IFT 530 Advanced Database Management Systems

Professor Robert Rucker

Written Document of Project

Project – Syllabus Generator

Team Members –

Sushmita Prafull Halasawade 1226584322

Trilok Navya Dubaka 1224487563

Navya Akshitha Sangishetty 1226626832

Synopsis

The proposed Syllabus Generator system is department centric and is intended to have least susceptibility to errors and mismanagements. Phase 1 of the project is developed using NORMA. The seven stages of Conceptual Schema Design Procedure are implemented, and corresponding SQL code is created. Stored procedures and triggers are defined. Data is loaded into the tables. After integration of all these the final database is created.

In ORM diagram, the subtypes are added. Course can be of type – iCourse and Inperson

Each course have two syllabus – Fall and Spring

Additionally, details of the professor like name, professor ID, phone number, syllabus. The ID of professors' are added as mandatorily unique so that data redundancy can be reduced.

A database is information that is set up for easy access, management and updating. Databases are used for storing, maintaining and accessing any sort of data. Information is gathered in one place so that it can be observed and analyzed. Databases can be thought of as an organized collection of information. We have used SQL Server Management Studio to create syllabus generator database and added the data to the tables. Also, we have represented Entity Relation diagram and executed queries against the data in SSMS.

We used Couchbase-NoSQL for implementing the Syllabus Generator database in JSON format. Here we created new buckets and inserted the data same as the SQL database but with a JSON file tables. NOSQL databases are classified according to their data model. Document, key-value, wide-column and graph are the most common types. NoSQL databases don't require a predetermined schema, you can interact with "unstructured data" more freely. The JSON stores semi-structured data: objects which contain their own relevant information sets that can be completely different from each other.

Section 1

Introduction and Importance

Syllabus Generator is a tool that generates and maintains the Syllabus of the courses in a university. It is a useful application for the instructors in the university to add, update or modify the existing syllabus under the supervision of Program Chair. Syllabus Generator is powerful, flexible, and easy to use and is designed and developed to provide real conceivable benefits to the university instructors.

The Project Syllabus Generator is based on SQL Database. SQL Server easily integrate data into applications and takes advantage of a broad set of cognitive services to leverage the potential of newly developing applications.

The purpose of this project is to automate the house keeping task of the professor which is updating or keeping track of the syllabus on day-to-day basis. Instructor can access the textbooks needed for the course as we have added Textbook table which stores the relevant textbooks of the courses. Syllabus generator provides the instructor to update the syllabus for the professor at any point in time and program chair is always aware of the changes that are being made to the syllabus by the professor.

In the traditional way, professor maintains the hardcopies of the syllabus for a class and then seek the approval to update the syllabus manually from the program chair and get the syllabus updated. It was a manual task to maintain the text books and keep track of the topics in the current syllabus. Some times when the syllabi have many topics professor might overlook a few topics.

After conduction a thorough research, we understood that Syllabus Generator concept is a ubiquitous concept. As we decided to develop a database system for professor for syllabus management, we have worked on identifying the entities and the relationships between them so that a meaningful and strong structure can be provided for the system without any flaws. The

importance of ORM is understood in this activity. As we have practised using NORMA during our lab activities in the coursework and have implemented the same in this Syllabus Generator project.

The relational view gave us a good understanding of joins between the tables. Moreover, we have implemented our SQL queries by having the complete knowledge of joins, stored procedures, triggers and user-defined functions.

Main Idea and Background:

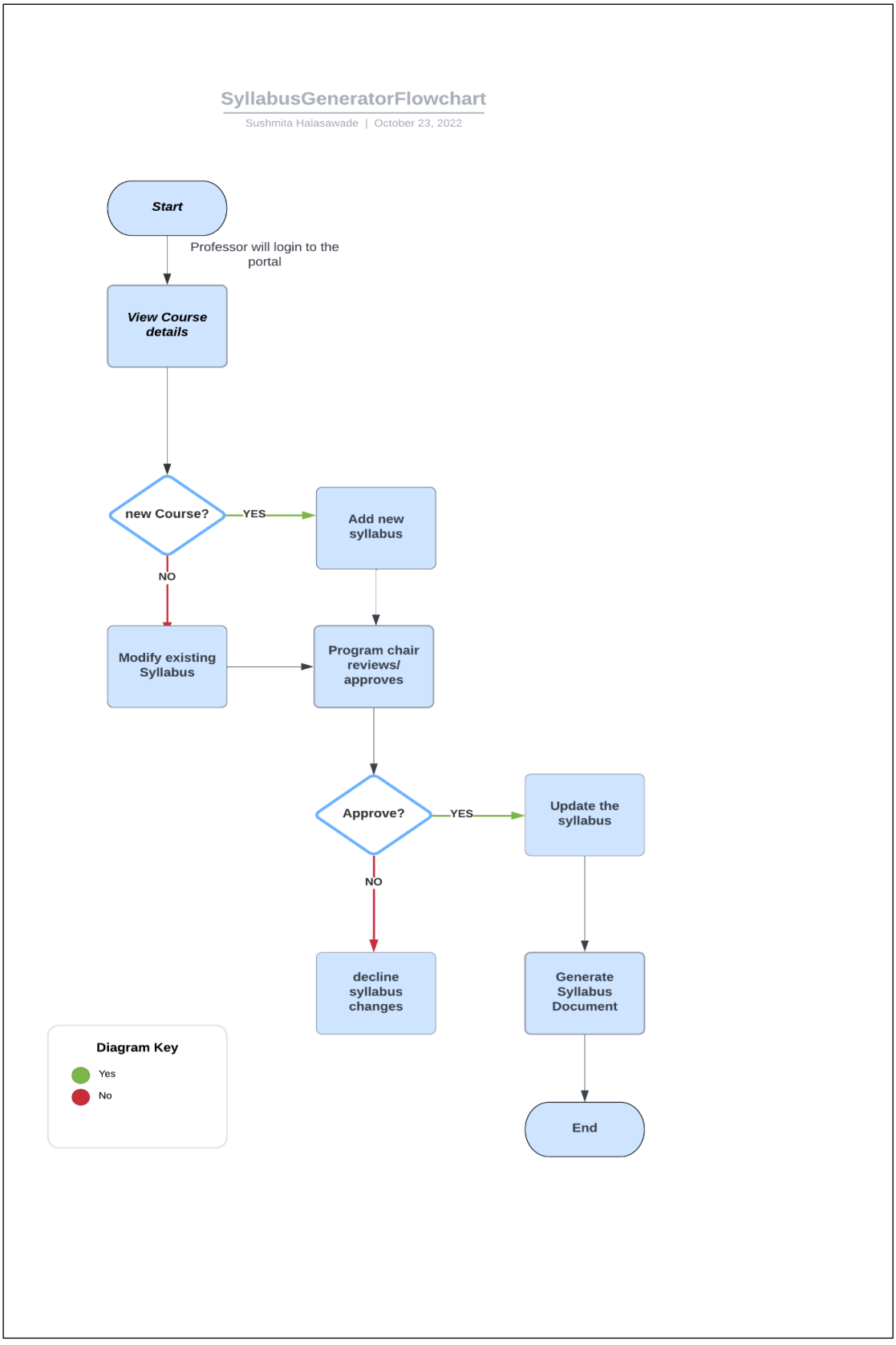
Syllabus Generator includes profile creation for the professors, storing their contacts details of the professors, textbooks required for the course, details for the program chair, existing syllabus.
The software is facilitated to assign a unique id to the department, program chair, course. Software includes the semester wise information ie., the course schedule for the in-person class and the icourse (offline course) and the semester wise schedule example - fall semester, spring semester and summer semester.
Syllabus Generator is designed for the universities to cover admin task of the course instructors and management process of the Syllabus for every course every semester. It is integrated end-to-end Syllabus management system that provides relevant information across the university to support effective decision making for carefully handcrafted syllabus for the students for their seamless academic success.
This software will help the university to be more efficient in handling the monotonous task of the professor and program chair. The purpose of the project is to give a complete requirement, documentation, design and implementation of the software.
Syllabus generator is a smart system designed to increase the quality of administrative services provided to department staff in a university. Recent attempts have been made to design and build a user-friendly and dependable database system to meet the needs of the department staff but this house keeping task of course instructors had been remained unexplored.

Database systems are required by all large businesses in order to manage information. Universities are one of them.

Data processing becomes more important in university courses due to large intake of the students and faculty members each year. In a university, program chair will have access to the syllabus and department. When ever professor add, update or modify the existing syllabus for a course, the program chair gets a notification that he/she would need to review and approve, decline or suggest modifications to the syllabus.

Once the Syllabus is approved by the program Chair the final Syllabus will be posted and the updated syllabus can be generated at any point in time by the professor. Finally, the same syllabus will be provided to the students.

Jobflow


SyllabusGeneratorFlowchart
Sushmita Halasawade | October 23, 2022
Start
Professor will login to the portal
View Course details
new Course?
YES
Add new syllabus
NO
Modify existing Syllabus
Program chair reviews/ approves
Approve?
YES
Update the syllabus
NO
decline syllabus changes
Generate Syllabus Document
End
Diagram Key
Yes
No

Database Description

Entities and Attributes:

1.Department Table

This entity represents the departments present in the university. Attributes of this entity are given below –

Attributes and its datatypes

DepartmentID: It is the primary key of the entity. It is a unique identifier and distinct from other departments in the university. Datatype is int.

DepartmentName: Every Department will have a name. This attribute stores the name of the department. Datatype is varchar.

DepartmentCode: Every Department will have a department code. This attribute stores the unique code of the department. Datatype is varchar.

2. Program Chair

Program Chair entity stores the details of the program chair. Attributes of this entity are given below –

ProgramChairID: This attribute holds the ID of the program chair.

DepartmentID: This attribute holds the ID of the Department.

Name: This attribute holds the name of the Program Chair

EmailID: This attribute holds the values for the email ID of the program chair

3. Professor Table

This table is for storing the details of the professor. Attributes of this table are given below –

Professor ID: This is the unique ID of the professor. It is the primary key in this table

DepartmentID: This is the foreign key. This is the primary key in Department table.

Name: This column holds the values for professor name. Datatype is varchar

EmailID: This column holds the values for professor's emailID. Datatype is varchar

ContactNumber: This is to store the contact number of the professor. Datatype is int.

4. TextBook Table

TextBook entity stores the details about the textbooks that professor can refer to design the syllabus for the class. Attributes of this entity are given below –

TextBookID: This attribute holds the values for TextBookID. Datatype is int. This is the primary key.

TextBookName: This attribute holds the values for TextBookName. Datatype is int.

Category: This attribute holds the value for domain of the text book. Datatype is varchar.

5. Course Table

This entity stores the information about the course details. Attributes of this entity are given below –

Attributes and its datatypes

CourseID: It is the primary key of the entity. It is a unique identifier and distinct from other departments in the university. Datatype is int.

DepartmentID: It is the foreign key of the entity. This is the primary key in Department table.

CourseName: This attribute stores the values for name of the course

CourseType: This attribute stores the values for the type of the course i.e., icourse or the in-person course.

Professor ID: It is the foreign key of the entity. This is the primary key in the professors table. This attribute stores the unique id of the professor

6. Syllabus Table

This entity stores the stores the information about syllabus. Attributes of this entity are given below –

SyllabusID: Syallabus ID is the unique ID for syllabus. Datatype is int.

CourseID: CourseID is the foreign key for syllabus table. This is a unique id for course. This is the primary key in course table. Datatype is int.

ProfessorID: It is the foreign key of the entity. This is the primary key in the professor table. This column stores the unique id of the professor. Datatype is int.

Semester: This attribute stores the values whether the semester is the fall semester or spring semester. Datatype is varchar.

LearningObjective: This attribute stores the values for Learning Objectives or value addition to the students by taking the course. Datatype is varchar.

Prerequisite: This attribute stores the values for prerequisites to take a particular course. Datatype is varchar.

TextBookID: This attribute stores the TextBook ID. Datatype is int. It is a foreign key. It is the primary key in TextBook Table

7. Course Schedule

This Course Schedule entity stores the details of the course schedule. Attributes of this entity are given below –

CourseScheduleID: This attribute holds the values for Course Schedule ID. This is the primary key in this entity. Datatype is int.

Syllabus ID : This attribute holds the values for Syllabus ID. This is the primary key in this entity. Datatype is int.

Course ID: This attribute holds the values for Course ID. This is the foreign key in this entity. Datatype is int.

Module: This attribute holds the values for the module name. Datatype is varchar.

CourseContent: This attribute holds the information about the course content. Datatype is varchar.

StartDate: This holds the values for the start date of the semester. Datatype is date.

EndDate: This holds the values for the end date of the semester. Datatype is date.

8. SyllabusAudit Table

This table is to store the syllabus that is deleted by the professor.

Questions

1) How to retrieve the Current Department Details
2) How to retrieve department wise Course Program Chair, Professor Details
3) How to retrieve the Professor name of Professor teaching particular course
4) How to retrieve Professor and program chair contact details
5) How to retrieve Course/Department Textbook list
6) How to retrieve start and end date of the semester

Triggers and Stored Procedures

Trigger

Delete Trigger to log the data/syllabus that is deleted by the professor.

Update Trigger – On updating the syllabus table, it will add entry in SyllabusAudit table with remark of updated field.

Stored Procedure

spFetchCourses - Stored procedure to fetch all courses of professor

spFetchSyllabusDetails - Stored procedure to fetch Syllabus details of course taught by professor

spInsertorUpdateSyllabus - Stored Procedure to insert or update the syllabus

spDeleteSyllabus - Stored Procedure to delete complete syllabus

User defined functions –

fnGettextbookdetails - Function to get textbook requirements of course

fnGetPrerequisite - Function to get prerequisite course list of course

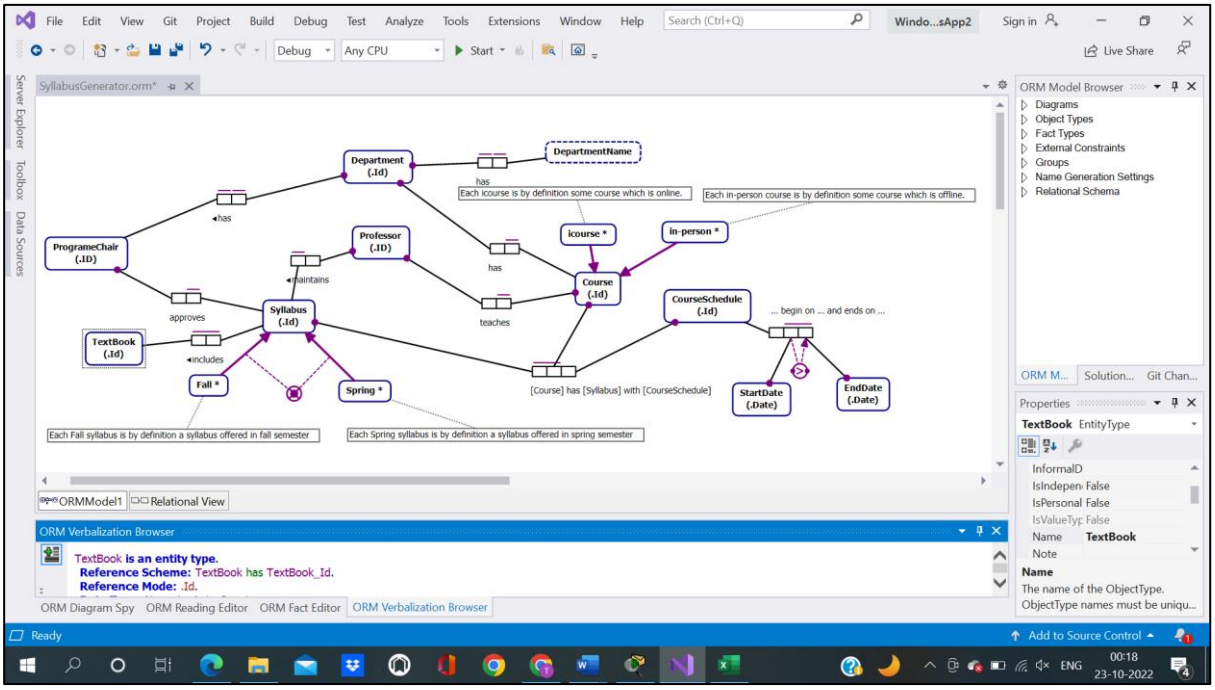
Section 2

Object Role Modeling more semantically expressive than all other conceptual modeling languages in existence today.

ORM diagrams are more semantically verbose, it is easy to translate ORM diagrams into other diagrams.

It is easy to map an injection or surjection from ORM to other modeling languages and it is likely not possible with the others.

The following ORM Diagram represents the Syllabus Generator model which includes the entity types. ORM Diagrams are used to depict the relationships between objects in a database. Through notions like objects and roles, ORM Diagrams provide a conceptual approach to database modeling.



1. Seven Entities that exist include Department, Course, Syllabus, Professor, Program Chair, Course Schedule, TextBook

2. One Many to Many Relationship exists between TextBook and Syllabus as one text book can be referred for more than one course syllabus and vice versa.

3. Two Ternary relations defined can be verbalized as follows:

a. Syllabus has a course Schedule that begin on start date and end on end date.

b. Every Course has a syllabus with defined course schedule

4. Two subtypes defined include:

a. Course can be of type – iCourse or Inperson

b. Each course will have two Syllabus – Fall and Spring

Verbalization

DepartmentName **is a value type**.
Data Type: Text: Variable Length (0).

Fact Types:
Department has DepartmentName.
Department has DepartmentName.
Each Department has **exactly one** DepartmentName.
For each DepartmentName, **at most one** Department has **that** DepartmentName.
Course **is an entity type**.
Reference Scheme: Course has Course_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Course has Course_Id.
Department has Course.
Professor teaches Course.
Course has Syllabus with CourseSchedule.
Each icourse is an instance of Course.
Each in-person is an instance of Course.
Department has Course.
Each Department has **some** Course.
For each Course, **at most one** Department has **that** Course.
It is possible that some Department has **more than one** Course.
ProgrameChair **is an entity type**.
Reference Scheme: ProgrameChair has ProgrameChair_ID.
Reference Mode: .ID.
Data Type: Numeric: Auto Counter.

Fact Types:
ProgrameChair has ProgrameChair_ID.
Department has ProgrameChair.
ProgrameChair approves Syllabus.
Department has ProgrameChair.
Each Department has **exactly one** ProgrameChair.
For each ProgrameChair, **at most one** Department has **that** ProgrameChair.
Professor **is an entity type**.
Reference Scheme: Professor has Professor_ID.
Reference Mode: .ID.
Data Type: Numeric: Auto Counter.

Fact Types:
Professor has Professor_ID.
Professor teaches Course.
Professor maintains Syllabus.
Professor teaches Course.
Each Professor teaches **some** Course.

For each Course, **exactly one** Professor teaches **that** Course.
It is possible that some Professor teaches **more than one** Course.
Syllabus **is an entity type**.
Reference Scheme: Syllabus has Syllabus_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Syllabus has Syllabus_Id.
Course has Syllabus with CourseSchedule.
Professor maintains Syllabus.
ProgrameChair approves Syllabus.
Each Fall is an instance of Syllabus.
Each Spring is an instance of Syllabus.
Syllabus includes TextBook.
CourseSchedule **is an entity type**.
Reference Scheme: CourseSchedule has CourseSchedule_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
CourseSchedule has CourseSchedule_Id.
Course has Syllabus with CourseSchedule.
CourseSchedule begin on StartDate and ends on EndDate.
Course has Syllabus with CourseSchedule.
For each Course **and** Syllabus,
that Course has **that** Syllabus with **at most one** CourseSchedule.
This association with Course, Syllabus **provides the preferred identification scheme for** CourseHasSyllabusWithCourseSchedule.
For each CourseSchedule,
some Course has **some** Syllabus with **that** CourseSchedule.
For each Syllabus,
some Course has **that** Syllabus with **some** CourseSchedule.
Each Course has **some** Syllabus with **some** CourseSchedule.
Professor maintains Syllabus.
Each Professor maintains **some** Syllabus.
For each Syllabus, **at most one** Professor maintains **that** Syllabus.
It is possible that some Professor maintains **more than one** Syllabus.
ProgrameChair approves Syllabus.
Each ProgrameChair approves **some** Syllabus.
For each Syllabus, **at most one** ProgrameChair approves **that** Syllabus.
It is possible that some ProgrameChair approves **more than one** Syllabus.
Icourse **is an entity type**.
Reference Scheme: Course has Course_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Each icourse is an instance of Course.
Derivation Note: *Each icourse is by definition some course which is online.*
In-person **is an entity type**.
Reference Scheme: Course has Course_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Each in-person is an instance of Course.
Derivation Note: *Each in-person course is by definition some course which is offline.*
Notes: *Each icourse is by definition some course which is online.*
Notes: *Each in-person course is by definition some course which is offline.*
Fall **is an entity type**.
Reference Scheme: Syllabus has Syllabus_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Each Fall is an instance of Syllabus.
Derivation Note: *Each Fall syllabus is by definition a syllabus offered in fall semester.*
Spring **is an entity type**.
Reference Scheme: Syllabus has Syllabus_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Each Spring is an instance of Syllabus.
Derivation Note: *Each Spring syllabus is by definition a syllabus offered in spring semester.*
Notes: *Each Fall syllabus is by definition a syllabus offered in fall semester.*
Notes: *Each Spring syllabus is by definition a syllabus offered in spring semester.*
StartDate **is an entity type**.
Reference Scheme: StartDate has StartDate_Date.
Reference Mode: .Date.
Data Type: Temporal: Date.

Fact Types:
StartDate has StartDate_Date.
CourseSchedule begin on StartDate and ends on EndDate.
EndDate **is an entity type**.
Reference Scheme: EndDate has EndDate_Date.
Reference Mode: .Date.
Data Type: Temporal: Date.

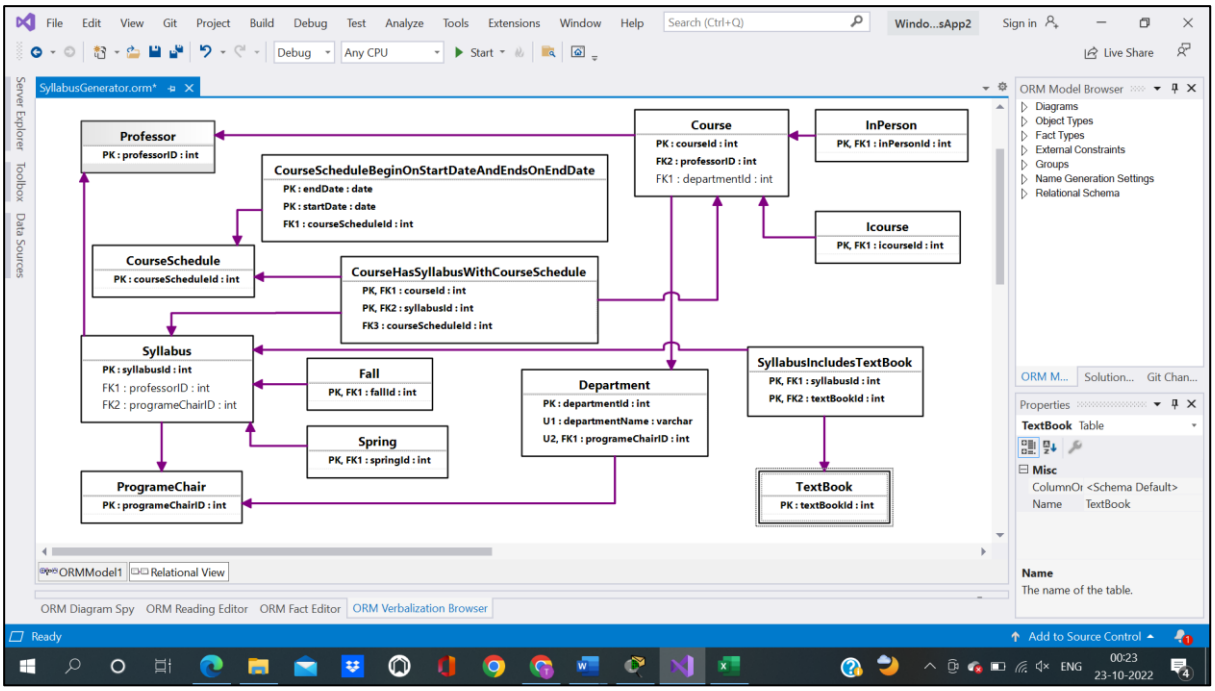
Fact Types:
EndDate has EndDate_Date.
CourseSchedule begin on StartDate and ends on EndDate.
CourseSchedule begin on StartDate and ends on EndDate.
For each StartDate,
some CourseSchedule begin on **that** StartDate and ends on **some** EndDate.
For each EndDate,
some CourseSchedule begin on **some** StartDate and ends on **that** EndDate.
For each StartDate **and** EndDate,
at most one CourseSchedule begin on **that** StartDate and ends on **that** EndDate.
This association with StartDate, EndDate **provides the preferred identification scheme for** CourseScheduleBeginOnStartDateAndEndsOnEndDate.
If some CourseSchedule begin on StartDate and ends on **some** EndDate
then StartDate **is greater than** EndDate.
For each Syllabus, **exactly one of the following holds:**
that Syllabus is **some** Fall;
that Syllabus is **some** Spring.
TextBook **is an entity type**.
Reference Scheme: TextBook has TextBook_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
TextBook has TextBook_Id.
Syllabus includes TextBook.
Syllabus includes TextBook.
It is possible that for some TextBook, **more than one** Syllabus includes **that** TextBook **and that some** Syllabus includes **more than one** TextBook.
In each population of Syllabus includes TextBook, **each** TextBook, Syllabus **combination occurs at most once**.
This association with TextBook, Syllabus **provides the preferred identification scheme for** SyllabusIncludesTextBook.
Department **is an entity type**.
Reference Scheme: Department has Department_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Department has Department_Id.
Department has DepartmentName.
Department has Course.
Department has ProgrameChair.

