EGE UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

INTRODUCTION TO DATABASES

ERASMUS CURRICULUM DATABASE INTEGRATION PROJECT

GROUP-2

PREPARED BY

05210000260-Kutlu Çağan Akın

05210000296-Ali Osman Taş

ANALYSIS

1- BAHÇEŞEHİR UNIVERSITY – SOFTWARE ENGINEERING DIAGRAM

The database system is comprehensively modeled to manage and integrate the academic curriculum of the Software Engineering Department at Bahçeşehir University (BAU). The design focus is on creating a high-fidelity representation of course structures and precise workload calculations to ensure full interoperability with Erasmus data exchange standards and MÜDEK accreditation requirements.

1. Dual-Role Instructor Administration The system moves beyond simple "Teacher-Course" links to reflect the complex operational reality of academic delivery. The INSTRUCTOR entity acts as a central academic node, linked to courses through two distinct and semantically significant relationships:

- **Administrative Responsibility (COORDINATES - 1:N):** This relationship enforces the rule that each course must have strictly one designated Coordinator. This role represents the legal authority responsible for syllabus approval, grade submission, and objection handling.
- **Instructional Delivery (TEACHES - M:N):** Recognizing the collaborative nature of modern engineering education, this relationship allows a course to be delivered by a diverse team (e.g., *Lead Lecturers, Lab Assistants, Guest Speakers*). This Many-to-Many structure enables the system to track every staff member involved in the teaching process without violating the single-point-of-responsibility principle.

2. Disjoint Curriculum Specialization Hierarchy To accurately model the department's flexible yet structured curriculum, a Disjoint Specialization (d) hierarchy is implemented on the COURSE entity. This ensures that every course falls into exactly one of the following distinct categories, preventing classification ambiguity:

- **MANDATORY_COURSE:** Compulsory core courses that form the backbone of the Software Engineering degree.
- **DEPARTMENTAL_ELECTIVE:** Technical courses offered specifically within the department to allow specialization in tracks like *Software Architecture* or *Data Engineering*.
- **NON_DEPARTMENTAL_ELECTIVE:** Courses taken from other engineering disciplines (e.g., *Industrial Engineering*) to foster interdisciplinary skills.
- **GENERAL_EDUCATION (GE)_ELECTIVE:** Courses from social sciences, arts, or sports, designed to provide a holistic education in line with university-wide requirements.

3. "Digital Syllabus" & Workload Compliance (Weak Entities) The system includes a cluster of existence-dependent weak entities that function as the "Digital

Syllabus" of a course. This detailed modeling is critical for automatic ECTS validation and course equivalency comparisons:

- **ECTS_WORKLOAD:** Instead of a static value, the system calculates the true student load dynamically. It tracks specific activities (e.g., *Homework, Quizzes, Projects*) by storing their Count, Duration, and deriving the Total Workload.
- **ASSESSMENT**: The grading architecture is defined by recording evaluation methods (e.g., *Midterm, Final, Project*) and their specific Contribution percentages (e.g., *Midterm 30%, Final 45%*), ensuring transparency.
- **WEEKLY_PLAN**: A granular semester schedule is stored week-by-week, linking specific Topics and required Preliminary Work (e.g., *Readings, Installations*) to each week number.
- **LEARNING_OUTCOME:** Specific competency targets are stored as distinct entries, facilitating outcome-based education (OBE) analysis.

4. Course Metadata and Semester Flow The COURSE entity is enriched with essential attributes such as T_Hour (Theoretical) and P_Hour (Practical) to define the delivery format. Additionally, ModeOfDelivery (e.g., *Face-to-Face, Hybrid*) is tracked to accommodate modern educational shifts. The curriculum is chronologically organized into SEMESTERS, characterized by a specific Season (*Fall/Spring*) and Semester Number, ensuring the logical flow of the 4-year program.

2- TOBB ETÜ - ARTIFICIAL INTELLIGENCE ENGINEERING DIAGRAM

The database system is meticulously designed to manage the complex academic curriculum and administrative structure of the **Artificial Intelligence Engineering Department** at **TOBB University of Economics and Technology**. This conceptual design addresses the university's unique educational model, characterized by a three-term (Trimester) system and a cooperative education program, distinguishing it from traditional two-semester institutions.

1. Comprehensive Instructor Profiling (Faculty Management) The system goes beyond standard instructor data storage by modeling a rich, CV-like profile for each faculty member. This is crucial for matching academic staff with specialized courses and facilitating Erasmus staff mobility.

- **Core Entity:** The INSTRUCTOR entity holds primary identification data (InstructorID) along with administrative details such as Title, Office Number, Phone Extension, and Personal Website.
- **Expertise Modeling (Multivalued Attribute):** To capture the diverse research interests of the faculty (e.g., *Computer Vision, Natural Language Processing, Game Theory*), a **multivalued attribute** named Research_Areas is employed. This allows the system to store multiple distinct specialization tags for a single instructor.
- **Academic & Professional History (Weak Entities):**
    - **EDUCATION:** A specific **weak entity** is created to track the academic background of instructors. It stores detailed records of their degrees (B.Sc., M.Sc., Ph.D.), the universities they attended, the specific departments, and graduation years.
    - **WORK_EXPERIENCE:** To reflect the industry experience of the faculty, another **weak entity** is modeled. This entity tracks previous roles, companies (e.g., *Aselsan, Microsoft*), and tenure periods, providing a complete professional timeline.

2. Course Structure and Recursive Prerequisites Courses are the central entities of the system, identified by a unique CourseCode. The entity structure is designed to hold granular details required for ECTS validation, such as the specific breakdown of **Theoretical (T_Hour)** and **Practical (P_Hour)** hours.

- **The Prerequisite Chain:** One of the most critical aspects of the TOBB ETÜ curriculum is its strict dependency chain (e.g., *Artificial Intelligence* requires *Data Structures*, which requires *Programming I*). This is modeled using a **Recursive Many-to-Many Relationship** named PREREQUISITE on the Course entity. This structure allows the system to validate whether a student has completed all necessary "Pre-Req" courses before registering for a "Main" course.

3. Curriculum Specialization (Disjoint Hierarchy) The curriculum requires students to follow a rigid path in the early years and specialize in a specific domain after the 5th term. This logic is implemented using a **Disjoint Specialization (d)** hierarchy:

- **MANDATORY_COURSE:** This category includes core engineering courses and the university's signature **Cooperative Education (Ortak Eğitim)** courses (OEG coded). These are full-time internships that are treated as credit-bearing mandatory courses within the system.
- **AI_ELECTIVE:** This category groups specialized technical courses selected from the Artificial Intelligence pool (e.g., *Deep Learning, Robotics*).
- **FINANCE_ELECTIVE:** Reflecting the interdisciplinary nature of the program, this category includes courses from the Finance pool (e.g., *Computational Finance, Risk Management*), allowing students to pursue a specific track.

