```sql
-- ============================================================
-- Author: Jesse Lecathelinais
-- Description: createDB.sql
-- Creates the database for UniversityX
-- ============================================================

--Drops all tables so they can be created again
DROP TABLE IF EXISTS ProgramEnrolment
DROP TABLE IF EXISTS StudentEnrolment
DROP TABLE IF EXISTS StudentTimetableSlot
DROP TABLE IF EXISTS TimetableSlot
DROP TABLE IF EXISTS ReasonType
DROP TABLE IF EXISTS PhysicalOffering
DROP TABLE IF EXISTS StudentCourseOffering
DROP TABLE IF EXISTS CourseOffering
DROP TABLE IF EXISTS Period
DROP TABLE IF EXISTS Facility
DROP TABLE IF EXISTS FacilityType
DROP TABLE IF EXISTS PhysicalCampus
DROP TABLE IF EXISTS Campus
DROP TABLE IF EXISTS AssignmentMajor
DROP TABLE IF EXISTS ProgramMajorMinor
DROP TABLE IF EXISTS ProgramStaff
DROP TABLE IF EXISTS OrganisationStaff
DROP TABLE IF EXISTS AcademicStaff
DROP TABLE IF EXISTS Person
DROP TABLE IF EXISTS ProgramCourse
DROP TABLE IF EXISTS Program
DROP TABLE IF EXISTS AssumedKnowledge
DROP TABLE IF EXISTS Course
DROP TABLE IF EXISTS MajorMinor
DROP TABLE IF EXISTS AssignmentType
DROP TABLE IF EXISTS SubOrganisationUnit
DROP TABLE IF EXISTS OrganisationUnit
go

    ---     Organisation Unit Data      ---

CREATE TABLE OrganisationUnit(
    unitCode    VARCHAR(8)      PRIMARY KEY,    --Code to identify the organisation unit
    unitName    VARCHAR(50)     NOT NULL,       --Name of the organisation unit
    description VARCHAR(200)    NOT NULL,       --Description of the organisation unit
    contactNo   VARCHAR(10)     NOT NULL,       --The contact number of the organisation unit
    UNIQUE(unitName),
);

INSERT INTO OrganisationUnit VALUES ('OU000001', 'Academic Division', 'A division for academics', '11111111');
INSERT INTO OrganisationUnit VALUES ('OU000002', 'Research Division', 'A division for research', '10101010');
INSERT INTO OrganisationUnit VALUES ('OU000003', 'College of Science and Engineering', 'Learn about the sciences and engineering in this college',
```

```
    '10293847');
50  INSERT INTO OrganisationUnit VALUES ('OU000004', 'College of Business and
    Law', 'Learn about the business and law in this college', '10010010');
51  INSERT INTO OrganisationUnit VALUES ('OU000005', 'School of Engineering',
    'Learn about engineering in this school', '1029384756');
52  INSERT INTO OrganisationUnit VALUES ('OU000006', 'School of Business', 'Learn
    about business in this school', '1001001010');
53  INSERT INTO OrganisationUnit VALUES ('OU000007', 'School of Science', 'Learn
    about science in this school', '1029384712');
54  INSERT INTO OrganisationUnit VALUES ('OU000008', 'School of Mathematics',
    'Learn about math', '66655511');
55  go
56
57      ---     Sub-Organisation Unit Data      ---
58
59  CREATE TABLE SubOrganisationUnit(
60      oUnit       VARCHAR(8)  NOT NULL,       --The main organisation unit
61      subOUnit    VARCHAR(8)  NOT NULL,       --The sub organisation unit of
        oUnit
62      PRIMARY KEY(oUnit, subOUnit),
63      CHECK(oUnit != subOUnit),           --Make sure that an organisation unit
        can't be its own sub orgnisation unit
64      FOREIGN KEY(oUnit) REFERENCES OrganisationUnit(unitCode)
65          ON UPDATE NO ACTION ON DELETE NO ACTION,
66      FOREIGN KEY(subOUnit) REFERENCES OrganisationUnit(unitCode)
67          ON UPDATE NO ACTION ON DELETE NO ACTION
68  );
69
70  INSERT INTO SubOrganisationUnit VALUES ('OU000003', 'OU000005');
71  INSERT INTO SubOrganisationUnit VALUES ('OU000004', 'OU000006');
72  INSERT INTO SubOrganisationUnit VALUES ('OU000003', 'OU000007');
73  go
74
75      ---     Type of assignment Data     ---
76
77  CREATE TABLE AssignmentType(
78      typeID  INT         PRIMARY KEY,    --Unique identifier
79      name    VARCHAR(50) NOT NULL,       --The name of the type (Major/Minor)
80      UNIQUE(name)
81  );
82
83  INSERT INTO AssignmentType VALUES (1, 'Directed');
84  INSERT INTO AssignmentType VALUES (2, 'Compulsory');
85  INSERT INTO AssignmentType VALUES (3, 'Other');
86  go
87
88      ---     Major/Minor Data        ---
89
90  CREATE TABLE MajorMinor(
91      mCode           VARCHAR(8)      PRIMARY KEY,    --Unique code for the
        major/minor
92      name            VARCHAR(50)     NOT NULL,       --Name for the major/minor
93      description     VARCHAR(200)    NOT NULL,       --Description for this
        major/minor
94      totalCredits    INT             NOT NULL,       --Total credits required
        to complete this major/minor
```

```
95      conditions      VARCHAR(200)    NOT NULL,       --The conditions that need
           to be met to complete this major/minor
96      isMajor         BIT             NOT NULL,       --Determines whether it's
          a major or a minor (1 if Major, 0 if Minor)
97      UNIQUE(name)
98  );
99
100 INSERT INTO MajorMinor VALUES ('M1', 'Pure Mathematics', 'Mathematics that is
      pure', 120, 'Be really good at maths', 1);
101 INSERT INTO MajorMinor VALUES ('m2', 'Impure Mathematics', 'Mathematics that
      is impure', 60, 'Be alright at maths', 0);
102 INSERT INTO MajorMinor VALUES ('M3', 'Applied Mathematics', 'Mathematics that
      is to be applied', 120, 'Apply maths somewhere', 1);
103 INSERT INTO MajorMinor VALUES ('M4', 'Data Science', 'The science behind the
      data', 160, 'Know what SQL is', 1);
104 go
105
106     ---     Course Data     ---
107
108 CREATE TABLE Course(
109     courseID        VARCHAR(9)      PRIMARY KEY,    --The identifier for the
          course
110     name            VARCHAR(50)     NOT NULL,       --The name of the course
111     numberCredits   INT             NOT NULL,       --The number of credits
          assigned for the course
112     description     VARCHAR(200)    NOT NULL,       --description of the
          course
113     UNIQUE(name)
114 );
115
116 INSERT INTO Course VALUES ('COMP3350', 'Advanced Database', 10, 'Continuation
      of COMP1140 with more stuff');
117 INSERT INTO Course VALUES ('COMP1140', 'Database and Information Management',
      10, 'Learn SQL and databases');
118 INSERT INTO Course VALUES ('MATH1510', 'Discrete Mathematics', 10,
      'Mathematics that is discrete');
119 INSERT INTO Course VALUES ('COMP3851A', 'CS and IT WIL Part A', 10, 'Off to
      work integrated learning!');
120 INSERT INTO Course VALUES ('COMP3851B', 'CS and IT WIL Part B', 10, 'Off to
      work integrated learning again!');
