Example 1 – Student Management System

Let's design a relational algebra query for a student management system. I'll outline a simple schema and then provide some example queries. You can adapt these to your specific needs.

Simplified Schema:

- **Student:** (StudentID, Name, Major, GPA)
- **Course:** (CourseID, CourseName, Credits)
- **Enrollment:** (StudentID, CourseID, Grade)

Example Queries and Relational Algebra:

1. **Find the names of all students majoring in Computer Science:**

   Code snippet

   ```
   π Name (σ Major = 'Computer Science' (Student))
   ```

   - σ (Selection): Filters the `Student` relation based on the condition `Major = 'Computer Science'`.
   - π (Projection): Keeps only the `Name` attribute from the result of the selection.
2. **Find the CourseID and CourseName of all courses worth 3 credits:**

   Code snippet

   ```
   π CourseID, CourseName (σ Credits = 3 (Course))
   ```

3. **Find the names of all students who have enrolled in the course with CourseID 'CS101':**

   Code snippet

   ```
   π Name (Student ⋈ StudentID (σ CourseID = 'CS101' (Enrollment)))
   ```

   - ⋈ (Natural Join): Joins the `Student` and `Enrollment` relations on the common attribute `StudentID`.
   - The selection `σ CourseID = 'CS101'` filters the joined relation to only include enrollments for the specified course.
   - The projection `π Name` extracts the names of the students. We don't need to explicitly specify `Student.StudentID` or `Enrollment.StudentID` in the join condition because it's a natural join.
4. **Find the names of all students who have a GPA greater than 3.5 and are majoring in Biology:**

   Code snippet

```
π Name (σ GPA > 3.5 ∧ Major = 'Biology' (Student))
```

- ∧ (Logical AND): Combines two conditions in the selection.

5. **Find the CourseName and Grade for all enrollments of student 'Alice':**

   Code snippet

   ```
   π CourseName, Grade (Course ⋈ CourseID (σ Name = 'Alice' (Student) ⋈ StudentID Enrollment))
   ```

   - This query involves two joins: first, joining `Student` and `Enrollment` on `StudentID` to find Alice's enrollments, and then joining the result with `Course` on `CourseID` to get the course names.

6. **Find the names of all students who have taken at least one course:**

   Code snippet

   ```
   π Name (Student ⋈ StudentID Enrollment)
   ```

   - This performs a join between `Student` and `Enrollment`. If a student has no enrollments, they will not be in the joined relation. Therefore, projecting the `Name` from the joined relation gives us the names of students who have taken at least one course.

7. **Find the names of all students who have not taken any courses:**

   Code snippet

   ```
   π Name (Student) - π Name (Student ⋈ StudentID Enrollment)
   ```

   - − (Set Difference): Subtracts the set of students who *have* taken courses (as in the previous query) from the set of *all* students.