RollCall Solutions

- Vishnu Vardhan Reddy Kothagadi
- Lekhya Reddy Mudda
- Pruthvi Shyam Billa
- Amuktha Malyada Nadipelli

Date - 12/02/2022

DATA BACKGROUND AND DATA SOURCE

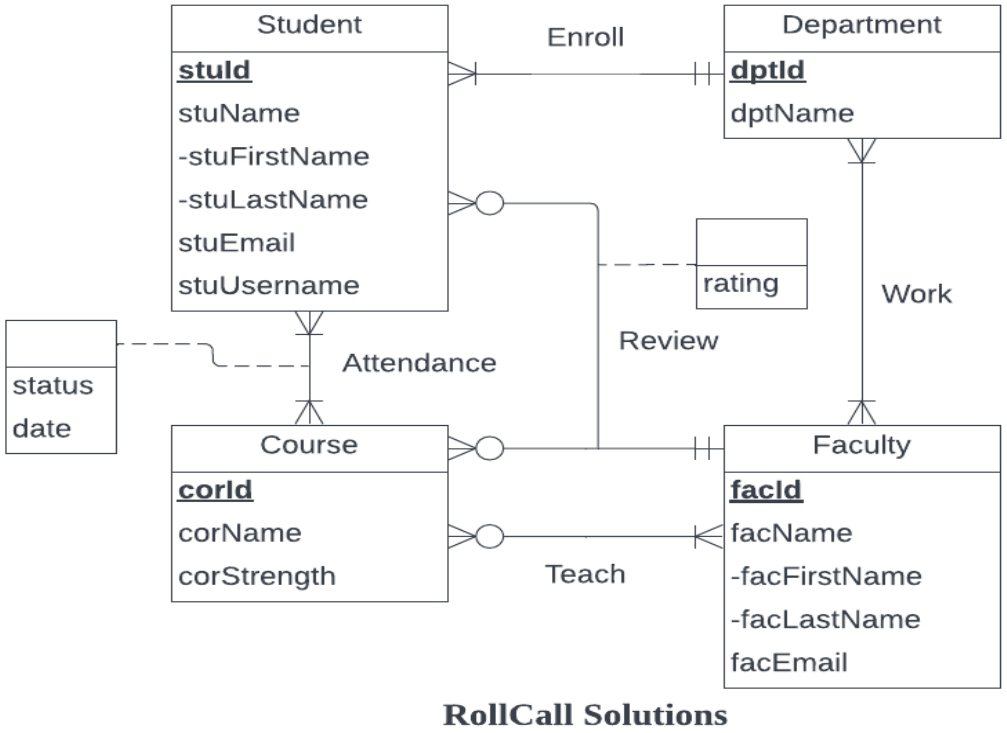
- Our data comprises of information pertaining to graduate students from specialty master's as well as faculty members at the Robert H. Smith School of Business.
- The student dataset has been extracted from the canvas website of the University of Maryland.
- This data is utilized to collect the attendance of students on a daily basis as well as store a particular faculty member's feedback for a given course.
- Data source link: https://umd.instructure.com/

MISSION STATEMENT & OBJECTIVES

To improve Master Program Office's database to develop the attendance system and provide transparent feedback for faculty from the students.

- To find the students who have attended at least one class for the registered courses but have least attendance in each department.
- To find the list of students present for the courses for the first day of the class.
- To find the average rating given by all students for each faculty.
- To find the department having the highest positive feedback given from their students.
- To find the average rating of courses offered based on department(eg Department Name = Information Systems).

ER Diagram



RollCall Solutions

RELATIONAL SCHEMA

Department (**dptId**, dptName)

Student (**stuId**, stuFirstName, stuLastName, stuEmail, stuUsername, *dptId*)

Course (**corId**, corName, corStrength)

Faculty (**facId**, facFirstName, facLastName, facEmail)

Work (***dptId***, ***facId***)

Teach (***facId***, ***corId***)

Attendance (***stuId***, ***corId***, status, atdDate)

Review (***stuId***, ***corId***, *facId*, rating)

SQL CREATE DEPARTMENT TABLE

```
CREATE TABLE [RollCall.Department] (

	dptId CHAR(10) NOT NULL,

	dptName VARCHAR(20),

	CONSTRAINT pk_Department_dptId PRIMARY KEY (dptId)

)
```