121 INSERT INTO Course VALUES ('SENG1110', 'Object-Oriented Programming', 10,
      'Learn Java');
122 INSERT INTO Course VALUES ('SENG1120', 'Data Structures', 10, 'Learn C++');
123 INSERT INTO Course VALUES ('MATH3820', 'Numerical Methods', 10, 'Learn
      Interpolation and other methods');
124 INSERT INTO Course VALUES ('MATH3120', 'Algebra', 10, 'Learn algebra');
125 INSERT INTO Course VALUES ('MATH2310', 'Calculus of Science and Engineering',
      10, 'Learn the calculus of the sciences');
126 INSERT INTO Course VALUES ('MATH1210', 'Mathematical Discovery 1', 10,
      'Discover math');
127 INSERT INTO Course VALUES ('MATH1220', 'Mathematical Discovery 2', 10,
      'Discover math again');
128 go
129
130     ---     Assumed Knowledge for Courses Data      ---
131
```

```
132 CREATE TABLE AssumedKnowledge(
133     course          VARCHAR(9)  NOT NULL,   --The main course
134     assumedCourse   VARCHAR(9)  NOT NULL,   --The assumed knowledge of the
        main course
135     isPrerequisite  BIT         NOT NULL,   --Flag that determines if
        assumedCourse has to be taken in order for the main course to be taken
136     PRIMARY KEY(course, assumedCourse),
137     CHECK(course != assumedCourse),         --Make sure that an organisation
        unit can't be its own sub orgnisation unit
138     FOREIGN KEY(course) REFERENCES Course(courseID)
139         ON UPDATE NO ACTION ON DELETE NO ACTION,
140     FOREIGN KEY(assumedCourse) REFERENCES Course(courseID)
141         ON UPDATE NO ACTION ON DELETE NO ACTION
142 );
143
144 INSERT INTO AssumedKnowledge VALUES ('COMP3350', 'COMP1140', 1);
145 INSERT INTO AssumedKnowledge VALUES ('COMP3350', 'SENG1110', 1);
146 INSERT INTO AssumedKnowledge VALUES ('COMP3851B', 'COMP3851A', 0);
147 INSERT INTO AssumedKnowledge VALUES ('MATH3820', 'MATH2310', 0);
148 INSERT INTO AssumedKnowledge VALUES ('MATH2310', 'MATH1220', 1);
149 INSERT INTO AssumedKnowledge VALUES ('MATH1220', 'MATH1210', 1);
150 INSERT INTO AssumedKnowledge VALUES ('SENG1120', 'SENG1110', 0);
151 go
152
153     ---     Program Data        ---
154
155 CREATE TABLE Program(
156     progCode        INT         PRIMARY KEY,    --Code for the program
157     name            VARCHAR(50) NOT NULL,       --Name of the program
158     oUnit           VARCHAR(8)  NOT NULL,       --The organisation unit that
        this program belongs to
159     totalCredits    INT         NOT NULL,       --Total credits required to
        complete this program
160     level           VARCHAR(20) NOT NULL,       --Level of the program (ie
        Bachelor)
161     certAchieved    VARCHAR(10) NOT NULL,       --Certification achieved once
        the program is completed (ie BSc)
162     UNIQUE(name),
163     FOREIGN KEY(oUnit) REFERENCES OrganisationUnit(unitCode)
164         ON UPDATE NO ACTION ON DELETE NO ACTION
165 );
166
167 INSERT INTO Program VALUES (10237, 'Bachelor of Mathematics', 'OU000008', 240,
    'Bachelor', 'BMath');
168 INSERT INTO Program VALUES (40103, 'Bachelor of Computer Science', 'OU000003',
    240, 'Bachelor', 'BCompSc');
169 INSERT INTO Program VALUES (11497, 'Bachelor of Information Technology',
    'OU000007', 240, 'Bachelor', 'BIT');
170 INSERT INTO Program VALUES (40177, 'Master of Information Technology',
    'OU000007', 120, 'Masters', 'BIT');
171 INSERT INTO Program VALUES (60238, 'PhD (Mathematics)', 'OU000008', 120,
    'PhD', 'PhD');
172 go
173
174     ---     Program Course Data     ---
175
```

```
176 CREATE TABLE ProgramCourse(
177     program     INT,                    --Code of the program
178     course      VARCHAR(9),             --Course that is featured in the
          program
179     startDate   DATE        NOT NULL,   --StartDate
180     endDate     DATE,
181     isCore      BIT         NOT NULL,
182     PRIMARY KEY(program, course, startDate),
183     CHECK(endDate > startDate),
184     FOREIGN KEY(program) REFERENCES Program(progCode)
185         ON UPDATE NO ACTION ON DELETE NO ACTION,
186     FOREIGN KEY(course) REFERENCES Course(courseID)
187         ON UPDATE NO ACTION ON DELETE NO ACTION
188 );
189
190 INSERT INTO ProgramCourse VALUES (10237, 'MATH1210', '2022-02-01', NULL, 1);
191 INSERT INTO ProgramCourse VALUES (10237, 'MATH1220', '2022-02-01', NULL, 1);
192 INSERT INTO ProgramCourse VALUES (10237, 'MATH2310', '2022-02-01', NULL, 0);
193 INSERT INTO ProgramCourse VALUES (10237, 'MATH1510', '2022-02-01', NULL, 0);
194 INSERT INTO ProgramCourse VALUES (10237, 'MATH3120', '2022-02-01', NULL, 1);
195 INSERT INTO ProgramCourse VALUES (10237, 'MATH3820', '2022-02-01', NULL, 1);
196 INSERT INTO ProgramCourse VALUES (60238, 'MATH3820', '2021-02-01',
      '2021-12-01', 1);
197 INSERT INTO ProgramCourse VALUES (60238, 'MATH3120', '2021-02-01',
      '2021-12-01', 1);
198 INSERT INTO ProgramCourse VALUES (40103, 'COMP3350', '2022-02-01', NULL, 0);
199 INSERT INTO ProgramCourse VALUES (40103, 'COMP1140', '2022-02-01', NULL, 1);
200 INSERT INTO ProgramCourse VALUES (40103, 'SENG1110', '2022-02-01', NULL, 1);
201 INSERT INTO ProgramCourse VALUES (40103, 'SENG1120', '2022-02-01', NULL, 1);
202 INSERT INTO ProgramCourse VALUES (40103, 'COMP3851A', '2022-02-01', NULL, 1);
203 INSERT INTO ProgramCourse VALUES (40103, 'COMP3851B', '2022-02-01', NULL, 1);
204 INSERT INTO ProgramCourse VALUES (11497, 'COMP3851A', '2022-02-01', NULL, 1);
205 INSERT INTO ProgramCourse VALUES (11497, 'COMP3851B', '2022-02-01', NULL, 1);
206 go
207
208     ---     Person Data     ---
209
210 CREATE TABLE Person(
211     personID        INT             PRIMARY KEY,    --Unique identifier for
          person
212     name            VARCHAR(100)    NOT NULL,       --The person's name
213     isStaff         BIT             NOT NULL,       --Flag to determine if the
          person is a staff member
214     isStudent       BIT             NOT NULL,       --Flag to determine if the
          person is a student
215     streetNum       VARCHAR(5)      NOT NULL,       --Street number of the
          address
216     street          VARCHAR(50)     NOT NULL,       --Street of the address
217     city            VARCHAR(50)     NOT NULL,       --City of the address
218     postcode        VARCHAR(5)      NOT NULL,       --Postcode of the address
219     contactNo       VARCHAR(10)     NOT NULL,       --Personal contact number
          of the person
220     staffContact    VARCHAR(10),                    --Staff contact number of
          the person (If the person is a staff member)
221     CHECK(isStaff = 1 OR isStudent = 1),                                --
