- **請以 ContosoUniversity 資料庫為主要資料來源**

  這個沒啥問題 有sql可以生個新的

- **須透過 DB First 流程建立 EF Core 實體資料模型**

  dotnet new mvc -n 名稱 && cd 名稱

  上三個套件
  dotnet add package Microsoft.EntityFrameworkCore.SqlServer
  dotnet add package Microsoft.EntityFrameworkCore.Design
  dotnet add package Microsoft.EntityFrameworkCore.Tools

  下指令生成Class和Context
  dotnet ef dbcontext scaffold "Server=(localdb)\MSSQLLocalDB;Database=ContosoUniversity;Trusted_Connection=True;MultipleActiveResultSets=true" Microsoft.EntityFrameworkCore.SqlServer -o Models

- **須對資料庫進行版控 (使用資料庫移轉方式)**

  生成migrations並取名Init
  dotnet ef migrations add Init

  更新資料庫
  dotnet ef datatbase update -v 失敗
  因為是初次版本記錄 需把cs檔內Up和Down更新過程指令全刪
  再次 dotnet ef database update -v 成功

- **須對每一個表格設計出完整的 CRUD 操作 APIs**

  用vistual studio 2019產生

- **針對 Departments 表格的 CUD 操作需用到預存程序**
  - 新增

    ~~json日期格式的問題~~
    ~~[JsonPropertyName("StartDate")]在Class日期欄位上套上這個~~
    ~~後來又發現不用??一整個鬼打牆..~~

    ~~_context.Department.FromSqlRaw($"Execute Department_Insert…~~
    ~~因只回傳ID 所以會發生錯誤~~
    ~~另在Models下做了一class只放ID(並加上[Key]屬性)~~

~~並以該class為DBSet在Dbcontext下 可以順利新增並接收ID了~~

~~return CreatedAtAction("GetDepartment", new { id = department.DepartmentId }, department);~~
~~一直無法回傳已新增的完整資料~~
~~只好以 var returnObj = _context.Department.FindAsync(rid) 接收後再回傳搞定~~
~~return CreatedAtAction("GetDepartment", returnObj);~~

~~過一天看StoredProcedure發現是會回傳兩個欄位~~
~~把兩個欄位都接收以後塞到department再return~~
~~省去再跑_context.Department.FindAsync(rid)~~

~~發現把Department這樣選Select(p => new { p.DepartmentId, p.RowVersion }就好~~
~~省去新增無謂的class~~

砍掉重練後 上面已成歷史 直接用下面的
var sql = _context.Department.FromSqlInterpolated($"EXECUTE dbo.Department_Insert {department.Name},{department.Budget},{department.StartDate},{department.InstructorId}");
var result = await sql.Select(r => new Department() { DepartmentId = r.DepartmentId, RowVersion = r.RowVersion }).ToListAsync();

department.DepartmentId = result[0].DepartmentId;
department.RowVersion = result[0].RowVersion;

- 修改

更新一直有問題 連原生的put也不行
~~可能是RowVersion問題 明天再試試@@DBTS && MIN_ACTIVE_ROWVERSION()~~

~~研究StoredProcedure 欄位RowVersion相同或空值都可以~~
~~空值我試不出來..用@RowVersion_Original={DBNull.Value}行不通~~
~~改採同值處理~~
~~byte[] btTsArray = department.RowVersion as byte[];~~
~~string strTimeSpan = "0x" + BitConverter.ToString(btTsArray).Replace("-", "");~~

砍掉重練後 上面已成歷史 直接用下面的
var sql = _context.Department.FromSqlInterpolated($"EXECUTE dbo.Department_Update {department.DepartmentId},{department.Name},{department.Budget},{department.StartDate},{department.InstructorId},{department.RowVersion}");
var result = await sql.Select(r => new Department() { RowVersion = r.RowVersion }).ToListAsync();

- 刪除

  ~~經歷過上兩個的磨難 一下就搞定~~
  ~~因為沒回傳值 所以用Select(p => new { })~~
  ~~後來連上一行也去掉~~

  砍掉重練後 上面已成歷史 直接用下面的 連轉換RowVersion都省了
  _context.Database.ExecuteSqlInterpolated($"Execute Department_Delete @DepartmentID ={department.DepartmentId},@RowVersion_Original ={department.RowVersion}");

- **請在 CoursesController 中設計 vwCourseStudents 與 vwCourseStudentCount 檢視表的 API 輸出**

  我猜是在練設定route啦..不過一開始我設定好沒回應
  發現自己耍笨 postman的http verb還在DELETE沒改回來

- **請用 Raw SQL Query 的方式查詢 vwDepartmentCourseCount 檢視表的內容**

  return await _context.VwDepartmentCourseCount.FromSqlRaw("Select * from vwDepartmentCourseCount").ToListAsync();
  跟DepartmentController.cs一樣

- **請修改 Course, Department, Person 表格，新增 DateModified 欄位(datetime) (請新增資料庫移轉紀錄)，並且這三個表格的資料透過 Web API 更新時，都要自動更新該欄位為更新當下的時間。**

  ~~public DateTime DateModified { get; set; } = DateTime.Now;~~
  ~~dotnet ef migrations add add_DateModified~~
  ~~dotnet ef database update -v~~