4. Syllabus and ECTS Workload (Weak Entities) To ensure full compliance with the **Bologna Process** and **Erasmus Data Exchange** standards, the system creates a detailed "digital twin" of the course syllabus through several existence-dependent **weak entities**:

- **ECTS_WORKLOAD:** This entity justifies the high ECTS credits (e.g., 8 ECTS) assigned to courses. It breaks down the student's workload into specific activities (e.g., *Lecture, Homework, Project, Exam Preparation*). It stores the **Count**, **Duration** (in hours), and the calculated **Workload** for each activity type.
- **WEEKLY_PLAN:** The trimester schedule is stored week-by-week (Weeks 1-12), linking specific **Topics** and required **Preliminary Work** to each week number.
- **ASSESSMENT:** The grading system is modeled by recording various evaluation methods (e.g., *Midterm, Final, Laboratory, Quiz*) and their strictly defined **Contribution** percentages to the final grade.
- **LEARNING_OUTCOME:** Specific skills and knowledge targets are stored as distinct entries linked to the course to facilitate curriculum mapping.

5. Trimester and Departmental Relations The system supports the university's three-term academic year (Fall, Spring, Summer) through the **SEMESTER** entity. Every course instance is linked to a specific semester plan. Additionally, the **DEPARTMENT** entity serves as the administrative root, managing course offerings and faculty employment.

3- EGE UNIVERSITY - COMPUTER ENGINEERING DIAGRAM

The database system is designed with a high degree of granularity to represent the comprehensive curriculum and credit structure of the Computer Engineering Department at Ege University. This conceptual design strictly adheres to the university's specific Bologna Information Package formats, focusing on the precise tracking of laboratory hours and a multi-layered elective course system.

1. "T+U+L" Course Structure and Rich Metadata Unlike standard models that group practical hours together, the system is engineered to reflect Ege University's unique credit calculation method, which distinguishes between Theoretical, Application (Practice), and Laboratory hours.

- **Granular Hour Tracking:** The COURSE entity explicitly stores three distinct attributes: Theoretical (T_Hour), Application (P_Hour), and Laboratory (L_Hour). This distinction is critical for generating accurate official transcripts and syllabus reports where these components are evaluated separately.
- **Extended Metadata:** To create a digital twin of the official Course Information Form, the entity includes rich descriptive fields such as Purpose, Content, InternshipStatus, and OtherConsiderations, ensuring no data loss from the physical documents.

2. Hierarchical (Nested) Curriculum Specialization The curriculum architecture is modeled using a Nested Disjoint Specialization hierarchy to manage the diverse elective pools found in the Ege University program structure.

- **Primary Level:** Courses are first categorized into MANDATORY (Core engineering courses) and ELECTIVE (Optional courses).
- **Secondary Level (Nested):** The ELECTIVE entity is further specialized into three specific sub-categories:
    - **TECHNICAL_ELECTIVE (TEG):** Advanced engineering topics (e.g., *Cloud Computing, Microservices*).
    - **SOCIAL_ELECTIVE (UNI.ELEC):** University-wide non-technical courses (e.g., *Global Politics*).
    - **PEDAGOGICAL_FORMATION:** Courses related to teaching methodologies and education sciences.

3. Advanced Syllabus Modeling (Weekly Plan) The system's WEEKLY_PLAN weak entity is significantly enhanced to match the university's detailed syllabus format. Instead of a single "Topic" field, the entity models the weekly content in three parallel streams:

- **T_Topic**: The theoretical subject covered in lectures.
- **U_Topic:** The specific problem sets or applications covered in practice sessions.
- **L_Topic**: The experiments conducted in the laboratory for that specific week.
- Additionally, attributes for Teaching_Method and Preliminary (preparation work) are included to fully represent the course execution plan.

4. ECTS Workload and Assessment Standards To validate the ECTS credits assigned to each course, the system uses existence-dependent weak entities that mirror the official "Reference Workload" and "Assessment" tables found in the course PDFs.

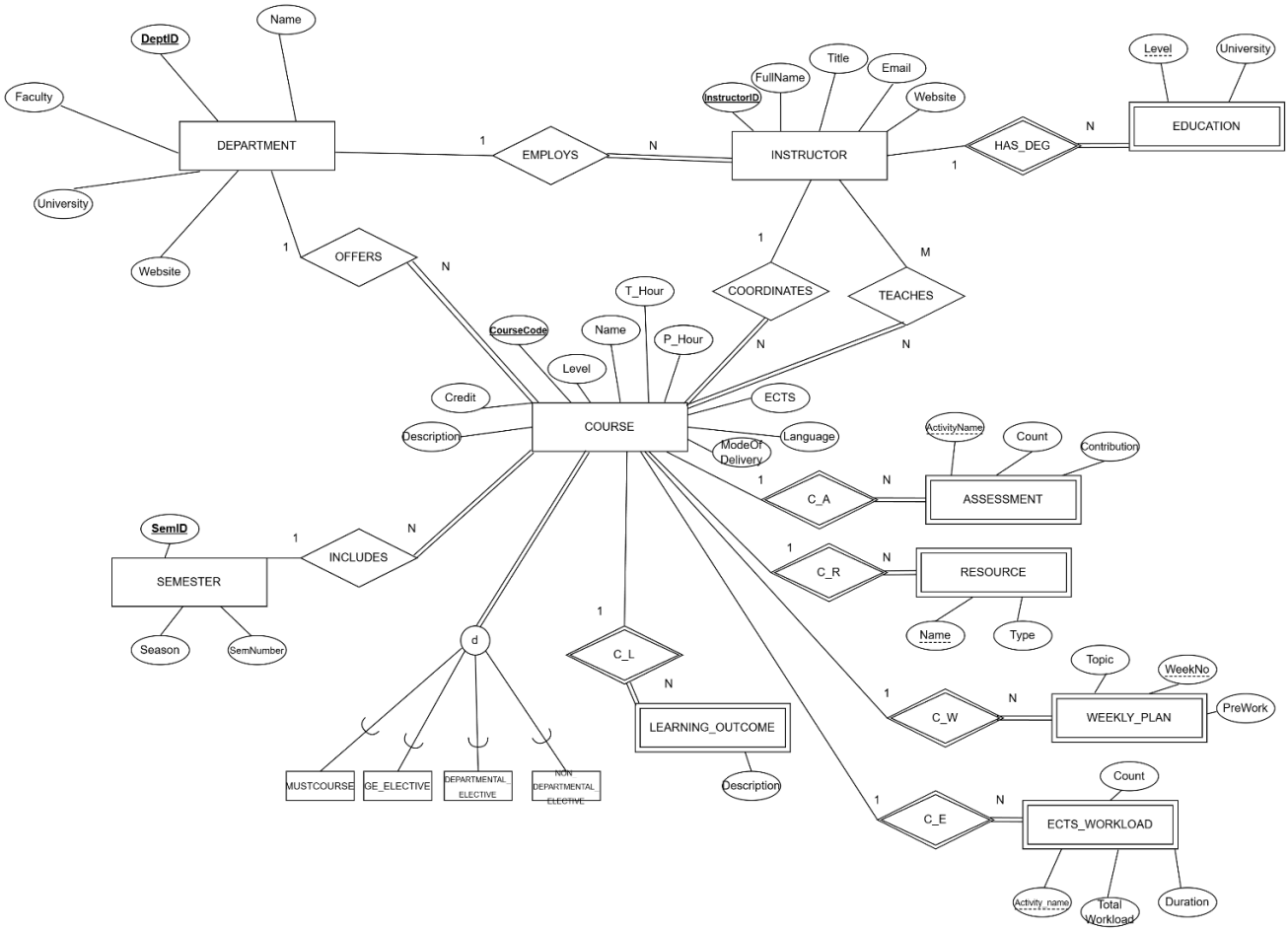
- **ECTS_WORKLOAD:** Tracks student effort across various activities (e.g., *Midterm Prep, Project, Lab Report*). It stores the Count, Duration (Hours), and the derived Workload for each item.
- **ASSESSMENT:** Defines the grading scheme by recording activities (e.g., *Midterm, Quiz, Final*) and their strictly defined Contribution percentages.

