

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.

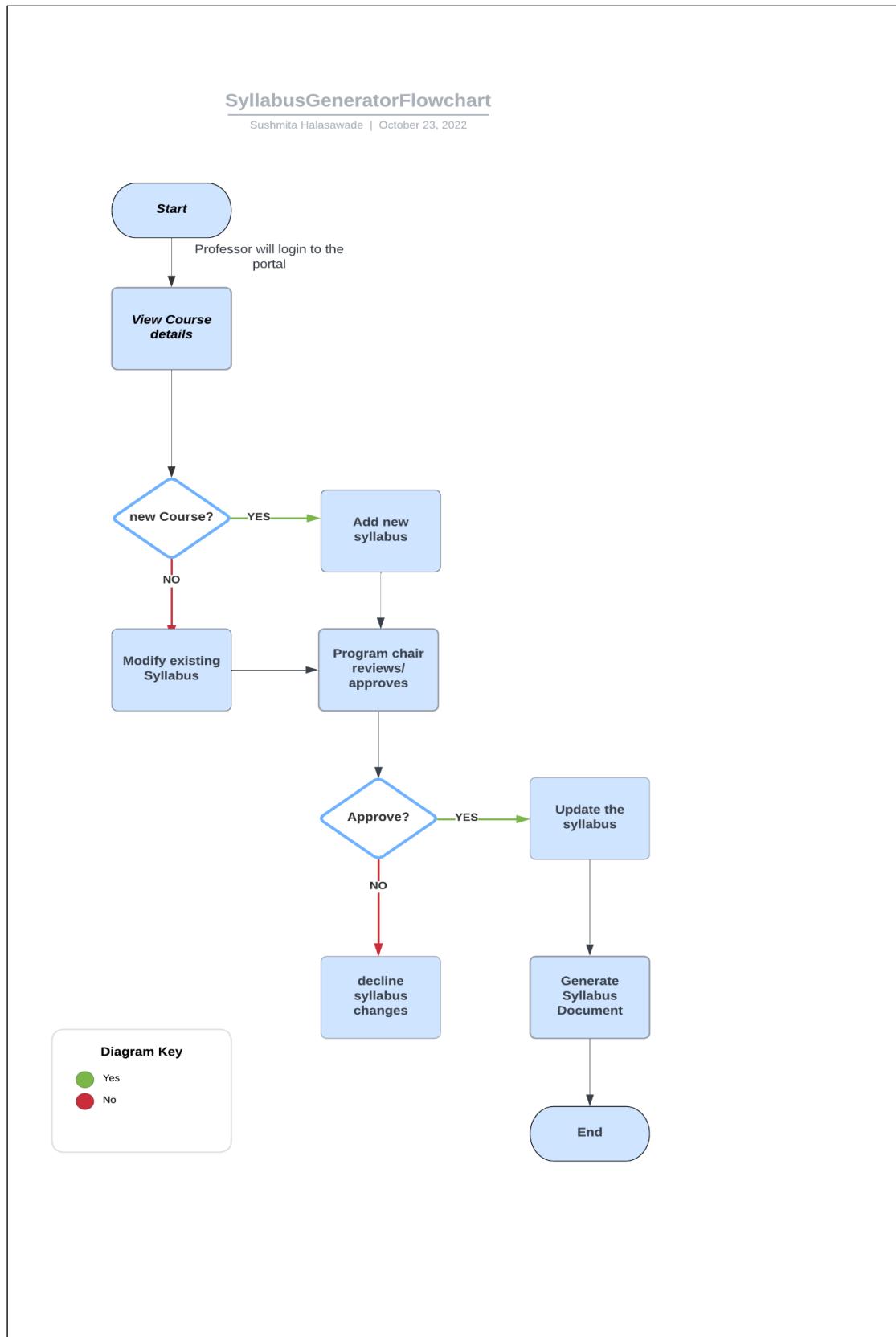
Syllabus generator is a smart system designed to increase the quality of administrative services provided to department staff in a university. Recent attempts have been made to design and build a user-friendly and dependable database system to meet the needs of the department staff but this house keeping 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



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

SyllabusID: Syllabus 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

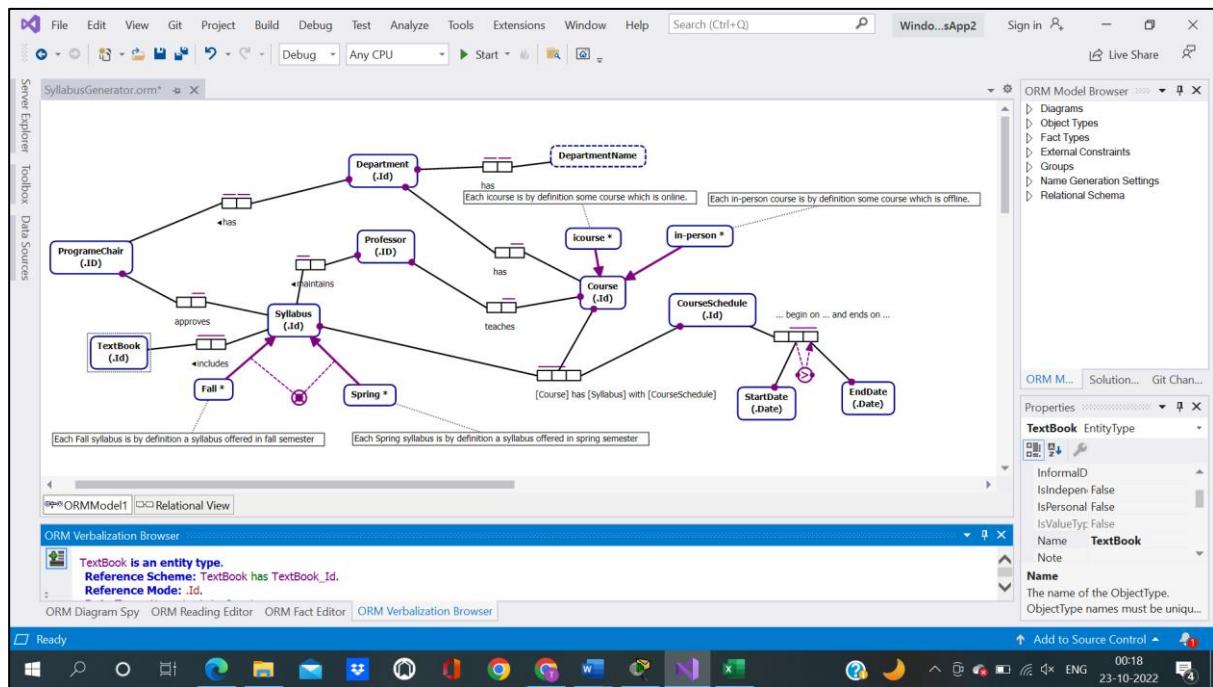
Section 2

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.

ProgramChair **is an entity type**.

Reference Scheme: ProgramChair has ProgramChair_ID.

Reference Mode: .ID.

Data Type: Numeric: Auto Counter.

Fact Types:

ProgramChair has ProgramChair_ID.

Department has ProgramChair.

ProgramChair approves Syllabus.

Department has ProgramChair.

Each Department has **exactly one** ProgramChair.

For each ProgramChair, **at most one** Department has **that** ProgramChair.

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.
ProgramChair 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.
ProgramChair approves Syllabus.
Each ProgramChair approves **some** Syllabus.
For each Syllabus, **at most one** ProgramChair approves **that** Syllabus.
It is possible that some ProgramChair 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.

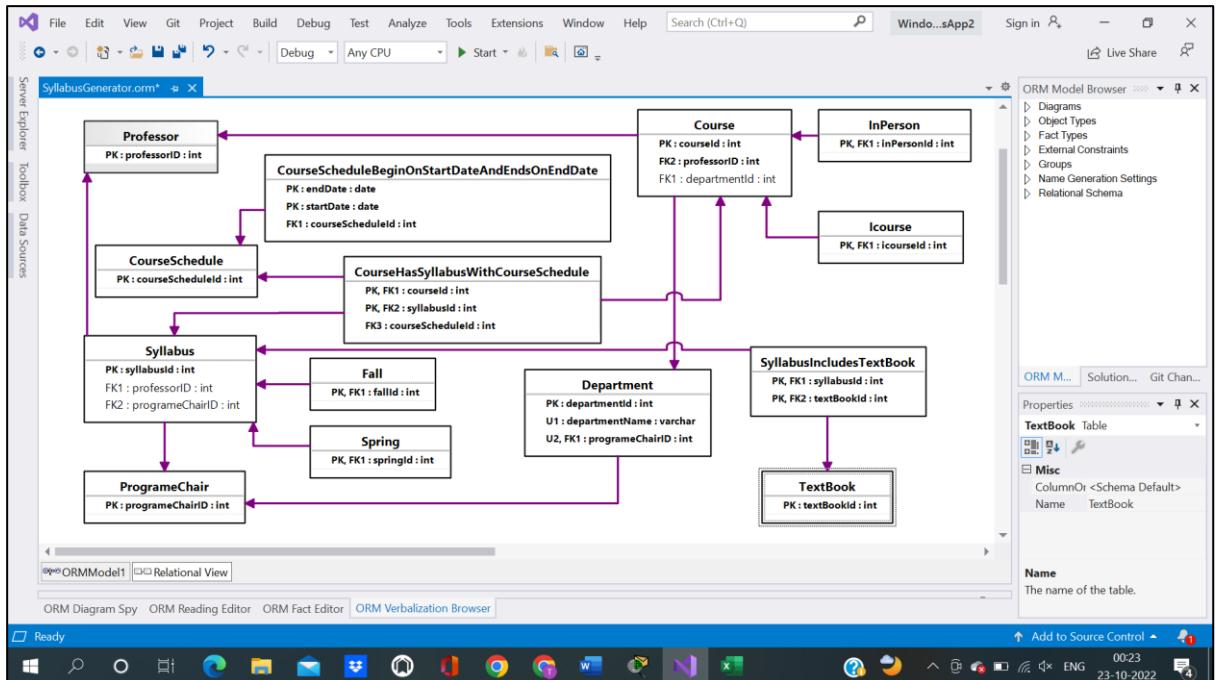
Data Type: Numeric: Auto Counter.