Section 3

Relational Schema Diagram



SQL Code

```
CREATE SCHEMA ORMModel1
GO

GO


CREATE TABLE ORMModel1.Department
(
	departmentId int IDENTITY (1, 1) NOT NULL,
	departmentName nvarchar(max) NOT NULL,
	programeChairID int NOT NULL,
	CONSTRAINT Department_PK PRIMARY KEY(departmentId),
	CONSTRAINT Department_UC1 UNIQUE(departmentName),
	CONSTRAINT Department_UC2 UNIQUE(programeChairID)
)
GO


CREATE TABLE ORMModel1.Course
(
	courseId int IDENTITY (1, 1) NOT NULL,
	professorID int NOT NULL,
	departmentId int,
	CONSTRAINT Course_PK PRIMARY KEY(courseId)
)
GO


CREATE TABLE ORMModel1.ProgrameChair
(
```

```
	programeChairID int IDENTITY (1, 1) NOT NULL,
	CONSTRAINT ProgrameChair_PK PRIMARY KEY(programeChairID)
)
GO


CREATE TABLE ORMModel1.Professor
(
	professorID int IDENTITY (1, 1) NOT NULL,
	CONSTRAINT Professor_PK PRIMARY KEY(professorID)
)
GO


CREATE TABLE ORMModel1.Syllabus
(
	syllabusId int IDENTITY (1, 1) NOT NULL,
	professorID int,
	programeChairID int,
	CONSTRAINT Syllabus_PK PRIMARY KEY(syllabusId)
)
GO


CREATE TABLE ORMModel1.CourseSchedule
(
	courseScheduleId int IDENTITY (1, 1) NOT NULL,
	CONSTRAINT CourseSchedule_PK PRIMARY KEY(courseScheduleId)
)
GO


CREATE TABLE ORMModel1.CourseHasSyllabusWithCourseSchedule
(
	courseId int NOT NULL,
	syllabusId int NOT NULL,
	courseScheduleId int NOT NULL,
	CONSTRAINT CourseHasSyllabusWithCourseSchedule_PK PRIMARY KEY(courseId,
syllabusId)
)
GO


CREATE TABLE ORMModel1.Icourse
(
	icourseId int NOT NULL,
	CONSTRAINT Icourse_PK PRIMARY KEY(icourseId)
)
GO


CREATE TABLE ORMModel1.InPerson
(
	inPersonId int NOT NULL,
	CONSTRAINT InPerson_PK PRIMARY KEY(inPersonId)
)
GO


CREATE TABLE ORMModel1.Fall
(
	fallId int NOT NULL,
```

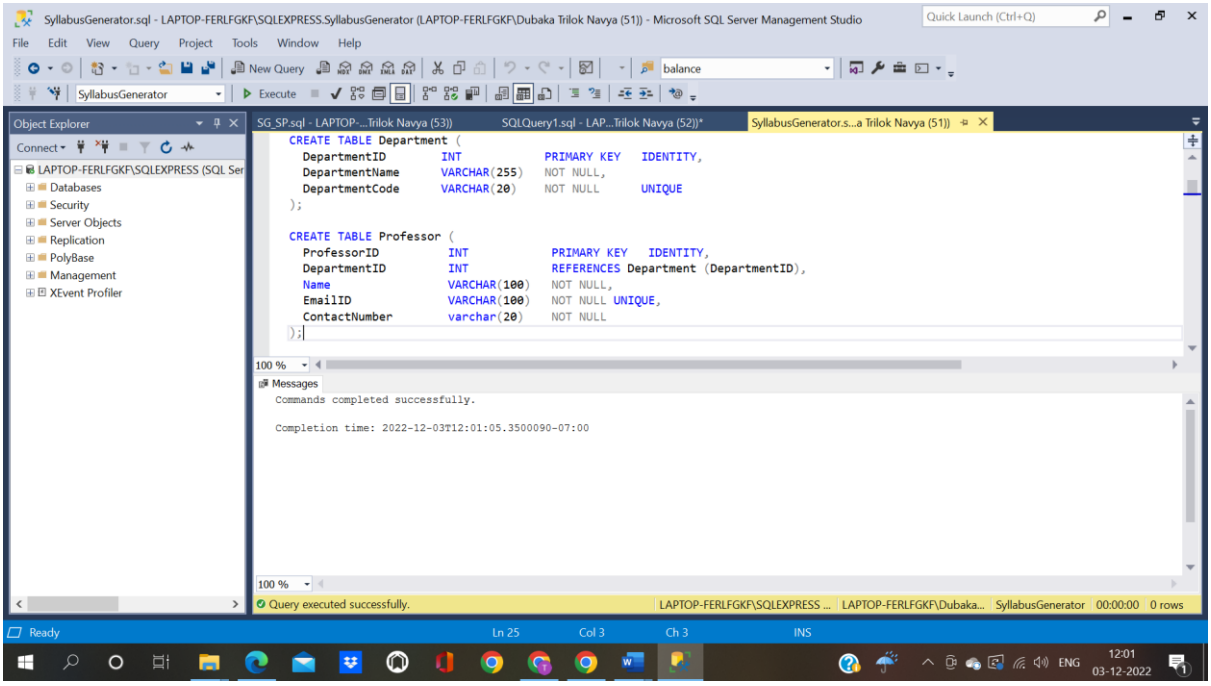
```
	CONSTRAINT Fall_PK PRIMARY KEY(fallId)
)
GO


CREATE TABLE ORMModel1.Spring
(
	springId int NOT NULL,
	CONSTRAINT Spring_PK PRIMARY KEY(springId)
)
GO


CREATE TABLE ORMModel1.CourseScheduleBeginOnStartDateAndEndsOnEndDate
(
	endDate date NOT NULL,
	startDate date NOT NULL,
	courseScheduleId int NOT NULL,
	CONSTRAINT CourseScheduleBeginOnStartDateAndEndsOnEndDate_PK PRIMARY
KEY(startDate, endDate)
)
GO


CREATE TABLE ORMModel1.TextBook
(
	textBookId int IDENTITY (1, 1) NOT NULL,
	CONSTRAINT TextBook_PK PRIMARY KEY(textBookId)
)
GO


CREATE TABLE ORMModel1.SyllabusIncludesTextBook
(
	syllabusId int NOT NULL,
	textBookId int NOT NULL,
	CONSTRAINT SyllabusIncludesTextBook_PK PRIMARY KEY(textBookId, syllabusId)
)
GO


ALTER TABLE ORMModel1.Department ADD CONSTRAINT Department_FK FOREIGN KEY
(programeChairID) REFERENCES ORMModel1.ProgrameChair (programeChairID) ON DELETE NO
ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.Course ADD CONSTRAINT Course_FK1 FOREIGN KEY (departmentId)
REFERENCES ORMModel1.Department (departmentId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.Course ADD CONSTRAINT Course_FK2 FOREIGN KEY (professorID)
REFERENCES ORMModel1.Professor (professorID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.Syllabus ADD CONSTRAINT Syllabus_FK1 FOREIGN KEY (professorID)
REFERENCES ORMModel1.Professor (professorID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
```

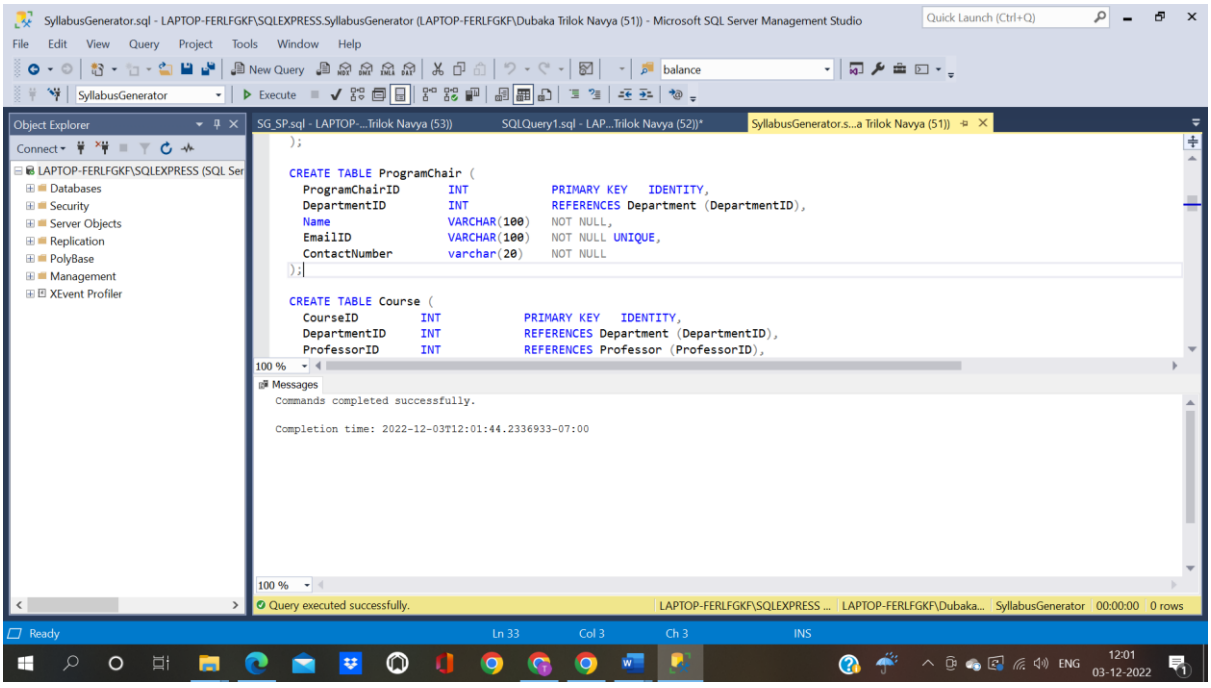
```
ALTER TABLE ORMModel1.Syllabus ADD CONSTRAINT Syllabus_FK2 FOREIGN KEY
(programeChairID) REFERENCES ORMModel1.ProgrameChair (programeChairID) ON DELETE NO
ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.CourseHasSyllabusWithCourseSchedule ADD CONSTRAINT
CourseHasSyllabusWithCourseSchedule_FK1 FOREIGN KEY (courseId) REFERENCES
ORMModel1.Course (courseId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.CourseHasSyllabusWithCourseSchedule ADD CONSTRAINT
CourseHasSyllabusWithCourseSchedule_FK2 FOREIGN KEY (syllabusId) REFERENCES
ORMModel1.Syllabus (syllabusId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.CourseHasSyllabusWithCourseSchedule ADD CONSTRAINT
CourseHasSyllabusWithCourseSchedule_FK3 FOREIGN KEY (courseScheduleId) REFERENCES
ORMModel1.CourseSchedule (courseScheduleId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.Icourse ADD CONSTRAINT Icourse_FK FOREIGN KEY (icourseId)
REFERENCES ORMModel1.Course (courseId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.InPerson ADD CONSTRAINT InPerson_FK FOREIGN KEY (inPersonId)
REFERENCES ORMModel1.Course (courseId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.Fall ADD CONSTRAINT Fall_FK FOREIGN KEY (fallId) REFERENCES
ORMModel1.Syllabus (syllabusId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.Spring ADD CONSTRAINT Spring_FK FOREIGN KEY (springId)
REFERENCES ORMModel1.Syllabus (syllabusId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.CourseScheduleBeginOnStartDateAndEndsOnEndDate ADD CONSTRAINT
CourseScheduleBeginOnStartDateAndEndsOnEndDate_FK FOREIGN KEY (courseScheduleId)
REFERENCES ORMModel1.CourseSchedule (courseScheduleId) ON DELETE NO ACTION ON UPDATE
NO ACTION
GO


ALTER TABLE ORMModel1.SyllabusIncludesTextBook ADD CONSTRAINT
SyllabusIncludesTextBook_FK1 FOREIGN KEY (syllabusId) REFERENCES ORMModel1.Syllabus
(syllabusId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO


ALTER TABLE ORMModel1.SyllabusIncludesTextBook ADD CONSTRAINT
SyllabusIncludesTextBook_FK2 FOREIGN KEY (textBookId) REFERENCES ORMModel1.TextBook
(textBookId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
```

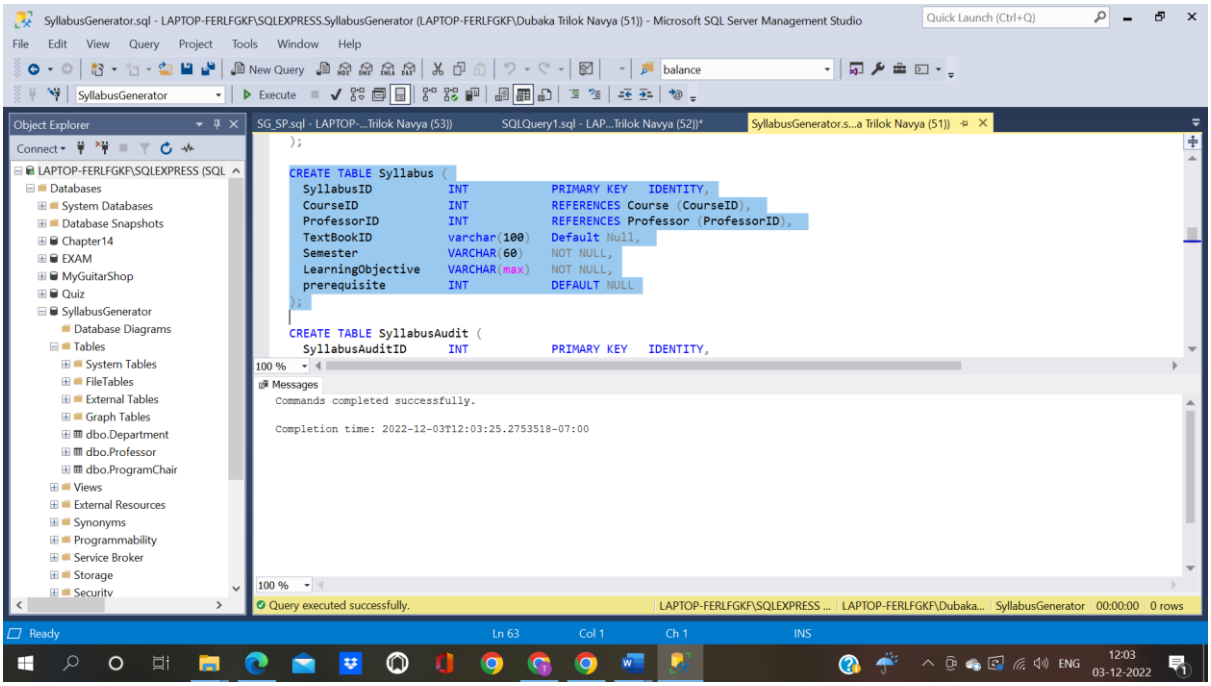
```
GO
```

Output

Tables Creation






SyllabusGenerator.sql - LAPTOP-FERLFGKF\SQLEXPRESS.SyllabusGenerator (LAPTOP-FERLFGKF\Dubaka Trilok Navya (51)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
balance
SyllabusGenerator
Execute
Object Explorer
Connect
LAPTOP-FERLFGKF\SQLEXPRESS (SQL
Databases
System Databases
Database Snapshots
Chapter14
EXAM
MyGuitarShop
Quiz
SyllabusGenerator
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Department
dbo.Professor
dbo.ProgramChair
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
SG_SP.sql - LAPTOP-...Trilok Navya (53))
SQLQuery1.sql - LAP...Trilok Navya (52))*
SyllabusGenerator.s...a Trilok Navya (51))
);
CREATE TABLE Syllabus (
SyllabusID INT PRIMARY KEY IDENTITY,
CourseID INT REFERENCES Course (CourseID),
ProfessorID INT REFERENCES Professor (ProfessorID),
TextBookID varchar(100) Default Null,
Semester VARCHAR(60) NOT NULL,
LearningObjective VARCHAR(max) NOT NULL,
prerequisite INT DEFAULT NULL
);
CREATE TABLE SyllabusAudit (
SyllabusAuditID INT PRIMARY KEY IDENTITY,
100 %
Messages
Commands completed successfully.
Completion time: 2022-12-03T12:03:25.2753518-07:00
100 %
Query executed successfully.
LAPTOP-FERLFGKF\SQLEXPRESS ...
LAPTOP-FERLFGKF\Dubaka...
SyllabusGenerator
00:00:00
0 rows
Ready
Ln 63
Col 1
Ch 1
INS
ENG
12:03
03-12-2022

Section 4

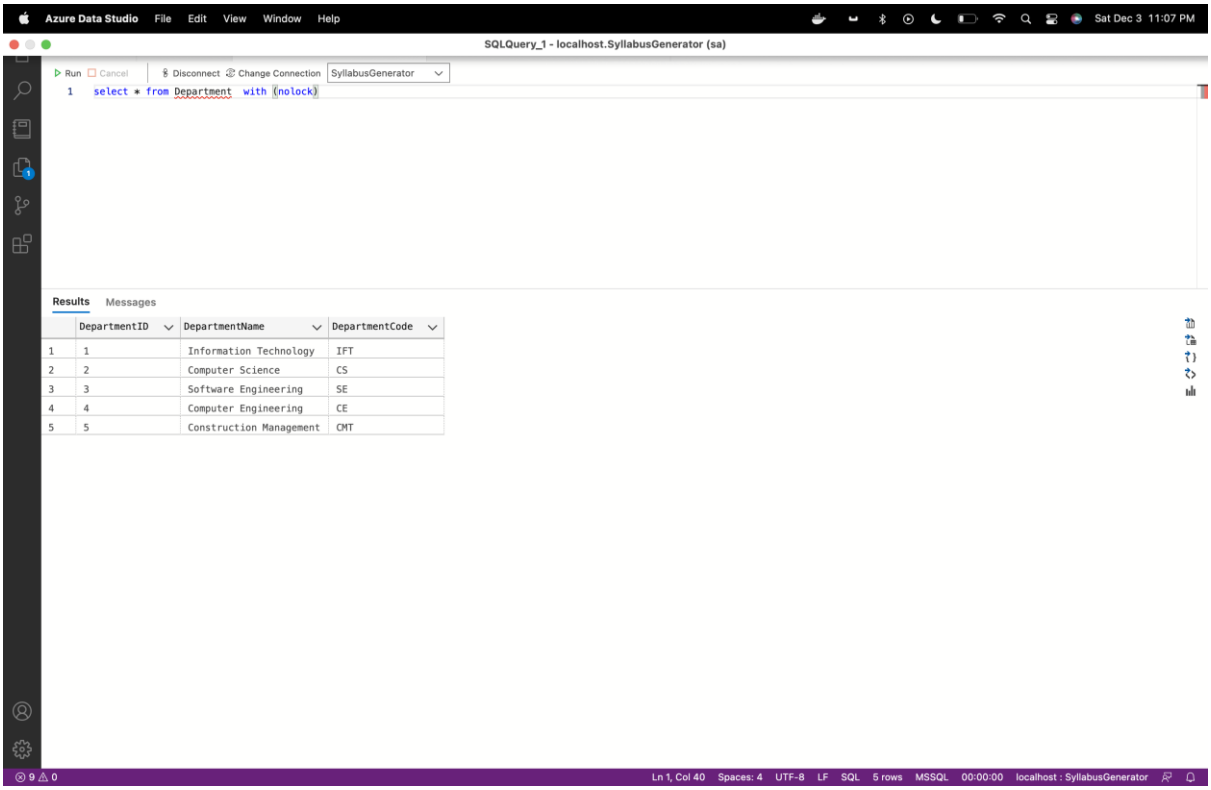
Relational Tables:

1)Department

```
CREATE TABLE Department (
  DepartmentID       INT          PRIMARY KEY   IDENTITY,
  DepartmentName     VARCHAR(255)   NOT NULL,
  DepartmentCode     VARCHAR(20)    NOT NULL     UNIQUE
);


SET IDENTITY_INSERT Department ON;
INSERT INTO Department (DepartmentID, DepartmentName, DepartmentCode) VALUES
(1, 'Information Technology', 'IFT'),
(2, 'Computer Science','CS'),
(3, 'Software Engineering','SE'),
(4, 'Computer Engineering', 'CE'),
(5, 'Construction Management', 'CMT');
SET IDENTITY_INSERT Department OFF;
```

Department:

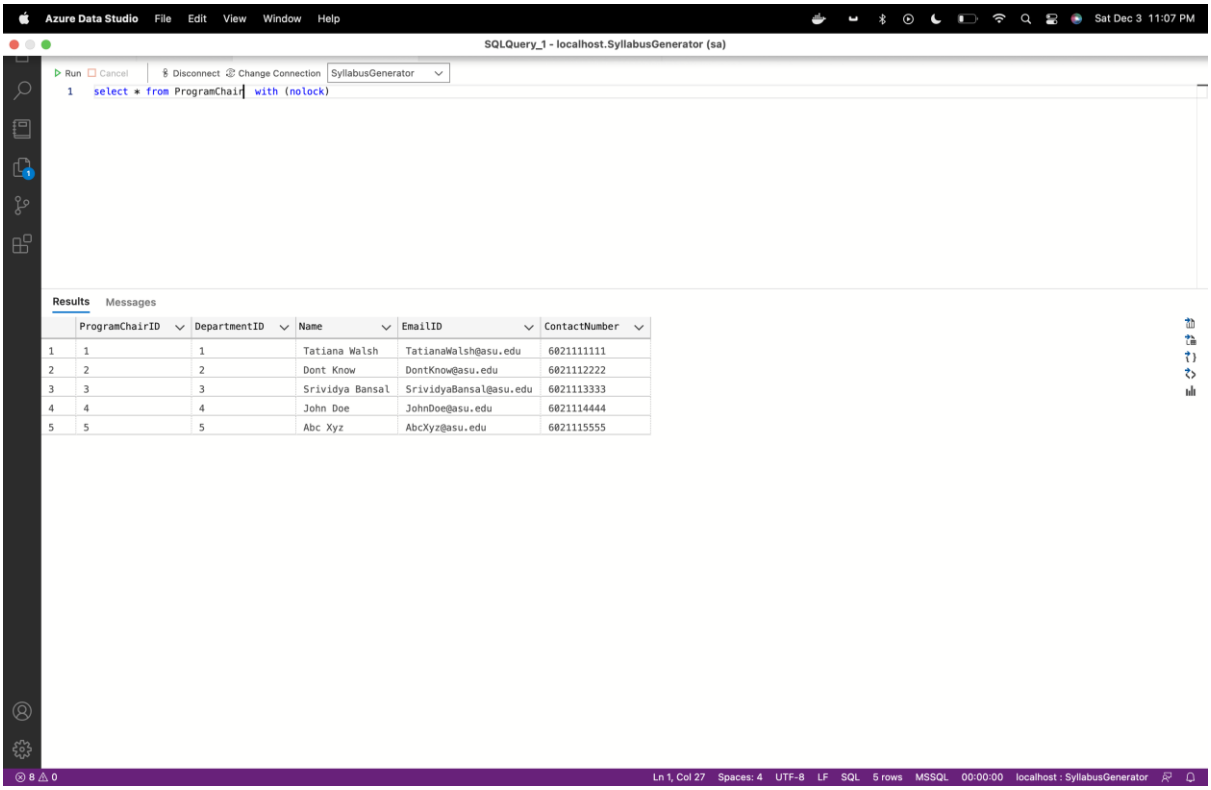
Query: `select * from Department  with (nolock)`