  砍掉重練後 上面已成歷史 直接用下面的
  public DateTime DateModified { get; set; }

  直接另新增一個ContosoUniversityExContext.cs
  partial class ContosoUniversityContext
  partial void OnModelCreatingPartial(ModelBuilder modelBuilder)
  {//抄原來ContosoUniversityContext

```
            modelBuilder.Entity<Department>().Property(e =>
e.DateModified).HasColumnName("DateModified");
            modelBuilder.Entity<Course>().Property(e =>
e.DateModified).HasColumnName("DateModified");
            modelBuilder.Entity<Person>().Property(e =>
e.DateModified).HasColumnName("DateModified");

        }
```

```
dotnet ef migrations add add_DateModified
dotnet ef database update -v
```

```
//下面抄來的 雖看得懂意思 叫我自己寫我不會 但卻是重點
public override int SaveChanges()
        {
            BeforeModify();
            return base.SaveChanges();
        }

        public override Task<int> SaveChangesAsync(bool
acceptAllChangesOnSuccess, CancellationToken cancellationToken = default)
        {
            BeforeModify();
            return base.SaveChangesAsync(acceptAllChangesOnSuccess,
cancellationToken);
        }

        private void BeforeModify()
        {
            foreach (var entry in ChangeTracker.Entries())
            {
                switch (entry.State)
                {
                    case EntityState.Added:
                        entry.CurrentValues["DateModified"] = DateTime.Now;
                        break;
                    case EntityState.Modified:
                        entry.CurrentValues["DateModified"] = DateTime.Now;
                        break;
                    case EntityState.Deleted:
                        break;
                }
            }
        }
```

最後修改stored procedure

- **請修改 Course, Department, Person 表格欄位，新增 IsDeleted 欄位 (bit) (請新增資料庫移轉紀錄)，且讓所有刪除這三個表格資料的 API 都不能真的刪除資料，而是標記刪除即可，標記刪除後，在 GET 資料的時候不能輸出該筆資料。**

~~public bool IsDeleted { get; set; }~~
~~dotnet ef migrations add add_DateModified~~
~~dotnet ef database update -vs~~

~~改了一下Stored Procedure~~

~~底下就網路抄來的...~~
~~modelBuilder.Entity<Department>().Property<bool>("IsDeleted");~~
~~modelBuilder.Entity<Department>().HasQueryFilter(m => EF.Property<bool>(m, "IsDeleted") == false);~~

~~modelBuilder.Entity<Person>().Property<bool>("IsDeleted");~~
~~modelBuilder.Entity<Person>().HasQueryFilter(m => EF.Property<bool>(m, "IsDeleted") == false);~~

~~modelBuilder.Entity<Course>().Property<bool>("IsDeleted");~~
~~modelBuilder.Entity<Course>().HasQueryFilter(m => EF.Property<bool>(m, "IsDeleted") == false);~~
~~public override int SaveChanges()~~
~~{~~
~~   UpdateSoftDeleteStatuses();~~
~~   return base.SaveChanges();~~
~~}~~
~~public override Task<int> SaveChangesAsync(bool acceptAllChangesOnSuccess, CancellationToken cancellationToken = default(CancellationToken))~~
~~{~~
~~   UpdateSoftDeleteStatuses();~~
~~   return base.SaveChangesAsync(acceptAllChangesOnSuccess, cancellationToken);~~
~~}~~
~~private void UpdateSoftDeleteStatuses()~~
~~{~~
~~   foreach (var entry in ChangeTracker.Entries())~~
~~   {~~
~~   switch (entry.State)~~
~~   {~~

```
        ~~case EntityState.Added:~~
            ~~entry.CurrentValues["IsDeleted"] = false;~~
            ~~break;~~
        ~~case EntityState.Deleted:~~
            ~~entry.State = EntityState.Modified;~~
            ~~entry.CurrentValues["IsDeleted"] = true;~~
            ~~break;~~
    ~~}~~
  ~~}~~
~~}~~
```

砍掉重練後　上面已成歷史　直接用下面的
public bool IsDeleted { get; set; }

然後在擴充的ContosoUniversityExContext.cs　OnModelCreatingPartial加上
modelBuilder.Entity<Department>().Property(e =>
e.IsDeleted).HasColumnName("IsDeleted");
modelBuilder.Entity<Course>().Property(e =>
e.IsDeleted).HasColumnName("IsDeleted");
modelBuilder.Entity<Person>().Property(e =>
e.IsDeleted).HasColumnName("IsDeleted");

modelBuilder.Entity<Department>().HasQueryFilter(m => EF.Property<bool>(m,
"IsDeleted") == false);
modelBuilder.Entity<Person>().HasQueryFilter(m => EF.Property<bool>(m,
"IsDeleted") == false);
modelBuilder.Entity<Course>().HasQueryFilter(m => EF.Property<bool>(m,
"IsDeleted") == false);

修改BeforeModify
```
case EntityState.Added:
    entry.CurrentValues["DateModified"] = DateTime.Now;
    entry.CurrentValues["IsDeleted"] = false;
      break;
case EntityState.Modified:
    entry.CurrentValues["DateModified"] = DateTime.Now;
      break;
case EntityState.Deleted:
    entry.CurrentValues["DateModified"] = DateTime.Now;
    entry.State = EntityState.Modified;
    entry.CurrentValues["IsDeleted"] = true;
      break;
```

```
dotnet ef migrations add add_DateModified
dotnet ef database update -v
```

結果使用stored procedure出現錯誤訊息
FromSqlRaw or FromSqlInterpolated was called with non-composable SQL and with a query composing over it. Consider calling `AsEnumerable` after the FromSqlRaw or FromSqlInterpolated method to perform the composition on the client side

照它的說法加了AsEmumerable也不行

直到看了同學找到IgnoreQueryFilters()方法才搞定