SQL CREATE STUDENT TABLE

```
CREATE TABLE [RollCall.Student] (
  stuId CHAR(10) NOT NULL,
  stuFirstName VARCHAR(20),
  stuLastName VARCHAR(20),
  stuEmail VARCHAR(20),
  stuUsername VARCHAR(20),
  dptId CHAR(10),
  CONSTRAINT pk_Student_stuId PRIMARY KEY (stuId),
  CONSTRAINT fk_Student_dptId FOREIGN KEY (dptId)
  REFERENCES [RollCall.Department] (dptId)
  ON DELETE CASCADE ON UPDATE CASCADE

 )
```

SQL CREATE COURSE TABLE

```
CREATE TABLE [RollCall.Course] (

        corId CHAR(10) NOT NULL,

        corName VARCHAR(70),

        corStrength INTEGER,

        CONSTRAINT pk_Course_corId PRIMARY KEY(corId)

)
```

SQL CREATE FACULTY TABLE

```
CREATE TABLE [RollCall.Faculty] (
        facId CHAR(10) NOT NULL,
        facFirstName VARCHAR(30),
        facLastName VARCHAR(30),
        facEmail VARCHAR(30),
        CONSTRAINT pk_Faculty_facId PRIMARY KEY(facId)
)
```

SQL CREATE WORK TABLE

```
CREATE TABLE [RollCall.Work](
   dptId CHAR (10) NOT NULL,
   facId CHAR (10) NOT NULL,
   CONSTRAINT pk_Work_dptId_facId PRIMARY KEY (dptId, facId),
   CONSTRAINT fk_Work_dptId FOREIGN KEY (dptId)
     REFERENCES [RollCall.Department] (dptId)
     ON DELETE NO ACTION ON UPDATE CASCADE,
   CONSTRAINT fk_Work_facId FOREIGN KEY (facId)
     REFERENCES [RollCall.Faculty] (facId)
     ON DELETE NO ACTION ON UPDATE CASCADE

)
```

SQL CREATE TECH TABLE

```
CREATE TABLE [RollCall.Teach] (
        facId CHAR(10) NOT NULL,
        corId CHAR(10) NOT NULL,
        CONSTRAINT pk_Teach_facId_corId PRIMARY KEY (facId, corId),
        CONSTRAINT fk_Teach_facId FOREIGN KEY (facId)
                REFERENCES [RollCall.Faculty] (facId)
                ON DELETE NO ACTION ON UPDATE CASCADE,
        CONSTRAINT fk_Teach_corId FOREIGN KEY (corId)
                REFERENCES [RollCall.Course] (corId)
                ON DELETE NO ACTION ON UPDATE CASCADE
)
```

SQL CREATE ATTENDANCE TABLE

```
CREATE TABLE [RollCall.Attendance] (
          stuId CHAR(10) NOT NULL,
          corId CHAR(10) NOT NULL,
          atdstatus VARCHAR(20),
          atddate DATE,
          CONSTRAINT pk_Attendance_stuId_corId PRIMARY KEY (stuId, corId),
          CONSTRAINT fk_Attendance_stuId FOREIGN KEY (stuId)
                    REFERENCES [RollCall.Student] (stuId)
                    ON DELETE NO ACTION ON UPDATE CASCADE,
          CONSTRAINT fk_Attendance_corId FOREIGN KEY (corId)
                    REFERENCES [RollCall.Course] (corId)
                    ON DELETE NO ACTION ON UPDATE CASCADE
)
```

SQL CREATE REVIEW TABLE

```
CREATE TABLE [RollCall.Review] (
          stuId CHAR (10) NOT NULL,
          corId CHAR (10) NOT NULL,
          facId CHAR (10) NOT NULL,
          rating INT,
          CONSTRAINT pk_Review_stuId_corId PRIMARY KEY (stuId, corId),
          CONSTRAINT fk_Review_stuId FOREIGN KEY (stuId)
                    REFERENCES [RollCall.Student] (stuId)
                    ON DELETE NO ACTION ON UPDATE CASCADE,
          CONSTRAINT fk_Review_corId FOREIGN KEY (corId)
                    REFERENCES [RollCall.Course] (corId)
                    ON DELETE NO ACTION ON UPDATE CASCADE,
          CONSTRAINT fk_Review_facId FOREIGN KEY (facId)
                    REFERENCES [RollCall.Faculty] (facId)
                    ON DELETE NO ACTION ON UPDATE CASCADE
)
```