2) Program Chair

```
SET IDENTITY_INSERT ProgramChair ON;

INSERT INTO ProgramChair (ProgramChairID, DepartmentID, Name,EmailID,ContactNumber) VALUES

(1, 1, 'Tatiana Walsh','TatianaWalsh@asu.edu','6021111111'),

(2, 2, 'Dont Know','DontKnow@asu.edu','6021112222'),

(3, 3, 'Srividya Bansal','SrividyaBansal@asu.edu','6021113333'),

(4, 4, 'John Doe','JohnDoe@asu.edu','6021114444'),

(5, 5, 'Abc Xyz','AbcXyz@asu.edu','6021115555');


SET IDENTITY_INSERT ProgramChair OFF;
```

Query: select * from ProgramChair with (nolock)



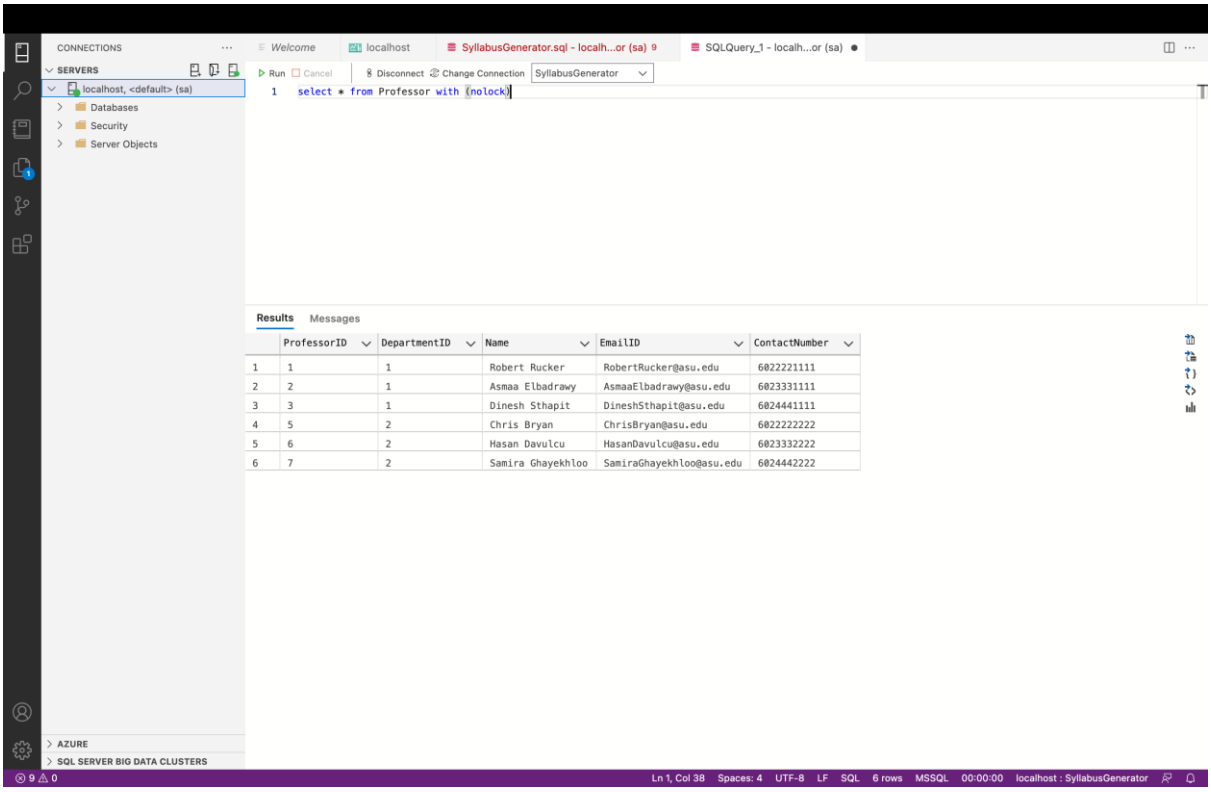
3)Professor

```
CREATE TABLE Professor (
  ProfessorID        INT          PRIMARY KEY   IDENTITY,
  DepartmentID       INT          REFERENCES Department (DepartmentID),
  Name               VARCHAR(100)   NOT NULL,
  EmailID            VARCHAR(100)   NOT NULL UNIQUE,
  ContactNumber      varchar(20)    NOT NULL
);
SET IDENTITY_INSERT Professor ON;
INSERT INTO Professor (ProfessorID, DepartmentID, Name,EmailID,ContactNumber) VALUES
(1, 1, 'Robert Rucker','RobertRucker@asu.edu','6022221111'),
(2, 1,'Asmaa Elbadrawy','AsmaaElbadrawy@asu.edu','6023331111'),
(3, 1,'Dinesh Sthapit','DineshSthapit@asu.edu','6024441111'),
(5, 2, 'Chris Bryan','ChrisBryan@asu.edu','6022222222'),
(6, 2, 'Hasan Davulcu','HasanDavulcu@asu.edu','6023332222'),
(7, 2, 'Samira Ghayekhloo','SamiraGhayekhloo@asu.edu','6024442222');
```

```
SET IDENTITY_INSERT Professor OFF;
```

Query:

```
select * from Professor with (nolock)
```



4)TextBook Table

```
CREATE TABLE TextBook (
  TextBookID       INT          PRIMARY KEY   IDENTITY,
  TextBookName     VARCHAR(255)   NOT NULL,
  Category         VARCHAR(255)   NOT NULL,

);
SET IDENTITY_INSERT TextBook ON;
INSERT INTO TextBook (TextBookID, TextBookName,Category) VALUES
(1, 'Murachs SQL server for developers','Database'),
(2, 'Halpin, Morgan( 2008) Information Modeling and Relational Databases 2nd edition, Kaufman','Database'),
(3, 'Data Science for Business by F. Provost and T. Fawcett','DataScience'),
```

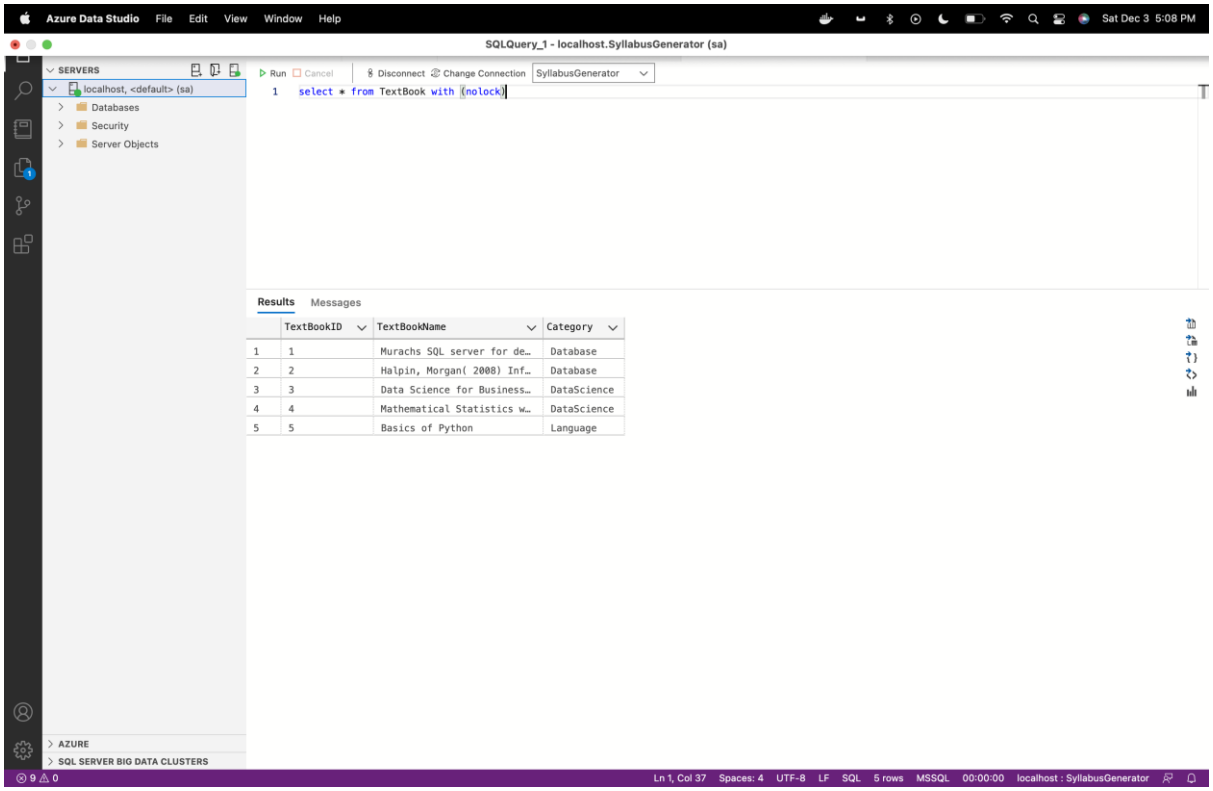
(4, 'Mathematical Statistics with Applications, 7th edition, by Wackerly, Mendenhall, and Scheaffer, Brooks/Cole, Cengage Learning, 2008.','DataScience'),

(5, 'Basics of Python','Language');

SET IDENTITY_INSERT TextBook OFF;

Query:

```
select * from TextBook with (nolock)
```



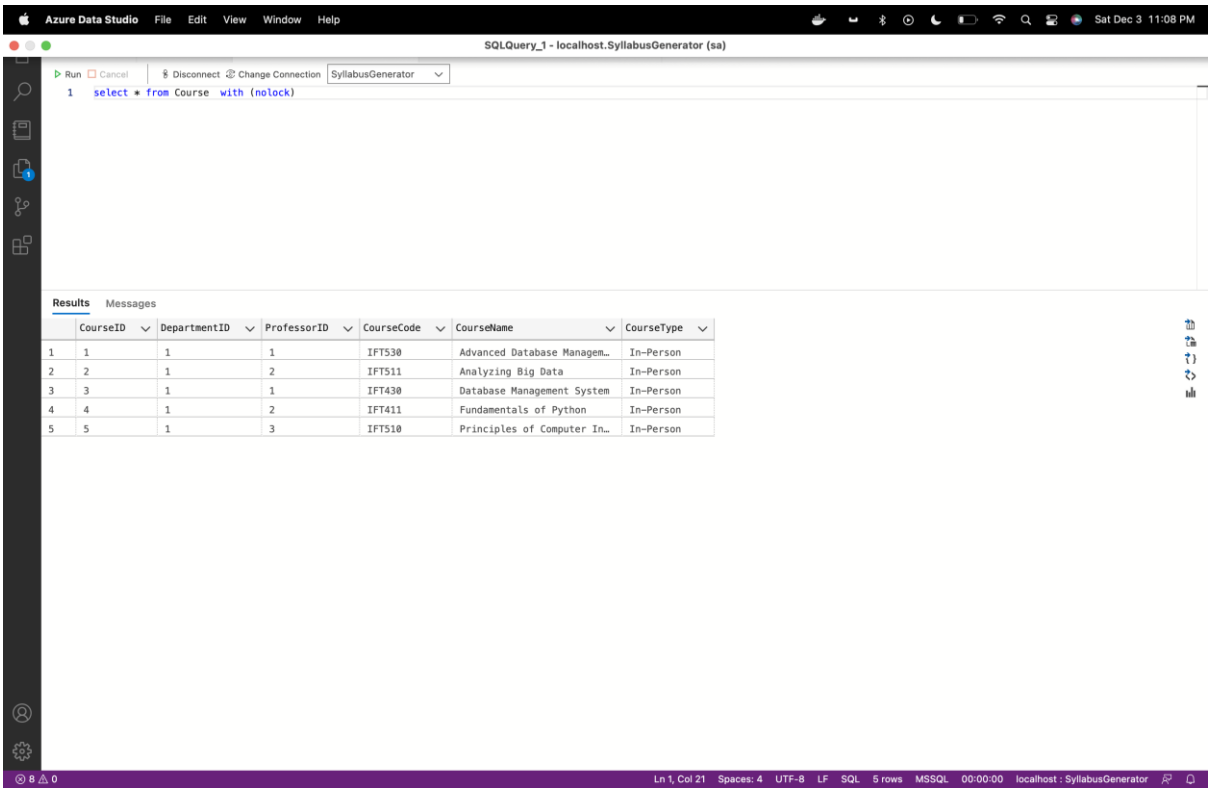
5)Course

```
CREATE TABLE Course (
  CourseID        INT          PRIMARY KEY   IDENTITY,
  DepartmentID    INT          REFERENCES Department (DepartmentID),
  ProfessorID     INT          REFERENCES Professor (ProfessorID),
  CourseCode      VARCHAR(50)   NOT NULL UNIQUE,
  CourseName      VARCHAR(50)   NOT NULL,
  CourseType      VARCHAR(10)   NOT NULL
);
```

```
SET IDENTITY_INSERT course ON;

INSERT INTO course (CourseID, DepartmentID, ProfessorID,CourseCode,CourseName,CourseType) VALUES

(1, 1, 1,'IFT530','Advanced Database Management Systems','In-Person'),

(2, 1,2, 'IFT511','Analyzing Big Data','In-Person'),

(3, 1,1, 'IFT430','Database Management System','In-Person'),

(4, 1,2, 'IFT411','Fundamentals of Python','In-Person');

(5, 1,3, 'IFT510','Principles of Computer Information','In-Person');


SET IDENTITY_INSERT course OFF;
```

Query:

```
select * from Course with (nolock)
```

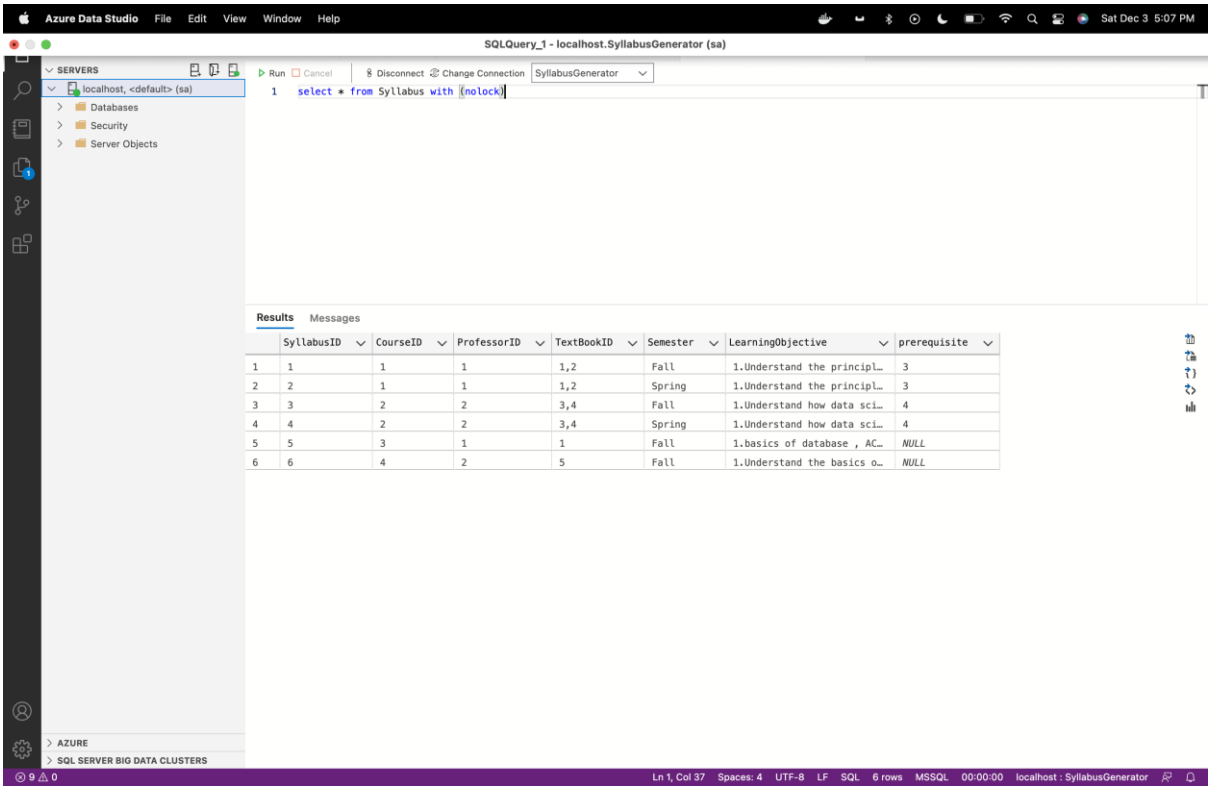


6)Syllabus

```
CREATE TABLE Syllabus (
  SyllabusID          INT          PRIMARY KEY   IDENTITY,
```

```
  CourseID          INT          REFERENCES Course (CourseID),
  ProfessorID       INT          REFERENCES Professor (ProfessorID),
  TextBookID        varchar(100)   Default Null,
  Semester          VARCHAR(60)    NOT NULL,
  LearningObjective   VARCHAR(max)   NOT NULL,
  prerequisite      varchar(max)          DEFAULT NULL
);
SET IDENTITY_INSERT Syllabus ON;
INSERT INTO Syllabus (SyllabusID, CourseID, ProfessorID,textbookid,Semester,LearningObjective,
prerequisite) VALUES
(1, 1, 1,'1,2','Fall','1.Understand the principles of conceptual modeling,2.understand the use of scripts when
implementing SQL code', '3'),
(2, 1, 1,'1,2','Spring','1.Understand the principles of conceptual modeling,2.understand the use of scripts when
implementing SQL code', '3'),
(3, 2, 2,'3,4','Fall','1.Understand how data science can be used as tools to analyze large amounts of data for the
purpose of extracting business value.', '4'),
(4, 2, 2,'3,4','Spring','1.Understand how data science can be used as tools to analyze large amounts of data for
the purpose of extracting business value.', '4'),
(5, 3, 1,'1','Fall','1.basics of database , ACID properties, CRUD methods, normalization', NULL),
(6, 4, 2,'5','Fall','1.Understand the basics of Python, Python libraries', NULL);

SET IDENTITY_INSERT Syllabus OFF;
```

Query: `select * from Syllabus with (nolock)`



7)Course Schedule

```
CREATE TABLE CourseSchedule (
  CourseScheduleID     INT          PRIMARY KEY  IDENTITY,
  SyllabusID           INT          REFERENCES Syllabus (SyllabusID),
  CourseID             INT          REFERENCES Course (CourseID),
  Module               Varchar(20)    NOT NULL,
  CourseContents       Varchar(max)   NOT NULL,
  StartDate            Date         NOT NULL,
  EndDate              Date         NOT NULL
);
SET IDENTITY_INSERT CourseSchedule ON;
INSERT INTO CourseSchedule (CourseScheduleID, SyllabusID, CourseID,Module,CourseContents,StartDate,
EndDate) VALUES
(1, 1, 1,'Module1','Understand the principles of conceptual modeling (ORM as an examplar).', '15 Aug 2022','30
Aug 2022'),
(2, 1, 1,'Module2','Basis of Primary keys, Uniqueness/ Mandatory constraints', '1 Sep 2022', '15 Sep 2022'),
(3, 1, 1,'Module3','basis for determining Foreign Keys, value and frequency constraints, dynamic SQL', '16 Sep
2022', '16 Oct 2022'),
```

(4, 1, 1,'Module4','Window functions, SQL++', '17 Oct 2022', '10 Nov 2022'),

(5, 1, 1,'Module5','Couchbase, Capella-DBaas', '11 Nov 2022', '30 Nov 2022'),

(6, 2, 1,'Module1','Understand the principles of conceptual modeling (ORM as an examplar).', '15 Aug 2022','30 Aug 2022'),

(7, 2, 1,'Module2','Basis of Primary keys, Uniqueness/ Mandatory constraints', '1 Sep 2022', '15 Sep 2022'),

(8, 2, 1,'Module3','basis for determining Foreign Keys, value and frequency constraints, dynamic SQL', '16 Sep 2022', '16 Oct 2022'),

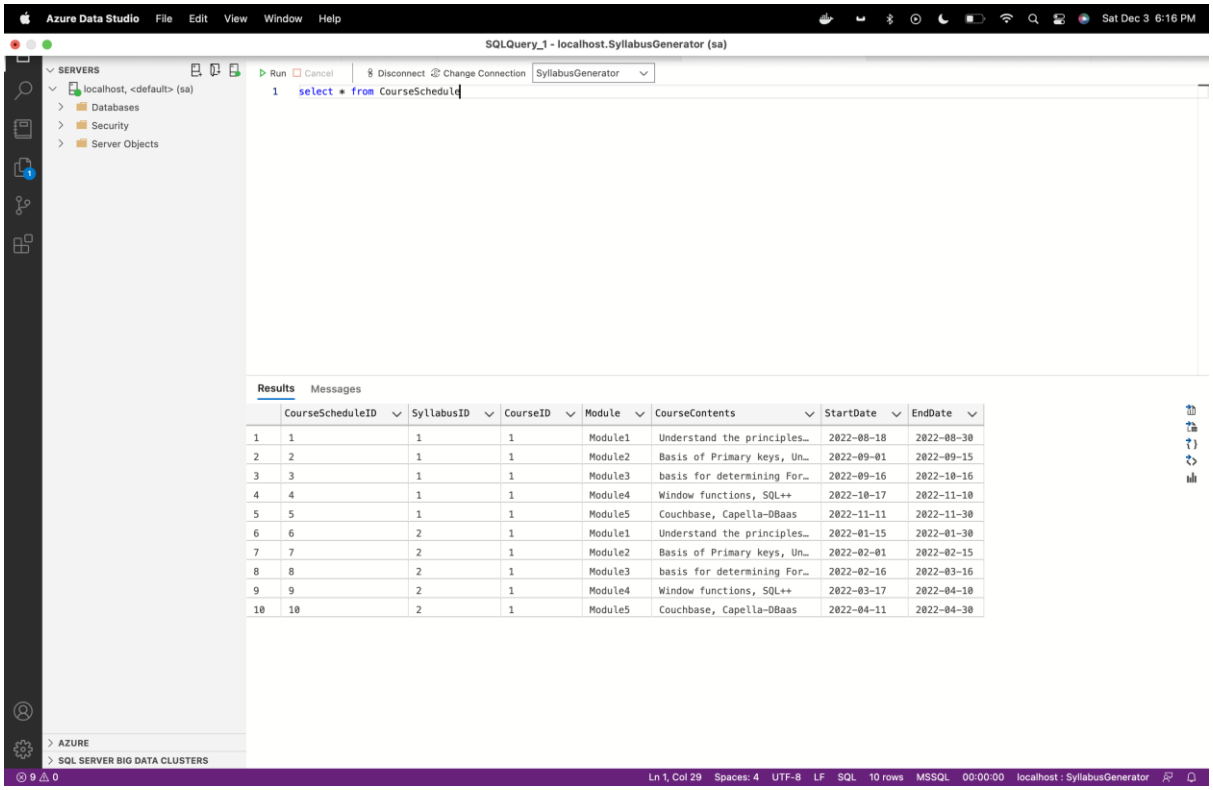
(9, 2, 1,'Module4','Window functions, SQL++', '17 Oct 2022', '10 Nov 2022'),