5. Faculty and Academic History The INSTRUCTOR entity is designed to serve not just as a nameholder but as a comprehensive academic profile. A specific weak entity, EDUCATION, is attached to the instructor to track their academic lineage (Degrees, Universities, Graduation Years). The system defines two key relationships:

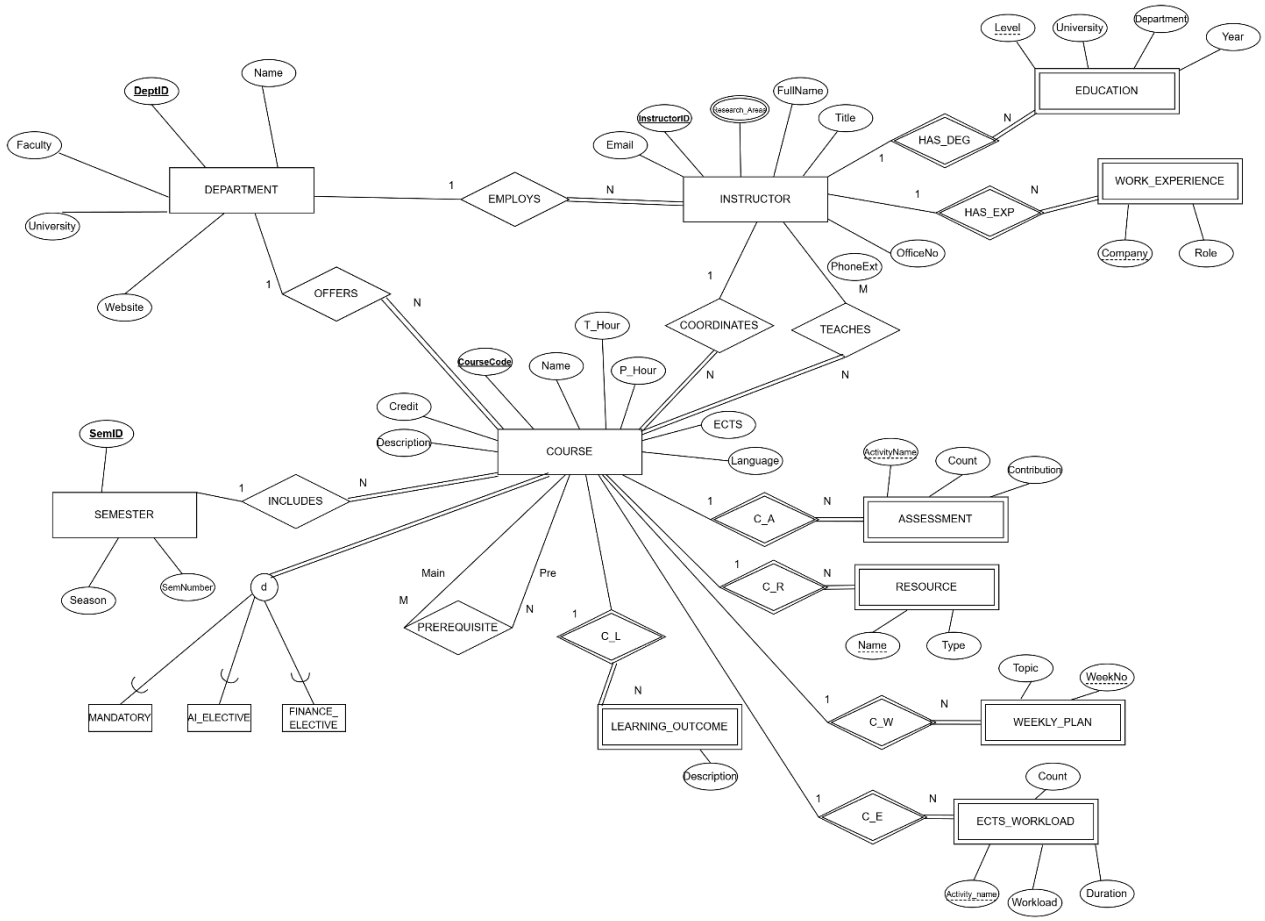
- **COORDINATES (1:N):** Identifies the single faculty member legally responsible for the course management.
- **TEACHES (M:N):** Allows for multiple staff members (including research assistants for laboratories) to be associated with the course delivery.