          Ensures that at least one of the flags is true
```

```
222     CHECK((isStaff = 0 AND staffContact IS NULL)            --Ensures that a
          non staff member doesn't have a staff contact number
223         OR (isStaff = 1 AND staffContact IS NOT NULL)),     --Ensures that a
            staff member has a staff contact number
224     UNIQUE(personID, isStaff),
225     UNIQUE(personID, isStudent)
226 );
227
228 INSERT INTO Person VALUES (1, 'Jesse Lecathelinais', 0, 1, '25', 'Sesame St.',
      'Newcastle', '2300', '2724228740', NULL);
229 INSERT INTO Person VALUES (2, 'Nathan Murphy', 0, 1, '123', 'Fake St.',
      'Billton', '1234', '9988776655', NULL);
230 INSERT INTO Person VALUES (3, 'Mitch Black', 0, 1, '37', 'Nelson St.',
      'Sydney', '8236', '3764283719', NULL);
231 INSERT INTO Person VALUES (4, 'Rukshan Athauda', 1, 0, '2', 'Apple St.',
      'Newcastle', '2300', '1234568712', '12345687');
232 INSERT INTO Person VALUES (5, 'Billy Joe', 1, 1, '5', 'Five St.', 'Jesmond',
      '2299', '0987654321', '09876543');
233 INSERT INTO Person VALUES (6, 'Joey Bill', 1, 1, '5', 'Six St.', 'Wallsend',
      '2287', '0487645321', '21984365');
234 go
235
236     ---     Academic Staff Data     ---
237
238 CREATE TABLE AcademicStaff(
239     staff   INT PRIMARY KEY,            --ID of the staff member
240     isStaff BIT CHECK(isStaff = 1),     --Flag to make sure that the staff
          member is a staff member
241     FOREIGN KEY(staff, isStaff) REFERENCES Person(personID, isStaff)
242         ON UPDATE CASCADE ON DELETE NO ACTION
243 );
244
245 INSERT INTO AcademicStaff VALUES (4, 1);
246 INSERT INTO AcademicStaff VALUES (6, 1);
247 go
248
249     ---     Organisation Unit/Staff Data        ---
250
251 CREATE TABLE OrganisationStaff(
252     oUnit       VARCHAR(8),                     --Organisation unit that the
          staff is working for
253     staff       INT,                            --Staff member ID
254     isStaff     BIT         CHECK(isStaff = 1), --Flag to make sure that the
          person referenced is a staff member
255     startDate   DATE        NOT NULL,           --Start date for the staff
          member at the organisation unit
256     endDate     DATE,                           --End date for the staff
          member at the organisation unit
257     role        VARCHAR(50) NOT NULL,           --The role of the staff member
           at the organisation unit
258     PRIMARY KEY(oUnit, staff, startDate),
259     CHECK(endDate > startDate),
260     FOREIGN KEY(oUnit) REFERENCES OrganisationUnit(unitCode)
261         ON UPDATE CASCADE ON DELETE NO ACTION,
262     FOREIGN KEY(staff, isStaff) REFERENCES Person(personID, isStaff)
263         ON UPDATE CASCADE ON DELETE NO ACTION
```

```
264 );
265
266 INSERT INTO OrganisationStaff VALUES ('OU000001', 4, 1, '2022-02-01', NULL,
      'Lecturer');
267 INSERT INTO OrganisationStaff VALUES ('OU000003', 4, 1, '2021-02-01',
      '2021-12-31', 'Lecturer');
268 INSERT INTO OrganisationStaff VALUES ('OU000002', 5, 1, '2018-02-01',
      '2019-02-01', 'Lecturer');
269 INSERT INTO OrganisationStaff VALUES ('OU000002', 5, 1, '2019-02-01',
      '2020-02-01', 'PVC');
270 INSERT INTO OrganisationStaff VALUES ('OU000007', 5, 1, '2020-02-01', NULL,
      'Tutor');
271 INSERT INTO OrganisationStaff VALUES ('OU000007', 6, 1, '2020-02-01',
      '2022-02-01', 'Lecturer');
272 INSERT INTO OrganisationStaff VALUES ('OU000001', 6, 1, '2022-02-01', NULL,
      'Lecturer');
273 go
274
275     ---     Program Staff Data      ---
276
277 CREATE TABLE ProgramStaff(
278     program     INT,                --Code of the program
279     convenor    INT,                --Course that is featured in the program
280     startDate   DATE    NOT NULL,   --Start date the staff member was convenor
        for the program
281     endDate     DATE,               --End date the staff member was convenor
        for the program
282     PRIMARY KEY(program, convenor, startDate),
283     CHECK(endDate > startDate),
284     FOREIGN KEY(program) REFERENCES Program(progCode)
285         ON UPDATE NO ACTION ON DELETE NO ACTION,
286     FOREIGN KEY(convenor) REFERENCES AcademicStaff(staff)
287         ON UPDATE NO ACTION ON DELETE NO ACTION
288 );
289
290 INSERT INTO ProgramStaff VALUES (10237, 4, '2022-02-01', NULL);
291 INSERT INTO ProgramStaff VALUES (10237, 6, '2021-02-01', '2022-02-01');
292 INSERT INTO ProgramStaff VALUES (11497, 6, '2021-02-01', '2022-02-01');
293 INSERT INTO ProgramStaff VALUES (60238, 6, '2021-02-01', NULL);
294 INSERT INTO ProgramStaff VALUES (40177, 6, '2021-02-01', NULL);
295 INSERT INTO ProgramStaff VALUES (40103, 4, '2021-02-01', NULL);
296 go
297
298     ---     Program Major Minor Data        ---
299
300 CREATE TABLE ProgramMajorMinor(
301     program     INT,            --Code of the program
302     majorMinor  VARCHAR(8),     --Code of the major that associates with the
        program
303     PRIMARY KEY(program, majorMinor),
304     FOREIGN KEY(program) REFERENCES Program(progCode)
305         ON UPDATE NO ACTION ON DELETE NO ACTION,
306     FOREIGN KEY(majorMinor) REFERENCES MajorMinor(mCode)
307         ON UPDATE NO ACTION ON DELETE NO ACTION
308 );
309
```

```
310 INSERT INTO ProgramMajorMinor VALUES (10237, 'M1');
311 INSERT INTO ProgramMajorMinor VALUES (10237, 'm2');
312 INSERT INTO ProgramMajorMinor VALUES (10237, 'M3');
313 INSERT INTO ProgramMajorMinor VALUES (40103, 'M4');
314 INSERT INTO ProgramMajorMinor VALUES (11497, 'M4');
315 go
316
317     ---     Assignment Major Data       ---
318
319 CREATE TABLE AssignmentMajor(
320     assignID    INT         PRIMARY KEY,    --Code of the program
321     majorMinor  VARCHAR(8)  NOT NULL,       --Code of the major that
        associates with the program
322     course      VARCHAR(9)  NOT NULL,       --course that is assigned to this
        major
323     type        INT         NOT NULL,       --type of assignment (eg Directed)
324     startDate   DATE        NOT NULL,       --start date the course was
        assigned to this major
325     endDate     DATE,                       --end date the course was assigned
         to this major
326     FOREIGN KEY(majorMinor) REFERENCES MajorMinor(mCode)
327         ON UPDATE NO ACTION ON DELETE NO ACTION,
328     FOREIGN KEY(course) REFERENCES Course(courseID)
329         ON UPDATE NO ACTION ON DELETE NO ACTION,
330     FOREIGN KEY(type) REFERENCES AssignmentType(typeID)