(10, 2, 1,'Module5','Couchbase, Capella-DBaas', '11 Nov 2022', '30 Nov 2022');

SET IDENTITY_INSERT CourseSchedule OFF;

Query:

```
select * from CourseSchedule with (nolock)
```



Section 5

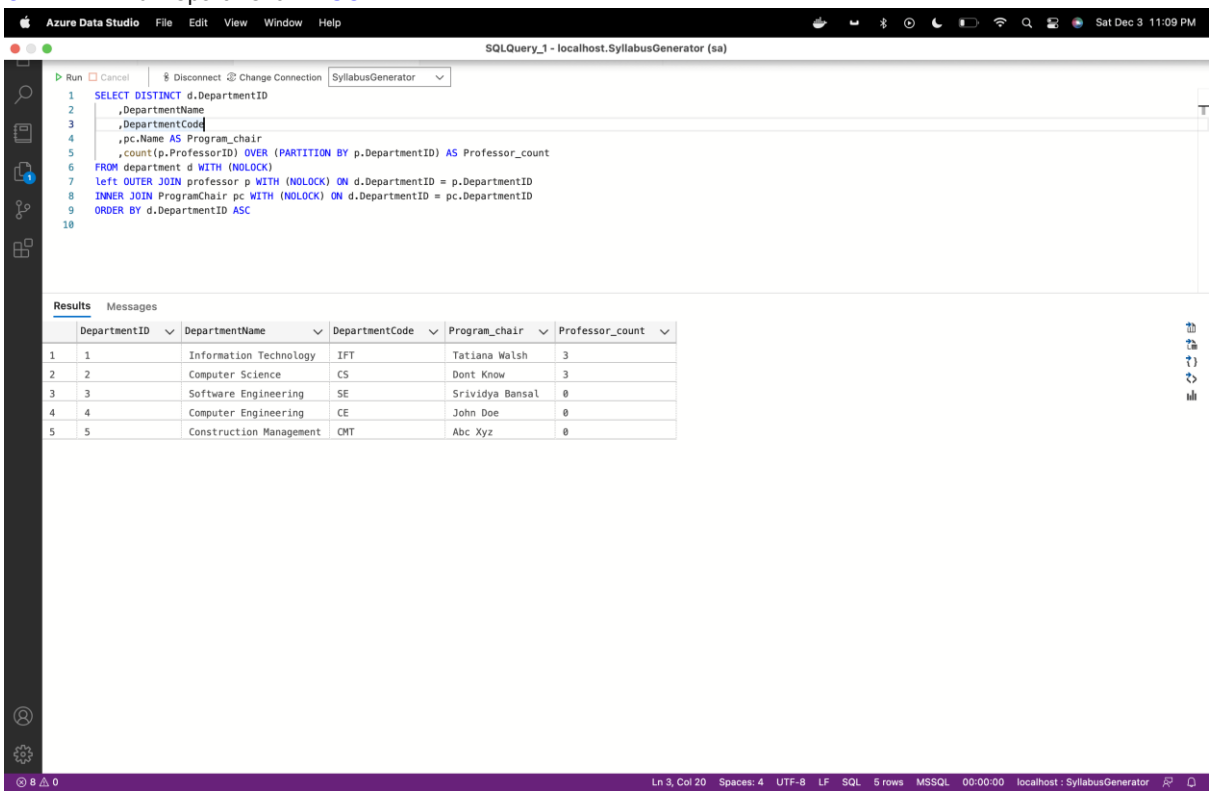
Section 5.1

SQL queries

Question1 :

Retrieving current department details

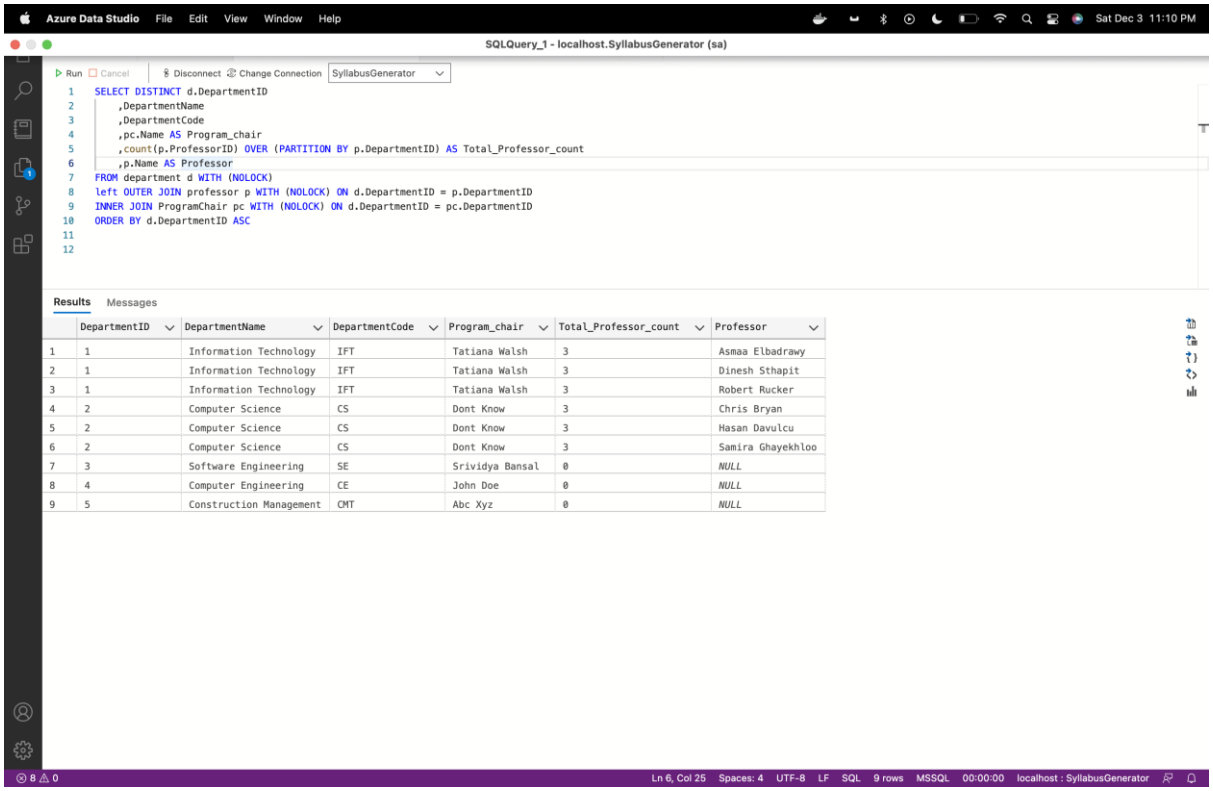
```
SELECT DISTINCT d.DepartmentID
    ,DepartmentName
    ,DepartmentCode
    ,pc.Name AS Program_chair
    ,count(p.ProfessorID) OVER (PARTITION BY p.DepartmentID) AS Professor_count
FROM department d WITH (NOLOCK)
left OUTER JOIN professor p WITH (NOLOCK) ON d.DepartmentID = p.DepartmentID
INNER JOIN ProgramChair pc WITH (NOLOCK) ON d.DepartmentID = pc.DepartmentID
ORDER BY d.DepartmentID ASC
```



Question2:

Department wise program chair and professor details

```
SELECT DISTINCT d.DepartmentID
    ,DepartmentName
    ,DepartmentCode
    ,pc.Name AS Program_chair
    ,count(p.ProfessorID) OVER (PARTITION BY p.DepartmentID) AS Total_Professor_count
    ,p.Name AS Professor
FROM department d WITH (NOLOCK)
left OUTER JOIN professor p WITH (NOLOCK) ON d.DepartmentID = p.DepartmentID
INNER JOIN ProgramChair pc WITH (NOLOCK) ON d.DepartmentID = pc.DepartmentID
ORDER BY d.DepartmentID ASC
```



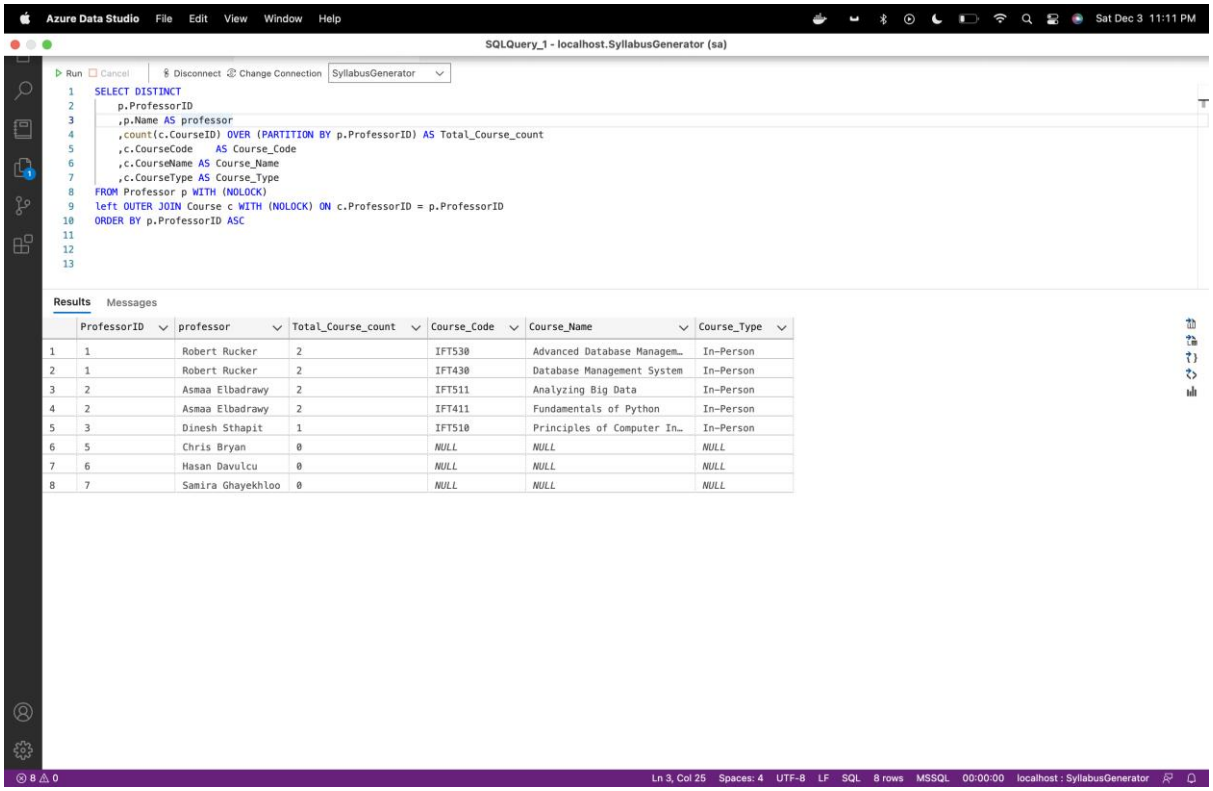
	DepartmentID	DepartmentName	DepartmentCode	Program_chair	Total_Professor_count	Professor
1	1	Information Technology	IFT	Tatiana Walsh	3	Asmaa Elbadrawy
2	1	Information Technology	IFT	Tatiana Walsh	3	Dinesh Sthapit
3	1	Information Technology	IFT	Tatiana Walsh	3	Robert Rucker
4	2	Computer Science	CS	Dont Know	3	Chris Bryan
5	2	Computer Science	CS	Dont Know	3	Hasan Davulcu
6	2	Computer Science	CS	Dont Know	3	Samira Ghayekhloo
7	3	Software Engineering	SE	Srividya Bansal	0	*NULL*
8	4	Computer Engineering	CE	John Doe	0	*NULL*
9	5	Construction Management	CMT	Abc Xyz	0	*NULL*

Question3:

Retrieving Professor teaching particular course

```
SELECT DISTINCT
    p.ProfessorID
    ,p.Name AS professor
```

```
    ,count(c.CourseID) OVER (PARTITION BY p.ProfessorID) AS Total_Course_count
    ,c.CourseCode    AS Course_Code
    ,c.CourseName AS Course_Name
    ,c.CourseType AS Course_Type
FROM Professor p WITH (NOLOCK)
left OUTER JOIN Course c WITH (NOLOCK) ON c.ProfessorID = p.ProfessorID
ORDER BY p.ProfessorID ASC
```

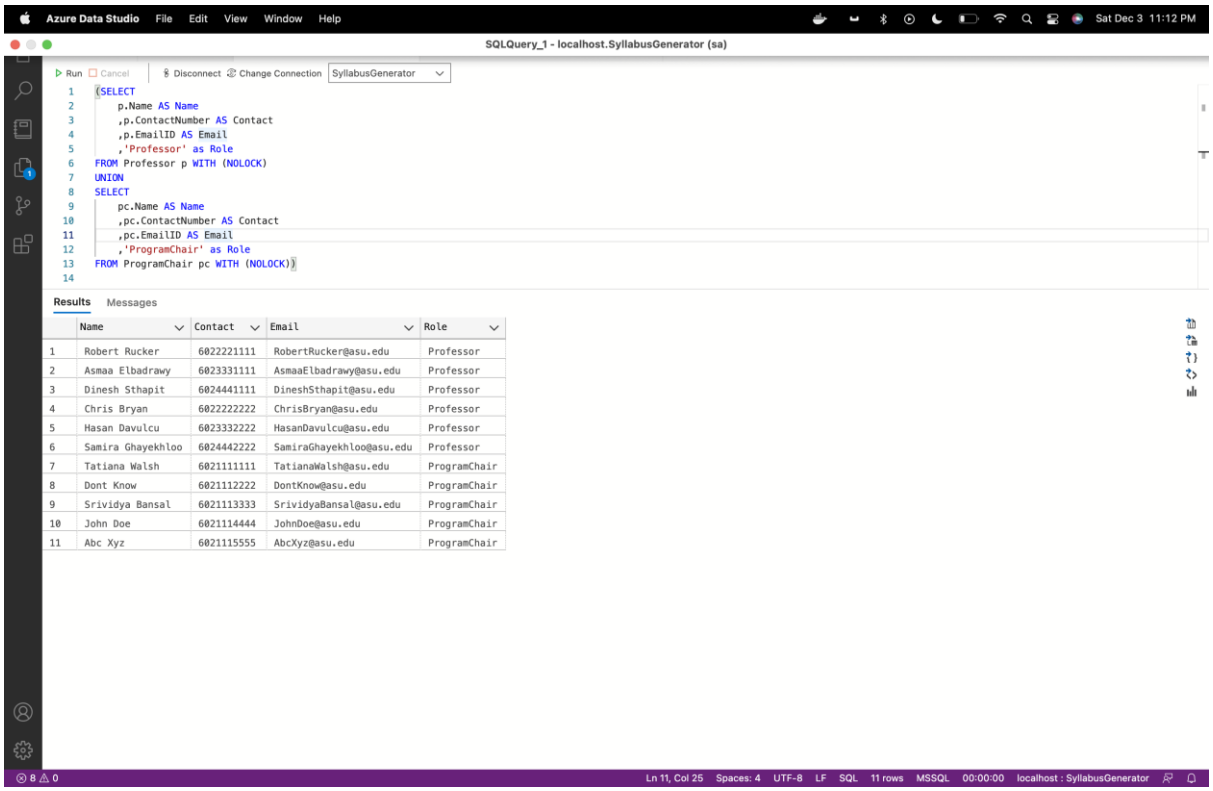


Question4:

Professor and Program chair contact details

```
(SELECT
    p.Name AS Name
    ,p.ContactNumber AS Contact
    ,p.EmailID AS Email
    ,'Professor' as Role
FROM Professor p WITH (NOLOCK)
```

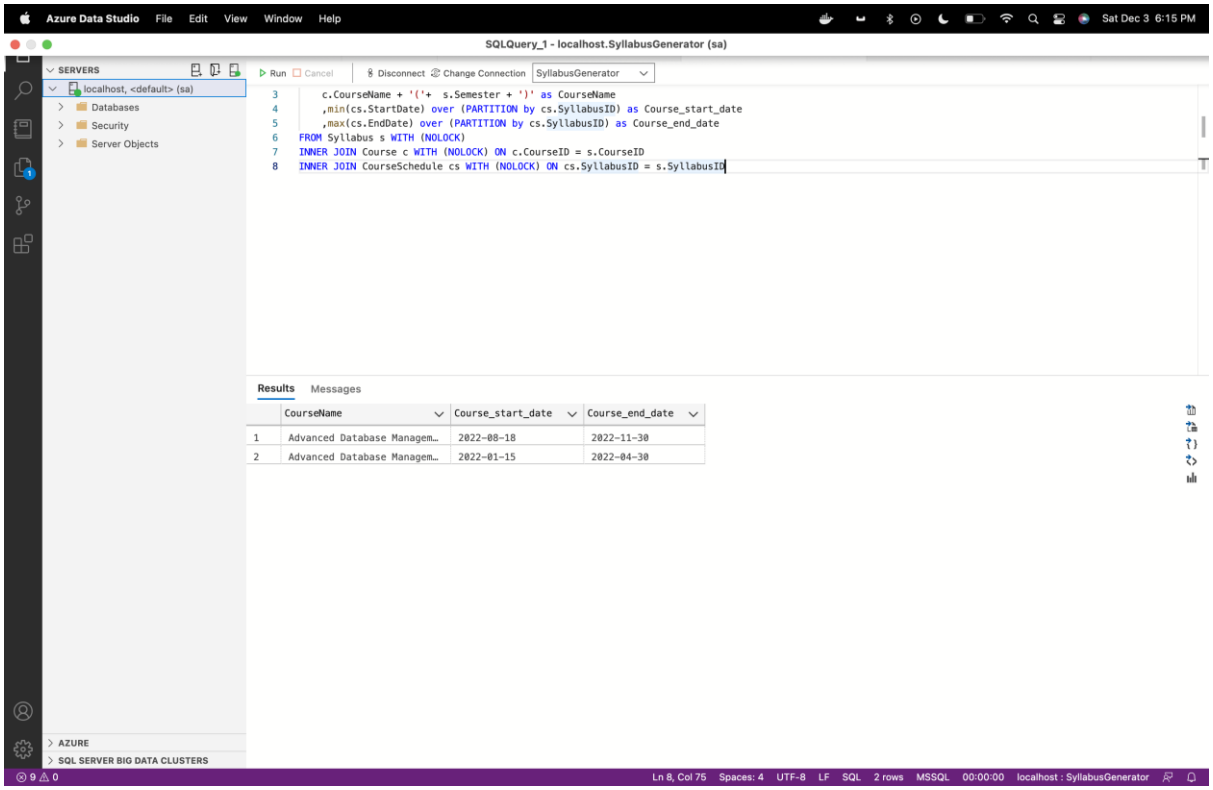
```
UNION
SELECT
    pc.Name AS Name
    ,pc.ContactNumber AS Contact
    ,pc.EmailID AS Email
    ,'ProgramChair' as Role
FROM ProgramChair pc WITH (NOLOCK))
```



Question5:

Retrieving start and end date of Syllabus

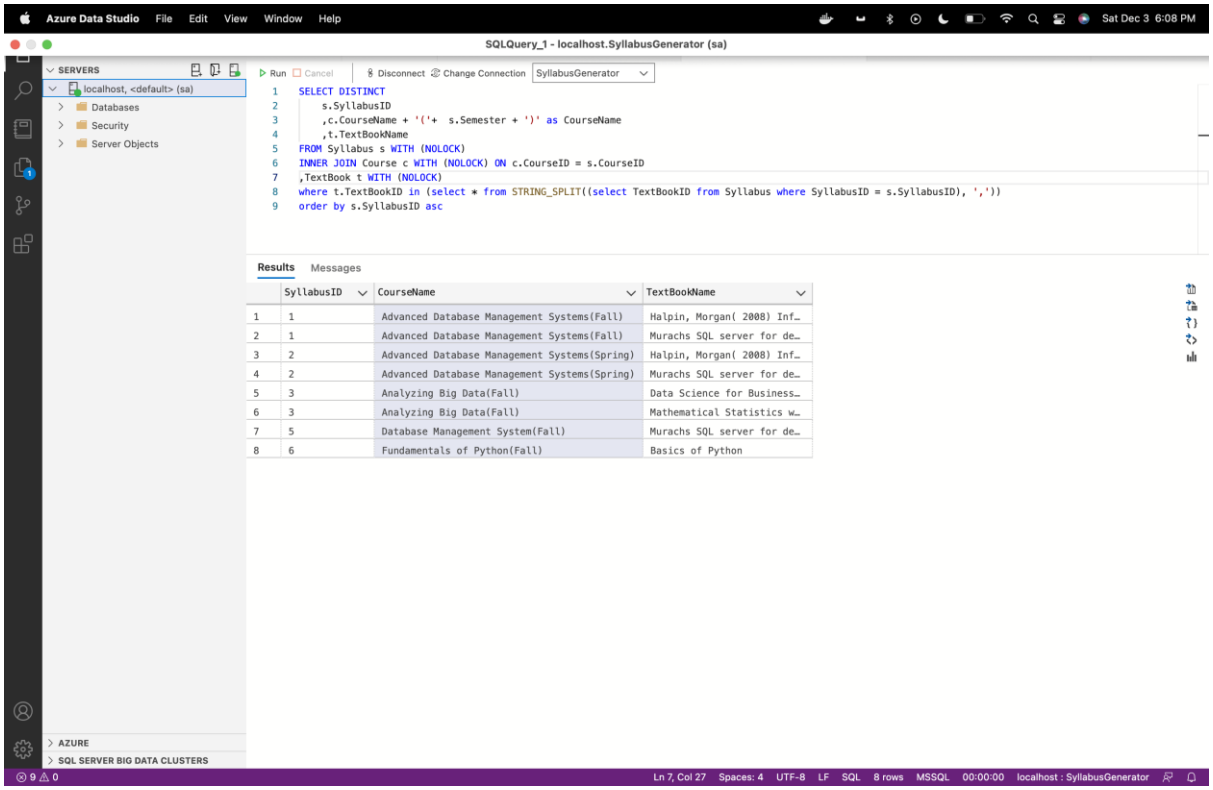
```
SELECT DISTINCT
    c.CourseName + '('+  s.Semester + ')' as CourseName
    ,min(cs.StartDate) over (PARTITION by cs.SyllabusID) as Course_start_date
    ,max(cs.EndDate) over (PARTITION by cs.SyllabusID) as Course_end_date
FROM Syllabus s WITH (NOLOCK)
INNER JOIN Course c WITH (NOLOCK) ON c.CourseID = s.CourseID
INNER JOIN CourseSchedule cs WITH (NOLOCK) ON cs.SyllabusID = s.SyllabusID
```



Question 6:

Course wise textbook list

```
SELECT DISTINCT
    s.SyllabusID
    ,c.CourseName + '('+  s.Semester + ')' as CourseName
    ,t.TextBookName
FROM Syllabus s WITH (NOLOCK)
INNER JOIN Course c WITH (NOLOCK) ON c.CourseID = s.CourseID
,TextBook t WITH (NOLOCK)
where t.TextBookID in (select * from STRING_SPLIT((select TextBookID from Syllabus where SyllabusID = s.SyllabusID), ','))
order by s.SyllabusID asc
```