Fact Types:

Department has Department_Id.
Department has DepartmentName.
Department has Course.
Department has ProgrammeChair.

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,
    programChairID int NOT NULL,
    CONSTRAINT Department_PK PRIMARY KEY(departmentId),
    CONSTRAINT Department_UC1 UNIQUE(departmentName),
    CONSTRAINT Department_UC2 UNIQUE(programChairID)
)
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.ProgramChair
(
)
```

```
    programChairID int IDENTITY (1, 1) NOT NULL,
    CONSTRAINT ProgramChair_PK PRIMARY KEY(programChairID)
)
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,
    programChairID 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
(programmeChairID) REFERENCES ORMModel1.ProgrammeChair (programmeChairID) 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  
(programmeChairID) REFERENCES ORMMODEL1.ProgrammeChair (programmeChairID) 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
```

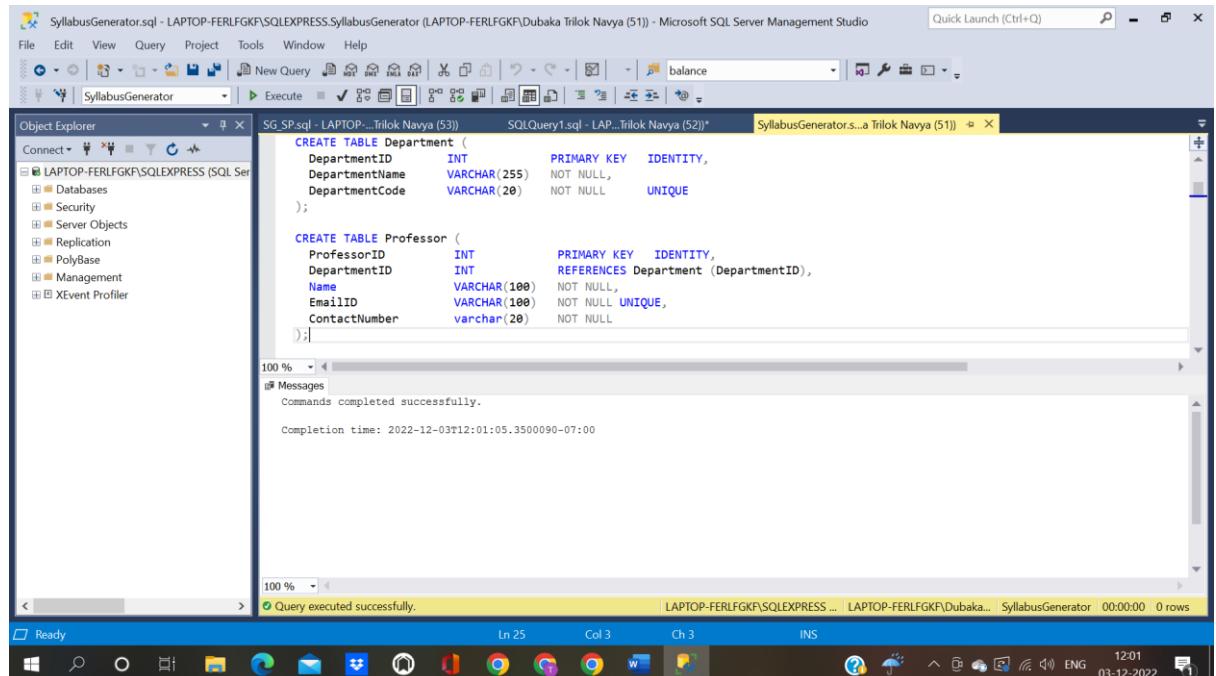
```
ALTER TABLE ORMMODEL1.SyllabusIncludesTextBook ADD CONSTRAINT  
SyllabusIncludesTextBook_FK1 FOREIGN KEY (syllabusId) REFERENCES ORMMODEL1.Syllabus  
(syllabusId) ON DELETE NO ACTION ON UPDATE NO ACTION  
GO
```

```
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



```
File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer
SyllabusGenerator
SG SP.sql - LAPTOP-FERLFGKF\SQLEXPRESS.SyllabusGenerator (LAPTOP-FERLFGKF\Dubaka Trilok Navya (51)) - Microsoft SQL Server Management Studio
CREATE TABLE Department (
    DepartmentID      INT          PRIMARY KEY   IDENTITY,
    DepartmentName    VARCHAR(255) NOT NULL,
    DepartmentCode    VARCHAR(20)  NOT NULL        UNIQUE
);

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
);
```

100 %

Messages

Commands completed successfully.

Completion time: 2022-12-03T12:01:05.3500090-07:00

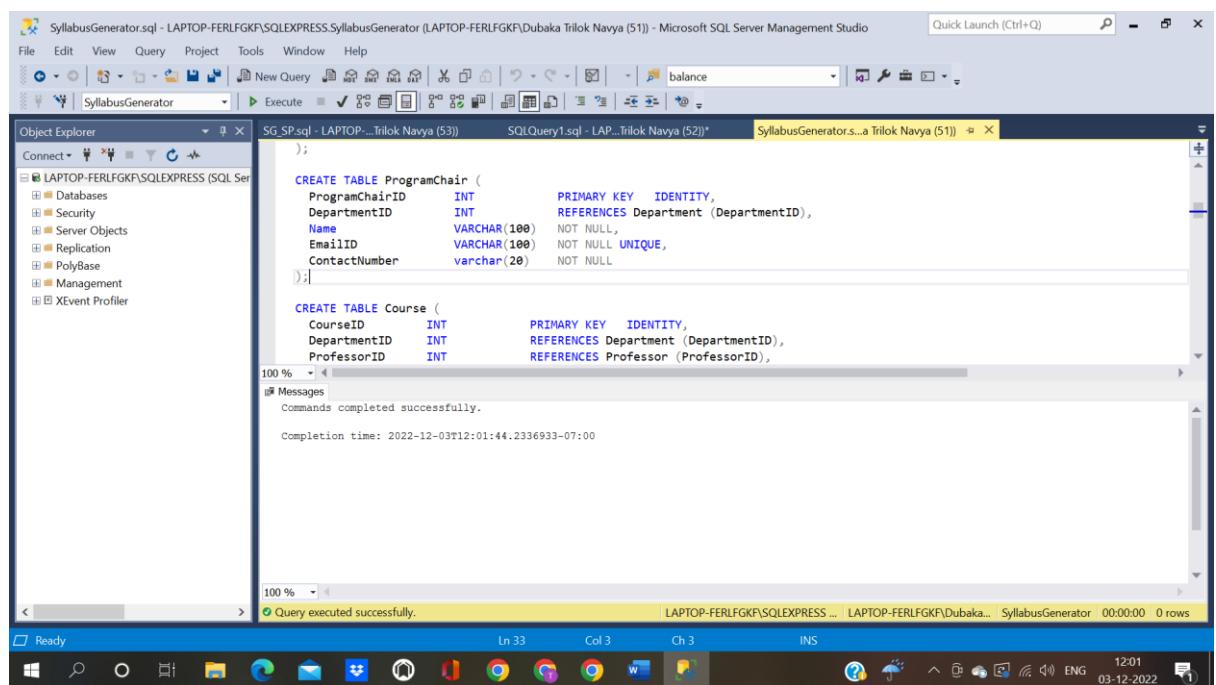
100 %

Query executed successfully.

LAPTOP-FERLFGKF\SQLEXPRESS ... LAPTOP-FERLFGKF\Dubaka... SyllabusGenerator 00:00:00 0 rows

Ready In 25 Col 3 Ch 3 INS

12:01 03-12-2022



```
File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer
SyllabusGenerator
SG SP.sql - LAPTOP-FERLFGKF\SQLEXPRESS.SyllabusGenerator (LAPTOP-FERLFGKF\Dubaka Trilok Navya (51)) - Microsoft SQL Server Management Studio
CREATE TABLE ProgramChair (
    ProgramChairID    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
);

CREATE TABLE Course (
    CourseID          INT          PRIMARY KEY   IDENTITY,
    DepartmentID     INT          REFERENCES Department (DepartmentID),
    ProfessorID       INT          REFERENCES Professor (ProfessorID),
    
```

100 %

Messages

Commands completed successfully.

Completion time: 2022-12-03T12:01:44.2336933-07:00

100 %

Query executed successfully.

LAPTOP-FERLFGKF\SQLEXPRESS ... LAPTOP-FERLFGKF\Dubaka... SyllabusGenerator 00:00:00 0 rows

Ready In 33 Col 3 Ch 3 INS

12:01 03-12-2022

SyllabusGenerator.sql - LAPTOP-FERLFGKF\SQLEXPRESS.SyllabusGenerator (LAPTOP-FERLFGKF\Dubaka Trilok Navya (51)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

Object Explorer

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,
 AuditDate DATETIME
)

100 % Commands completed successfully.
Completion time: 2022-12-03T12:03:25.2753518-07:00

100 % Query executed successfully.

LN 63 COL 1 CH 1 INS

LAPTOP-FERLFGKF\SQLEXPRESS ... | LAPTOP-FERLFGKF\Dubaka... SyllabusGenerator 00:00:00 0 rows

Ready 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)`

A screenshot of the Azure Data Studio interface. The title bar shows "Azure Data Studio" and the connection details "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The main area displays a query window with the following content:

```
1 select * from Department with (nolock)
```

The results pane shows a table with the following data:

	DepartmentID	DepartmentName	DepartmentCode
1	1	Information Technology	IFT
2	2	Computer Science	CS
3	3	Software Engineering	SE
4	4	Computer Engineering	CE
5	5	Construction Management	CMT

At the bottom of the interface, there is a status bar with the following information: "Ln 1, Col 40 Spaces: 4 UTF-8 LF SQL 5 rows MSSQL 00:00:00 localhost : SyllabusGenerator".