331         ON UPDATE NO ACTION ON DELETE NO ACTION
332 );
333
334 INSERT INTO AssignmentMajor VALUES (1, 'M1', 'MATH2310', 3, '1999-01-01',
      NULL);
335 INSERT INTO AssignmentMajor VALUES (2, 'M1', 'MATH3820', 1, '1999-01-01',
      NULL);
336 INSERT INTO AssignmentMajor VALUES (3, 'M1', 'MATH3120', 1, '1999-01-01',
      NULL);
337 INSERT INTO AssignmentMajor VALUES (4, 'm2', 'MATH3820', 1, '2003-01-01',
      NULL);
338 INSERT INTO AssignmentMajor VALUES (5, 'm2', 'MATH1510', 2, '2003-01-01',
      '2005-02-02');
339 INSERT INTO AssignmentMajor VALUES (6, 'm2', 'MATH1210', 2, '2001-01-01',
      NULL);
340 INSERT INTO AssignmentMajor VALUES (7, 'm2', 'MATH1220', 2, '2003-01-01',
      NULL);
341 INSERT INTO AssignmentMajor VALUES (8, 'm2', 'MATH3820', 1, '2005-01-01',
      NULL);
342 INSERT INTO AssignmentMajor VALUES (9, 'M3', 'MATH3820', 1, '2003-01-01',
      NULL);
343 INSERT INTO AssignmentMajor VALUES (10, 'M3', 'MATH3120', 1, '2008-01-01',
      NULL);
344 INSERT INTO AssignmentMajor VALUES (11, 'M4', 'COMP3350', 2, '1989-01-01',
      NULL);
345 INSERT INTO AssignmentMajor VALUES (12, 'M4', 'MATH1220', 2, '1979-01-01',
      NULL);
346 INSERT INTO AssignmentMajor VALUES (13, 'M4', 'COMP3851A', 2, '2018-01-01',
      NULL);
347 INSERT INTO AssignmentMajor VALUES (14, 'M4', 'COMP3851B', 2, '2018-01-01',
      NULL);
```

```
348 go
349
350     ---     Campus Data     ---
351
352 CREATE TABLE Campus(
353     campID      INT         PRIMARY KEY,    --ID for the campus
354     name        VARCHAR(50) NOT NULL,       --Name of the campusID
355 );
356
357 INSERT INTO Campus VALUES (1, 'UniversityX, Callaghan Campus');
358 INSERT INTO Campus VALUES (2, 'UniversityX, Newcastle City Campus');
359 INSERT INTO Campus VALUES (3, 'UniversityX, Ourimbah Campus');
360 INSERT INTO Campus VALUES (4, 'UniversityX, Online Campus');
361 INSERT INTO Campus VALUES (5, 'UniversityX, Covid Campus');
362 INSERT INTO Campus VALUES (6, 'UniversityX, American Campus');
363 go
364
365     ---     Physical Campus Data        ---
366
367 CREATE TABLE PhysicalCampus(
368     campID      INT         PRIMARY KEY,    --ID for the campus
369     city        VARCHAR(50) NOT NULL,       --City that the campus is located ↵
        in
370     country     VARCHAR(50) NOT NULL,       --Country that the campus is      ↵
        located in
371     FOREIGN KEY(campID) REFERENCES Campus(campID)
372         ON UPDATE NO ACTION ON DELETE NO ACTION
373 );
374
375 INSERT INTO PhysicalCampus VALUES (1, 'Callaghan', 'Australia');
376 INSERT INTO PhysicalCampus VALUES (2, 'Newcastle', 'Australia');
377 INSERT INTO PhysicalCampus VALUES (3, 'Ourimbah', 'Australia');
378 INSERT INTO PhysicalCampus VALUES (6, 'Los Angeles', 'USA');
379 go
380
381     ---     Type of Facility Data       ---
382
383 CREATE TABLE FacilityType(
384     typeID  INT         PRIMARY KEY,    --Unique identifier
385     name    VARCHAR(50) NOT NULL,       --The name of the type (Facility)
386     UNIQUE(name)
387 );
388
389 INSERT INTO FacilityType VALUES (1, 'Room');
390 INSERT INTO FacilityType VALUES (2, 'Classroom');
391 INSERT INTO FacilityType VALUES (3, 'Computer Lab');
392 go
393
394     ---     Facility Data       ---
395
396 CREATE TABLE Facility(
397     facID           INT         PRIMARY KEY,    --ID for the facility
398     campus          INT         NOT NULL,       --Campus that the facility     ↵
        resides in
399     roomNumber      INT         NOT NULL,       --room number of the facility
400     buildingName    VARCHAR(50) NOT NULL,       --name of the building the     ↵
```

```
          facility is in
401       capacity          INT          NOT NULL,       --capacity for the facility
402       type              INT          NOT NULL,       --type of facility
403       UNIQUE(campus, roomNumber, buildingName),
404       FOREIGN KEY(campus) REFERENCES PhysicalCampus(campID)
405           ON UPDATE NO ACTION ON DELETE NO ACTION,
406       FOREIGN KEY(type) REFERENCES FacilityType(typeID)
407           ON UPDATE NO ACTION ON DELETE NO ACTION
408   );
409
410   INSERT INTO Facility VALUES (1, 1, 001, 'Social Sciences', 50, 2);
411   INSERT INTO Facility VALUES (2, 1, 002, 'General Purpose', 30, 2);
412   INSERT INTO Facility VALUES (3, 1, 209, 'Engineering Science', 70, 3);
413   INSERT INTO Facility VALUES (4, 1, 205, 'Mathematics', 50, 2);
414   INSERT INTO Facility VALUES (5, 1, 306, 'CT Building', 30, 3);
415   INSERT INTO Facility VALUES (6, 1, 013, 'Physics', 20, 3);
416   INSERT INTO Facility VALUES (7, 1, 201, 'General Purpose', 100, 2);
417   INSERT INTO Facility VALUES (8, 1, 107, 'Mathematics', 100, 2);
418   INSERT INTO Facility VALUES (9, 2, 201, 'CT Building', 100, 1);
419   INSERT INTO Facility VALUES (10, 3, 201, 'CT Building', 70, 1);
420   INSERT INTO Facility VALUES (11, 6, 407, 'USA Building', 407, 3);
421   INSERT INTO Facility VALUES (12, 6, 704, 'USA Building', 70, 3);
422   go
423
424       ---     Period Data     ---
425
426   CREATE TABLE Period(
427       periodID    INT         PRIMARY KEY,    --ID for the facility
428       name        VARCHAR(10) NOT NULL,       --Name of period
429       startDate   DATE        NOT NULL,       --start date of period
430       endDate     DATE,                       --end date of period
431       year        VARCHAR(5)  NOT NULL,       --year of the semester
432       CHECK(name = 'Semester' OR name = 'Trimester')
433   );
434
435   INSERT INTO Period VALUES (1, 'Semester', '2020-01-01', '2020-06-30', '2020');
436   INSERT INTO Period VALUES (2, 'Semester', '2020-07-01', '2020-12-31', '2020');
437   INSERT INTO Period VALUES (3, 'Semester', '2021-07-01', '2021-12-31', '2021');
438   INSERT INTO Period VALUES (4, 'Semester', '2021-07-01', '2021-12-31', '2021');
439   INSERT INTO Period VALUES (5, 'Semester', '2022-07-01', '2022-12-31', '2022');
440   INSERT INTO Period VALUES (6, 'Trimester', '2020-01-01', '2020-04-30',
        '2020');
441   INSERT INTO Period VALUES (7, 'Trimester', '2020-05-01', '2020-08-31',
        '2020');
442   INSERT INTO Period VALUES (8, 'Trimester', '2020-09-01', '2020-12-31',
        '2020');
443   INSERT INTO Period VALUES (9, 'Trimester', '2021-01-01', '2021-04-30',
        '2020');
444   INSERT INTO Period VALUES (10, 'Trimester', '2021-05-01', '2021-08-31',
        '2020');
445   INSERT INTO Period VALUES (11, 'Trimester', '2021-09-01', '2021-12-31',
        '2020');
446   INSERT INTO Period VALUES (12, 'Trimester', '2022-01-01', '2022-04-30',