Business Transaction 1

```
-- 1. To find the students who have attended atleast one class for the registered courses but have least attendance in each department.

SELECT ac.stuId AS 'Student Id',
        ac.[Student Name] AS 'Student Name',
        ac.attendanceCount AS 'Attendance Count',
        ac.dptName AS 'Department Name'
        FROM (SELECT s.stuId,
                    CONCAT(s.stuFirstName,' ',s.stuLastName) AS 'Student Name',
                    COUNT(a.atdstatus) AS attendanceCount,
                    d.dptId,
                    d.dptName,
                    DENSE_RANK() OVER (PARTITION BY d.dptName ORDER BY COUNT(a.atdstatus)) as atd_rnk
                FROM [RollCall.Student] s, [RollCall.Attendance] a, [RollCall.Department] d
                WHERE a.stuId = s.stuId AND s.dptId=d.dptId AND a.atdstatus = 'Present'
                GROUP BY s.stuId, s.stuFirstName, s.stuLastName, d.dptId, d.dptName
        )ac
WHERE ac.atd_rnk=1 -- rank is used to determine the students with least attendance
ORDER BY [Attendance Count]
```

Student Id	Student Name	Attendance Count	Department Name
112276379	Malik McDay	1	Business Analytics
119262030	Mohak Verma	1	Business Analytics
119340639	Lohith Maddula	1	Business Analytics
119419847	Mohit Buddha	1	Business Analytics
119071379	Michaela Bracken	1	Marketing Analytics
119597517	Anirudh Anandh	1	Marketing Analytics
119718278	Papitha Lakshmanan	1	Supply Chain
119406267	Rishikesh Baskaran	2	Information Systems

Business Transaction 2

```
-- 2. To find the list of students present for the courses for the first day of the class.

SELECT s.stuId AS 'Student Id',
       CONCAT(s.stuFirstName,' ',s.stuLastName) AS 'Student Name',
       s.stuEmail AS 'Student Email',
       s.stuUsername AS 'Student Username',
       s.dptId AS 'Department Id'
FROM [RollCall.Student] s
WHERE s.stuId IN (
              SELECT a.stuId
              FROM [RollCall.Attendance] a, (
                                    SELECT MIN(a.atddate) AS firstDay
                                    FROM [RollCall.Attendance] a) fd
              WHERE a.atddate = fd.firstDay)
GO
```

Student Id	Student Name	Student Email	Student Username	Department Id
112276379	Malik McDay	mmcday@umd.edu	mmcday	D000000002
116279379	Samuel Chapis	schapis@umd.edu	schapis	D000000005
117355284	Alexandra Lesia Dakhniuk	adakhniuk@gmail.com	adakhniuk	D000000002
117567557	Maliha Bukhari	mbukhar3@umd.edu	mbukhar3	D000000002
118267557	Janarthan Anuraag	januraag@umd.edu	januraag	D000000005
118572697	Oluwabukunmi Adubi	oadubi1@umd.edu	oadubi1	D000000001
118572819	Akhil Aenugu	aaenugu@umd.edu	aaenugu	D000000001
118602400	Aishwarya Sadagopan	aishsada@umd.edu	aishsada	D000000002
119071379	Michaela Bracken	mbracken@umd.edu	mbracken	D000000005
119075722	Srikar Alluri	salluri1@umd.edu	salluri1	D000000002
119132906	Manas Bhat	manasmb@umd.edu	manasmb	D000000001
119149676	Apurv Chauhan	apurv13@umd.edu	apurv13	D000000001
119205600	Yu Hsiang Cheng	ycheng88@umd.edu	ycheng88	D000000001
119251638	Vincent Brown	vjgbrown@umd.edu	vjgbrown	D000000001
119262030	Mohak Verma	mverma12@umd.edu	mverma12	D000000002
119268278	Sai Bobba	sbobba1@umd.edu	sbobba1	D000000001
119340639	Lohith Maddula	lohith88@umd.edu	lohith88	D000000002
119381311	Parichay Bajaj	pbajaj@umd.edu	pbajaj	D000000002
119406267	Rishikesh Baskaran	rishi01@umd.edu	rishi01	D000000001
119410134	Sai Varanasi	ssvaran9@umd.edu	svaran9	D000000002
119419847	Mohit Buddha	mbuddha@umd.edu	mbuddha	D000000002
119597517	Anirudh Anandh	aananth@umd.edu	aananth	D000000005
119718078	Akhib Ahmed	aakhib@umd.edu	aakhib	D000000006
119718278	Papitha Lakshmanan	lpapitha@umd.edu	lpapitha	D000000006