DESIGN-CONCEPTUAL DESIGN

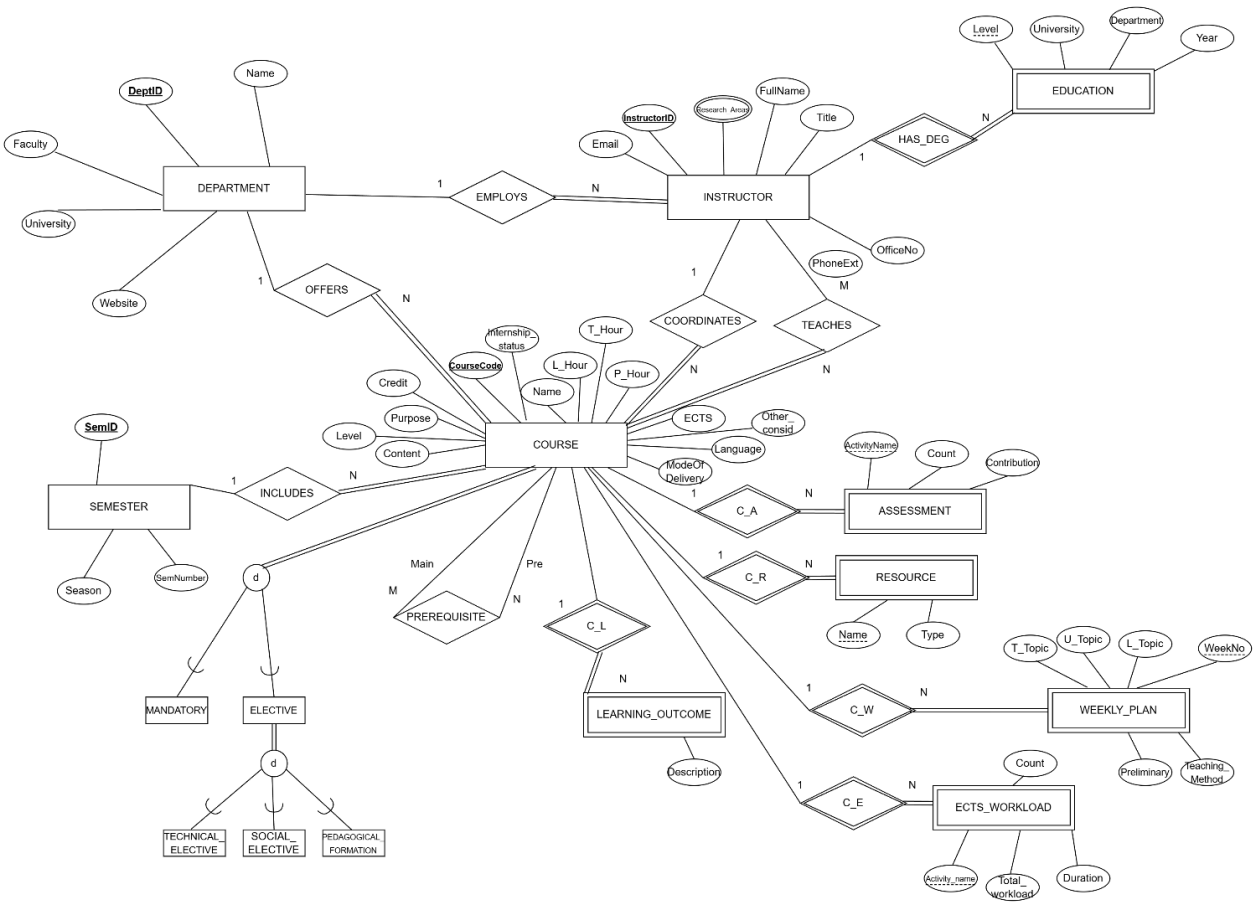
1- BAHÇEŞEHİR UNIVERSITY SOFTWARE ENGINEERING DIAGRAM



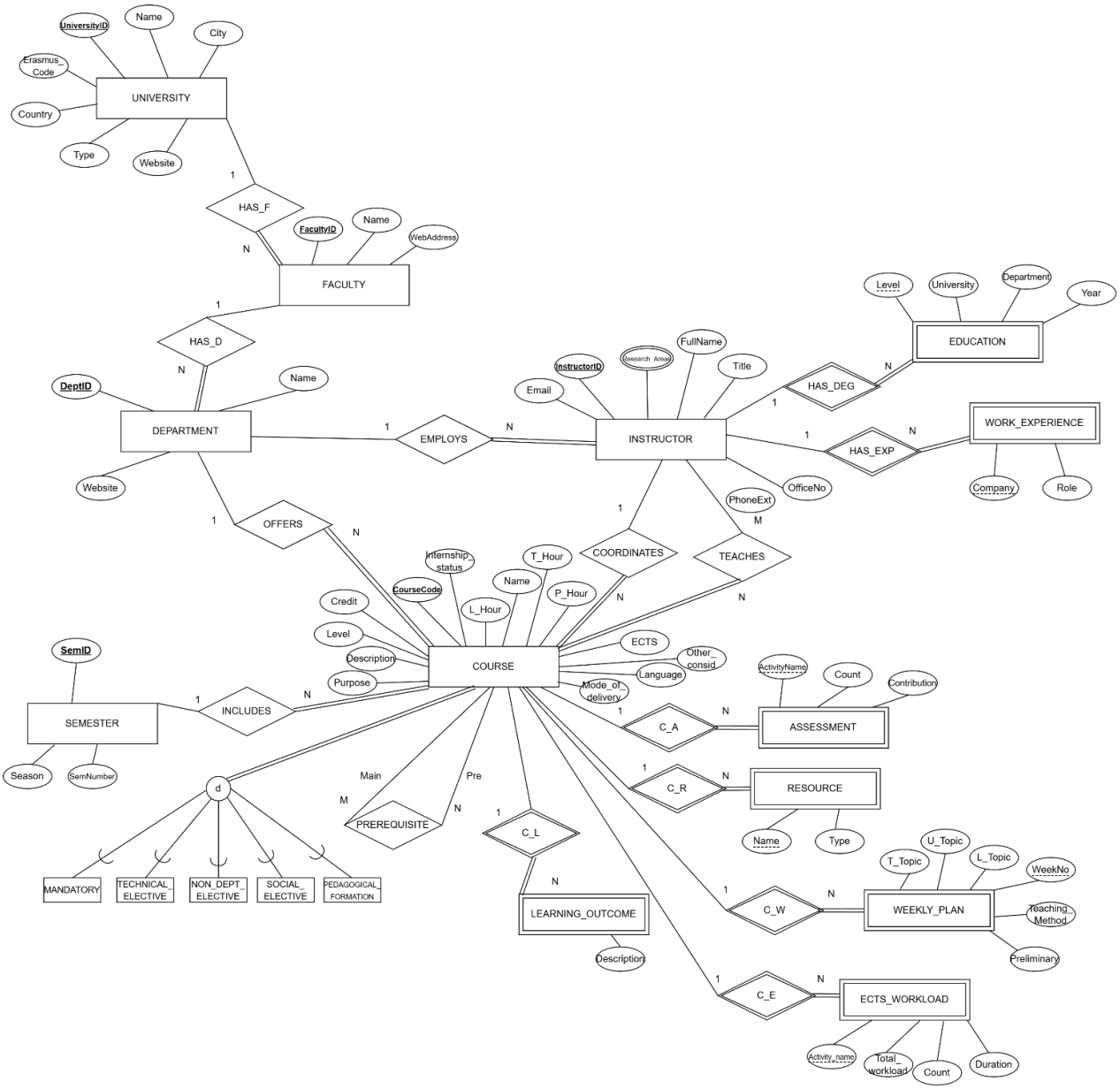
2- TOBB ETÜ ARTIFICIAL INTELLIGENCE ENGINEERING DIAGRAM



3- EGE UNIVERSITY COMPUTER ENGINEERING DIAGRAM



4- COMBINED DIAGRAM



INTEGRATION STRATEGY

The final EER diagram represents a unified database system capable of managing the diverse academic structures of Bahçeşehir, TOBB ETÜ, and Ege Universities simultaneously. To achieve this, a "Superset Integration Strategy" was applied. This approach involves creating a comprehensive schema that includes the union of all attributes and relationships from the three individual models, ensuring that no granular data—essential for Erasmus equivalency checks—is lost.