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)`

A screenshot of the Azure Data Studio interface. The title bar shows "Azure Data Studio" and the connection details "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The main area displays a SQL query in the editor pane:

```
1 select * from ProgramChair with (nolock)
```

The results pane shows a table with the following data:

	ProgramChairID	DepartmentID	Name	EmailID	ContactNumber
1	1	1	Tatiana Walsh	TatianaWalsh@asu.edu	6021111111
2	2	2	Dont Know	DontKnow@asu.edu	6021112222
3	3	3	Srividya Bansal	SrividyaBansal@asu.edu	6021113333
4	4	4	John Doe	JohnDoe@asu.edu	6021114444
5	5	5	Abc Xyz	AbcXyz@asu.edu	6021115555

At the bottom of the interface, status information includes "Ln 1, Col 27", "Spaces: 4", "UTF-8", "LF", "SQL", "5 rows", "MSSQL", "00:00:00", "localhost : SyllabusGenerator", and a refresh icon.

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)
```

ProfessorID	DepartmentID	Name	EmailID	ContactNumber
1	1	Robert Rucker	RobertRucker@asu.edu	6022221111
2	1	Asmaa Elbadrawy	AsmaaElbadrawy@asu.edu	6023331111
3	1	Dinesh Sthapit	DineshSthapit@asu.edu	6024441111
4	2	Chris Bryan	ChrisBryan@asu.edu	6022222222
5	2	Hasan Davulcu	HasanDavulcu@asu.edu	6023332222
6	2	Samira Ghayekhlo	SamiraGhayekhlo@asu.edu	6024442222

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'),
```

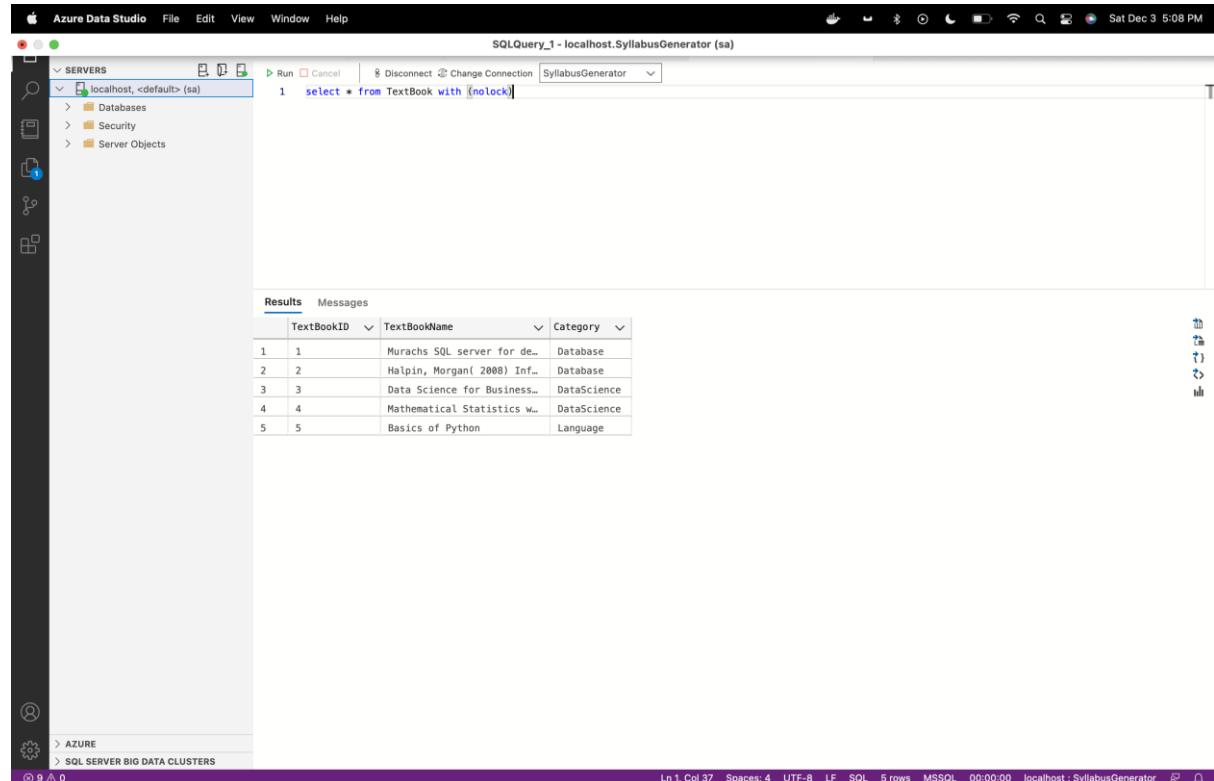
(4, 'Mathematical Statistics with Applications, 7th edition, by Wackerly, Mendenhall, and Scheaffer, Brooks/Cole, Cengage Learning, 2008.', 'DataScience'),

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

```
SET IDENTITY_INSERT TextBook OFF;
```

Query:

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



A screenshot of the Azure Data Studio interface. The top navigation bar includes 'Azure Data Studio', 'File', 'Edit', 'View', 'Window', and 'Help'. The title bar shows 'SQLQuery_1 - localhost.SyllabusGenerator (sa)' and the date 'Sat Dec 3 5:08 PM'. The left sidebar has a 'Servers' section with 'localhost, <default> (sa)' selected, showing 'Databases', 'Security', and 'Server Objects'. The main area contains a query editor with the following text:

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

The results pane displays a table with the following data:

	TextBookID	TextBookName	Category
1	1	Murachs SQL server for de...	Database
2	2	Halpin, Morgan (2008) Inf...	Database
3	3	Data Science for Business...	DataScience
4	4	Mathematical Statistics w...	DataScience
5	5	Basics of Python	Language

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  
);
```

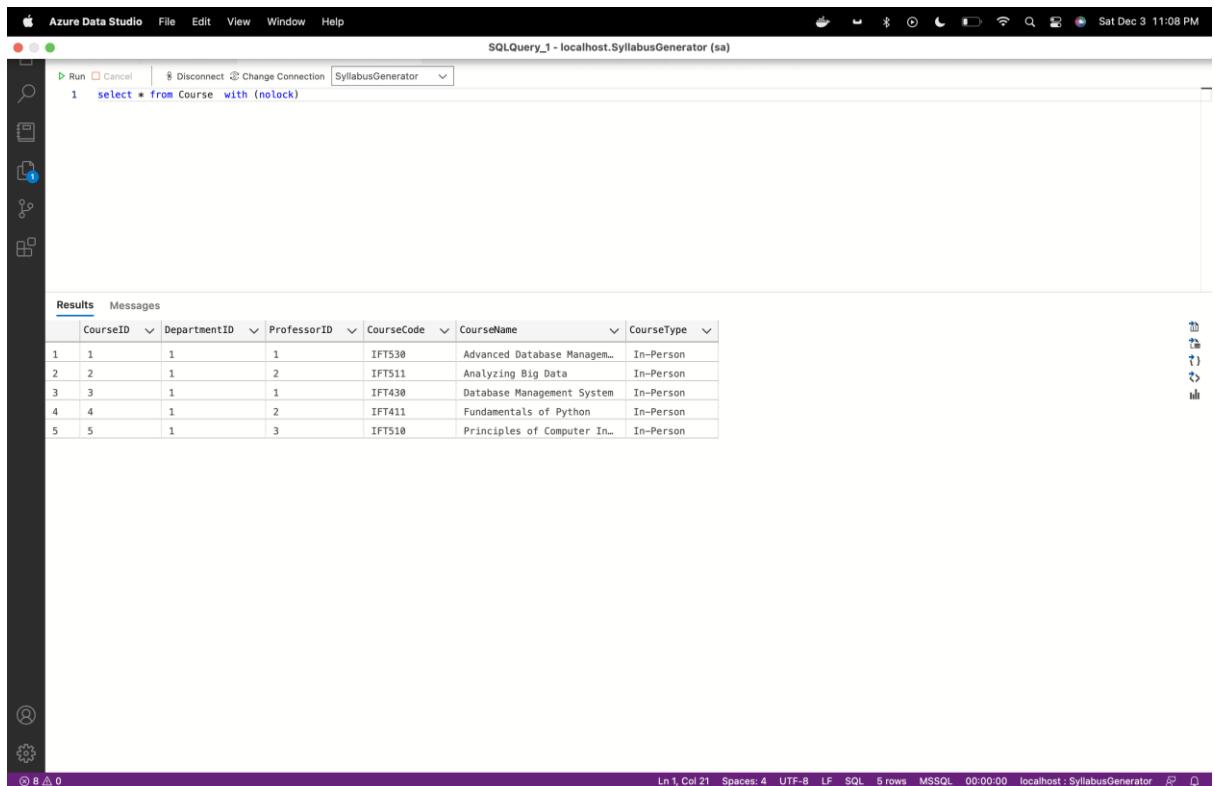
```
SET IDENTITY_INSERT course ON;
```

```
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)
```



A screenshot of the Azure Data Studio interface. The title bar shows "Azure Data Studio" and "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The main area has a toolbar on the left with icons for search, refresh, and other database operations. A code editor window is open with the following SQL query:

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

The results pane displays the following data:

	CourseID	DepartmentID	ProfessorID	CourseCode	CourseName	CourseType
1	1	1	1	IFT530	Advanced Database Managem...	In-Person
2	2	1	2	IFT511	Analyzing Big Data	In-Person
3	3	1	1	IFT430	Database Management System	In-Person
4	4	1	2	IFT411	Fundamentals of Python	In-Person
5	5	1	3	IFT510	Principles of Computer In...	In-Person

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)

The screenshot shows the Azure Data Studio interface with a SQL query window titled 'SQLQuery_1 - localhost.SyllabusGenerator (sa)'. The query is:

```
1  select * from Syllabus with (nolock)
```

The results grid displays the following data:

SyllabusID	CourseID	ProfessorID	TextBookID	Semester	LearningObjective	prerequisite
1	1	1	1,2	Fall	1.Understand the principl...	3
2	1	1	1,2	Spring	1.Understand the principl...	3
3	2	2	3,4	Fall	1.Understand how data sci...	4
4	2	2	3,4	Spring	1.Understand how data sci...	4
5	3	1	1	Fall	1.basics of database , AC...	NULL
6	4	2	5	Fall	1.Understand the basics o...	NULL

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;

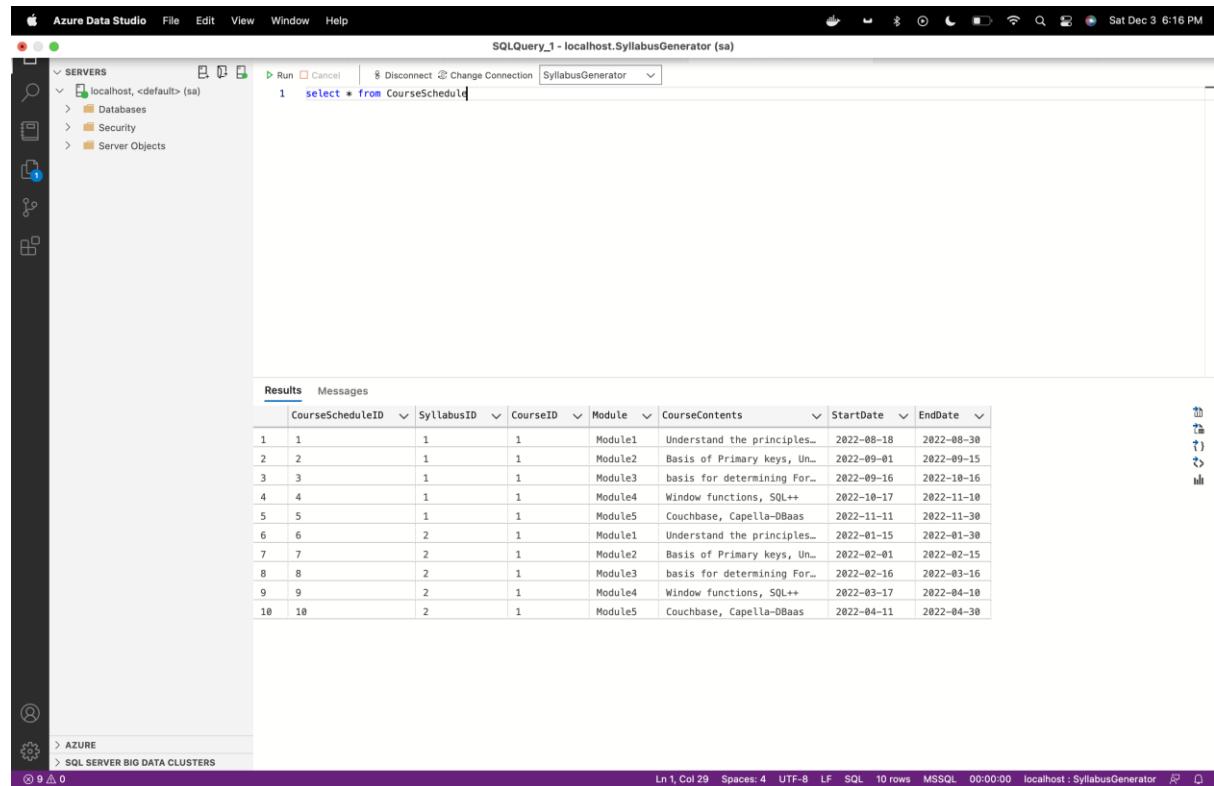
INSERT INTO CourseSchedule (CourseScheduleID, SyllabusID, CourseID,Module,CourseContents,StartDate,EndDate) VALUES
(1, 1, 1,'Module1','Understand the principles of conceptual modeling (ORM as an exemplar).', '15 Aug 2022','30 Aug 2022'),
(2, 1, 1,'Module2','Basis of Primary keys, Uniqueness/ Mandatory 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 exemplar).', '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)`



The screenshot shows the Azure Data Studio interface with a query window titled "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The query is:

```
select * from CourseSchedule
```

The results pane displays the following data:

	CourseScheduleID	SyllabusID	CourseID	Module	CourseContents	StartDate	EndDate
1	1	1	1	Module1	Understand the principles...	2022-08-18	2022-08-30
2	2	1	1	Module2	Basis of Primary keys, Un...	2022-09-01	2022-09-15
3	3	1	1	Module3	basis for determining For...	2022-09-16	2022-10-16
4	4	1	1	Module4	Window functions, SQL++	2022-10-17	2022-11-10
5	5	1	1	Module5	Couchbase, Capella-DBaaS	2022-11-11	2022-11-30
6	6	2	1	Module1	Understand the principles...	2022-01-15	2022-01-30
7	7	2	1	Module2	Basis of Primary keys, Un...	2022-02-01	2022-02-15
8	8	2	1	Module3	basis for determining For...	2022-02-16	2022-03-16
9	9	2	1	Module4	Window functions, SQL++	2022-03-17	2022-04-10
10	10	2	1	Module5	Couchbase, Capella-DBaaS	2022-04-11	2022-04-30

Section 5

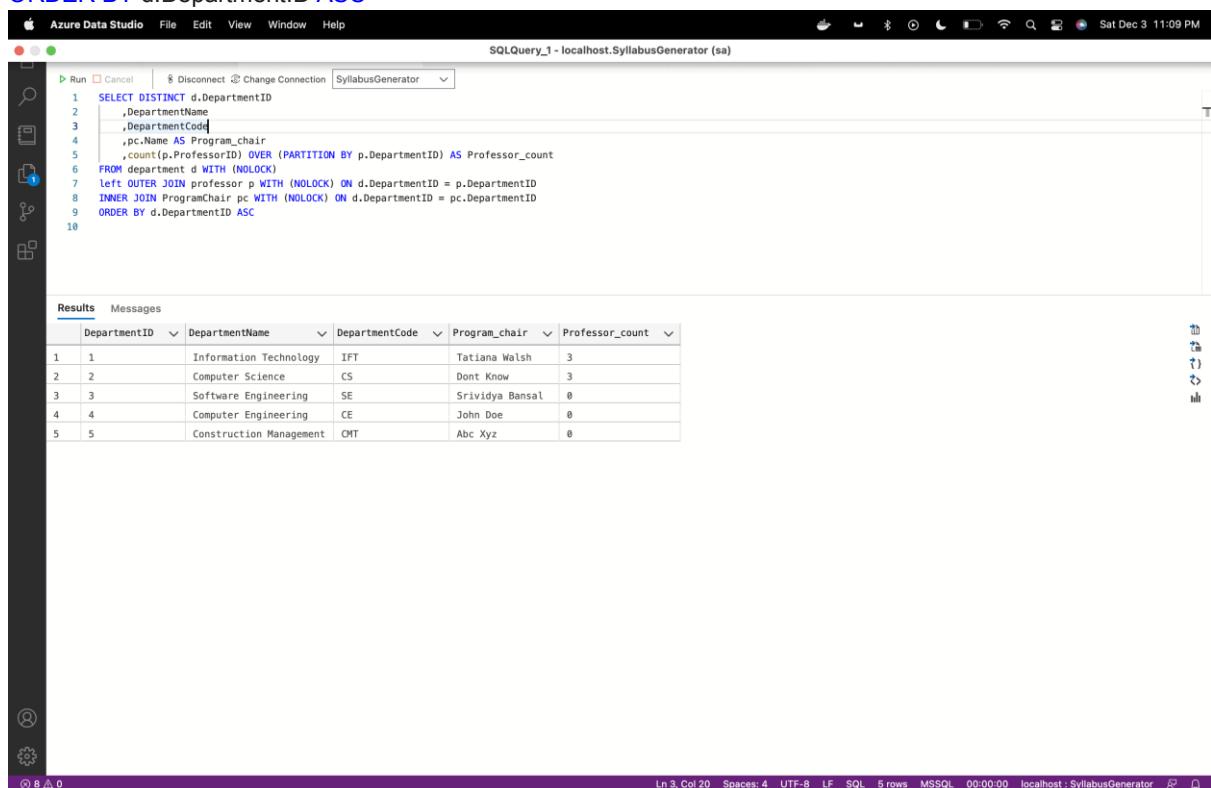
Section 5.1

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
```