Section 5

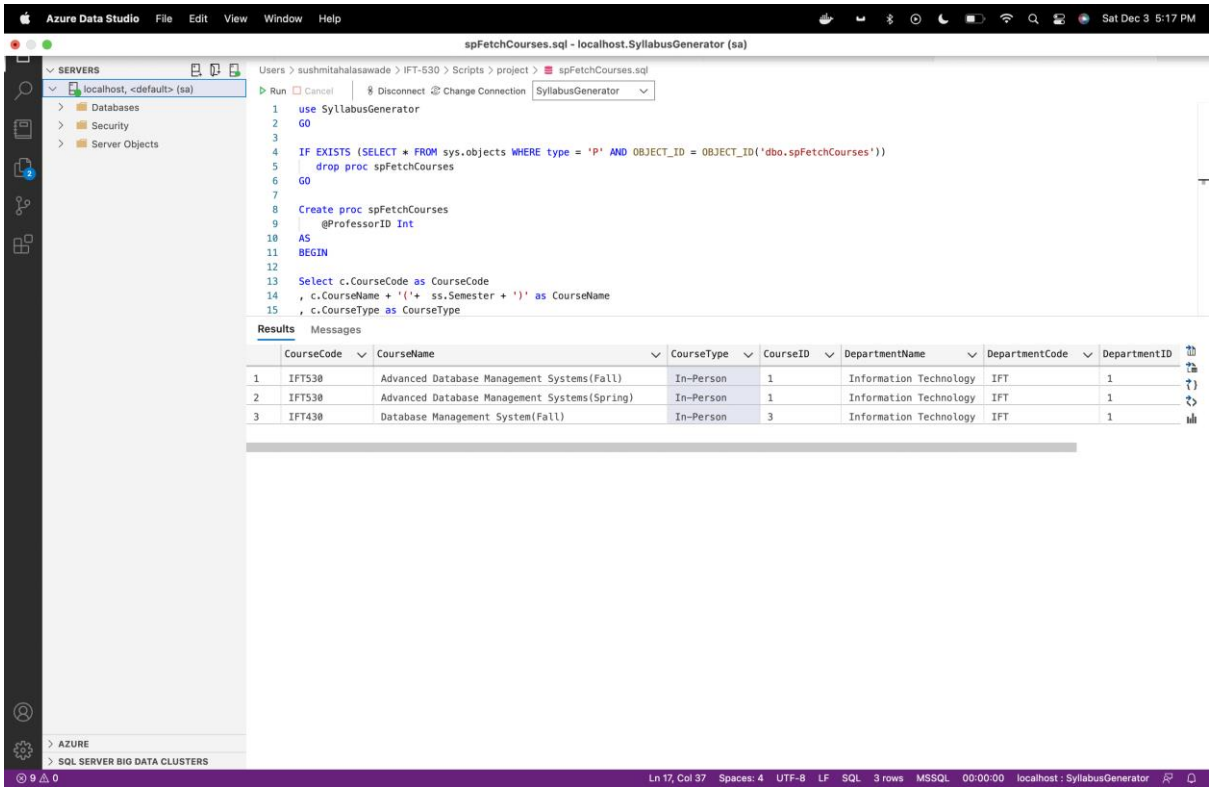
Additional Constraints

1.Stored procedure to fetch all courses of professor:

Query:

```
use SyllabusGenerator

GO


IF EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID =
OBJECT_ID('dbo.spFetchCourses'))

  drop proc spFetchCourses

GO


Create proc spFetchCourses

   @ProfessorID Int

AS

BEGIN
```

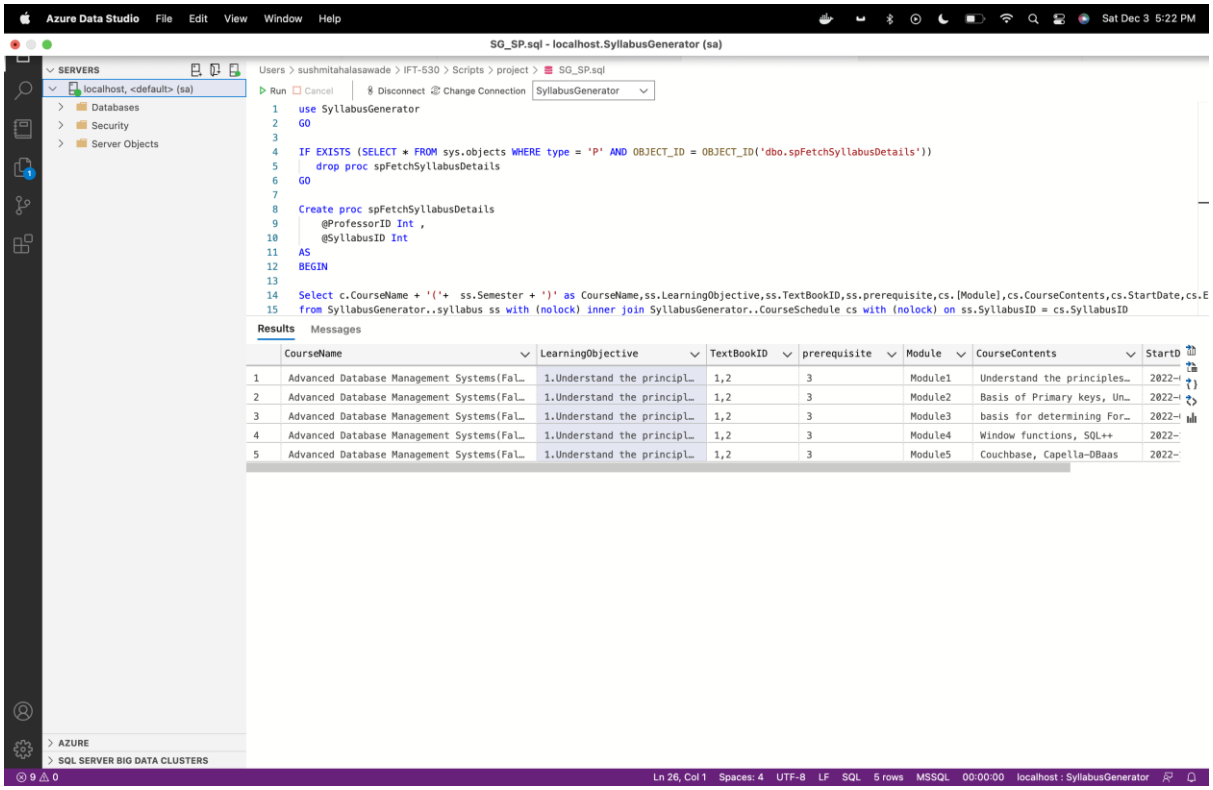
```
Select c.CourseCode as CourseCode
, c.CourseName + '('+  ss.Semester + ')' as CourseName
, c.CourseType as CourseType
, c.CourseID as CourseID
, d.DepartmentName as DepartmentName
, d.DepartmentCode as DepartmentCode
, d.DepartmentID as DepartmentID
, ss.SyllabusID
from SyllabusGenerator..course c with (nolock) inner join SyllabusGenerator..Syllabus ss with (nolock) on c.CourseID = ss.CourseID
inner join SyllabusGenerator..Department d with (nolock) on d.DepartmentID = c.DepartmentID
where  c.ProfessorID =  @ProfessorID



END

exec spFetchCourses 1
```



2.Stored procedure to fetch Syllabus details of course taught by professor:

Query:

```
use SyllabusGenerator
GO

IF EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID =
OBJECT_ID('dbo.spFetchSyllabusDetails'))
  drop proc spFetchSyllabusDetails
GO

Create proc spFetchSyllabusDetails
   @ProfessorID Int ,
   @SyllabusID Int
AS
BEGIN

Select c.CourseName + '('+  ss.Semester + ')' as
CourseName,ss.LearningObjective,ss.TextBookID,ss.prerequisite,cs.[Module],cs.CourseContents,cs.StartDate,c
s.EndDate
from SyllabusGenerator..syllabus ss with (nolock) inner join SyllabusGenerator..CourseSchedule cs with (nolock)
on ss.SyllabusID = cs.SyllabusID
inner join SyllabusGenerator..course c with (nolock) on  c.CourseID = ss.CourseID
where ss.SyllabusID = @SyllabusID and ss.ProfessorID = @ProfessorID

END

--
--exec spFetchSyllabusDetails 1, 1
```



3. Stored Procedure to insert or update the syllabus.

Query:

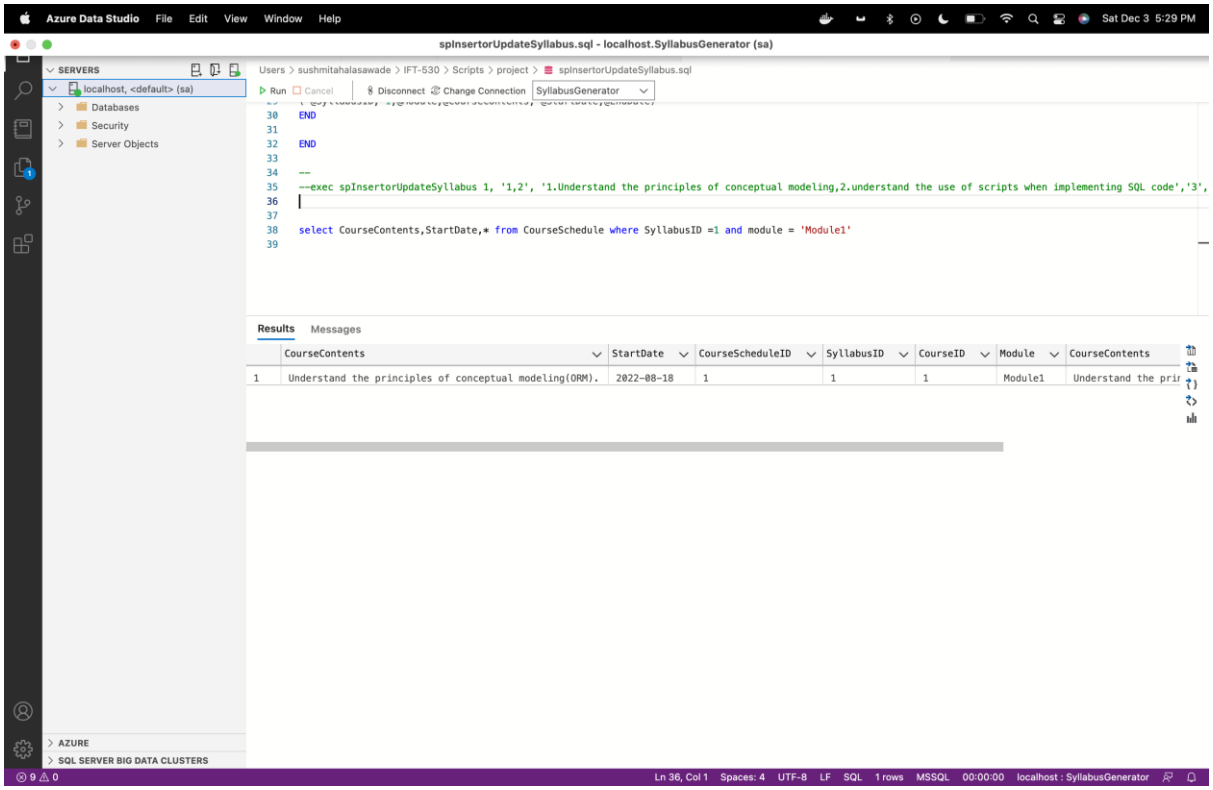
```
use SyllabusGenerator
GO

IF EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID =
OBJECT_ID('dbo.spInsertorUpdateSyllabus'))
  drop proc spInsertorUpdateSyllabus
GO

Create proc spInsertorUpdateSyllabus
   @SyllabusID varchar(10),
   @textbookid varchar(100),
   @LearningObjective varchar(max),
   @prerequisite varchar(100),
   @Module varchar(20),
   @CourseContents varchar(max),
   @StartDate DATE,
```

```
    @EndDate DATE
AS
BEGIN

update Syllabus with (rowlock) set textbookid = @textbookid, LearningObjective= @LearningObjective,
prerequisite= @prerequisite where SyllabusID = @SyllabusID

if Exists(select * from CourseSchedule where SyllabusID = @SyllabusID and module = @Module)
BEGIN
update CourseSchedule with (rowlock) set CourseContents = @CourseContents, StartDate= @StartDate,
EndDate= @EndDate where SyllabusID = @SyllabusID and module = @Module
END
ELSE
BEGIN
INSERT INTO CourseSchedule ( SyllabusID, CourseID,Module,CourseContents,StartDate, EndDate) VALUES
( @SyllabusID, 1,@Module,@CourseContents, @StartDate,@EndDate)
END

END
```

Test:

```
--exec spInsertorUpdateSyllabus 1, '1,2', '1.Understand the principles of conceptual modeling,2.understand the
use of scripts when implementing SQL code','3','Module1','Understand the principles of conceptual
modeling(ORM).','18 Aug 2022','30 Aug 2022'

select CourseContents,StartDate,* from CourseSchedule where SyllabusID =1 and module = 'Module1'
```

This will update existing module content and start date.



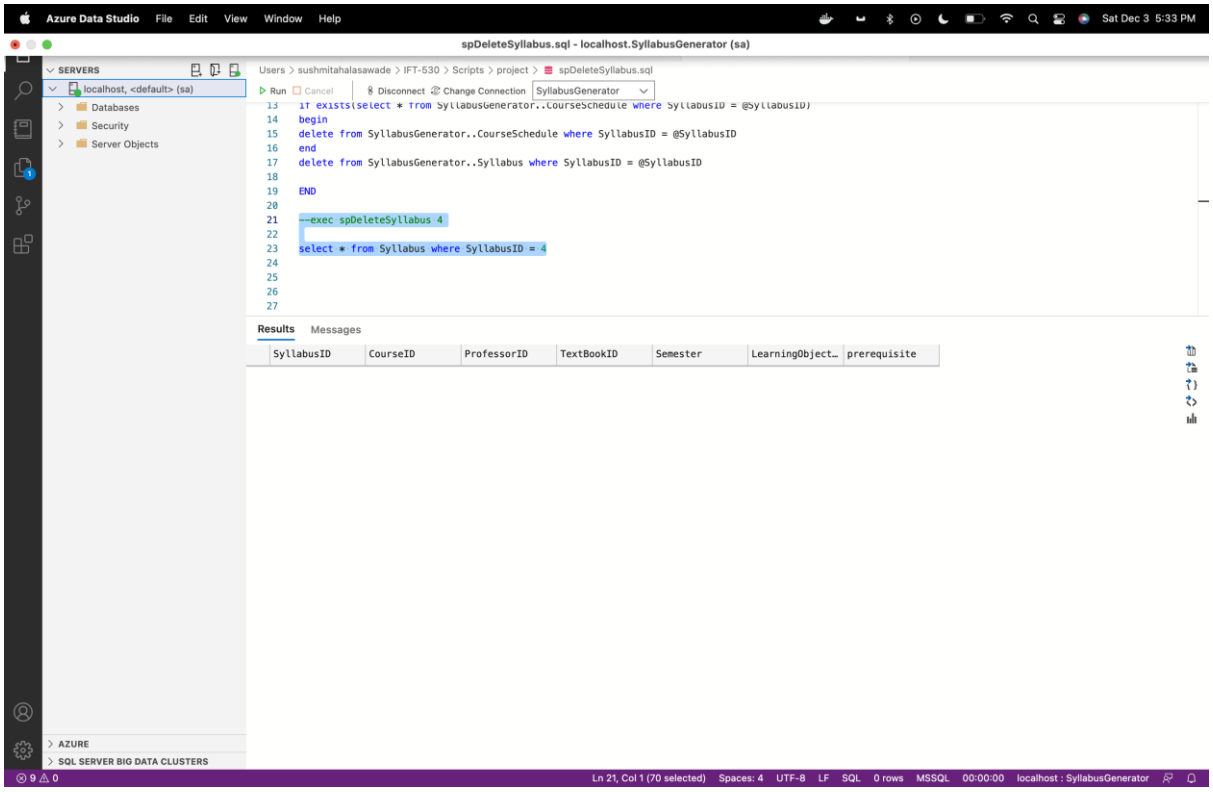
4. Stored Procedure to delete complete syllabus.

```
use SyllabusGenerator
GO

IF EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID =
OBJECT_ID('dbo.spDeleteSyllabus'))
   drop proc spDeleteSyllabus
GO

Create proc spDeleteSyllabus
    @SyllabusID Int
AS
BEGIN

if exists(select * from SyllabusGenerator..CourseSchedule where SyllabusID = @SyllabusID)
begin
delete from SyllabusGenerator..CourseSchedule where SyllabusID = @SyllabusID
end
```

```
delete from SyllabusGenerator..Syllabus where SyllabusID = @SyllabusID


END
```

Test:

```
--exec spDeleteSyllabus 4


select * from Syllabus where SyllabusID = 4
```



Functions

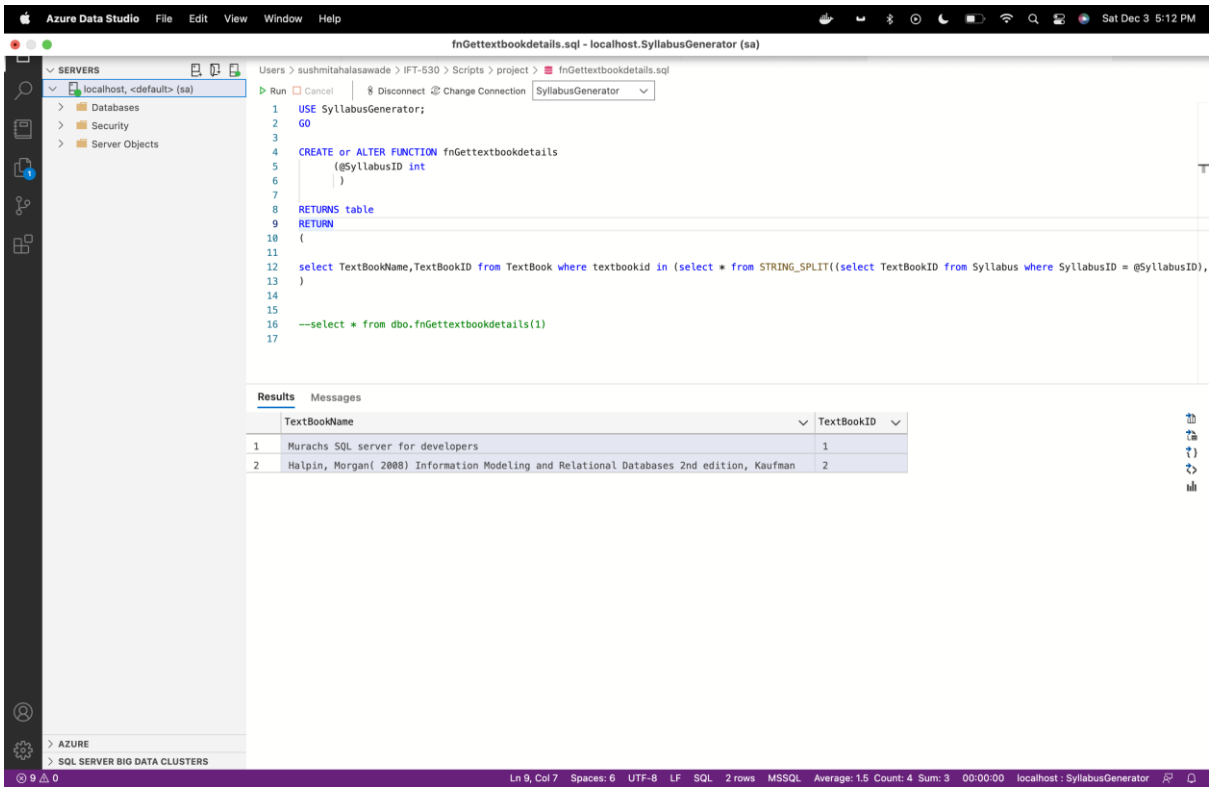
1. **Get textbook requirements of course:**

Query:

```
USE SyllabusGenerator;

GO


CREATE or ALTER FUNCTION fnGettextbookdetails

    (@SyllabusID int

    )
```

```
RETURNS table

RETURN

(

select TextBookName,TextBookID from TextBook where textbookid in (select * from STRING_SPLIT((select TextBookID from Syllabus where SyllabusID = @SyllabusID), ','))

)


--select * from dbo.fnGettextbookdetails(1)
```

Output:

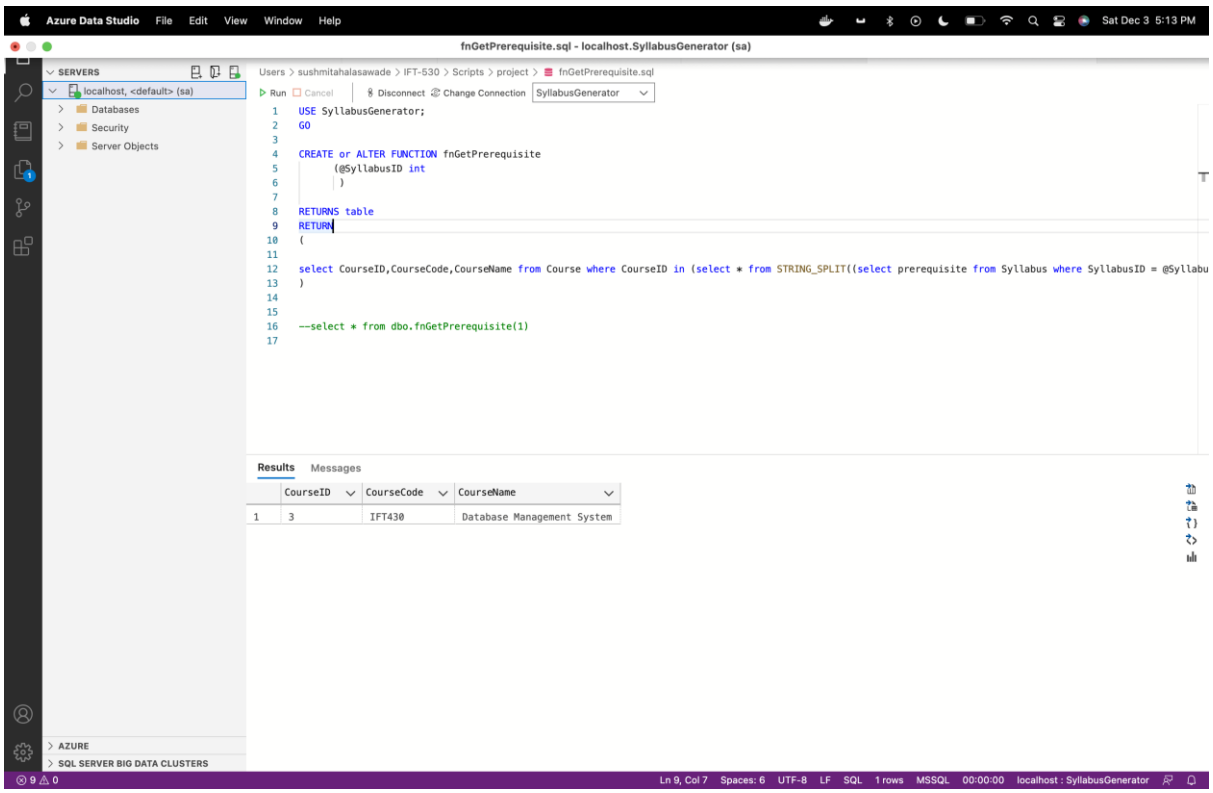


1. **Get prerequisite course list of course:**
   Query:

```
USE SyllabusGenerator;

GO


CREATE or ALTER FUNCTION fnGetPrerequisite

    (@SyllabusID int
```

```
)

RETURNS table

RETURN

(

select CourseID,CourseCode,CourseName from Course where CourseID in (select * from
STRING_SPLIT((select prerequisite from Syllabus where SyllabusID = @SyllabusID), ','))

)

--select * from dbo.fnGetPrerequisite(1)
```

Output:



Trigger :

When professor deletes already existing Syllabus there should be some place where we can store the deleted syllabus. If professor wants to review the syllabus, he/she deleted they can always refer back to this SyllabusAudit table.