The integration process was governed by the following four core design rules:

1. The "Root-Hierarchy" Rule (University Entity) To support the specific business scenario where a student queries course equivalencies between a "Source University" and a "Target University," the scope of the database was expanded.

- **Root Entity:** A UNIVERSITY entity was introduced as the root of the hierarchy, storing international identification data such as ErasmusCode, Country, and Type.
- **Hierarchy Flow:** The structure follows a strict UNIVERSITY (1) → FACULTY (N) → DEPARTMENT (N) relationship chain, allowing the system to host multiple institutions within a single database instance.

2. The "Maximal Detail" Rule (Course & Syllabus) For an accurate Erasmus assessment, the system must support the most detailed curriculum format available among the three universities.

- **Superset Attributes:** The COURSE entity includes the union of all specific attributes: TOBB's T_Hour and P_Hour combined with Ege's L_Hour (Laboratory), Purpose, Content, and InternshipStatus.
- **Digital Twin Syllabus:** Ege University's granular WEEKLY_PLAN structure was adopted as the standard. Instead of a single topic field, the combined diagram tracks T_Topic (Theory), U_Topic (Application), and L_Topic (Lab) separately for each week. This allows the system to compare lab content specifically, which is often a deciding factor in engineering course equivalency.
- **Prerequisite Chain:** TOBB ETÜ's recursive PREREQUISITE relationship was retained to model dependency chains, which remains optional (partial participation) for universities that do not enforce strict prerequisites.

3. The "Rich Profile" Rule (Instructor) To facilitate Erasmus staff mobility and research matching, the simpler instructor models were upgraded to match the most detailed one (TOBB ETÜ).

- The INSTRUCTOR entity is now supported by two weak entities: EDUCATION (tracking degrees and alma maters) and WORK_EXPERIENCE (tracking industry background).
- A multivalued attribute Research_Areas is included to allow queries based on faculty expertise (e.g., "Find professors specializing in Artificial Intelligence").

4. Semantic Mapping (Flat Specialization) Since each university uses different terminology for similar course categories (e.g., *AI Elective* vs. *Departmental Elective*), a Semantic Mapping approach was applied. The nested hierarchies were flattened into a single Disjoint Specialization structure with standardized categories to enable cross-university queries.
The table below illustrates how the specific terminologies of each university are mapped to the unified entities in the Combined EER Diagram:

Database Entity Name	**EGE Equivalent**	**TOBB Equivalent**	**BAU Equivalent**
MANDATORY	Mandatory	Mandatory / OEG (Co-op)	Must Course
TECHNICAL_ELECTIVE	**TEG** (Technical Elective)	**AI Elective**	**Departmental Elective**
NON_DEPT_ELECTIVE	*(Not in Ege)*	**Finance Elective**	**Non-Departmental Elective**
SOCIAL_ELECTIVE	**UNI.ELEC** (University Elec)	*(In TOBB usually free)*	**GE Elective** (General Ed.)
PEDAGOGICAL_FORMATION	**PFSDG** (Pedagogic For.)	*(Not in TOBB)*	*(Not in BAU)*

DESIGN-LOGICAL MODEL

EER TO RELATIONAL MAPPING

1st ITERATION

STEP 1

UNIVERSITY (UniversityID, Name, City, Country, Erasmus_Code, Type, Website)

FACULTY (FacultyID, Name, WebAddress)

DEPARTMENT (DeptID, Name, Website)

SEMESTER (SemID, Season, SemNumber)

INSTRUCTOR (InstructorID, FullName, Title, Email, PhoneExt, OfficeNo)

COURSE (CourseCode, Name, ECTS, Credit, Level, Language, ModeOfDelivery, Description, Purpose, InternshipStatus, T_Hour, P_Hour, L_Hour, OtherConsid)

STEP 2

WEEKLY_PLAN (COURSE.CourseCode, WeekNo, T_Topic, U_Topic, L_Topic, TeachingMethod, Preliminary)

ASSESSMENT (COURSE.CourseCode, ActivityName, Count, Contribution)

ECTS_WORKLOAD (COURSE.CourseCode, ActivityName, Count, Duration, TotalWorkload)

RESOURCE (COURSE.CourseCode, Name, Type)

LEARNING_OUTCOME (COURSE.CourseCode, Description)

EDUCATION (INSTRUCTOR.InstructorID, Level, University, Department, Year)

WORK_EXPERIENCE (INSTRUCTOR.InstructorID, Company, Role)

STEP 3

-

STEP 4

FACULTY (FacultyID, UNIVERSITY.UniversityID, Name, WebAddress)

DEPARTMENT (DeptID, FACULTY.FacultyID, Name, Website)

INSTRUCTOR (InstructorID, DEPARTMENT.DeptID, FullName, Title, Email, PhoneExt, OfficeNo)

COURSE (CourseCode, DEPARTMENT.DeptID, INSTRUCTOR.InstructorID, SEMESTER.SemID, Name, ECTS, Credit, Level, Language, ModeOfDelivery, Description, Purpose, InternshipStatus, T_Hour, P_Hour, L_Hour, OtherConsid)

STEP 5

TEACHES (INSTRUCTOR.InstructorID, COURSE.CourseCode)

PREREQUISITE (COURSE.MainCourseCode, COURSE.PreCourseCode)

***FK1:** MainCourseCode references COURSE.CourseCode*

***FK2:** PreCourseCode references COURSE.CourseCode*

STEP 6

INSTRUCTOR_RESEARCH_AREAS (INSTRUCTOR.InstructorID, Area_Name)

STEP 7

-

STEP 8

MANDATORY (COURSE.CourseCode)

TECHNICAL_ELECTIVE (COURSE.CourseCode)