        '2020');
447   go
448
```

```
449     ---     Course Offering Data        ---
450
451 CREATE TABLE CourseOffering(
452     offeringID  INT         PRIMARY KEY,    --ID for the course offering
453     course      VARCHAR(9)  NOT NULL,       --Course that is being offered
454     coordinator INT         NOT NULL,       --Coordinator of the course
        offering
455     isStaff     BIT CHECK(isStaff = 1),     --Flag that indicates that the
        coordinator is a staff member
456     campus      INT         NOT NULL,       --Campus that the course offering
        is being held
457     period      INT         NOT NULL,       --Period that the course offering
        is being held
458     FOREIGN KEY(course) REFERENCES Course(courseID)
459         ON UPDATE NO ACTION ON DELETE NO ACTION,
460     FOREIGN KEY(coordinator, isStaff) REFERENCES Person(personID, isStaff)
461         ON UPDATE NO ACTION ON DELETE NO ACTION,
462     FOREIGN KEY(campus) REFERENCES Campus(campID)
463         ON UPDATE NO ACTION ON DELETE NO ACTION,
464     FOREIGN KEY(period) REFERENCES Period(periodID)
465         ON UPDATE NO ACTION ON DELETE NO ACTION
466 );
467
468 INSERT INTO CourseOffering VALUES (1, 'COMP1140', 4, 1, 1, 1);
469 INSERT INTO CourseOffering VALUES (2, 'COMP3350', 4, 2, 1, 2);
470 INSERT INTO CourseOffering VALUES (3, 'COMP3851A', 4, 1, 1, 3);
471 INSERT INTO CourseOffering VALUES (4, 'COMP3851B', 6, 1, 1, 4);
472 INSERT INTO CourseOffering VALUES (5, 'MATH1210', 4, 6, 1, 5);
473 INSERT INTO CourseOffering VALUES (6, 'MATH1220', 5, 1, 1, 6);
474 INSERT INTO CourseOffering VALUES (7, 'MATH1510', 6, 5, 1, 7);
475 INSERT INTO CourseOffering VALUES (8, 'MATH2310', 6, 4, 1, 8);
476 INSERT INTO CourseOffering VALUES (9, 'MATH3120', 4, 1, 1, 9);
477 INSERT INTO CourseOffering VALUES (10, 'MATH3820', 5, 1, 1, 10);
478 INSERT INTO CourseOffering VALUES (11, 'SENG1110', 5, 3, 1, 11);
479 INSERT INTO CourseOffering VALUES (12, 'SENG1120', 5, 2, 1, 12);
480 go
481
482     ---     Student/Course Offering Data        ---
483
484 CREATE TABLE StudentCourseOffering(
485     student         INT         NOT NULL,       --ID of the student
486     isStudent       BIT CHECK(isStudent = 1),   --Flag that determines that
        this person is a student
487     offering        INT         NOT NULL,       --Course that is being offered
488     dateRegistered  DATE        NOT NULL,       --Date the student registered
        for the course
489     finalMark       INT,                        --Final mark of the course
490     finalGrade      VARCHAR(2),                 --Final grade of the course
491     isCompleted     BIT         NOT NULL,       --Flag that indicates that the
          course has been successfully completed (no fail)
492     PRIMARY KEY(student, offering),
493     FOREIGN KEY(student, isStudent) REFERENCES Person(personID, isStudent)
494         ON UPDATE NO ACTION ON DELETE NO ACTION,
495     FOREIGN KEY(offering) REFERENCES CourseOffering(offeringID)
496         ON UPDATE NO ACTION ON DELETE NO ACTION
497 );
```

```
498
499 INSERT INTO StudentCourseOffering VALUES (1, 1, 1, '2020-01-05', 87, 'HD', 1);
500 INSERT INTO StudentCourseOffering VALUES (1, 1, 3, '2022-01-05', NULL, NULL,
      0);
501 INSERT INTO StudentCourseOffering VALUES (1, 1, 2, '2022-01-05', NULL, NULL,
      0);
502 INSERT INTO StudentCourseOffering VALUES (1, 1, 6, '2018-01-05', 77, 'D', 1);
503 INSERT INTO StudentCourseOffering VALUES (1, 1, 7, '2020-07-05', 100, 'HD',
      1);
504 INSERT INTO StudentCourseOffering VALUES (1, 1, 8, '2020-01-05', 98, 'HD', 1);
505 INSERT INTO StudentCourseOffering VALUES (1, 1, 9, '2021-01-05', 81, 'D', 1);
506 INSERT INTO StudentCourseOffering VALUES (1, 1, 10, '2021-01-05', 96, 'HD',
      1);
507 INSERT INTO StudentCourseOffering VALUES (1, 1, 11, '2019-01-05', 93, 'HD',
      1);
508 INSERT INTO StudentCourseOffering VALUES (2, 1, 1, '2020-01-05', 100, 'HD',
      1);
509 INSERT INTO StudentCourseOffering VALUES (2, 1, 3, '2022-01-05', NULL, NULL,
      0);
510 INSERT INTO StudentCourseOffering VALUES (2, 1, 11, '2021-01-05', 67, 'C', 1);
511 INSERT INTO StudentCourseOffering VALUES (3, 1, 7, '2021-01-05', 90, 'HD', 1);
512 INSERT INTO StudentCourseOffering VALUES (3, 1, 11, '2020-01-05', 87, 'HD',
      1);
513 INSERT INTO StudentCourseOffering VALUES (3, 1, 12, '2021-07-05', 69, 'C', 1);
514 INSERT INTO StudentCourseOffering VALUES (5, 1, 5, '2021-07-05', 60, 'P', 1);
515 INSERT INTO StudentCourseOffering VALUES (5, 1, 1, '2021-07-05', 60, 'P', 1);
516 INSERT INTO StudentCourseOffering VALUES (6, 1, 3, '2020-01-05', 48, 'F', 0);
517 INSERT INTO StudentCourseOffering VALUES (6, 1, 4, '2021-07-05', 60, 'P', 1);
518 go
519
520     ---     Physical Offering Data      ---
521
522 CREATE TABLE PhysicalOffering(
523     offeringID  INT PRIMARY KEY,            --ID of the course offering
524     FOREIGN KEY(offeringID) REFERENCES CourseOffering(offeringID)
525         ON UPDATE NO ACTION ON DELETE NO ACTION
526 );
527
528 INSERT INTO PhysicalOffering VALUES (1);
529 INSERT INTO PhysicalOffering VALUES (2);
530 INSERT INTO PhysicalOffering VALUES (3);
531 INSERT INTO PhysicalOffering VALUES (4);
532 INSERT INTO PhysicalOffering VALUES (8);
533 INSERT INTO PhysicalOffering VALUES (10);
534 go
535
536     ---     Type of Reason Data     ---
537
538 CREATE TABLE ReasonType(
539     typeID  INT         PRIMARY KEY,    --Unique identifier
540     name    VARCHAR(50) NOT NULL,       --The name of the type (Reason)
541     UNIQUE(name)
542 );
543
544 INSERT INTO ReasonType VALUES (1, 'Lecture');
545 INSERT INTO ReasonType VALUES (2, 'Lab');
```

```
546 INSERT INTO ReasonType VALUES (3, 'Workshop');
547 go
548
549     ---     Timetable Slot Data     ---
550
551 CREATE TABLE TimetableSlot(
552     slotID      INT     PRIMARY KEY,    --ID of the timetable slot
553     facility    INT     NOT NULL,       --Facility that this timetable slot is
        taken in
554     offering    INT     NOT NULL,       --The offering that this slot is apart
        of
555     staff       INT     NOT NULL,       --Staff member teaching in this slot
556     isStaff     BIT CHECK(isStaff = 1), --Guarentees person is staff
557     date        DATE    NOT NULL,       --Date of timetable slot
558     startTime   TIME    NOT NULL,       --Start time of the timetable slot
559     endTime     TIME    NOT NULL,       --End time of the timetable slot
560     reason      INT     NOT NULL,       --The reason for this timetable slot