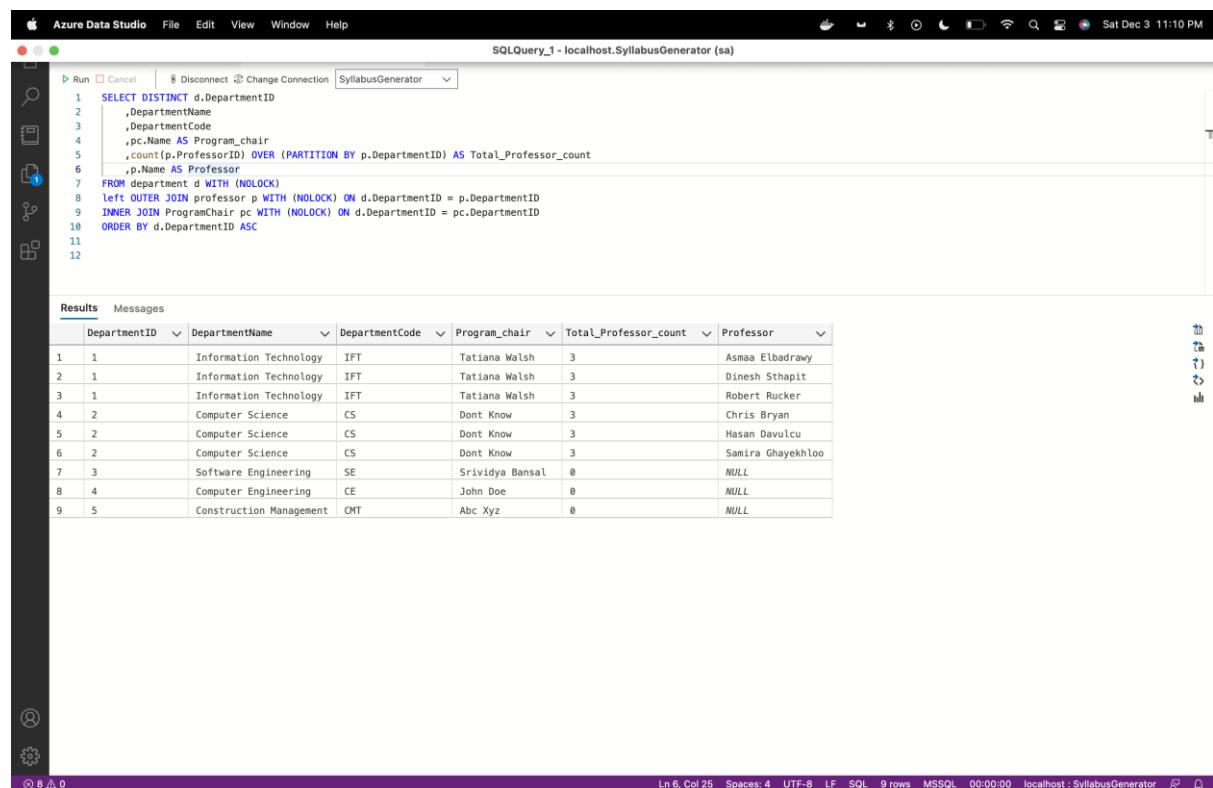
The screenshot shows the Azure Data Studio interface with a query window titled "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The query itself is the one provided above, selecting department details and professor counts. Below the query, the results pane displays a table with five rows of data:

	DepartmentID	DepartmentName	DepartmentCode	Program_chair	Professor_count
1	1	Information Technology	IFT	Tatiana Walsh	3
2	2	Computer Science	CS	Dont Know	3
3	3	Software Engineering	SE	Srividya Bansal	0
4	4	Computer Engineering	CE	John Doe	0
5	5	Construction Management	CMT	Abc Xyz	0

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
```



The screenshot shows the Azure Data Studio interface with a query window titled "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The query is the one provided above, selecting department details, professor counts, and program chairs. Below the query, the results are displayed in a table with columns: DepartmentID, DepartmentName, DepartmentCode, Program_chair, Total_Professor_count, and Professor.

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

```

The screenshot shows the Azure Data Studio interface with a query window titled "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The query is a SELECT statement that retrieves professor information and course details, including course counts. The results are displayed in a table with columns: ProfessorID, professor, Total_Course_count, Course_Code, Course_Name, and Course_Type.

	ProfessorID	professor	Total_Course_count	Course_Code	Course_Name	Course_Type
1	1	Robert Rucker	2	IFT530	Advanced Database Managem...	In-Person
2	1	Robert Rucker	2	IFT430	Database Management System	In-Person
3	2	Asmaa Elbadrawy	2	IFT511	Analyzing Big Data	In-Person
4	2	Asmaa Elbadrawy	2	IFT411	Fundamentals of Python	In-Person
5	3	Dinesh Sthapit	1	IFT510	Principles of Computer In...	In-Person
6	5	Chris Bryan	0	NULL	NULL	NULL
7	6	Hasan Davulcu	0	NULL	NULL	NULL
8	7	Samira Ghayekhloo	0	NULL	NULL	NULL

Question4:

Professor and Program chair contact details

(SELECT

```

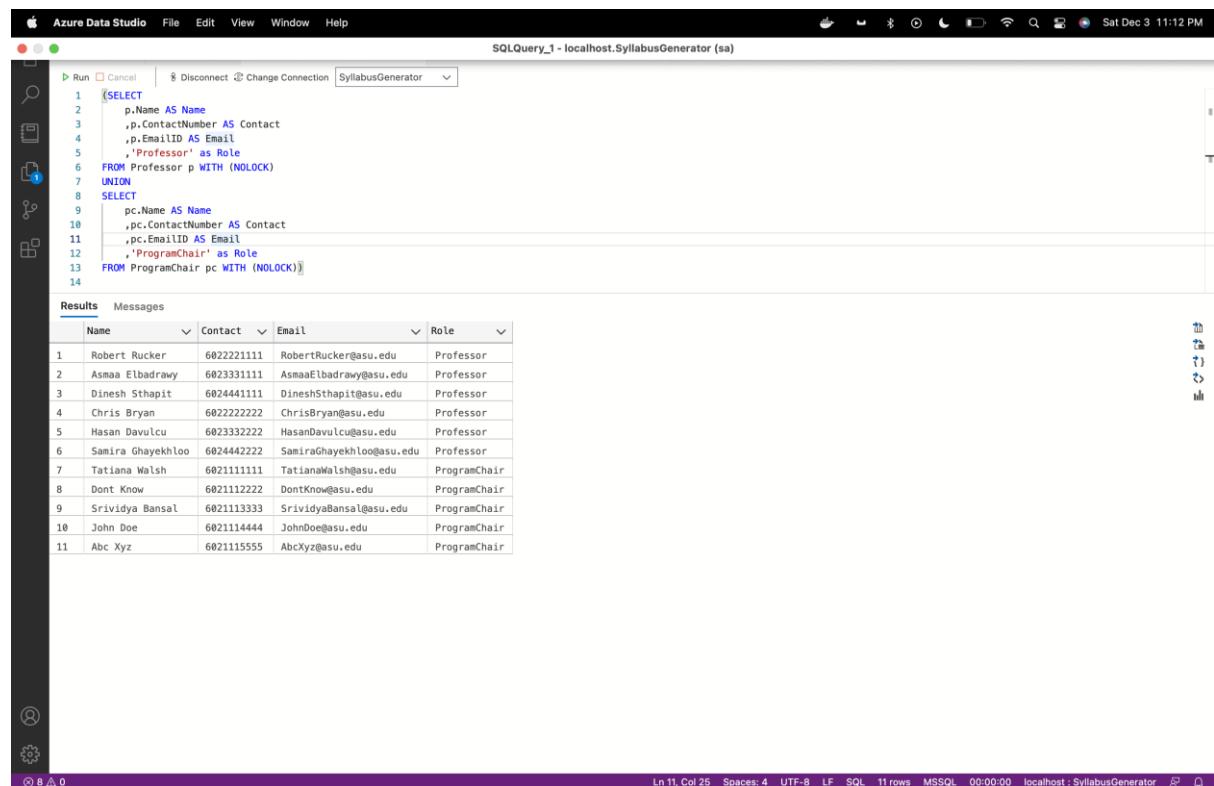
p.Name AS Name
,p.ContactNumber AS Contact
,p.EmailID AS Email
,'Professor' as Role
FROM Professor p WITH (NOLOCK)

```

UNION

SELECT

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



```
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)
```

	Name	Contact	Email	Role
1	Robert Rucker	6022221111	RobertRucker@asu.edu	Professor
2	Asmaa Elbadrawy	6023331111	AsmaaElbadrawy@asu.edu	Professor
3	Dinesh Sthapit	6024441111	DineshSthapit@asu.edu	Professor
4	Chris Bryan	6022222222	ChrisBryan@asu.edu	Professor
5	Hasan Davulcu	6023332222	HasanDavulcu@asu.edu	Professor
6	Samira Ghayekhloo	6024442222	SamiraGhayekhloo@asu.edu	Professor
7	Tatiana Walsh	6021111111	TatianaWalsh@asu.edu	ProgramChair
8	Dont Know	6021112222	DontKnow@asu.edu	ProgramChair
9	Srividya Bansal	6021113333	SrividyaBansal@asu.edu	ProgramChair
10	John Doe	6021114444	JohnDoe@asu.edu	ProgramChair
11	Abc Xyz	6021115555	AbcXyz@asu.edu	ProgramChair

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
```

The screenshot shows the Azure Data Studio interface. In the top navigation bar, the title is "SQLQuery_1 - localhost.SyllabusGenerator (sa)". The left sidebar shows the "SERVERS" node expanded, with "localhost, <default> (sa)" selected. The main area contains a SQL query:

```

3     c.CourseName + '('+ s.Semester + ')' as CourseName
4     ,min(cs.StartDate) over (PARTITION by cs.SyllabusID) as Course_start_date
5     ,max(cs.EndDate) over (PARTITION by cs.SyllabusID) as Course_end_date
6   FROM Syllabus s WITH (NOLOCK)
7   INNER JOIN Course c WITH (NOLOCK) ON c.CourseID = s.CourseID
8   INNER JOIN CourseSchedule cs WITH (NOLOCK) ON cs.SyllabusID = s.SyllabusID

```

The "Results" tab is selected, displaying the following table:

CourseName	Course_start_date	Course_end_date
Advanced Database Management System	2022-08-18	2022-11-30
Advanced Database Management System	2022-01-15	2022-04-30

At the bottom of the interface, there are various status indicators and connection information.

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

```

```

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

```

SyllabusID	CourseName	TextBookName
1	Advanced Database Management Systems(Fall)	Halpin, Morgan(2008) Inf...
2	Advanced Database Management Systems(Fall)	Murach's SQL server for de...
3	Advanced Database Management Systems(Spring)	Halpin, Morgan(2008) Inf...
4	Advanced Database Management Systems(Spring)	Murach's SQL server for de...
5	Analyzing Big Data(Fall)	Data Science for Business...
6	Analyzing Big Data(Fall)	Mathematical Statistics w...
7	Database Management System(Fall)	Murach's SQL server for de...
8	Fundamentals of Python(Fall)	Basics of Python