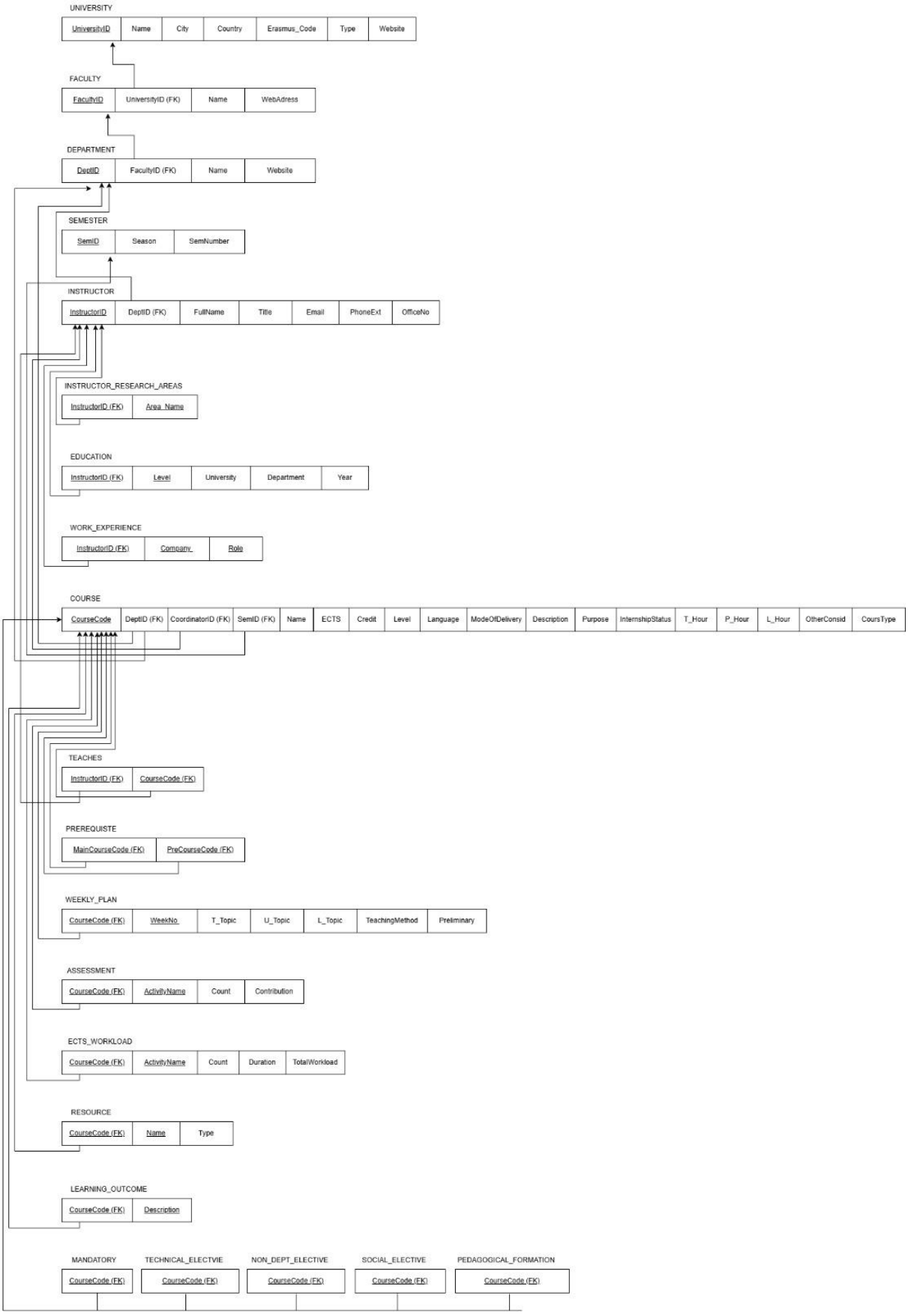
NON_DEPT_ELECTIVE (COURSE.CourseCode)

SOCIAL_ELECTIVE (COURSE.CourseCode)

PEDAGOGICAL_FORMATION (COURSE.CourseCode)

STEP 9

-


UNIVERSITY
UniversityID
Name
City
Country
Erasmus_Code
Type
Website
FACULTY
FacultyID
UniversityID (FK)
Name
WebAdress
DEPARTMENT
DeptID
FacultyID (FK)
Name
Website
SEMESTER
SemID
Season
SemNumber
INSTRUCTOR
InstructorID
DeptID (FK)
FullName
Title
Email
PhoneExt
OfficeNo
INSTRUCTOR_RESEARCH_AREAS
InstructorID (FK)
Area_Name
EDUCATION
InstructorID (FK)
Level
University
Department
Year
WORK_EXPERIENCE
InstructorID (FK)
Company
Role
COURSE
CourseCode
DeptID (FK)
CoordinatorID (FK)
SemID (FK)
Name
ECTS
Credit
Level
Language
ModeOfDelivery
Description
Purpose
InternshipStatus
T_Hour
P_Hour
L_Hour
OtherConsid
CoursType
TEACHES
InstructorID (FK)
CourseCode (FK)
PREREQUISTE
MainCourseCode (FK)
PreCourseCode (FK)
WEEKLY_PLAN
CourseCode (FK)
WeekNo
T_Topic
U_Topic
L_Topic
TeachingMethod
Preliminary
ASSESSMENT
CourseCode (FK)
ActivityName
Count
Contribution
ECTS_WORKLOAD
CourseCode (FK)
ActivityName
Count
Duration
TotalWorkload
RESOURCE
CourseCode (FK)
Name
Type
LEARNING_OUTCOME
CourseCode (FK)
Description
MANDATORY
CourseCode (FK)
TECHNICAL_ELECTVIE
CourseCode (FK)
NON_DEPT_ELECTIVE
CourseCode (FK)
SOCIAL_ELECTIVE
CourseCode (FK)
PEDAGOGICAL_FORMATION
CourseCode (FK)