561     UNIQUE(slotID, offering),
562     FOREIGN KEY(facility) REFERENCES Facility(facID)
563         ON UPDATE NO ACTION ON DELETE NO ACTION,
564     FOREIGN KEY(offering) REFERENCES PhysicalOffering(offeringID)
565         ON UPDATE NO ACTION ON DELETE NO ACTION,
566     FOREIGN KEY(reason) REFERENCES ReasonType(typeID)
567         ON UPDATE NO ACTION ON DELETE NO ACTION,
568     FOREIGN KEY(staff, isStaff) REFERENCES Person(personID, isStaff)
569         ON UPDATE NO ACTION ON DELETE NO ACTION
570 );
571
572 INSERT INTO TimetableSlot VALUES (1, 2, 2, 4, 1, '2022-02-22', '15:00:00',
      '17:00:00', 1);
573 INSERT INTO TimetableSlot VALUES (2, 3, 2, 4, 1, '2022-02-22', '17:00:00',
      '19:00:00', 3);
574 INSERT INTO TimetableSlot VALUES (3, 7, 3, 4, 1, '2022-02-25', '09:00:00',
      '11:00:00', 1);
575 INSERT INTO TimetableSlot VALUES (4, 8, 3, 5, 1, '2022-02-25', '13:00:00',
      '15:00:00', 3);
576 INSERT INTO TimetableSlot VALUES (5, 11, 1, 5, 1, '2022-02-25', '10:00:00',
      '12:00:00', 2);
577 INSERT INTO TimetableSlot VALUES (6, 12, 1, 5, 1, '2022-02-23', '8:00:00',
      '10:00:00', 1);
578 INSERT INTO TimetableSlot VALUES (7, 1, 4, 6, 1, '2022-02-21', '13:00:00',
      '15:00:00', 1);
579 INSERT INTO TimetableSlot VALUES (8, 5, 4, 6, 1, '2022-02-24', '11:00:00',
      '12:00:00', 3);
580 INSERT INTO TimetableSlot VALUES (9, 1, 4, 6, 1, '2022-02-22', '15:00:00',
      '17:00:00', 3);
581 INSERT INTO TimetableSlot VALUES (10, 7, 8, 4, 1, '2022-02-24', '11:00:00',
      '14:00:00', 2);
582 INSERT INTO TimetableSlot VALUES (11, 10, 10, 6, 1, '2022-02-23', '15:00:00',
      '17:00:00', 1);
583 INSERT INTO TimetableSlot VALUES (12, 6, 10, 4, 1, '2022-02-22', '11:00:00',
      '13:00:00', 3);
584 INSERT INTO TimetableSlot VALUES (13, 1, 10, 6, 1, '2022-02-23', '12:00:00',
      '14:00:00', 2);
585 INSERT INTO TimetableSlot VALUES (14, 10, 10, 5, 1, '2022-02-23', '12:00:00',
      '14:00:00', 2);
```

```
586 go
587
588     ---     Student Course Offering x Timetable slot data       ---
589
590 CREATE TABLE StudentTimetableSlot(
591     student     INT     NOT NULL,       --ID of the student
592     offering    INT     NOT NULL,       --The offering that this slot is apart
        of
593     slot        INT     NOT NULL,       --Timetable slot that is being held
594     PRIMARY KEY(student, slot, offering),
595     FOREIGN KEY(student, offering) REFERENCES StudentCourseOffering(student,
        offering)
596         ON UPDATE NO ACTION ON DELETE NO ACTION,
597     FOREIGN KEY(slot, offering) REFERENCES TimetableSlot(slotID, offering)
598         ON UPDATE NO ACTION ON DELETE NO ACTION
599 );
600
601 INSERT INTO StudentTimetableSlot VALUES (1, 2, 1);
602 INSERT INTO StudentTimetableSlot VALUES (1, 2, 2);
603 INSERT INTO StudentTimetableSlot VALUES (1, 3, 3);
604 INSERT INTO StudentTimetableSlot VALUES (1, 3, 4);
605 INSERT INTO StudentTimetableSlot VALUES (1, 8, 10);
606 INSERT INTO StudentTimetableSlot VALUES (1, 10, 11);
607 INSERT INTO StudentTimetableSlot VALUES (1, 10, 12);
608 INSERT INTO StudentTimetableSlot VALUES (1, 10, 13);
609 INSERT INTO StudentTimetableSlot VALUES (2, 3, 3);
610 INSERT INTO StudentTimetableSlot VALUES (2, 3, 4);
611 INSERT INTO StudentTimetableSlot VALUES (5, 1, 5);
612 INSERT INTO StudentTimetableSlot VALUES (5, 1, 6);
613 INSERT INTO StudentTimetableSlot VALUES (6, 3, 3);
614 INSERT INTO StudentTimetableSlot VALUES (6, 3, 4);
615 INSERT INTO StudentTimetableSlot VALUES (6, 4, 7);
616 go
617
618     ---     Student Enrolment Data      ---
619
620 CREATE TABLE StudentEnrolment(
621     enrolID             INT             PRIMARY KEY,    --ID of the student
        enrolment
622     student             INT             NOT NULL,       --Student that is
        enrolling for program(s)
623     isStudent           BIT CHECK(isStudent = 1),       --Flag that makes sure
         that this person is a student
624     period              INT             NOT NULL,       --The period this
        student is offering in
625     dateEnrolled        DATE            NOT NULL,       --The date the student
         is enrolling
626     dateCompletion  DATE,                           --The date the student
        completed their enrolment
627     status              VARCHAR(100)    NOT NULL,       --Status of the
        student enrolment
628     FOREIGN KEY(student, isStudent) REFERENCES Person(personID, isStudent)
629         ON UPDATE NO ACTION ON DELETE NO ACTION,
630     FOREIGN KEY(period) REFERENCES Period(periodID)
631         ON UPDATE NO ACTION ON DELETE NO ACTION
632 );
```

```
633
634 INSERT INTO StudentEnrolment VALUES (1, 1, 1, 2, '2019-01-15', NULL,
      'Enrolled');
635 INSERT INTO StudentEnrolment VALUES (2, 2, 1, 3, '2018-07-15', NULL,
      'Enrolled');
636 INSERT INTO StudentEnrolment VALUES (3, 3, 1, 5, '2011-01-15', '2021-12-31',
      'Completed');
637 go
638
639     ---     Program Enrolment Data      ---
640
641 CREATE TABLE ProgramEnrolment(
642     enrolID INT NOT NULL,   --ID of the student enrolment
643     program INT NOT NULL,   --program that is being enrolled in
644     PRIMARY KEY(enrolID, program),
645     FOREIGN KEY(enrolID) REFERENCES StudentEnrolment(enrolID)
646         ON UPDATE NO ACTION ON DELETE NO ACTION,
647     FOREIGN KEY(program) REFERENCES Program(progCode)
648         ON UPDATE NO ACTION ON DELETE NO ACTION
649 );
650
651 INSERT INTO ProgramEnrolment VALUES (1, 10237);
652 INSERT INTO ProgramEnrolment VALUES (1, 40103);
653 INSERT INTO ProgramEnrolment VALUES (2, 11497);
654 INSERT INTO ProgramEnrolment VALUES (3, 10237);
655 go
```

```
1  --  ============================================================
2  --  Author: Jesse Lecathelinais
3  --  Description: create_tr_timetableclashcheck.sql
4  --  Enforces no student or staff member has any timetable clashes
5  --  ============================================================
6
7  DROP TRIGGER IF EXISTS timetableClashCheckStaff
8  DROP TRIGGER IF EXISTS timetableClashCheckStudent
9  go
10
11
12
13 CREATE TRIGGER timetableClashCheckStudent       --  Trigger name
14 ON StudentTimetableSlot                        --  Table name
15 FOR INSERT, UPDATE                     --  Actions when trigger is executed
16 AS
17 BEGIN
18     --  Check if the Business Rule is violated
19     --  If violated, print an error message
20     --  Cancel (rollback) the transaction
21
22     DECLARE studentCursor CURSOR            --  Cursor for timetableclashcheck
          trigger for students