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

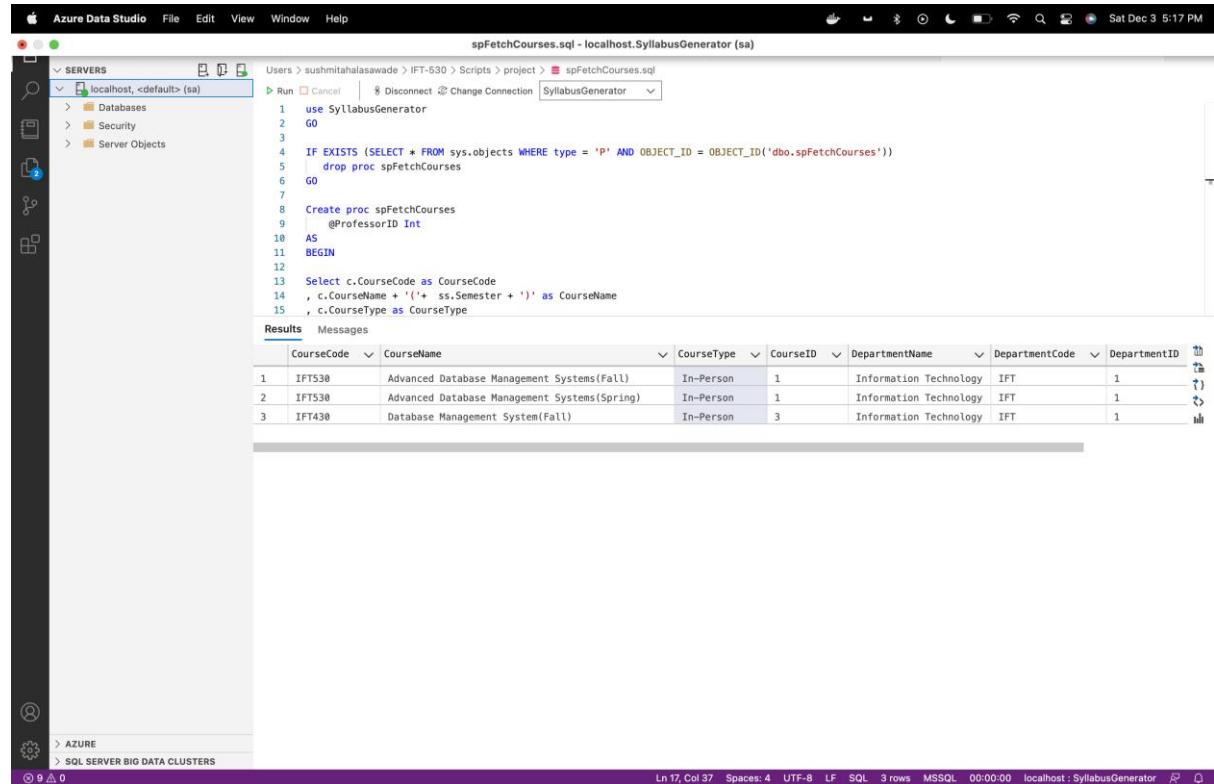
inner join SyllabusGenerator..Department d with (nolock) on d.DepartmentID = c.DepartmentID

where c.ProfessorID = @ProfessorID

```

END

```
exec spFetchCourses 1
```



The screenshot shows the Azure Data Studio interface with the following details:

- Servers:** localhost, <default> (sa)
- Script:** spFetchCourses.sql - localhost.SyllabusGenerator (sa)
- Code:**

```

1 use SyllabusGenerator
2 GO
3
4 IF EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID = OBJECT_ID('dbo.spFetchCourses'))
5 drop proc spFetchCourses
6 GO
7
8 Create proc spFetchCourses
9   @ProfessorID Int
10  AS
11  BEGIN
12
13  Select c.CourseCode as CourseCode
14 , c.CourseName + '('+ ss.Semester + ')' as CourseName
15 , c.CourseType as CourseType

```

- Results:** Messages

CourseCode	CourseName	CourseType	CourseID	DepartmentName	DepartmentCode	DepartmentID
1 IFT530	Advanced Database Management Systems(Fall)	In-Person	1	Information Technology	IFT	1
2 IFT538	Advanced Database Management Systems(Spring)	In-Person	1	Information Technology	IFT	1
3 IFT430	Database Management System(Fall)	In-Person	3	Information Technology	IFT	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
```

```

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,cs.E
    from SyllabusGenerator..syllabus ss with (nolock) inner join SyllabusGenerator..CourseSchedule cs with (nolock) on ss.SyllabusID = cs.SyllabusID

```

CourseName	LearningObjective	TextBookID	prerequisite	Module	CourseContents	StartDate
1 Advanced Database Management Systems(Fal...	1.Understand the princip...	1,2	3	Module1	Understand the principles...	2022-1
2 Advanced Database Management Systems(Fal...	1.Understand the princip...	1,2	3	Module2	Basis of Primary keys, Un...	2022-1
3 Advanced Database Management Systems(Fal...	1.Understand the princip...	1,2	3	Module3	basis for determining For...	2022-1
4 Advanced Database Management Systems(Fal...	1.Understand the princip...	1,2	3	Module4	Window functions, SQL++	2022-
5 Advanced Database Management Systems(Fal...	1.Understand the princip...	1,2	3	Module5	Couchbase, Capella-DBaaS	2022-

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,
```

```

@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.

```

spInsertorUpdateSyllabus.sql - localhost.SyllabusGenerator (sa)
Users > sushmitahalasawade > IFT-530 > Scripts > project > spInsertorUpdateSyllabus.sql

30 END
31
32 END
33
34 --
35 --exec spInsertorUpdateSyllabus 1, '1,2', '1.Understand the principles of conceptual modeling,2.understand the use of scripts when implementing SQL code','3',
36
37
38 select CourseContents,StartDate,* from CourseSchedule where SyllabusID =1 and module = 'Module1'
39

```

CourseContents	StartDate	CourseScheduleID	SyllabusID	CourseID	Module	CourseContents
Understand the principles of conceptual modeling(ORM).	2022-08-18	1	1	1	Module1	Understand the pr

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
```

The screenshot shows the Azure Data Studio interface. The title bar reads "Azure Data Studio - spDeleteSyllabus.sql - localhost.SyllabusGenerator (sa)". The left sidebar shows the "Servers" node with "localhost, <default> (sa)" selected. The main pane displays the SQL script "spDeleteSyllabus.sql". The script contains the following code:

```
13 if exists(select * from SyllabusGenerator..Courseschedule where SyllabusID = @SyllabusID)
14 begin
15 delete from SyllabusGenerator..CourseSchedule where SyllabusID = @SyllabusID
16 end
17 delete from SyllabusGenerator..Syllabus where SyllabusID = @SyllabusID
18
19
20
21 --exec spDeleteSyllabus 4
22
23 select * from Syllabus where SyllabusID = 4
24
25
26
27
```

The "Results" tab is selected at the bottom, showing a table with columns: SyllabusID, CourseID, ProfessorID, TextBookID, Semester, and LearningObject... prerequisite. There are no rows displayed.

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:

The screenshot shows the Azure Data Studio interface with a query window titled "fnGettextbookdetails.sql - localhost.SyllabusGenerator (sa)". The code pane contains the following T-SQL script:

```
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)
```

The results pane shows the output of the function execution:

TextBookName	TextBookID
Murachs SQL server for developers	1
Halpin, Morgan (2008) Information Modeling and Relational Databases 2nd edition, Kaufman	2

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:

The screenshot shows the Azure Data Studio interface. The left sidebar displays the 'Servers' node under 'localhost, <default> (sa)'. The main area shows a SQL script named 'fnGetPrerequisite.sql' with the following content:

```
1 USE SyllabusGenerator;
2 GO
3
4 CREATE or ALTER FUNCTION fnGetPrerequisite
5     (@SyllabusID int
6     )
7
8 RETURNS table
9 RETURN
10 (
11     select CourseID,CourseCode,CourseName from Course where CourseID in (select * from STRING_SPLIT((select prerequisite from Syllabus where SyllabusID = @SyllabusID), ','))
12 )
13
14
15
16 --select * from dbo.fnGetPrerequisite(1)
17
```

Below the script, the 'Results' tab is selected, showing a table with one row of data:

	CourseID	CourseCode	CourseName
1	3	IFT430	Database Management System

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.

The screenshot shows the Azure Data Studio interface. The top navigation bar includes File, Edit, View, Window, Help, and a date/time stamp of Sat Dec 3 5:37 PM. The left sidebar shows SERVERS with localhost, <default> (sa) selected. The main area displays a script named SG_Trigger.sql with the following content:

```
-- test  
/*  
select * from syllabus where syllabusid = 4;  
select * from syllabusaudit ;  
*/
```

The Results tab is active, showing a table with the following data:

SyllabusAuditID	SyllabusID	CourseID	ProfessorID	TextBookID	Semester	LearningObjective	prerequisite	Audit_Act	
1	1	4	2	2	3,4	Spring	1.Understand how data sci...	4	DELETE

At the bottom of the interface, status information includes Ln 39, Col 1 (74 selected), Spaces: 4, UTF-8, LF, SQL, 1 rows, MSSQL, 00:00:00, localhost : SyllabusGenerator.

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)