IMPLEMENTATION-PYHSICAL MODEL

```sql
CREATE DATABASE ErasmusProjectDB;
GO
USE ErasmusProjectDB;
GO


CREATE TABLE UNIVERSITY (
    UniversityID INT PRIMARY KEY,
    Name NVARCHAR(100),
    City NVARCHAR(50),
    Country NVARCHAR(50),
    Erasmus_Code VARCHAR(20),
    Type VARCHAR(20),
    Website VARCHAR(100)
);

CREATE TABLE SEMESTER (
    SemID INT PRIMARY KEY,
    Season VARCHAR(10),
    SemNumber INT
);


CREATE TABLE FACULTY (
    FacultyID INT PRIMARY KEY,
    Name NVARCHAR(100),
    WebAddress VARCHAR(100),
    UniversityID INT,
    FOREIGN KEY (UniversityID) REFERENCES UNIVERSITY(UniversityID)
);

CREATE TABLE DEPARTMENT (
    DeptID INT PRIMARY KEY,
    Name NVARCHAR(100),
    Website VARCHAR(100),
    FacultyID INT,
    FOREIGN KEY (FacultyID) REFERENCES FACULTY(FacultyID)
);
```

```sql
CREATE TABLE INSTRUCTOR (
    InstructorID INT PRIMARY KEY,
    FullName NVARCHAR(100),
    Title NVARCHAR(50),
    Email VARCHAR(100),
    PhoneExt VARCHAR(20),
    OfficeNo VARCHAR(20),
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES DEPARTMENT(DeptID)
);


CREATE TABLE COURSE (
    CourseCode VARCHAR(20) PRIMARY KEY,
    Name NVARCHAR(150),
    ECTS INT,
    Credit INT,
    Level VARCHAR(20),
    Language VARCHAR(20),
    ModeOfDelivery VARCHAR(50),
    Description NVARCHAR(MAX),
    Purpose NVARCHAR(MAX),
    InternshipStatus NVARCHAR(50),
    T_Hour INT,
    P_Hour INT,
    L_Hour INT,
    OtherConsid NVARCHAR(200),
    -- Foreign Keys
    DeptID INT,
    SemID INT,
    CoordinatorID INT, -- Alias kullandık (Instructor tablosuna gider)
    Course_Type VARCHAR(50), -- Discriminator (Mandatory, Elective vs.)

    FOREIGN KEY (DeptID) REFERENCES DEPARTMENT(DeptID),
    FOREIGN KEY (SemID) REFERENCES SEMESTER(SemID),
    FOREIGN KEY (CoordinatorID) REFERENCES INSTRUCTOR(InstructorID)
);
```

```sql
CREATE TABLE WEEKLY_PLAN (
    CourseCode VARCHAR(20),
    WeekNo INT,
    T_Topic NVARCHAR(200),
    U_Topic NVARCHAR(200),
    L_Topic NVARCHAR(200),
    TeachingMethod NVARCHAR(100),
    Preliminary NVARCHAR(200),
    PRIMARY KEY (CourseCode, WeekNo),
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE ASSESSMENT (
    CourseCode VARCHAR(20),
    ActivityName NVARCHAR(100),
    Count INT,
    Contribution INT,
    PRIMARY KEY (CourseCode, ActivityName),
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE ECTS_WORKLOAD (
    CourseCode VARCHAR(20),
    ActivityName NVARCHAR(100),
    Count INT,
    Duration INT,
    TotalWorkload INT,
    PRIMARY KEY (CourseCode, ActivityName),
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE RESOURCE (
    CourseCode VARCHAR(20),
    Name NVARCHAR(255),
    Type VARCHAR(50),
    PRIMARY KEY (CourseCode, Name),
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);
```

```sql
CREATE TABLE LEARNING_OUTCOME (
    CourseCode VARCHAR(20),
    Description NVARCHAR(MAX),
    -- SQL Server'da NVARCHAR(MAX) Primary Key olamaz, o yüzden Identity ID ekledik
    OutcomeID INT IDENTITY(1,1),
    PRIMARY KEY (OutcomeID),
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE INSTRUCTOR_EDUCATION (
    InstructorID INT,
    Year INT,
    University NVARCHAR(100),
    Degree_Level VARCHAR(50),
    DepartmentName NVARCHAR(100),
    PRIMARY KEY (InstructorID, Year, Degree_Level),
    FOREIGN KEY (InstructorID) REFERENCES INSTRUCTOR(InstructorID)
);

CREATE TABLE INSTRUCTOR_WORK_EXP (
    InstructorID INT,
    Company NVARCHAR(100),
    Role NVARCHAR(100),
    PRIMARY KEY (InstructorID, Company, Role),
    FOREIGN KEY (InstructorID) REFERENCES INSTRUCTOR(InstructorID)
);


CREATE TABLE TEACHES (
    InstructorID INT,
    CourseCode VARCHAR(20),
    PRIMARY KEY (InstructorID, CourseCode),
    FOREIGN KEY (InstructorID) REFERENCES INSTRUCTOR(InstructorID),
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE PREREQUISITE (
    MainCourseCode VARCHAR(20),
    PreCourseCode VARCHAR(20),
    PRIMARY KEY (MainCourseCode, PreCourseCode),
    FOREIGN KEY (MainCourseCode) REFERENCES COURSE(CourseCode),
    FOREIGN KEY (PreCourseCode) REFERENCES COURSE(CourseCode)
);
```

```sql
CREATE TABLE INSTRUCTOR_RESEARCH_AREAS (
    InstructorID INT,
    Area_Name NVARCHAR(100),
    PRIMARY KEY (InstructorID, Area_Name),
    FOREIGN KEY (InstructorID) REFERENCES INSTRUCTOR(InstructorID)
);


CREATE TABLE MANDATORY (
    CourseCode VARCHAR(20) PRIMARY KEY,
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE TECHNICAL_ELECTIVE (
    CourseCode VARCHAR(20) PRIMARY KEY,
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE NON_DEPT_ELECTIVE (
    CourseCode VARCHAR(20) PRIMARY KEY,
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE SOCIAL_ELECTIVE (
    CourseCode VARCHAR(20) PRIMARY KEY,
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);

CREATE TABLE PEDAGOGICAL_FORMATION (
    CourseCode VARCHAR(20) PRIMARY KEY,
    FOREIGN KEY (CourseCode) REFERENCES COURSE(CourseCode)
);
```

