birth" field is earlier than that.

2.1.e - Retrieve the list of courses taught by a specific instructor.

Query:
```
SELECT course_name FROM courses
    WHERE instructor = "Prof.Ashish";
```

Expected output:

Course_id	Course_name
102	Engineering Graphics
103	Fluid Mechanics

Explanation: To filter the courses according to the instructor's name the WHERE clause is used, in order to provide a list of the courses that "Prof.Ashish" has taught.

2.1.f - Retrieve the total number of students enrolled in each major.

Query:
```
SELECT major,  COUNT(*)
        AS total_students
        FROM students
        GROUP BY major;
```

Expected Output:

major	total_students
Anatomy	1
Surveying	2
Management&Planning	1
Rock Mechanics	1

Explanation: The GROUP BY clause in this query groups students according to their major. It then determines the total number of students in each major using the COUNT (*) function. The outcome displays the major together with the number of students enrolled in that major.

2.1.g - Retrieve the course with the highest number of credit hours.

Query:
```
SELECT * FROM courses
         ORDER BY credit_hours
         DESC LIMIT 1;
```

Expected Output:

Course_id	Course_name	Credit_hours	instructor
104	Mineral Exploration	4	Prof.Trihan

Explanation: This Query is used to order the results based on the "credit_hours" column in descending order (highest to lowest), so that the one with the highest number of credit hours comes first. At last LIMIT 1 clause is used to retrieve only first row (the highest one).

2.1.h- Retrieve the oldest and youngest students in the database.

Query: for Oldest student

```
SELECT * FROM students

ORDER BY date_of_birth

ASC LIMIT 1;
```

Expected Output:

student_id	student_name	date_of_birth	email	major
102	Levi	1995-08-07	captainLevi02@gmail.com	Surveying

Query: for youngest students

```
SELECT * FROM students

ORDER BY date_of_birth

DESC LIMIT 1;
```

Expected Output:

student_id	student_name	date_of_birth	email	major
103	Mikasa	1998-01-02	ackerman03@gmail.com	Surveying

Explanation: The oldest and youngest students in the table are retrieved individually by these two queries. The first query sorts students according to date of birth in ascending order (oldest first) using ORDER BY date_of_birth ASC. LIMIT 1 makes sure that just the top record is returned, giving the oldest student. The second query sorts students by date of birth in decreasing order (youngest first) using ORDER BY date_of_birth DESC. LIMIT 1 makes sure that just the top record is returned, displaying the youngest student.

2.2: Update database

2.2.a - Add a new course to the Courses table

Query:
```
INSERT INTO

    Courses (course_id, course_name, credit_hours, instructor)

    VALUES (105, "History", 5, "Prof.Tauqeer");
```

Expected Output:

Course_id	Course_name	Credit_hours	instructor
101	Geology	3	Prof.Rathi
102	Engineering Graphics	2	Prof.Ashish
103	Fluid Mechanics	3	Prof.Ashish
104	Mineral Exploration	4	Prof.Trihan
105	History	5	Prof.Tauqeer

Explanation: In this query “INSERT INTO” statement is used to add a new course with its details into the “courses” table.

2.2.b - Enroll a new student in the students table.

Query:

```
INSERT INTO

  Students (student_id, student_name, date_of_birth, email, major)

  VALUES (105, "Reiner", "1995-05-02", "reiner133@gmail.com", "Rock Mechanics");
```

Expected Output:

student_id	student_name	date_of_birth	email	major
101	Eren	1997-08-07	erenYeager01@gmail.com	Anatomy
102	Levi	1995-08-07	captainLevi02@gmail.com	Surveying
103	Mikasa	1998-01-02	ackerman03@gmail.com	Surveying
104	Armin	1997-02-01	armin116@gmail.com	Management&Planning
105	Reiner	1995-02-01	reiner133@gmail.com	Rock Mechanics

Explanation: In this query "INSERT INTO" statement is used to add the details of a new student into the "students" table.

2.3: Nested Queries

2.3.a - Retrieve the list of students who are enrolled in a specific course.

Query:

```
SELECT * FROM Students

            WHERE student_id IN (

            SELECT student_id

            FROM enrollments

            WHERE course_id = 101

       );
```

Students table:

student_id	student_name	date_of_birth	email	major
101	Eren	1997-08-07	erenYeager01@gmail.com	Anatomy
102	Levi	1995-08-07	captainLevi02@gmail.com	Surveying
103	Mikasa	1998-01-02	ackerman03@gmail.com	Surveying
104	Armin	1997-02-01	armin116@gmail.com	Management&Planning
105	Reiner	1995-02-01	reiner133@gmail.com	Rock Mechanics

Enrollments table:

enrollment_id	student_id	course_id
1	101	101
2	102	101
3	103	102
4	104	101
5	105	102

Expected Output:

student_id	student_name	date_of_birth	email	major
101	Eren	1997-08-07	erenYeager01@gmail.com	Anatomy
102	Levi	1995-08-07	captainLevi02@gmail.com	Surveying
104	Armin	1997-02-01	armin116@gmail.com	Management&Planning

Explanation: To obtain the list of students enrolled in a certain course having “course_id = 101” , this query employs a nested query. All “student_ids” from a hypothetical "enrollments" table where the course_ id matches 101 are selected by the inner query. Then, the outer query retrieves every entry from the "students" table where the student_id appears in the inner query's result using the IN operator.

2.3.b - Retrieve the list of courses with enrollments greater than a certain number

Query:

```
SELECT course_id, course_name FROM courses
    WHERE course_id IN (
    SELECT course_id
    FROM enrollments
    GROUP BY course_id
    HAVING COUNT (student_id) > 2
);
```

courses table:

Course_id	Course_name	Credit_hours	instructor
101	Geology	3	Prof.Rathi
102	Engineering Graphics	2	Prof.Ashish
103	Fluid Mechanics	3	Prof.Ashish
104	Mineral Exploration	4	Prof.Trihan
105	History	5	Prof.Tauqeer

enrollments table:

enrollment_id	student_id	course_id
1	101	101
2	102	101
3	103	102
4	104	101
5	105	102
6	106	103

Expected Output:

Course_id	Course_name
101	Geology
102	Engineering Graphics

Explanation: To obtain the list of courses with enrollments more than two, this query employs a nested query. SELECT course_id FROM enrollments is the inner query. Grouping rows from the hypothetical "enrollments" table by course_id, then filtering out groups with a count greater than two (GROUP BY course_id HAVING COUNT (student_id) > 2). Next, the outer query retrieves every row from the "courses" table where the course_id appears in the inner query's result by using the IN operator.

END