```
34 Insert into SyllabusAudit (SyllabusID, CourseID, ProfessorID, textbookid,Semester,LearningObjective, prerequisite,Audit_action, Action_by, Action_At, Remark)
35 values(@SyllabusID,@CourseID,@ProfessorID,@textbookid,@Semester,@LearningObjective,@prerequisite,'UPDATE',@ProfessorID,GETDATE(),@description)
36
37
38 -- test
39 /*
40 update syllabus with (rowlock) set textbookid = '1,2' where syllabusid =1;
41 select * from Syllabus where syllabusid =1;
42 select top 1 remark, * from SyllabusAudit where syllabusid =1 order by action_at desc;
43 */
44
45
```

Results Messages

	SyllabusID	CourseID	ProfessorID	TextBookID	Semester	LearningObjective	prerequisite	Audit
1	1	1	1	1,2	Fall	1.Understand the princip...	3	UPD

	remark	SyllabusAuditID	SyllabusID	CourseID	ProfessorID	TextBookID	Semester	LearningObjective	prerequisite	Audit
1	Updated textbookid of the syllabus	3	1	1	1	1,2	Fall	1.Understand the princip...	3	UPD

Ln 38, Col 1 Spaces: 4 UTF-8 LF SQL 2 rows MSSQL 00:00:00 localhost: SyllabusGenerator

Section 6

Couchbase – NoSQL

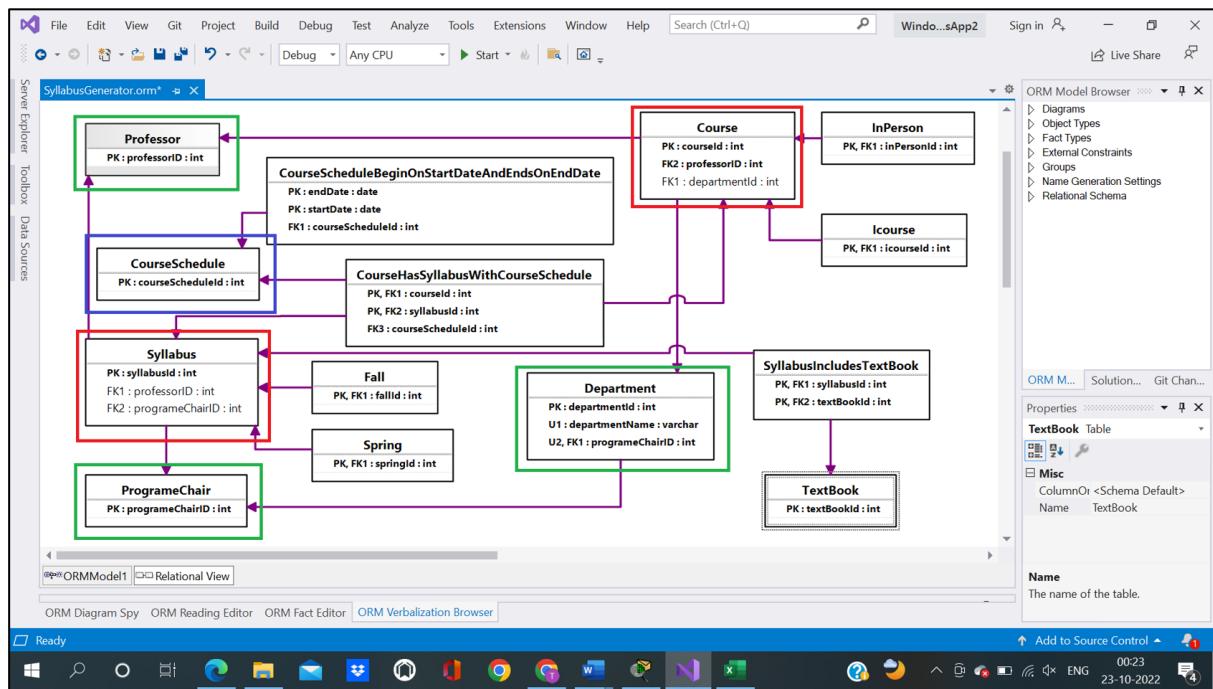
Section 6.1 ORM

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

The screenshot shows a web browser window for the ADBMS application. The URL in the address bar is 127.0.0.1. The main page displays a list of existing buckets: 'Course' (551KiB), 'Enrollment' (555KiB), 'Student' (555KiB), 'customers' (583KiB), and 'orders' (583KiB). All buckets are associated with 'Documents' and 'Scopes & Collections'. On the left, a sidebar menu includes 'Dashboard', 'Servers', 'Buckets' (selected), 'Backup', 'XDCR', 'Security', 'Settings', 'Logs', 'Documents', 'Query', 'Indexes', 'Search', 'Analytics', 'Eventing', and 'Views'. A warning message at the bottom states: 'Warning: At least two servers with the data service are required to provide replication.' A modal dialog box titled 'Add Data Bucket' is open in the center. It contains fields for 'Name' (set to 'SyllabusGenerator_Department'), 'Bucket Type' (set to 'Couchbase'), 'Storage Backend' (set to 'CouchStore'), and 'Memory Quota' (set to '100 MiB'). Below these fields is a progress bar indicating disk usage: 'other buckets (500MiB)', 'this bucket (100MiB)', and 'available (424MiB)'. At the bottom of the dialog are 'Cancel' and 'Add Bucket' buttons.

Course Bucket : SyllabusGenerator_Course

The screenshot shows the ADBMS interface with the 'Buckets' tab selected. A modal window titled 'Add Data Bucket' is open, prompting for a bucket name ('SyllabusGenerator_Course'), bucket type ('Couchbase'), storage backend ('CouchStore'), and memory quota (set to 100 MiB). A warning message at the bottom of the modal states: 'Warning: At least two servers with the data service are required to provide replication.'

Course Schedule Bucket: SyllabusGenerator_CourseSchedule

The screenshot shows the ADBMS interface with the 'Buckets' tab selected. A modal window titled 'Add Data Bucket' is open, prompting for a bucket name ('SyllabusGenerator_CourseSchedule'), bucket type ('Couchbase'), storage backend ('CouchStore'), and memory quota (set to 100 MiB). A warning message at the bottom of the modal states: 'Warning: At least two servers with the data service are required to provide replication.'

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"
    }
]
}

);

```

The screenshot shows the ADBMS Query Editor interface. On the left, there's a sidebar with various navigation links like Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, Analytics, Eventing, and Views. The 'Query' link is currently selected. The main area has tabs for Workbench, Monitor, and UDF. The current tab is 'Query Editor'. In the editor, the following SQL code is written:

```

1 insert into SyllabusGenerator_Department (key, value)
2 values ("1",
3
4     { "DepartmentID": "1",
5         "DepartmentName": "Information Technology",
6         "DepartmentCode": "IFT",
7         "ProgramChair":
8             { "ProgramChairID": "1",
9                 "Name": "Tatiana Walsh",
10                "EmailID": "TatianaWalsh@asu.edu",
11                "ContactNumber": "6021111111"
12            },
13            "Professor": [
14                { "ProfessorID": "1",

```

Below the code, there are buttons for Execute, Run as TX, Index Advisor, Explain, and a status message indicating SUCCESS just now | 52.8ms | 2 mutations. The results section shows a single row of data:

```

1+ [
2     "results": []
3 ]

```

On the right side, there's an 'Explore Your Data' panel with sections for Course, Enrollment, Student, customers, and orders. It also includes a note about replication: 'Warning: At least two servers with the data service are required to provide replication.' A Refresh button is located at the bottom right of the results area.

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": "1,2",
      "LearningObjective": "1.Understand the principles of conceptual modeling,2.understand the use of scripts when implementing SQL code",
      "Prerequisite": "3"
    }
  ]
})
```

"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": "Data Science Fundamentals",
 "CourseCode": "DSF101",
 "CourseType": "Online",
 "ProfessorID": "3",
 "Syllabus": ""
})

```

    "CourseName": "Fundamentals of Python",
    "CourseCode": "IFT411",
    "CourseType": "In-Person",
    "ProfessorID": "3",
    "Syllabus": """
}

);

```

The screenshot shows the ADBMS Query Editor interface. On the left, there's a sidebar with various navigation links like Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query (which is selected), Indexes, Search, Analytics, Eventing, and Views. The main area is titled 'Query Editor' and contains a code editor with the following SQL-like query:

```

1 insert into SyllabusGenerator_Course (key, value)
2 values ("1",
3
4   { "CourseID": "1",
5     "DepartmentID": "1",
6     "CourseName": "Advanced Database Management Systems",
7     "CourseCode": "IFT530",
8     "CourseType": "In-Person",
9     "ProfessorID": "2",
10    "Syllabus": [
11      { "SyllabusID": "1",
12        "Semester": "Fall",
13        "TextbookID": "1,2",
14        "LearningObjective": "1.Understand the principles of conceptual modeling,2.understand the use of script"
]

```

Below the code editor, there are buttons for 'Execute', 'Run as TX', 'Index Advisor', and 'Explain'. A status bar indicates 'success just now | 28.6ms | 4 mutations'. To the right, there's a 'Explore Your Data' section showing document types and sample values from collections like 'Course', 'Enrollment', 'Student', 'customers', and 'orders'. At the bottom, there's a results table with one row of data and a warning message: 'Warning: At least two servers with the data service are required to provide replication.' A 'Refresh' button is also present.

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 exemplar).",
  "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 exemplar).",
    "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"
  }
]
```

}

);

The screenshot shows the ADBMS Query Editor interface. On the left, a sidebar lists various database management tasks like Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query (which is selected), Indexes, Search, Analytics, Eventing, and Views. The main area has tabs for Workbench, Monitor, and UDF. The current tab is 'Query'. In the 'Query Editor' pane, there is a code editor containing the following SQL-like query:

```
1 insert into SyllabusGenerator_CourseSchedule (key, value)
2 values ('1',
3
4   { "CourseScheduleID": "1",
5     "SyllabusID": "1",
6     "CourseID": "1",
7     "Module": [
8       { "ModuleName": "Module1",
9         "CourseContent": "Understand the principles of conceptual modeling (ORM as an exemplar).",
10        "StartDate": "15 Aug 2022",
11        "EndDate": "30 Aug 2022"
12      },
13      { "ModuleName": "Module2",
14        "CourseContent": "Basis of Primary keys, Uniqueness/ Mandatory constraints",
15      }
16    ]
17  }
18 }
```

Below the code editor are buttons for Execute, Run as TX, Index Advisor, Explain, and a status message indicating success just now | 26ms | 2 mutations. The 'Results' pane shows the output of the query:

```
1+ []
2 "results": []
3 []
```

There is a warning message at the bottom left: "Warning: At least two servers with the data service are required to provide replication." On the right side, there is a sidebar titled "Explore Your Data" with a list of collections: Course, Enrollment, Student, customers, and orders, with a note that 1 collection is available.