23     FOR
24     SELECT  slotID, student, date, startTime, endTime
25     FROM    TimetableSlot ts, StudentTimetableSlot sts
26     WHERE   ts.slotID = sts.slot
27     FOR READ ONLY
28
29     DECLARE @slot       INT
30     DECLARE @student    INT
31     DECLARE @date       DATE
32     DECLARE @startTime  TIME
33     DECLARE @endTime    TIME
34
35     DECLARE @finalCount INT
36     SET     @finalCount = 0
37     DECLARE @currCount  INT
38     SET     @currCount = 0
39
40     OPEN studentCursor
41     FETCH NEXT FROM studentCursor INTO @slot, @student, @date, @startTime,
         @endTime
42
43     WHILE @@FETCH_STATUS = 0
44     BEGIN
45
46         --  Find any overlapping time slots
47         SELECT @currCount = COUNT(*)
48         FROM    Inserted i,             --The timetable slot data that is to
           be inserted
49                 TimetableSlot ts,
50                 PhysicalOffering po,
51                 CourseOffering c,
52                 StudentCourseOffering sc,
53                 Period pe
```

```
54          WHERE   i.slot = ts.slotID
55              AND i.slot != @slot
56              AND i.offering = ts.offering
57              AND ts.offering = po.offeringID     --  Check offering's all
                  connected
58              AND po.offeringID = c.offeringID
59              AND sc.offering = c.offeringID
60              AND i.student = sc.student
61              AND i.student = @student
62              AND c.period = pe.periodID
63              AND ts.date = @date
64              AND (           -- Timetable clash check section
65                      (ts.startTime >= @startTime AND ts.startTime < @endTime)
66                  OR  (ts.endTime > @startTime AND ts.endTime <= @endTime)
67                  )
68
69          --Update final counter
70          IF @currCount > @finalCount
71          BEGIN
72              SET @finalCount = @currCount
73          END
74
75          FETCH NEXT FROM studentCursor INTO @slot, @student, @date, @startTime,
               @endTime
76      END
77
78      DEALLOCATE studentCursor
79
80      IF @finalCount > 0
81      BEGIN
82          RAISERROR('Time table clash encountered for a student. The command is
              terminated', 11, 1)
83          ROLLBACK TRANSACTION
84      END
85
86  END
87  go
88
89  CREATE TRIGGER timetableClashCheckStaff --  Trigger to prevent staff timetable
       clashes
90  ON TimetableSlot                        --  Table name
91  FOR INSERT, UPDATE                      --  Actions when trigger is executed
92  AS
93  BEGIN
94      --  Check if the Business Rule is violated
95      --  If violated, print an error message
96      --  Cancel (rollback) the transaction
97
98      DECLARE staffCursor CURSOR          --  Cursor for timetableclashcheck
            trigger
99      FOR
100     SELECT  slotID, staff, date, startTime, endTime
101     FROM    TimetableSlot
102     FOR READ ONLY
103
104     DECLARE @slot       INT
```

```
105     DECLARE @staff      INT
106     DECLARE @date       DATE
107     DECLARE @startTime  TIME
108     DECLARE @endTime    TIME
109
110     DECLARE @finalCount INT
111     SET     @finalCount = 0
112     DECLARE @currCount  INT
113     SET     @currCount = 0
114
115     OPEN staffCursor
116     FETCH NEXT FROM staffCursor INTO @slot, @staff, @date, @startTime,
          @endTime
117
118     WHILE @@FETCH_STATUS = 0
119     BEGIN
120
121         --  Find any overlapping time slots
122         SELECT @currCount = COUNT(*)
123         FROM    Inserted i,             --The timetable slot data that is to
          be inserted
124                 PhysicalOffering po,
125                 CourseOffering c,
126                 StudentCourseOffering sc,
127                 Period pe
128         WHERE   i.offering = po.offeringID      --  Check offering's all
          connected
129             AND po.offeringID = c.offeringID
130             AND sc.offering = c.offeringID
131             AND c.period = pe.periodID
132             AND i.staff = @staff
133             AND i.slotID != @slot
134             AND i.date = @date
135             AND (           -- Timetable clash check section
136                     (i.startTime >= @startTime AND i.startTime < @endTime)
137                 OR  (i.endTime > @startTime AND i.endTime <= @endTime)
138                 )
139
140         IF @currCount > @finalCount
141         BEGIN
142             SET @finalCount = @currCount
143         END
144
145         FETCH NEXT FROM staffCursor INTO  @slot, @staff, @date, @startTime,
          @endTime
146     END
147
148     DEALLOCATE staffCursor
149
150     IF @finalCount > 0
151     BEGIN
152         RAISERROR('Time table clash encountered of a staff member. The command
            is terminated', 11, 1)
153         ROLLBACK TRANSACTION
154     END
155
```

```
156 END
157 go
```

```sql
--  ============================================================
--  Author: Jesse Lecathelinais
--  Description: test_tr_timetableclashcheck_5.sql
--  Data to test that no timetable clashes occur for staff or students
--  ============================================================

--Shouldn't work for TimetableSlot (Staff clash)
INSERT INTO TimetableSlot VALUES (15, 11, 1, 5, 1, '2022-02-25', '10:00:00',
  '12:00:00', 2);
INSERT INTO TimetableSlot VALUES (16, 11, 1, 4, 1, '2022-02-25', '8:00:00',
  '10:00:00', 3);
INSERT INTO TimetableSlot VALUES (17, 1, 4, 6, 1, '2022-02-24', '11:00:00',
  '12:00:00', 3);
INSERT INTO TimetableSlot VALUES (18, 2, 8, 4, 1, '2022-02-25', '10:00:00',
  '12:00:00', 1);
INSERT INTO TimetableSlot VALUES (19, 9, 8, 4, 1, '2022-02-22', '14:00:00',
  '16:00:00', 2);
INSERT INTO TimetableSlot VALUES (20, 4, 10, 4, 1, '2022-02-25', '08:00:00',
  '10:00:00', 1);

--Should work for TimetableSlot (Staff doesn't clash)
INSERT INTO TimetableSlot VALUES (21, 4, 10, 4, 1, '2022-02-24', '14:00:00',
  '16:00:00', 2);

--Shouldn't work for StudentTimetableSlot (Student clash)
INSERT INTO StudentTimetableSlot VALUES (1, 10, 14);

--Should work for StudentTimetableSlot (Student doesn't clash)
INSERT INTO StudentTimetableSlot VALUES (6, 4, 8);
INSERT INTO StudentTimetableSlot VALUES (6, 4, 9);
```

```sql
-- ============================================
-- Author:      Nathan Murphy
-- Description: usp_RegisterForCourses.sql
-- Creation SQL for stored procedure to be called by a front end system to register
-- Procedure Flow:
-- *Check if Valid Student Number, If invalid present error and exit.