Business Transaction 3

```
-- 3. To find the average rating given by all students for each faculty.

SELECT r.facId AS 'Faculty Id',
        CONCAT(f.facFirstName,' ',f.facLastName) AS 'Faculty Name',
        AVG(r.rating) AS 'Average Rating'
FROM [RollCall.Review] r, [RollCall.Faculty] f
WHERE r.facId = f.facId
GROUP BY r.facId, f.facFirstName, f.facLastName, r.rating
ORDER BY r.facId ASC
GO
```

Faculty Id	Faculty Name	Average Rating
F000000001	Adam Lee	5
F000000002	Sujin Kim	4
F000000003	John Bono	5
F000000004	Tejwansh Anand	5
F000000006	Suresh Acharya	5
F000000007	Kunpeng Zhang	5
F000000008	Michel Wedel	5
F000000009	Amna Kirmani	5
F000000010	Judy Frels	5
F000000011	Martin Dresner	5

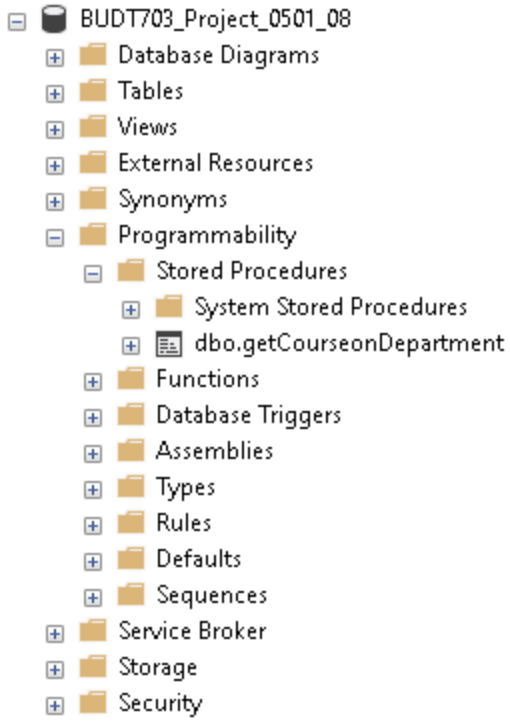
Business Transaction 4

```
-- 4. To find the department having the highest positive feedback given from their students.

SELECT d.dptId AS 'Department Id',
        d.dptName AS 'Department Name',
        AVG(r.rating) AS 'Average Rating'
FROM [RollCall.Department] d, [RollCall.Review] r, [RollCall.Student] s
WHERE s.dptId = d.dptId AND r.stuId = s.stuId
GROUP BY d.dptId, d.dptName
ORDER BY 'Average Rating' DESC
```

Department Id	Department Name	Average Rating
D000000002	Business Analytics	5
D000000005	Marketing Analytics	5
D000000006	Supply Chain	5
D000000001	Information Systems	4

Business Transaction 5




```sql
--5. Finding the average rating of courses offered based on department(eg Department Name = Information Systems).

CREATE OR ALTER PROCEDURE getCourseonDepartment
    @DepartmentName nvarchar(50)
AS

    SET NOCOUNT ON;
    SELECT ar.dptName AS 'Department Name',
           ar.corName AS 'Course Name',
           ar.avg_rating as 'Average Rating' FROM
           (SELECT d.dptName,
                   c.corName,
                   AVG(r.rating) as avg_rating,
                   DENSE_RANK() OVER (PARTITION BY d.dptName order by AVG(r.rating) DESC) as course_rnk
           FROM [RollCall.Department] d, [RollCall.Review] r, [RollCall.Student] s, [RollCall.Course] c
           WHERE s.dptId = d.dptId AND r.stuId = s.stuId and r.corId = c.corId
           GROUP BY d.dptName, c.corName)ar
    WHERE ar.dptName = @DepartmentName
    ORDER BY ar.avg_rating DESC

EXECUTE getCourseonDepartment @DepartmentName = N'Information Systems';
```

Department Name	Course Name	Average Rating
Information Systems	Data Processing And Analytics in Python	5
Information Systems	Database Management Systems	5
Information Systems	Digital Business Transformation	5
Information Systems	Data Models And Decisions	4

Thank You

UNIVERSITY OF MARYLAND