Creating dataset on SyllabusGenerator_Department

Query:

Create dataset on SyllabusGenerator_Department

The screenshot shows the ADBMS Analytics interface in a Safari browser. The left sidebar has 'Analytics' selected. The main area has a 'Query Editor' tab open with the query: '1 create dataset on SyllabusGenerator_Department;'. Below it is a 'Query Results' section showing the response: '1 [{"data_not_cached": "Hit execute to rerun query"}]'. On the right, there's a sidebar titled 'Analytics Scopes, Links, & Collections' with sections for 'Default' and 'Local' collections like Course, Enrollment, Student, customers, orders, sphcustomers, and sphorders.

Creating dataset on SyllabusGenerator_Course

Query:

create dataset on SyllabusGenerator_Course

The screenshot shows the ADBMS Analytics interface in a Safari browser. The left sidebar has 'Analytics' selected. The main area has a 'Query Editor' tab open with the query: '1 create dataset on SyllabusGenerator_Course;'. Below it is a 'Query Results' section showing the response: '1 [{"success": true, "elapsed": 668.10ms, "execution": 649.87ms, "docs_scanned": 0}]'. On the right, there's a sidebar titled 'Analytics Scopes, Links, & Collections' with sections for 'Default' and 'Local' collections like Course, Enrollment, Student, SyllabusGenerator_Department, customers, orders, sphcustomers, and sphorders.

Creating dataset on:

Query

create dataset on SyllabusGenerator_CourseSchedule;

The screenshot shows the ADBMS Analytics interface running in a Safari browser. The title bar indicates the page is at 127.0.0.1. The main area has a blue header with the ADBMS logo and 'Analytics'. On the left, a sidebar menu is open under 'Analytics', showing options like 'Dashboard', 'Servers', 'Buckets', 'Backup', 'XDCR', 'Security', 'Settings', 'Logs', 'Documents', 'Query', 'Indexes', 'Search', 'Eventing', and 'Views'. The 'Analytics' option is currently selected. The central workspace is divided into two main sections: 'Query Editor' and 'Query Results'. In the 'Query Editor', there is a single line of code: '1 create dataset on SyllabusGenerator_CourseSchedule;'. Below the code, the status bar shows 'Execute' (highlighted in blue), 'Explain', '✓ success | elapsed: 539.12ms | execution: 526.94ms | docs scanned: 0', and a 'format' dropdown. The 'Query Results' section shows a table with one row and no data. It includes tabs for 'JSON', 'Table' (selected), 'Chart', 'Plan', and 'Plan Text'. To the right of the workspace is a sidebar titled 'Analytics Scopes, Links, & Collections'. It shows a tree structure with 'Default' selected, containing nodes for 'Local' (with 'Course', 'Enrollment', 'Student', 'SyllabusGenerator_CourseSchedule', 'SyllabusGenerator_Department', 'customers', 'orders', 'sphcustomers', and 'sphorders'), and '+ remote link'. At the bottom of the interface, a yellow warning message box says 'Warning: At least two servers with the data service are required to provide replication.' with a close button 'X'. A 'Refresh' button is located in the bottom right corner of the workspace.

Section 6.3

Screenshots of JSON Documents from CouchBase

Course Bucket

The screenshot shows the ADBMS interface for the 'Course Bucket'. The left sidebar has 'Documents' selected. The main area displays a table of course documents with columns 'id' and '_source'. There are four results listed:

	id	_source
	1	{"CourseCode": "IFT530", "CourseID": "1", "CourseName": "Advanced Database Management Systems", "CourseType": "In-Person", "DepartmentID": "1", "ProfessorID": "1", "Syllabus": [{"LearningObjective": "1.Unde..."}]}
	2	{"CourseCode": "IFT511", "CourseID": "2", "CourseName": "Analyzing Big Data", "CourseType": "In-Person", "DepartmentID": "1", "ProfessorID": "2", "Syllabus": [{"LearningObjective": "1.Understand how data s..."}]}
	3	{"CourseCode": "IFT430", "CourseID": "3", "CourseName": "Database Management System", "CourseType": "In-Person", "DepartmentID": "1", "ProfessorID": "1", "Syllabus": [{"LearningObjective": "1.basics of data..."}]}
	4	{"CourseCode": "IFT411", "CourseID": "4", "CourseName": "Fundamentals of Python", "CourseType": "In-Person", "DepartmentID": "1", "ProfessorID": "2", "Syllabus": [{"LearningObjective": "1.Understand the bas..."}]}

A warning message at the bottom left says: "Warning: At least two servers with the data service are required to provide replication."

Department Bucket

The screenshot shows the ADBMS interface for the 'Department Bucket'. The left sidebar has 'Documents' selected. The main area displays a table of department documents with columns 'id' and '_source'. There are two results listed:

	id	_source
	1	{"DepartmentCode": "IFT", "DepartmentID": "1", "DepartmentName": "Information Technology", "Professor": [{"ContactNumber": "6022221111", "EmailID": "RobertRucker@asu.edu", "Name": "Robert Rucker", "ProfessorID": "1..."}]}
	2	{"DepartmentCode": "CS", "DepartmentID": "2", "DepartmentName": "Computer Science", "Professor": [{"ContactNumber": "6022222222", "EmailID": "ChrisBryan@asu.edu", "Name": "Chris Bryan", "ProfessorID": "5"}, {"Contac..."}]}

A warning message at the bottom left says: "Warning: At least two servers with the data service are required to provide replication."

Course Schedule Bucket

The screenshot shows the ADBMS - Enterprise Edition 7.1.2 build 3454 interface. The top navigation bar includes Safari, File, Edit, View, History, Bookmarks, Window, Help, and a date/time indicator (Sat Dec 3 11:55 PM). The main header displays the title "ADBMS > Documents". On the right, there are buttons for "ADD DOCUMENT", "activity", "help", and "Administrator". Below the header, there's a search bar with "N1QL WHERE" and a "no indexes available..." message. The left sidebar has a "Documents" section selected, listing "Query", "Indexes", "Search", "Analytics", "Eventing", and "Views". Other sections like "Dashboard", "Servers", "Buckets", "Backup", "XDCR", "Security", "Settings", and "Logs" are also visible. The main content area shows a table with two results for the query "SyllabusGenerator_CourseSchedule._default._default, limit: 10, offset: 0". The table columns include "id" and some JSON data. A warning message at the bottom left says "Warning: At least two servers with the data service are required to provide replication.".

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 exemplar).",  
            "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 exemplar).",  
            "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"  
    }  
]  
}  
]
```

The screenshot shows the ADBMS Analytics interface. On the left, there's a sidebar with various navigation options like Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, Analytics (which is selected), Eventing, and Views. The main area has tabs for 'Query Editor' and 'Query Results'. In the 'Query Editor' tab, a query is being run:

```

1 FROM SyllabusGenerator_Course AS o, o.Syllabus AS i, SyllabusGenerator_CourseSchedule AS cs
2 where i.Semester = 'Fall' and cs.SyllabusID = i.SyllabusID
3 SELECT o.CourseName, o.CourseCode, i.Semester, i.LearningObjective, cs.Module

```

The 'Query Results' tab shows the output in JSON format:

```

1 [
2   {
3     "CourseName": "Advanced Database Management Systems",
4     "CourseCode": "IFT530",
5     "Semester": "Fall",
6     "LearningObjective": "1.Understand the principles of conceptual modeling,2.understand the use of scripts when implementing SQL code",
7     "Module": [
8       {
9         "CourseContent": "Understand the principles of conceptual modeling (ORM as an example).",
10        "EndDate": "30 Aug 2022",
11        "ModuleName": "Module1",
12        "StartDate": "15 Aug 2022"
13      },
14      {
15        "CourseContent": "Basis of Primary keys, Uniqueness/ Mandatory constraints",
16        "EndDate": "15 Sept 2022",
17        "ModuleName": "Module2",
18        "StartDate": "1 Sept 2022"
19      },
20      {
21        "CourseContent": "basis for determining Foreign Keys, value and frequency constraints, dynamic SQL",
22        "End": "Module3",
23      }
24    ]
25  ]

```

A warning message at the bottom left says: "Warning: At least two servers with the data service are required to provide replication." On the right side, there's a sidebar titled "Analytics Scopes, Links, & Collections" with a "Default" section and a "Local" section listing various collections like Course, Enrollment, Student, etc.

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

127.0.0.1

ADBMS > Analytics

Workbench Monitor

IMPORT EXPORT

Dashboard Servers Buckets Backup XDCR Security Settings Logs Documents Query Indexes Search Analytics Eventing Views

Query Editor

```
1 FROM SyllabusGenerator_Dept 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

query context

Query Results

```
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",
23    "CourseCode": "IFT411",
24    "DepartmentName": "Information Technology"
25  }
]
```

JSON Table Chart Plan Plan Text

Analytics Scopes, Links, & Collections

Map From Data Service

Default Local cb.local + collection

- Course
- Enrollment
- Student
- SyllabusGenerator_Course
- SyllabusGenerator_CourseSchedule
- SyllabusGenerator_Department
- customers
- orders
- sphcustomers
- sphorders

Hide empty analytics scopes

Refresh

Warning: At least two servers with the data service are required to provide replication.

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 constraints 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
- Once 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>