-- *Create Cursor and begin iteration over CourseOfferingList
-- *Every iteration :
-- *Check if courseid exists in courses, If not RaiseError & continue.
-- *Check if student is already
-- ============================================
DROP PROCEDURE IF EXISTS usp_RegisterForCourses
DROP TYPE IF EXISTS CourseOfferingList
go

CREATE TYPE CourseOfferingList AS TABLE -- Table type for the list of offerings.
(
    courseID INT -- Course ID passed into the procedure.
)
go

--Sample Data
-- @studentNumber ID 2 = Nathan Murphy

CREATE PROCEDURE usp_RegisterForCourses
    @studentNumber INT,
    @CourseOfferingList AS CourseOfferingList READONLY

AS
BEGIN
    DECLARE @CurrOfferingID VARCHAR(10)

    --Checking weather the Student is valid within the Pearson Table.
    -- Check if the count is less then 1 as 0 will be the result ending in error.
    IF (SELECT COUNT(personID) FROM Person WHERE personID = @studentNumber) < 1
        BEGIN
            RAISERROR('Invalid Student Number. Cannot proceed with enrollment.',11,1)
            RETURN -- Exit out of the procedure to provident incorrect data being passed forward.
        END
    ELSE IF (SELECT COUNT(personID) FROM Person WHERE personID = @studentNumber AND isStudent = 1) < 1
        BEGIN
            RAISERROR('Staff member not a student. Cannot proceed with enrollment.',11,1)
            RETURN -- Exit out of the procedure to provident incorrect data being passed forward.
        END
    ELSE
        BEGIN -- Begin iteration over the table of course offerings passed into the procedure.
```

```
47              DECLARE insert_Cursor CURSOR FOR
48
49              SELECT * -- Populate the cursor with all elements within Course
                  Offering List.
50              FROM @CourseOfferingList
51
52              OPEN insert_Cursor -- open cursor.
53
54              FETCH NEXT FROM insert_Cursor INTO @CurrOfferingID -- Get the
                  first record from the table.
55
56              WHILE @@FETCH_STATUS = 0 --Intiate the cursor status.
57
58              BEGIN
59                  BEGIN TRY
60
61                      --Used to record the total amount of prerequisite courses
                        for the selected offering
62                      DECLARE @temp INT
63
64                      SELECT @temp = COUNT(*)
65                      FROM AssumedKnowledge a, CourseOffering co
66                      WHERE   a.course = co.course
67                          AND co.offeringID = @CurrOfferingID
68
69
70                      -- Check if the current course offering is a valid course.
71                      IF (
72                          SELECT COUNT(offeringID)
73                          FROM CourseOffering
74                          WHERE @CurrOfferingID = offeringID
75                      ) < 1
76                          RAISERROR('Invalid Course ID, This Course does not
                        exist as a offered course.',11,1)
77
78                      -- Check if student already completed the course
79                      ELSE IF (
80                          SELECT COUNT(offering)
81                          FROM StudentCourseOffering
82                          WHERE   offering = @CurrOfferingID
83                              AND student = @studentNumber
84                              AND isCompleted = 1
85                      ) > 0
86                          RAISERROR('Student has already completed this
                        course.',16,1)
87
88                      -- Check if student is already enrolled in a course.
89                      ELSE IF (
90                          SELECT COUNT(offering)
91                          FROM StudentCourseOffering
92                          WHERE   offering = @CurrOfferingID
93                              AND student = @studentNumber
94                      ) > 0
95                          RAISERROR('Student is already enrolled in this
                        course.',16,1)
96
```

```
 97
 98
 99                      --Check if all prerequisite courses have been completed
100                      ELSE IF (
101                          SELECT COUNT(*)
102                          FROM StudentCourseOffering sc, CourseOffering co
103                          WHERE   sc.offering = co.offeringID
104                              AND sc.student = @studentNumber
105                              AND sc.isCompleted = 1
106                              AND co.course IN (
107                                  SELECT a.assumedCourse
108                                  FROM AssumedKnowledge a, CourseOffering co
109                                  WHERE   a.course = co.course
110                                      AND co.offeringID = @CurrOfferingID
111                              )
112                          ) < @temp
113                          RAISERROR('Student has not completed all prerequisite courses for this course.',16,1)
114
115                      ELSE
116                      -- Insert enrolment record for student in an offering, allocated to today's date.
117                          INSERT INTO StudentCourseOffering(student, isStudent, offering, dateRegistered, finalMark, finalGrade, isCompleted)
118                          VALUES (@studentNumber, 1, @CurrOfferingID, (SELECT GETDATE()), NULL, NULL, 0)
119                  END TRY
120                  -- Catch Statement to output any Error messages that happen
121                  BEGIN CATCH
122                      --Catch any Errors using the following
123                      DECLARE @Message NVARCHAR(250);
124                      DECLARE @Severity INT;
125                      DECLARE @State INT;
126
127                      SELECT
128                          @Message = ERROR_MESSAGE(),
129                          @Severity = ERROR_SEVERITY(),
130                          @State = ERROR_STATE();
131
132                      RAISERROR(@Message, @Severity, @State);
133                  END CATCH
134              -- Fetch next record.
135              FETCH NEXT FROM insert_Cursor INTO @CurrOfferingID
136              END
137          END
138  END
139  go
```

```sql
-- ============================================
-- Author:      Nathan Murphy
-- Description: test_usp_RegisterForCourses.sql
-- Script for testing the stored prcoedure (usp_RegisterForCourses)
-- ============================================

SET NOCOUNT ON;
DECLARE @StudentID INT
DECLARE @OfferingsTable CourseOfferingList;

SET @StudentID = 2 --  Valid Student

INSERT INTO @OfferingsTable VALUES(2) -- COMP3350 (Will enter for student 2)
INSERT INTO @OfferingsTable VALUES(1) -- COMP1140
INSERT INTO @OfferingsTable VALUES(1) -- COMP1140
INSERT INTO @OfferingsTable VALUES(20) -- Not a valid Course
INSERT INTO @OfferingsTable VALUES(11) -- SENG1110
INSERT INTO @OfferingsTable VALUES(3) -- COMP3851A
INSERT INTO @OfferingsTable VALUES(5) -- MATH1210

SET NOCOUNT OFF;

PRINT('First Exec')
EXEC usp_RegisterForCourses @StudentID, @OfferingsTable --

SET @StudentID = 200 -- Invalid Student

PRINT(' ')
PRINT('Second Exec')
EXEC usp_RegisterForCourses @StudentID, @OfferingsTable

SET @StudentID = 4 -- Staff member that isn't a student

PRINT(' ')
PRINT('Third Exec')
EXEC usp_RegisterForCourses @StudentID, @OfferingsTable
```