Latin School of San Diego Project

Introduction

The Latin School of San Diego is a fictional language school that offers courses in Latin language and culture. It is a small school that has been in operation for about 8 years. They offer new classes on a quarterly basis. Up to now they have been using Excel workbooks to track course offerings, student enrollments and instructor payments. The process has become increasingly unwieldly and the school wants to upgrade to a database backend.

You have been hired to create a new SQL Server database and migrate the existing Excel data into the database. There are two Excel workbooks, each covering about a 4 year period. You will need to import both workbooks into the database.

Creating a frontend interface would normally be expected in the real world, however it is beyond the scope of this project. Common solutions could be a website, a Visual Studio application, or linking the database to a Microsoft Access frontend.

The Database Structure

It is assumed you had earlier meetings with Latin School management to discuss the business logic behind the database. In these discussions you identified the primary entities and the relationships between them.

There are 4 primary entities to be managed: Courses, Sections, Persons and Faculty. Course records describe the content of what will be taught in a course as well as the expected cost and duration. Section records are a subset of a Course record. They contain information on when and where a course will be taught as well as who will be teaching it. Person records contain information students enrolled in courses but also people who are members of the Latin School. Not all members enroll in courses. Faculty records contain information about the instructors who teach classes for the school.

The relationships between these entities are as follows.

- A Course can have multiple Sections, but a Section is related to only one Course
- A Section can have multiple Faculty and Faculty can teach multiple Sections
- A Section can have multiple Students and Students can be enrolled in multiple Sections
- Because there are 2 many-to-many relationships, a linking table will need to be created for each relationship

There will be 2 lookup/crosswalk tables that are related to the Section table: A Term table that contains additional information about the term (aka quarter) in which a section is offered. A Room table that contains information about the location of a section.

Finally, the school only captures a single address for a person, but they want the ability to capture multiple addresses per person going forward. Each Address record should have an address type of “home” or “work”. All existing addresses will be assigned a “home” value.

Based on this information a preliminary ERD (Entity Relationship Diagram) was created and attached to the end of this document.

Latin School of San Diego Project

Building the Database

When you create your database, assign it a name in the following format: LSP_[initials]. For example my database name would be "LSP_edw". You will need to create your database with the proper table relationships and columns. It is up to you to identify what columns should be assigned to each table. All the data contained in the Excel spreadsheets should be included. However do not introduce duplicate data to the database (e.g. The student name should only be in the Person table).

Modify the column datatypes to be practical for each column. If a column contains numbers or dates then the datatype should reflect that. Don't use the nvarchar datatype for strings, and set the string length to a reasonable size for the data it contains.

Notes on Primary and Foreign keys. Each table requires a Primary Key that consists of a single column. You will also need to assign the appropriate foreign keys. Some of the data already has a column that works as a primary key (e.g. SectionID) while other data will require a primary key created from scratch. My preference is to use numeric identity columns as primary keys, however this is not a requirement for the project.

Additional Database Objects

Once you have created your tables and imported the data from the Excel workbooks, you will need to create a few views and stored procedures for reporting purposes.

1. Create a view that shows the number of times each Course has been offered in the history of the school. The view should include the course code and course title, count of sections, the total gross revenue, and the average revenue per section. Do not include cancelled sections in the count and average (i.e. don't include records where the SectionStatus = CN).

2. Create a view that displays the gross revenue from tuition as well as faculty payments for each Academic Year.

3. Create a procedure that displays the course history for a selected person. The procedure will take a parameter that equals the Primary Key of the person. For each record returned include following:
    a. Student Name
    b. Section ID
    c. Course Code
    d. Course Title
    e. Primary Instructor Name
    f. Term Code
    g. Start Date
    h. Tuition Paid
    i. Grade

4. Create a procedure that adds a new person to the database. The procedure will need to populate both the Person and Address tables. Populate the following columns:
    a. Person table: First Name, Last Name
    b. Address table: Address Type, Address Line, City

Latin School of San Diego Project

Submission Format

To earn full credit on the project, the following documents will need to be submitted to the online portal.

1. Screenshots of the output from views 1-2 and procedures 3-4.
2. A .sql file script of the entire database and all database objects.
3. A screen shot of an entity relationship diagram showing all table and column names, and the relationships between tables.