SQL Trigger Code:

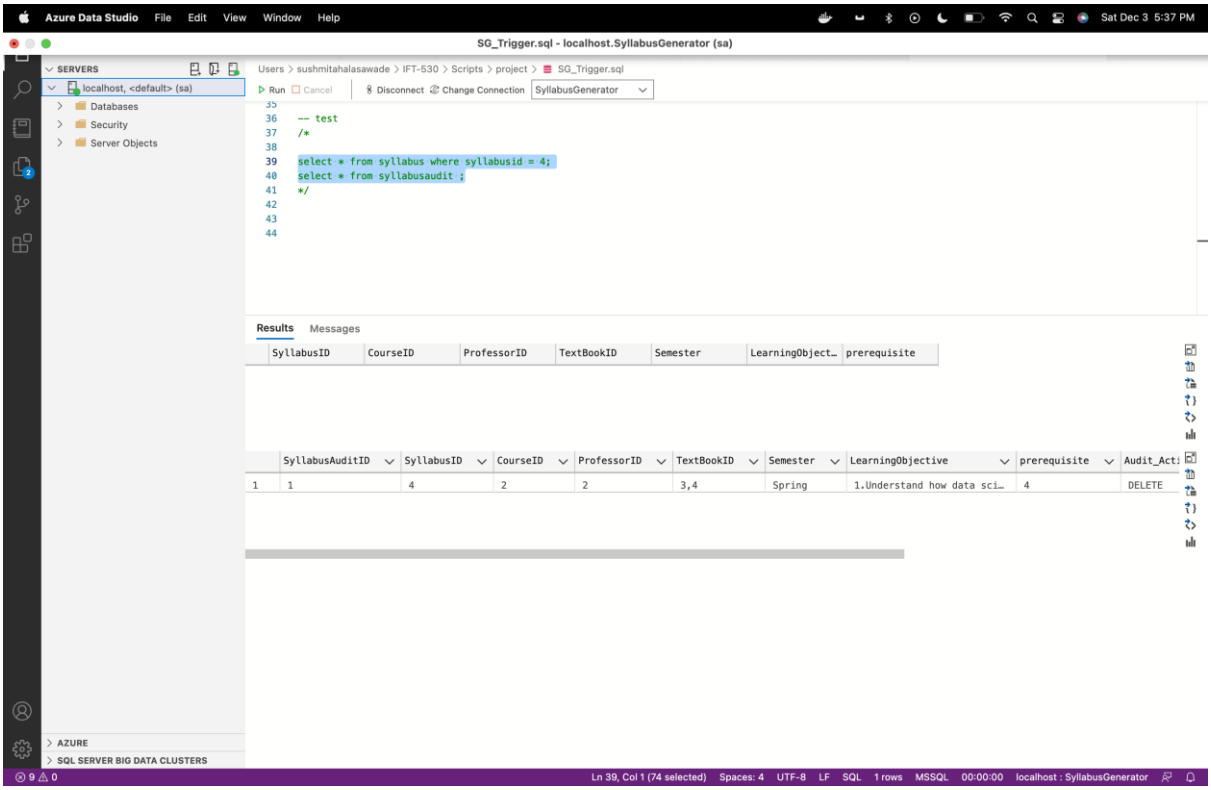
1. **Delete trigger to audit deleted syllabus.**

```
USE syllabusGenerator
GO

CREATE or alter TRIGGER trInsteadofDeleteSyllabus
on Syllabus
After DELETE
AS

declare @SyllabusID INT
declare @CourseID int
declare @ProfessorID int
declare @textbookid nvarchar(100)
declare @Semester nvarchar(10)
declare @LearningObjective nvarchar(max)
declare @prerequisite nvarchar(50)

select @SyllabusID= d.SyllabusID from deleted as d;
select @CourseID= d.CourseID from deleted as d;
select @ProfessorID= d.ProfessorID from deleted as d;
select @textbookid= d.TextBookID from deleted as d;
select @Semester= d.Semester from deleted as d;
select @LearningObjective= d.LearningObjective from deleted as d;
select @prerequisite= d.prerequisite from deleted as d;

Insert into SyllabusAudit (SyllabusID, CourseID, ProfessorID,textbookid,Semester,LearningObjective,
prerequisite,Audit_action, Action_by, Action_At, Remark)
values(@SyllabusID,@CourseID,@ProfessorID,@textbookid,@Semester,@LearningObjective,@prerequisite,'D
ELETE',@ProfessorID,GETDATE(),'Deleted Syllabus')
print('Record deleted -Instead of Delete trigger')
```

Test:

```
select * from syllabus where syllabusid = 4;

select * from syllabusaudit ;
```

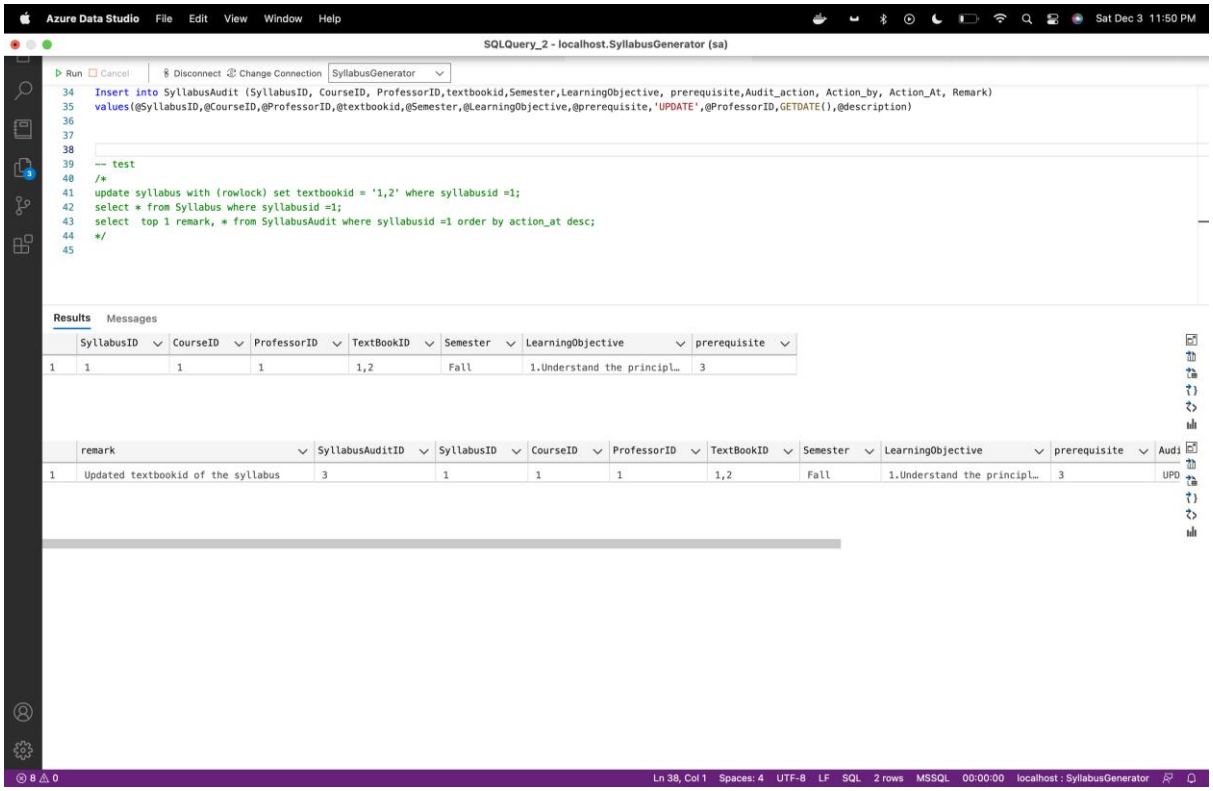
When we executed delete syllabys stored procedure for Syllabus with id = 4, the audit record is added in the table with after delete trigger.



Update Trigger

```
CREATE TRIGGER trAfterUpdateSyllabus
on dbo.Syllabus
For UPDATE
AS

declare @textbookid varchar(100)
declare @LearningObjective varchar(max)
declare @prerequisite varchar(100)
declare @description varchar(50)
declare @SyllabusID int
```

```
declare @CourseID int

declare @ProfessorID int

declare @Semester int


select @textbookid= i.textbookid from inserted as i;

select @LearningObjective= i.LearningObjective from inserted as i;

select @prerequisite= i.prerequisite from inserted as i;

select @SyllabusID= i.SyllabusID from inserted as i;

select @CourseID= i.CourseID from inserted as i;

select @ProfessorID= i.ProfessorID from inserted as i;

select @Semester= i.Semester from inserted as i;


IF UPDATE(textbookid)

    SET @description = 'Updated textbookid of the syllabus'


IF UPDATE(LearningObjective)

    SET @description = 'Updated LearningObjective of the syllabus'


IF UPDATE(prerequisite)

    SET @description = 'Updated prerequisite of the syllabus'


Insert into SyllabusAudit (SyllabusID, CourseID, ProfessorID,textbookid,Semester,LearningObjective,
prerequisite,Audit_action, Action_by, Action_At, Remark)

values(@SyllabusID,@CourseID,@ProfessorID,@textbookid,@Semester,@LearningObjective,@prerequisite,'U
PDATE',@ProfessorID,GETDATE(),@description)



-- test

/*

update syllabus with (rowlock) set textbookid = '1,2' where syllabusid =1;

select * from Syllabus where syllabusid =1;

select  top 1 remark, * from SyllabusAudit where syllabusid =1 order by action_at desc;

*/
```


Azure Data Studio File Edit View Window Help
Sat Dec 3 11:50 PM
SQLQuery_2 - localhost.SyllabusGenerator (sa)
Run Cancel
Disconnect Change Connection
SyllabusGenerator
Insert into SyllabusAudit (SyllabusID, CourseID, ProfessorID,textbookid,Semester,LearningObjective, prerequisite,Audit_action, Action_by, Action_At, Remark)
values(@SyllabusID,@CourseID,@ProfessorID,@textbookid,@Semester,@LearningObjective,@prerequisite,'UPDATE',@ProfessorID,GETDATE(),@description)
-- test
/*
update syllabus with (rowlock) set textbookid = '1,2' where syllabusid =1;
select * from Syllabus where syllabusid =1;
select top 1 remark, * from SyllabusAudit where syllabusid =1 order by action_at desc;
*/
Results
Messages
SyllabusID CourseID ProfessorID TextBookID Semester LearningObjective prerequisite
1 1 1 1 1,2 Fall 1.Understand the principl… 3
remark SyllabusAuditID SyllabusID CourseID ProfessorID TextBookID Semester LearningObjective prerequisite Audi
1 Updated textbookid of the syllabus 3 1 1 1 1,2 Fall 1.Understand the principl… 3 UPD
Ln 38, Col 1 Spaces: 4 UTF-8 LF SQL 2 rows MSSQL 00:00:00 localhost : SyllabusGenerator

Section 6

Couchbase – NoSQL

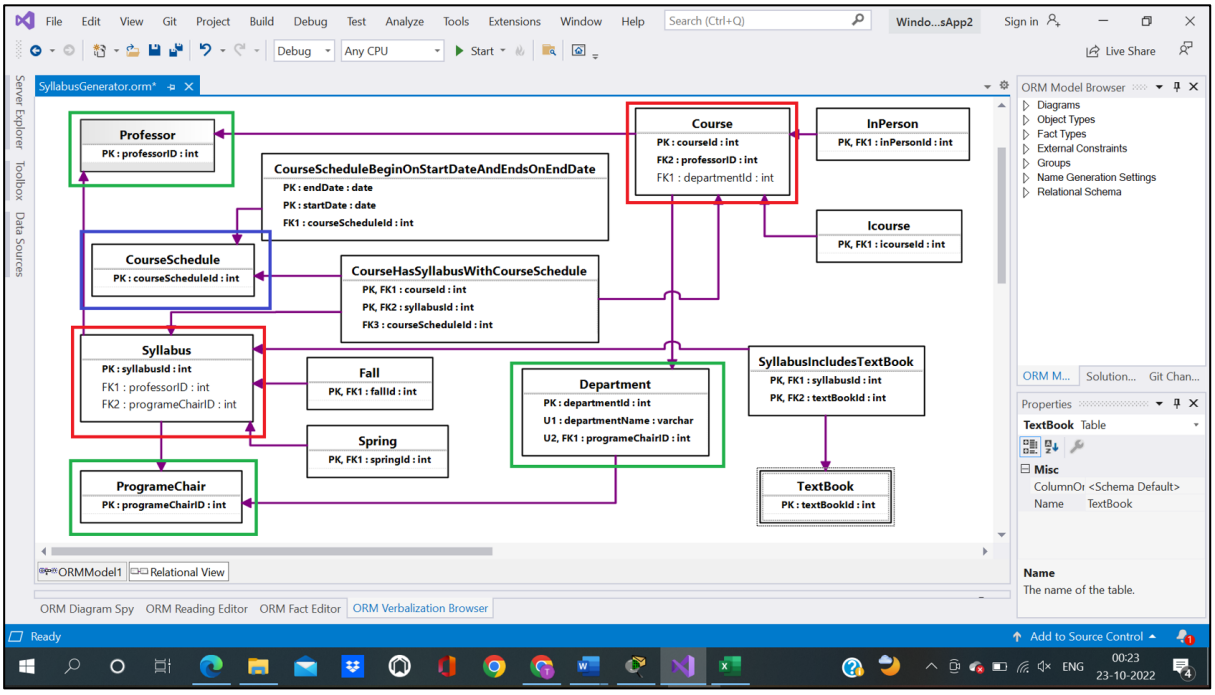
Section 6.1 ORM

Section 6.1 (ORM diagram from a NoSQL perspective)

List which entities of your original ORM diagram you are extending by nesting.

We are extending the entities

- Course and Syllabus – Syllabus as nested document inside course document (represented with red boxes in screenshot below)
- Department, ProgramChair and Professor – Program chair as document inside department document((represented with green boxes in screenshot below)
- Course Schedule(represented in blue)



Section 6.2 Buckets Creation

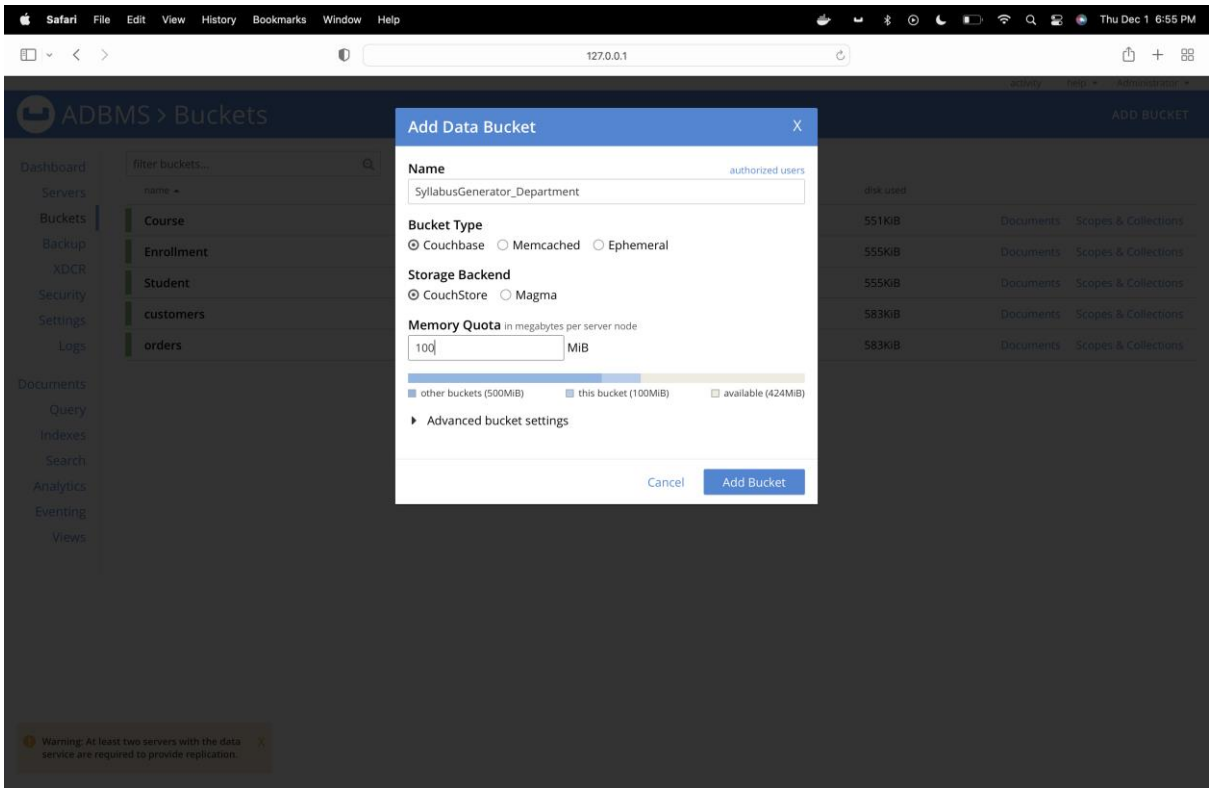
We have created the buckets

1) Department

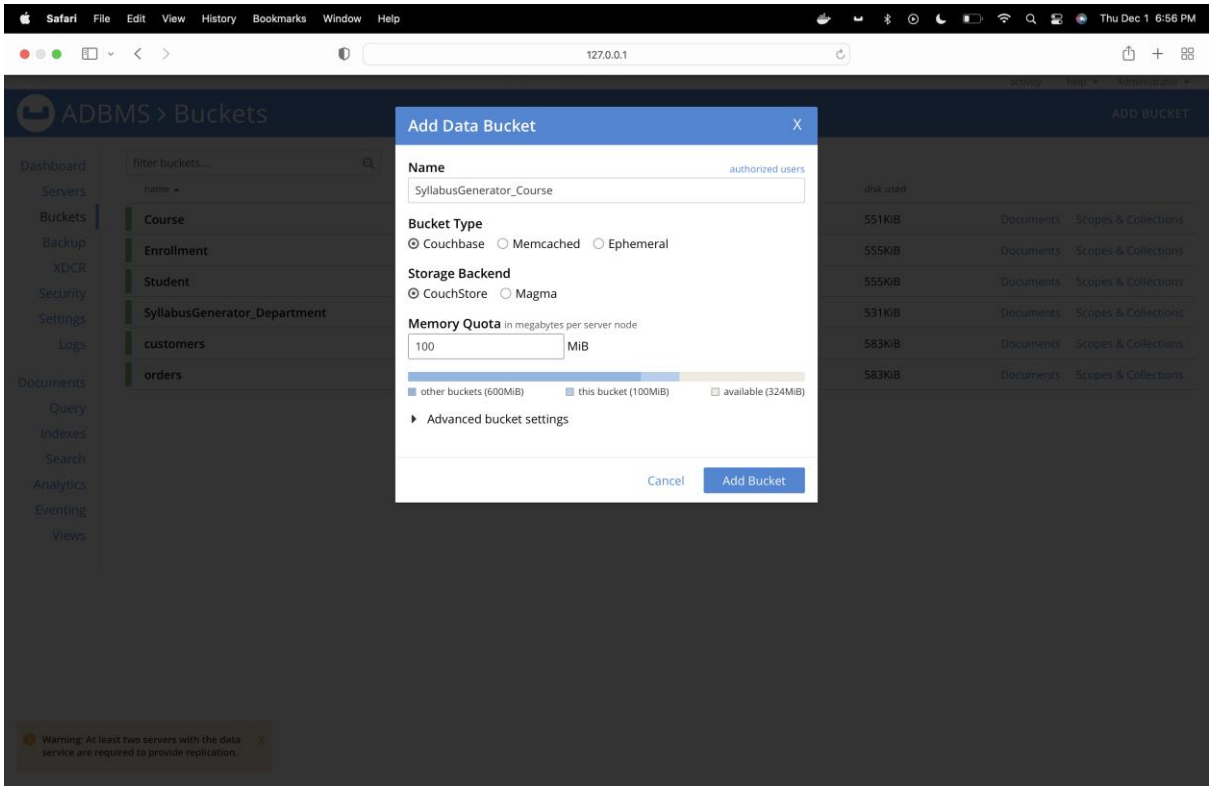
2) Course

3) Course Schedule

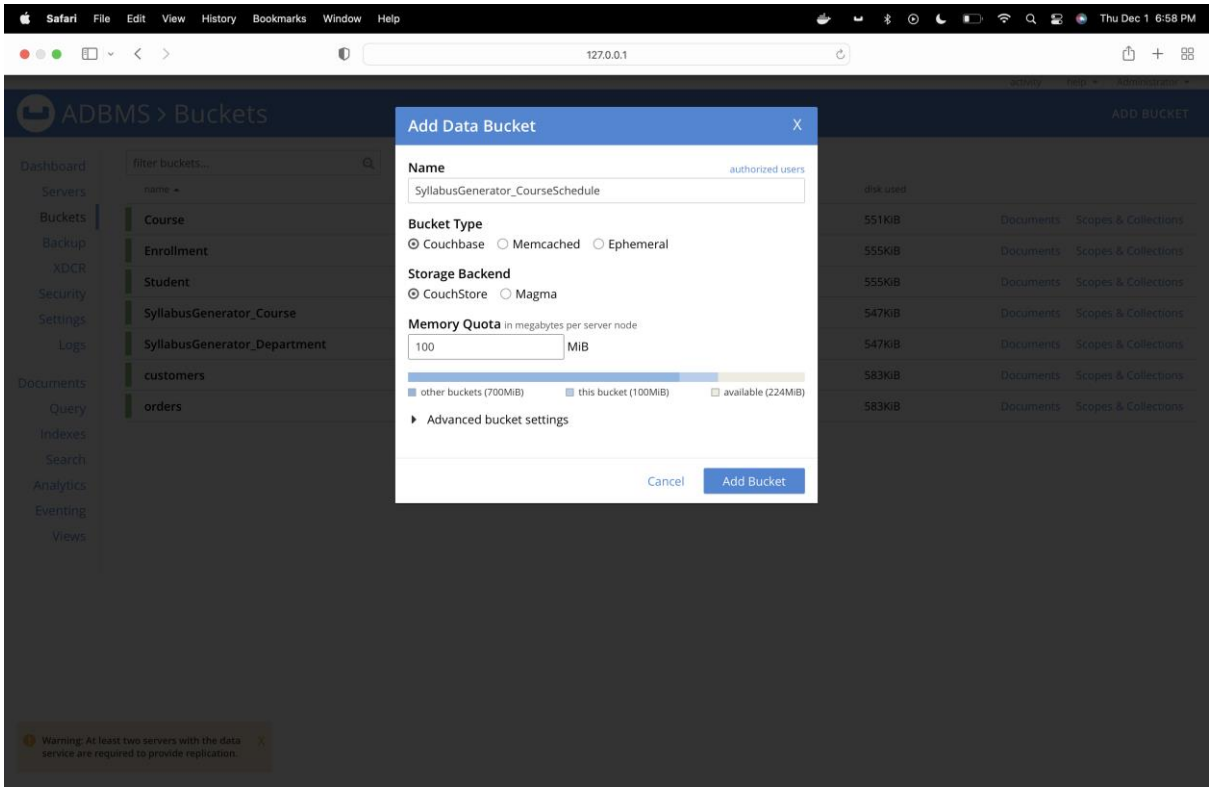
Department Bucket : SyllabusGenerator_Department



Course Bucket : SyllabusGenerator_Course



Course Schedule Bucket: SyllabusGenerator_CourseSchedule



Adding Data

Department Bucket

Json:

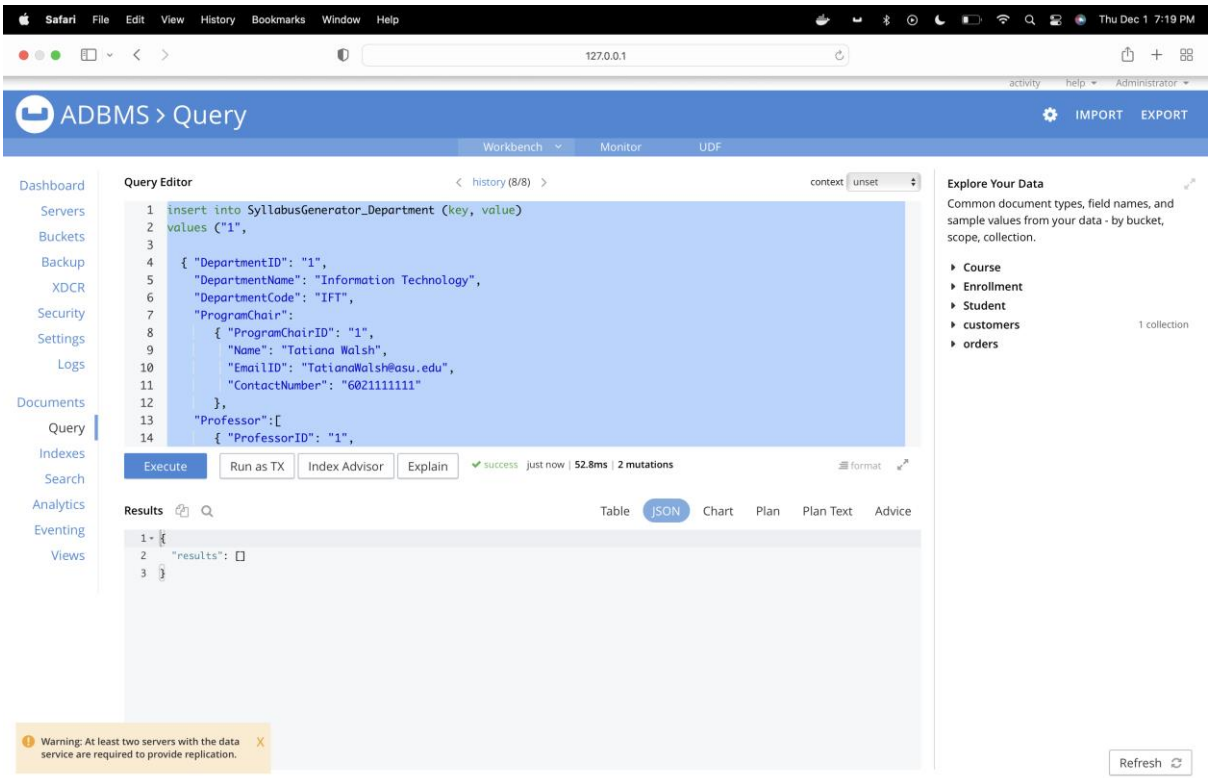
insert into SyllabusGenerator_Department (key, value)

values ("1",


```
 { "DepartmentID": "1",
   "DepartmentName": "Information Technology",
   "DepartmentCode": "IFT",
   "ProgramChair":
     { "ProgramChairID": "1",
       "Name": "Tatiana Walsh",
       "EmailID": "TatianaWalsh@asu.edu",
       "ContactNumber": "6021111111"
     },
   "Professor":[
     { "ProfessorID": "1",
       "Name": "Robert Rucker",
       "EmailID": "RobertRucker@asu.edu",
       "ContactNumber": "6022221111"
     },
     { "ProfessorID": "2",
       "Name": "Asmaa Elbadrawy",
       "EmailID": "AsmaaElbadrawy@asu.edu",
       "ContactNumber": "6023331111"
     },
     { "ProfessorID": "3",
       "Name": "Dinesh Sthapit",
```

```
        "EmailID": "DineshSthapit@asu.edu",
        "ContactNumber": "6024441111"
      }
    ]
  }

), ("2",

  { "DepartmentID": "2",
    "DepartmentName": "Computer Science",
    "DepartmentCode": "CS",
    "ProgramChair":
      { "ProgramChairID": "4",
        "Name": "Dont Know",
        "EmailID": "DontKnow@asu.edu",
        "ContactNumber": "6021112222"
      },
    "Professor":[
      { "ProfessorID": "5",
        "Name": "Chris Bryan",
        "EmailID": "ChrisBryan@asu.edu",
        "ContactNumber": "6022222222"
      },
      { "ProfessorID": "6",
        "Name": "Hasan Davulcu",
        "EmailID": "HasanDavulcu@asu.edu",
        "ContactNumber": "6023332222"
      },
      { "ProfessorID": "7",
        "Name": "Samira Ghayekhloo",
        "EmailID": "SamiraGhayekhloo@asu.edu",
```

```
        "ContactNumber": "6024442222"
      }
   ]
 }

);
```



Course Bucket :

Adding course documents:

Json:

```
insert into SyllabusGenerator_Course (key, value)
values ("1",

 { "CourseID": "1",
   "DepartmentID": "1",
   "CourseName": "Advanced Database Management Systems",
```

```
   "CourseCode": "IFT530",
   "CourseType": "In-Person",
   "ProfessorID": "2",
   "Syllabus":[
     { "SyllabusID": "1",
       "Semester": "Fall",
       "TextbookID": "1,2",
       "LearningObjective": "1.Understand the principles of conceptual modeling,2.understand the use of scripts when implementing SQL code",
       "Prerequisite": "3"
     },
     { "SyllabusID": "2",
       "Semester": "Spring",
       "TextbookID": "1,2",
       "LearningObjective": "1.Understand the principles of conceptual modeling,2.understand the use of scripts when implementing SQL code",
       "Prerequisite": "3"
     }
   ]
 }

), ("2",
{
   "CourseID": "2",
   "DepartmentID": "1",
   "CourseName": "Analyzing Big Data",
   "CourseCode": "IFT511",
   "CourseType": "In-Person",
   "ProfessorID": "3",
   "Syllabus":[
     { "SyllabusID": "1",
       "Semester": "Fall",
```

```
        "TextbookID": "3,4",
        "LearningObjective": "1.Understand how  data science can be used as tools to analyze large amounts of data for the purpose of extracting business value.",
        "Prerequisite": "4"
      },
      { "SyllabusID": "2",
        "Semester": "Spring",
        "TextbookID": "3,4",
        "LearningObjective": "1.Understand how  data science can be used as tools to analyze large amounts of data for the purpose of extracting business value.",
        "Prerequisite": "4"
      }
    ]
  }

),

("3",
{
    "CourseID": "3",
    "DepartmentID": "1",
    "CourseName": "Database Management System",
    "CourseCode": "IFT430",
    "CourseType": "In-Person",
    "ProfessorID": "2",
    "Syllabus":""
  }
    ),
("4",
{
    "CourseID": "4",
    "DepartmentID": "1",
```

"CourseName": "Fundamentals of Python",

"CourseCode": "IFT411",

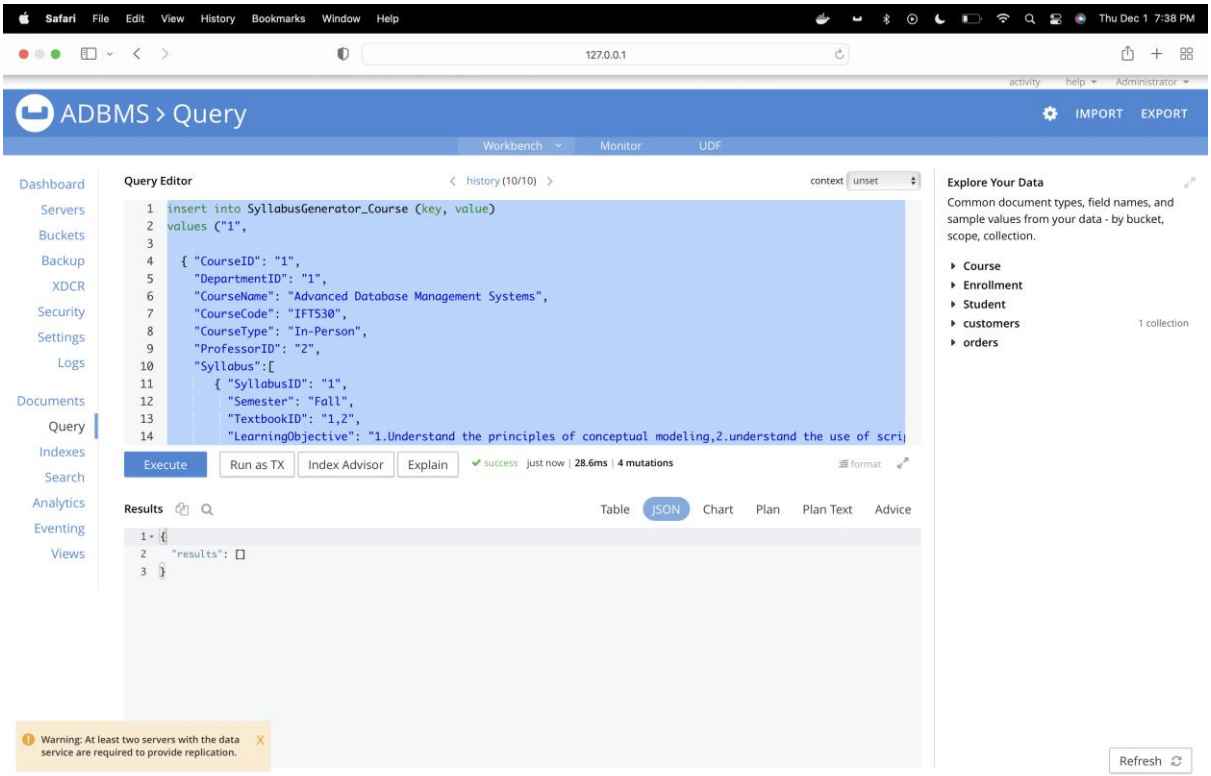
"CourseType": "In-Person",

"ProfessorID": "3",

"Syllabus":""

}

);



Adding Course Schedule Documents

JSON:

insert into SyllabusGenerator_CourseSchedule (key, value)

values ("1",

{ "CourseScheduleID": "1",

"SyllabusID": "1",

"CourseID": "1",

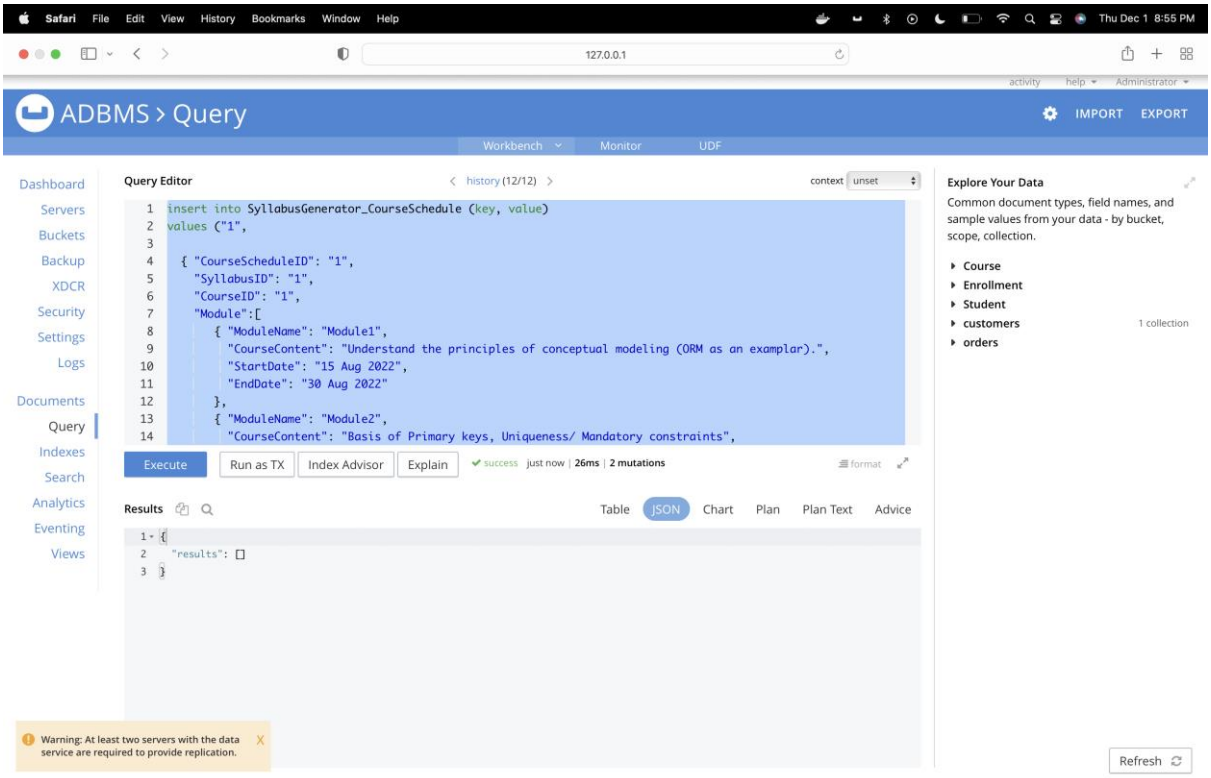
"Module":[

```
    { "ModuleName": "Module1",
      "CourseContent": "Understand the principles of conceptual modeling (ORM as an examplar).",
      "StartDate": "15 Aug 2022",
      "EndDate": "30 Aug 2022"
    },
    { "ModuleName": "Module2",
      "CourseContent": "Basis of Primary keys, Uniqueness/ Mandatory constraints",
      "StartDate": "1 Sept 2022",
      "EndDate": "15 Sept 2022"
    },
    { "ModuleName": "Module3",
      "CourseContent": "basis for determining Foreign Keys, value and frequency constraints,
dynamic SQL",
      "StartDate": "16 Sept 2022",
      "EndDate": "16 Oct 2022"
    },
    { "ModuleName": "Module4",
      "CourseContent": "Window functions, SQL++",
      "StartDate": "17 Oct 2022",
      "EndDate": "10 Nov 2022"
    },
    { "ModuleName": "Module5",
      "CourseContent": "Couchbase, Capella-DBaas",
      "StartDate": "11 Nov 2022",
      "EndDate": "30 Nov 2022"
    }
  ]
 }

), ("2",
{
```

```
  "CourseScheduleID": "2",
  "SyllabusID": "2",
  "CourseID": "1",
  "Module":[
    { "ModuleName": "Module1",
      "CourseContent": "Understand the principles of conceptual modeling (ORM as an examplar).",
      "StartDate": "15 Jan 2022",
      "EndDate": "30 Jan 2022"
    },
    { "ModuleName": "Module2",
      "CourseContent": "Basis of Primary keys, Uniqueness/ Mandatory constraints",
      "StartDate": "1 Feb 2022",
      "EndDate": "15 Feb 2022"
    },
    { "ModuleName": "Module3",
      "CourseContent": "basis for determining Foreign Keys, value and frequency constraints,
dynamic SQL",
      "StartDate": "16 Feb 2022",
      "EndDate": "16 Mar 2022"
    },
    { "ModuleName": "Module4",
      "CourseContent": "Window functions, SQL++",
      "StartDate": "17 Mar 2022",
      "EndDate": "10 Apr 2022"
    },
    { "ModuleName": "Module5",
      "CourseContent": "Couchbase, Capella-DBaas",
      "StartDate": "11 Apr 2022",
      "EndDate": "30 Apr 2022"
    }
  ]
```

}

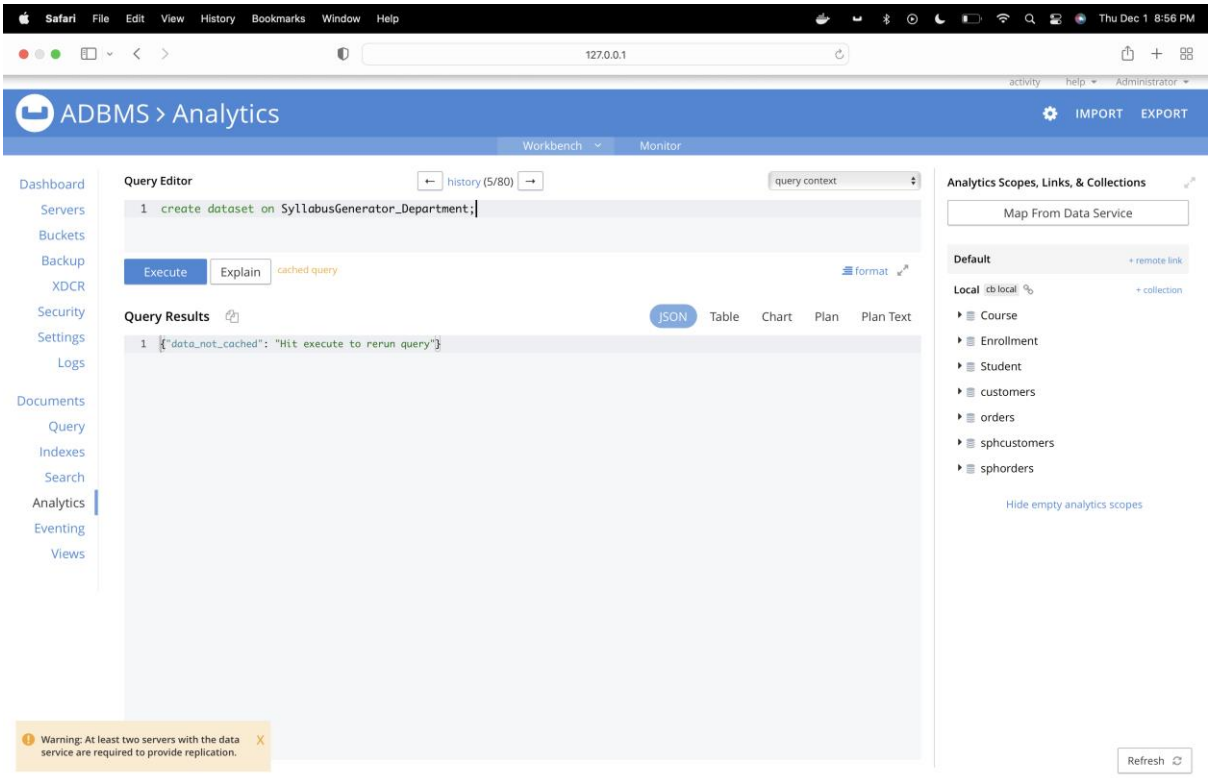
);



Creating dataset on SyllabusGenerator_Department

Query:

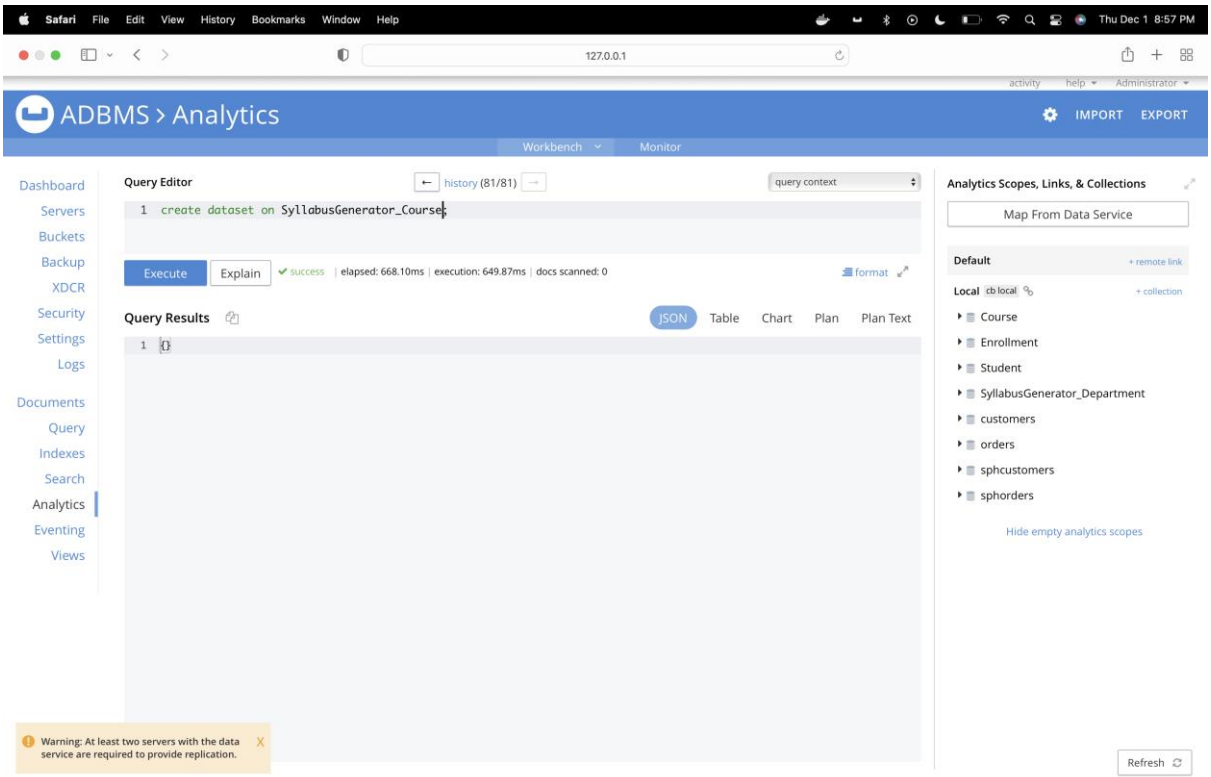
Create dataset on SyllabusGenerator_Department



Creating dataset on SyllabusGenerator_Course

<u>Query:</u>

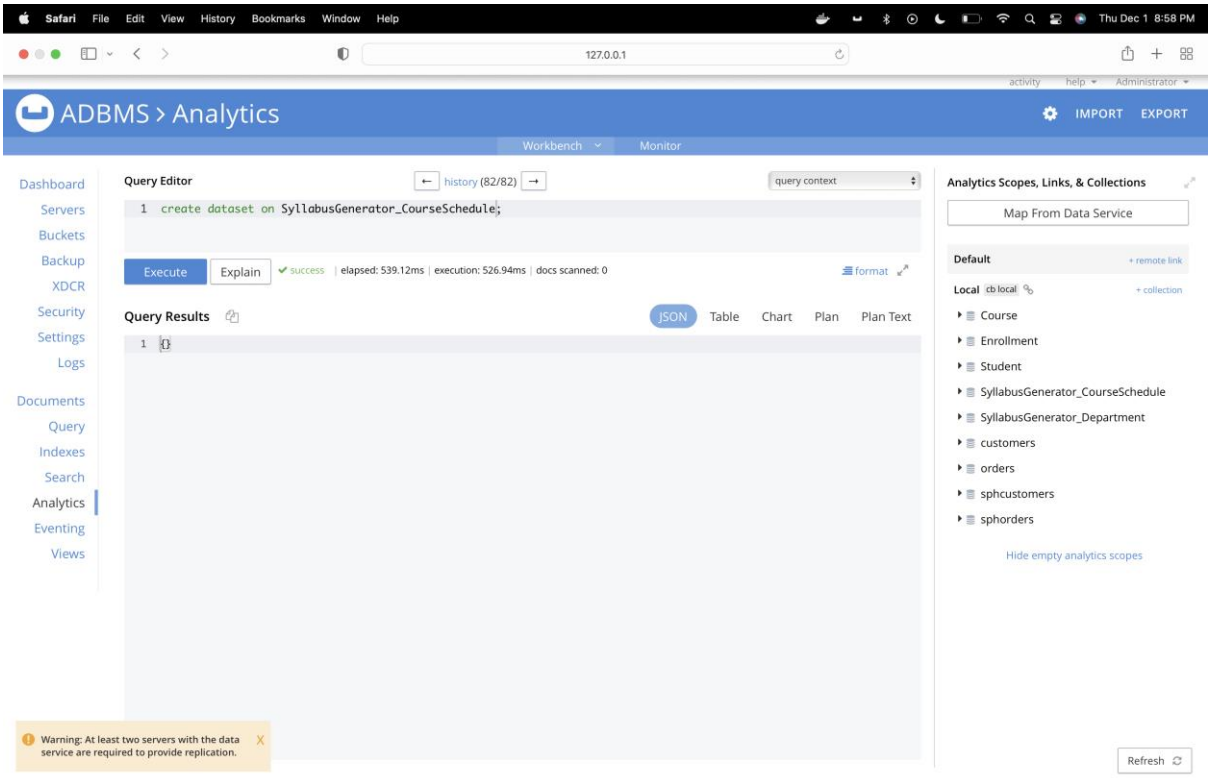
create dataset on SyllabusGenerator_Course