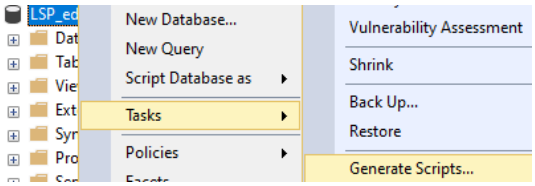
Output Screenshots

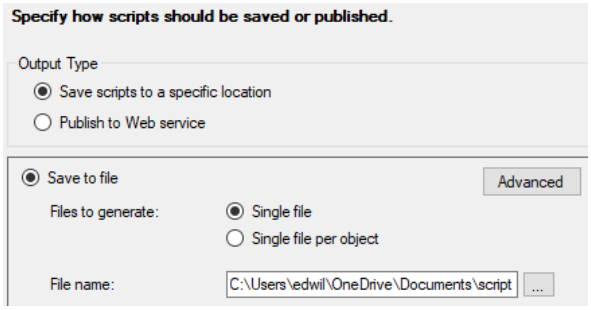
I used the following scripts to get the output for my view and procedure screenshots. Note that your object and column names may not be the same as mine.

```
SELECT * FROM CourseRevenue_v ORDER BY CourseCode
SELECT * FROM AnnualRevenue_v ORDER BY AcademicYear
EXEC StudentHistory_p 1400
EXEC InsertPerson_p 'Eric','Williamson','work','500 Elm St.','North Pole'
SELECT TOP 1 * FROM Person ORDER BY PersonID DESC
SELECT TOP 1 * FROM Address ORDER BY AddressID DESC
```

Script Database

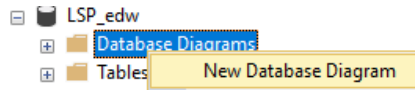
You can script the database by right-clicking on the database name in the Object Explorer and selecting Tasks >> Generate Scripts. Click Next to skip the Introduction page. On the Choose Objects page select "Script entire database and all database objects" and click Next. On the Set Scripting Options page ensure save to a single file is selected then click Next two more times to create the .sql file.

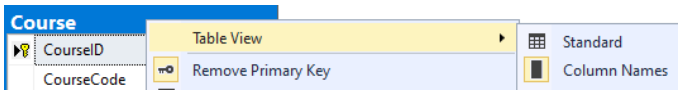




Entity Relationship Diagram Screenshot

You can create a database diagram by right-clicking on the Database Diagrams folder underneath your database name in the Object Explorer window. Select the tables you want to view in the diagram and click Add. You can drag the tables on the screen and resize as needed. It should be the default but make sure each table has "Column Names" selected as the table view. You can find this feature by right-clicking on a table in the diagram.





Latin School of San Diego Project

Tips while working on your Project

- Load your Excel data staging tables in a separate database from the Latin School Project database. This way you won't have to reload Excel data if you delete your project database.

- Review and possibly clean up your data before trying to import it into your destination tables. Your data comes from two separate Excel workbooks and there may be duplicates.

- Set the primary and foreign keys on your destination tables before importing data. Referential integrity will prevent mistakes like duplicates and orphaned records from being introduced.

- Consider how you will create the primary and foreign keys for a record if one doesn't already exist. You may need to join to a substitute column in order to get the data you need.

- The SQL Server import tool might truncate Excel text columns with more than 255 characters. This is because the import tool only evaluates the first 8 rows of an Excel spreadsheet when determining the length to assign. If all of the first 8 rows are 255 characters or less then nvarchar(255) is assigned. If a least one of the 8 rows has more than 255 characters, then nvarchar(max) is assigned. If you have columns that might exceed 255 characters, sorting by the longest lengths at the top will ensure your data doesn't get truncated. Hint: Look at the Course Description.

- If you want to insert data into a column that is also an identity column, you need to temporarily disable the identity. You can do this with the following script before your INSERT statement:

  ```
  SET IDENTITY_INSERT [Table Name] ON
  ```
  Don't forget to re-enable the identity once you are done inserting
  ```
  SET IDENTITY_INSERT [Table Name] OFF
  ```

- Use an OUTPUT clause if you need to capture the identity value for an inserted record. Storing the output in a temp table can be effective.

- I set up my script to drop and recreate the database and its objects every time I execute it. The advantage is as you incrementally build your script, you can easily correct earlier mistakes. The execution will automatically wipe out the old errors when rebuilding your database. The lesson 1 homework gives an example of this process. It's important to have your Excel file imports in a separate database if you adopt this method.

- Give yourself time to work on this project. It took me about 3 hours just to write my code, and I had the advantage of knowing the answer beforehand.

Latin School of San Diego Project

Entity Relationship Diagram

Below is a snapshot of an ERD you can potentially leverage for the Latin School project. The ERD only shows primary and foreign keys. Your final ERD will need to include all column names. Note that your column names do not have to mirror the ones I used in this diagram.