Creating dataset on:

Query

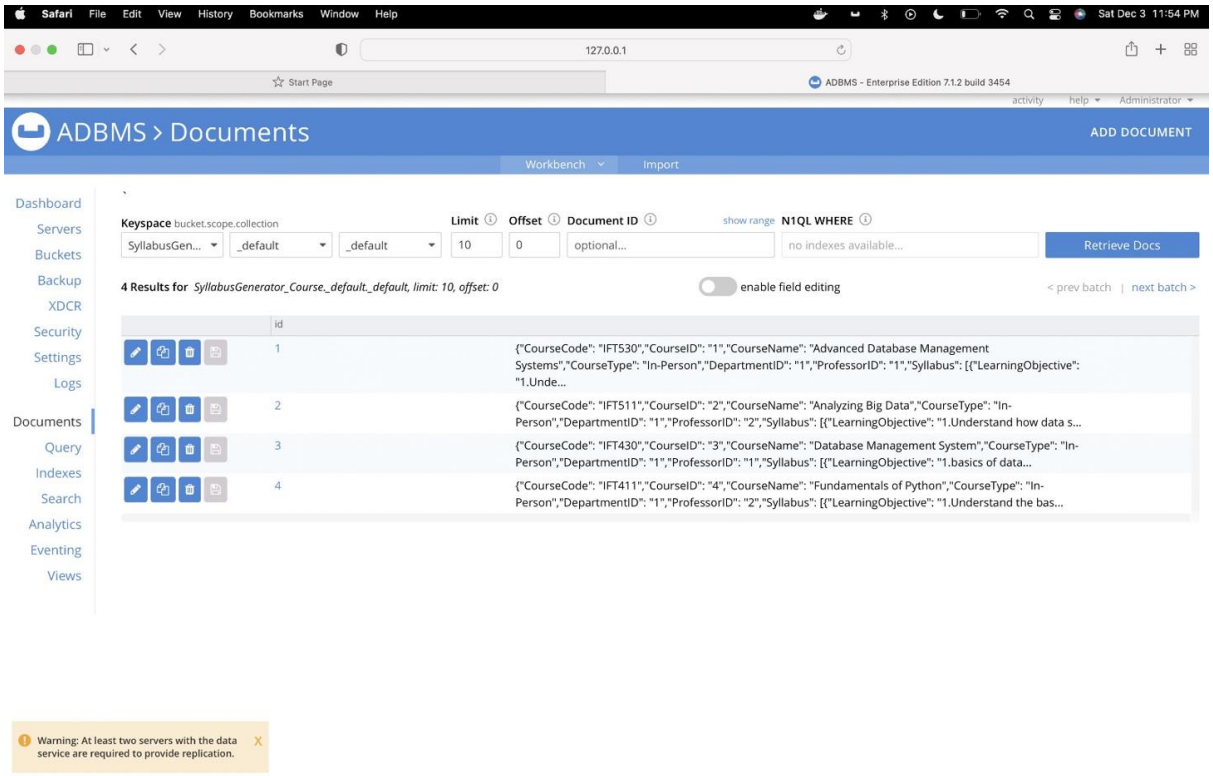
create dataset on SyllabusGenerator_CourseSchedule;



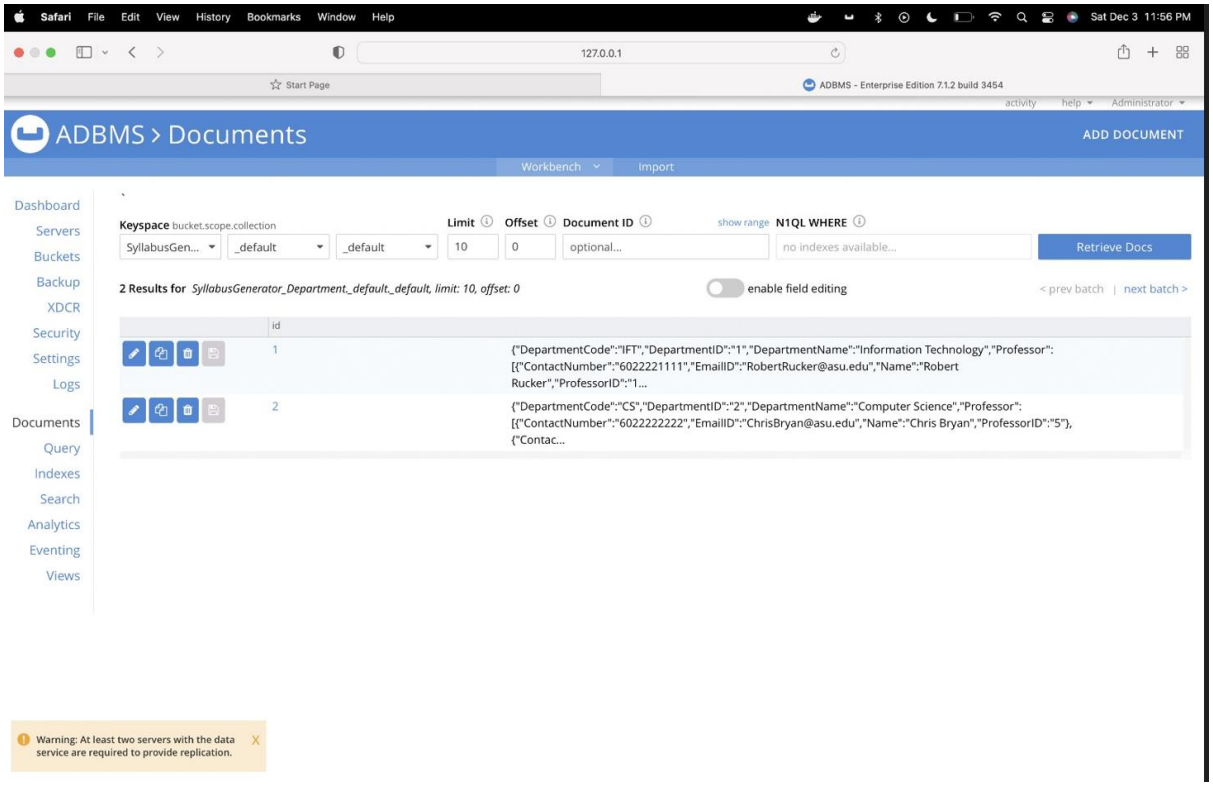
Section 6.3

Screenshots of JSON Documents from CouchBase

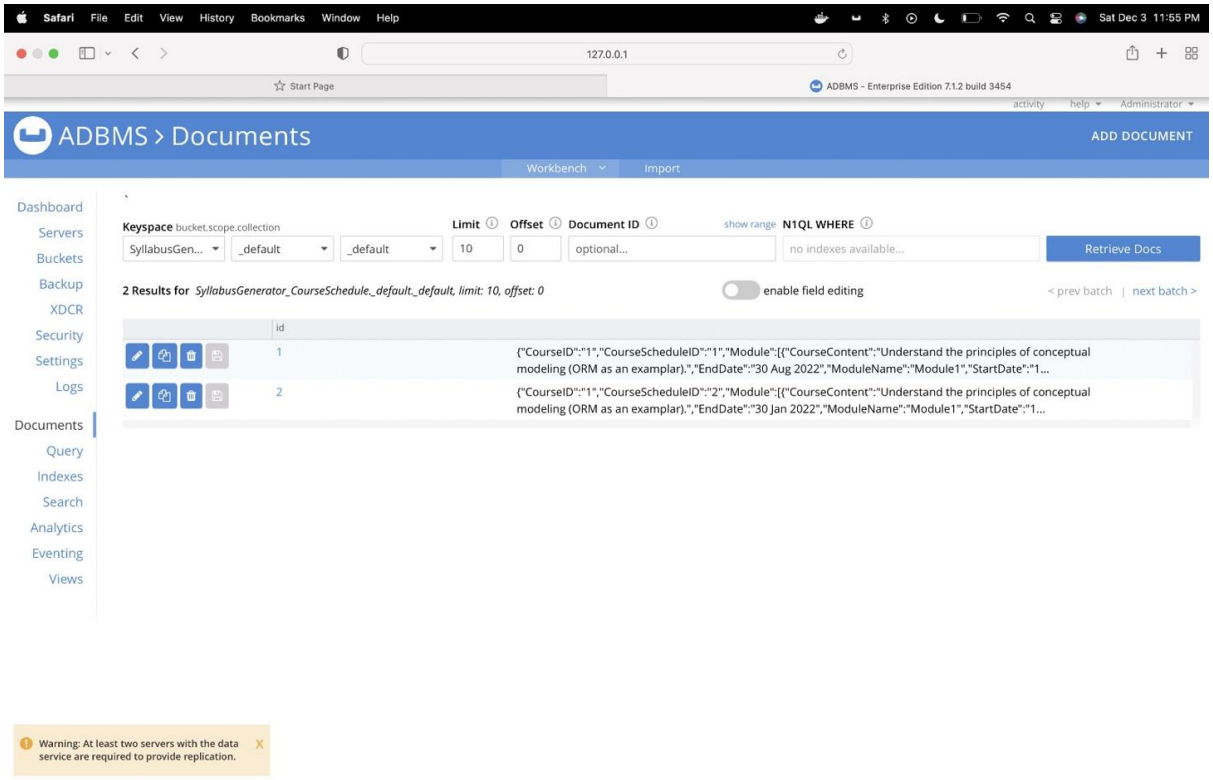
Course Bucket



Department Bucket



Course Schedule Bucket



Section 6.4

Query to get all fall courses detailed Syllabus

Query 1:

FROM SyllabusGenerator_Course AS o, o.Syllabus AS i, SyllabusGenerator_CourseSchedule AS cs

where i.Semester = 'Fall' and cs.SyllabusID = i.SyllabusID

SELECT o.CourseName,o.CourseCode,i.Semester,i.LearningObjective,cs.Module

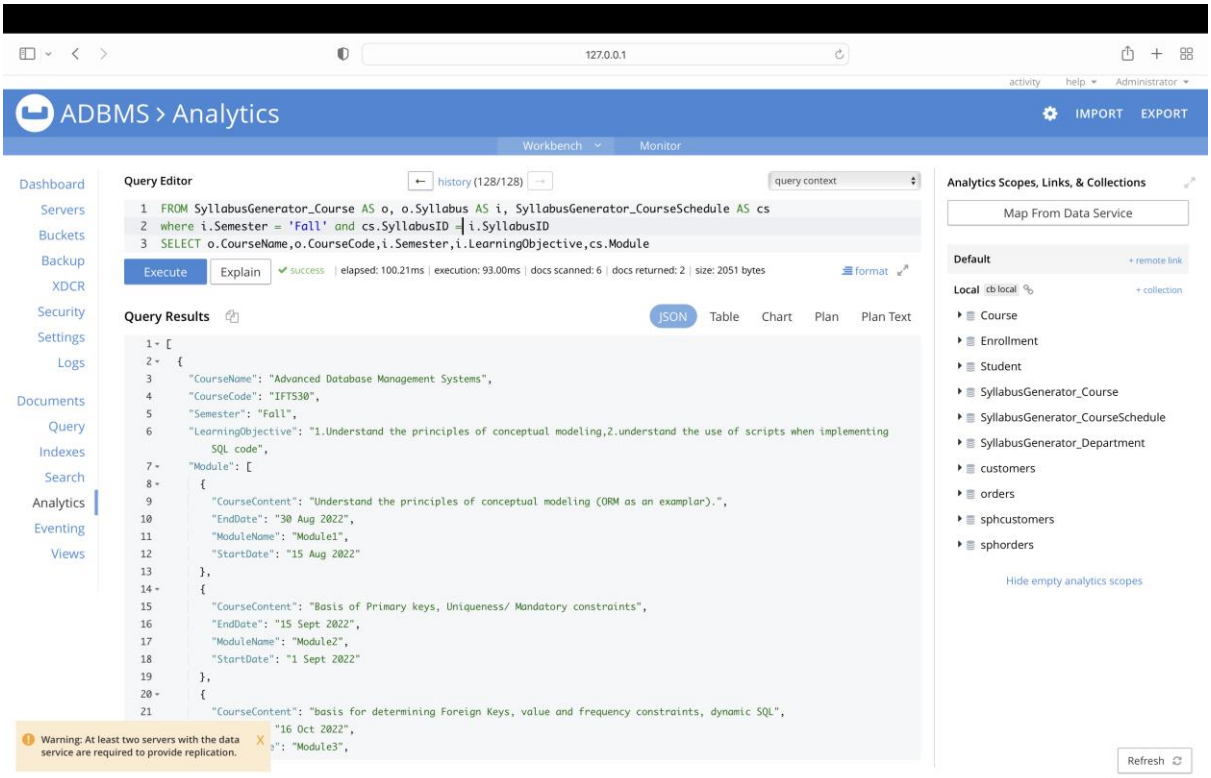
Output:

[

```
{
  "CourseName": "Advanced Database Management Systems",
  "CourseCode": "IFT530",
  "Semester": "Fall",
  "LearningObjective": "1.Understand the principles of conceptual modeling,2.understand the use of scripts when implementing SQL code",
  "Module": [
    {
      "CourseContent": "Understand the principles of conceptual modeling (ORM as an examplar).",
      "EndDate": "30 Aug 2022",
      "ModuleName": "Module1",
      "StartDate": "15 Aug 2022"
    },
    {
      "CourseContent": "Basis of Primary keys, Uniqueness/ Mandatory constraints",
      "EndDate": "15 Sept 2022",
      "ModuleName": "Module2",
      "StartDate": "1 Sept 2022"
    },
    {
      "CourseContent": "basis for determining Foreign Keys, value and frequency constraints, dynamic SQL",
      "EndDate": "16 Oct 2022",
      "ModuleName": "Module3",
      "StartDate": "16 Sept 2022"
    },
    {
      "CourseContent": "Window functions, SQL++",
      "EndDate": "10 Nov 2022",
      "ModuleName": "Module4",
      "StartDate": "17 Oct 2022"
    },
```

```
    {
      "CourseContent": "Couchbase, Capella-DBaas",
      "EndDate": "30 Nov 2022",
      "ModuleName": "Module5",
      "StartDate": "11 Nov 2022"
    }
  ]
 },
 {
  "CourseName": "Analyzing Big Data",
  "CourseCode": "IFT511",
  "Semester": "Fall",
  "LearningObjective": "1.Understand how  data science can be used as tools to analyze large amounts of data for the purpose of extracting business value.",
  "Module": [
    {
      "CourseContent": "Understand the principles of conceptual modeling (ORM as an examplar).",
      "EndDate": "30 Aug 2022",
      "ModuleName": "Module1",
      "StartDate": "15 Aug 2022"
    },
    {
      "CourseContent": "Basis of Primary keys, Uniqueness/ Mandatory constraints",
      "EndDate": "15 Sept 2022",
      "ModuleName": "Module2",
      "StartDate": "1 Sept 2022"
    },
    {
      "CourseContent": "basis for determining Foreign Keys, value and frequency constraints, dynamic SQL",
      "EndDate": "16 Oct 2022",
      "ModuleName": "Module3",
```

```
      "StartDate": "16 Sept 2022"
    },
    {
      "CourseContent": "Window functions, SQL++",
      "EndDate": "10 Nov 2022",
      "ModuleName": "Module4",
      "StartDate": "17 Oct 2022"
    },
    {
      "CourseContent": "Couchbase, Capella-DBaas",
      "EndDate": "30 Nov 2022",
      "ModuleName": "Module5",
      "StartDate": "11 Nov 2022"
    }
  ]
 }
]
```



Query 2

Get list of courses with their professor: (using nested documents for join)

Query:

FROM SyllabusGenerator_Department as d, d.Professor as p join SyllabusGenerator_Course as c on p.ProfessorID = c.ProfessorID

select p.Name, c.CourseName, c.CourseCode, d.DepartmentName

output:

[

{

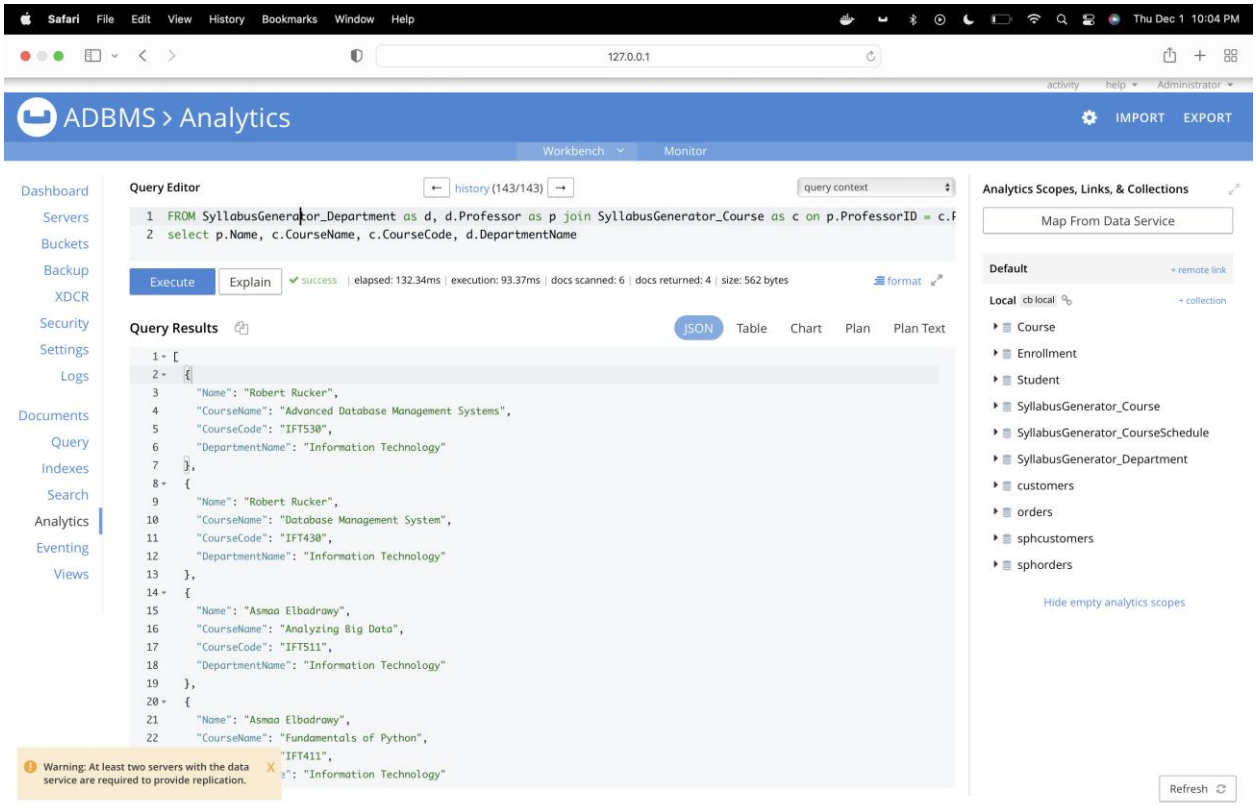
"Name": "Robert Rucker",

"CourseName": "Advanced Database Management Systems",

"CourseCode": "IFT530",

"DepartmentName": "Information Technology"

```
  },
  {
    "Name": "Robert Rucker",
    "CourseName": "Database Management System",
    "CourseCode": "IFT430",
    "DepartmentName": "Information Technology"
  },
  {
    "Name": "Asmaa Elbadrawy",
    "CourseName": "Analyzing Big Data",
    "CourseCode": "IFT511",
    "DepartmentName": "Information Technology"
  },
  {
    "Name": "Asmaa Elbadrawy",
    "CourseName": "Fundamentals of Python",
    "CourseCode": "IFT411",
    "DepartmentName": "Information Technology"
  }
]
```


Safari File Edit View History Bookmarks Window Help
Thu Dec 1 10:04 PM
127.0.0.1
activity help Administrator
ADBMS > Analytics
IMPORT EXPORT
Workbench Monitor
Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views
Query Editor
history (143/143)
query context
1 FROM SyllabusGenerator_Department as d, d.Professor as p join SyllabusGenerator_Course as c on p.ProfessorID = c.F
2 select p.Name, c.CourseName, c.CourseCode, d.DepartmentName
Execute
Explain
success | elapsed: 132.34ms | execution: 93.37ms | docs scanned: 6 | docs returned: 4 | size: 562 bytes
format
Query Results
JSON
Table
Chart
Plan
Plan Text
1 [
2 {
3 "Name": "Robert Rucker",
4 "CourseName": "Advanced Database Management Systems",
5 "CourseCode": "IFT530",
6 "DepartmentName": "Information Technology"
7 },
8 {
9 "Name": "Robert Rucker",
10 "CourseName": "Database Management System",
11 "CourseCode": "IFT430",
12 "DepartmentName": "Information Technology"
13 },
14 {
15 "Name": "Asmaa Elbadrawy",
16 "CourseName": "Analyzing Big Data",
17 "CourseCode": "IFT511",
18 "DepartmentName": "Information Technology"
19 },
20 {
21 "Name": "Asmaa Elbadrawy",
22 "CourseName": "Fundamentals of Python",
"IFT411",
": "Information Technology"
Warning: At least two servers with the data service are required to provide replication.
Analytics Scopes, Links, & Collections
Map From Data Service
Default
+ remote link
Local cb local
+ collection
Course
Enrollment
Student
SyllabusGenerator_Course
SyllabusGenerator_CourseSchedule
SyllabusGenerator_Department
customers
orders
sphcustomers
sphorders
Hide empty analytics scopes
Refresh

Section – 7

Summary

- Collected dataset to clearly define the entities in the University.
- We created the SQL tables, stored procedures, and triggers for the Syllabus generator. Professors can easily retrieve all the information regarding Syllabus pertaining to icourse or the in-person course, Fall semester course, Summer Semester course etc.,
- Identified relationship between the entities and included primary and foreign keys.
- Included the Object Role Modeling Diagram. Added the fall semester and spring semester as subtypes under the entity syllabus.
- Added the subtypes icourse and in-person course as subtypes under course.
- Included constrains and verified the verbalization for each entity. Generated relational view for the diagram.
- Added the delete trigger that will log the deleted action in the audit table so whenever professor deletes the syllabus, the deleted syllabus log is updated in the trigger. This way we can keep track of the deleted data.
- We have added triggers – update trigger and delete trigger
- Added Stored Procedure
- Added user defined functions
- One the Program Chair approves the syllabus designed by the professor the syllabus will be finalized and published.

Conclusion

In conclusion, a database is an efficient mechanism to store and organize the data. It allows for a centralized facility that can be modified and quickly shared among the multiple users. Syllabus Generator will help in quickly generating the syllabus for the professors and it can be implemented university wide to reduce the effort required to modify the syllabus.

It is one stop solution to the tedious task of maintaining syllabus. It is the convenient for both the professor and the program chair to look at this repository for any concerns regarding the program syllabus.

References –

[1]Object-Role Modeling Fundamentals A Practical Guide to Data Modeling with ORM (Terry Halpin)

[2]Murach's SQL Server 2016 for Developers (Joel Murach) (z-lib.org)

[3]https://victormorgante.medium.com/why-learn-object-role-modeling-part-ii-8f29726c253c