POPULATE THE DATABASE -INSERT

```
-- 1. ADIM: HOCA TANIMLAMA
IF NOT EXISTS (SELECT * FROM INSTRUCTOR WHERE InstructorID = 3002)
BEGIN
    INSERT INTO INSTRUCTOR (InstructorID, FullName, Title, Email, PhoneExt, OfficeNo, DeptID)
    VALUES (3002, 'Murat Osman Unalir', 'Prof. Dr.', 'murat.unalir@ege.edu.tr', '+90 232 311 2020', 'E-205', 204);
END
```

```
-- 2. ADIM: DERS KARTI (INTRODUCTION TO DATABASES)
IF NOT EXISTS (SELECT * FROM COURSE WHERE CourseCode = '501003222023')
BEGIN
    INSERT INTO COURSE (
        CourseCode, Name, ECTS, Credit, Level, Language, ModeOfDelivery,
        Description, Purpose, InternshipStatus, T_Hour, P_Hour, L_Hour, OtherConsid,
        DeptID, SemID, CoordinatorID, Course_Type
    )
    VALUES (
        '501003222023',
        'INTRODUCTION TO DATABASES',
        6, -- ECTS
        4, -- Kredi
        'Undergraduate',
        'English', --
        'Face-to-Face',
        'Relational Data Model, SQL, Conceptual Modeling, Database Normalization, Query Processing, Transaction Proce
        'The aim of the course is to let the students to know and understand database concept, relational model, SQL
        'None',
        3, -- Teori
        2, -- Uygulama (Pratik)
        0,
        'None',
        204, -- Ege Computer Engineering
        20241, -- Güz Dönemi
        3002, -- Prof. Dr. Murat Osman Ünalır
        'Mandatory'
    );
END
```

```
-- Zorunlu Ders Tablosuna Ekleme
IF NOT EXISTS (SELECT * FROM MANDATORY WHERE CourseCode = '501003222023')
BEGIN
    INSERT INTO MANDATORY (CourseCode) VALUES ('501003222023');
END

-- Hoca ile Ders Eşleşmesi
IF NOT EXISTS (SELECT * FROM TEACHES WHERE CourseCode = '501003222023')
BEGIN
    INSERT INTO TEACHES (InstructorID, CourseCode) VALUES (3002, '501003222023');
END
```

```sql
-- 3. ADIM: HAFTALIK PLAN (WEEKLY_PLAN)
INSERT INTO WEEKLY_PLAN (CourseCode, WeekNo, T_Topic, TeachingMethod) VALUES
('501003222023', 1, 'Database System Concepts and Architecture', 'Lecture'),
('501003222023', 2, 'Relational Data Model and Relational Database Constraints', 'Lecture'),
('501003222023', 3, 'Basic SQL', 'Lecture'),
('501003222023', 4, 'Advanced SQL: Complex Queries, Triggers, Views', 'Lecture'),
('501003222023', 5, 'Relational Algebra and Relational Calculus', 'Lecture'),
('501003222023', 6, 'Data Modeling with ER', 'Lecture'),
('501003222023', 7, 'EER Modeling', 'Lecture'),
('501003222023', 8, 'Midterm Exam', 'Exam'),
('501003222023', 9, 'Practical Database Design Methodology and UML Diagrams', 'Lecture'),
('501003222023', 10, 'Fundamentals of Functional Dependency and Normalization', 'Lecture'),
('501003222023', 11, 'Relational Database Design Algorithms and Advanced Dependencies', 'Lecture'),
('501003222023', 12, 'Algorithms for Query Processing and Optimization', 'Lecture'),
('501003222023', 13, 'Introduction to Transaction Processing and Theory', 'Lecture'),
('501003222023', 14, 'Concurrency Control Techniques', 'Lecture'),
('501003222023', 15, 'Introduction to Information Retrieval and Web Crawling', 'Lecture'),
('501003222023', 16, 'Final Exam', 'Exam');
```

```sql
-- 4. ADIM: DEĞERLENDİRME
INSERT INTO ASSESSMENT (CourseCode, ActivityName, [Count], Contribution) VALUES
('501003222023', 'Term Project', 1, 28),
('501003222023', 'Midterm Exam', 1, 28),
('501003222023', 'Quizzes', 6, 14),
('501003222023', 'Final Exam', 1, 30);
```

```sql
-- 5. ADIM: İŞ YÜKÜ
INSERT INTO ECTS_WORKLOAD (CourseCode, ActivityName, [Count], Duration, TotalWorkload) VALUES
('501003222023', 'Midterm Exam', 1, 2, 2),
('501003222023', 'Final Exam', 1, 2, 2),
('501003222023', 'Attending Lectures', 14, 3, 42),
('501003222023', 'Practice', 14, 2, 28),
('501003222023', 'Problem Solving', 14, 1, 14),
('501003222023', 'Study for Midterm', 1, 22, 22),
('501003222023', 'Study for Final', 1, 28, 28),
('501003222023', 'Reading', 14, 3, 42);
```

```sql
-- 6. ADIM: KAYNAKLAR
INSERT INTO RESOURCE (CourseCode, Name, Type) VALUES
('501003222023', 'Ramez Elmasri and Shamkant Navathe - Fundamentals of Database Systems: Global Edition, 6/E', 'Book
```

```sql
-- 7. ADIM: ÖĞRENME ÇIKTILARI
INSERT INTO LEARNING_OUTCOME (CourseCode, Description) VALUES
('501003222023', 'To explain the concept of database.'),
('501003222023', 'To list the components of database architecture.'),
('501003222023', 'To describe the concepts of relational model and to use them.'),
('501003222023', 'To apply relational database operations.'),
('501003222023', 'To explain relationship type, set, role and structural constraints.'),
('501003222023', 'To use notational details related to ER diagrams.'),
('501003222023', 'To list the rules and constraints of normal forms.'),
('501003222023', 'To apply selectivity and cost computing in query optimization.');
```

TRIGGERS

```sql
USE ErasmusProjectDB;
GO

-- ============================================================
-- TRIGGER 1: Otomatik İş Yükü Hesaplama
-- Tablo: ECTS_WORKLOAD
-- Amaç: Kullanıcı 'Count' ve 'Duration' girdiğinde, sistem
-- 'TotalWorkload' alanını otomatik olarak (Count * Duration) hesaplar.
-- ============================================================

CREATE OR ALTER TRIGGER trg_CalculateTotalWorkload
ON ECTS_WORKLOAD
AFTER INSERT, UPDATE
AS
BEGIN
    -- Sonsuz döngüyü (Recursive Trigger) engellemek için kontrol
    IF TRIGGER_NESTLEVEL() > 1 RETURN;

    -- Inserted tablosundan gelen verilerle güncelleme yap
    UPDATE ECTS_WORKLOAD
    SET TotalWorkload = i.Count * i.Duration
    FROM ECTS_WORKLOAD w
    INNER JOIN inserted i ON w.CourseCode = i.CourseCode AND w.ActivityName = i.ActivityName;

    PRINT '>>> TRIGGER 1 CALISTI: Toplam is yuku otomatik hesaplandi.';
END;
GO
```

```sql
USE ErasmusProjectDB;
GO
-- TRIGGER 2: Değerlendirme Puanı Kontrolü
-- Tablo: ASSESSMENT
-- Amaç: Bir dersin sınav/ödev katkı payları toplamı 100'ü geçemez.
-- Geçerse hata verir ve işlemi iptal eder.

CREATE OR ALTER TRIGGER trg_CheckAssessmentTotal
ON ASSESSMENT
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @CourseCode NVARCHAR(50);
    DECLARE @TotalContribution INT;

    -- İşlem yapılan dersin kodunu al
    SELECT @CourseCode = CourseCode FROM inserted;

    -- O dersin toplam puanını hesapla
    SELECT @TotalContribution = SUM(Contribution)
    FROM ASSESSMENT
    WHERE CourseCode = @CourseCode;

    -- Kontrol: 100'ü geçti mi?
    IF @TotalContribution > 100
    BEGIN
        RAISERROR ('HATA: Bir dersin toplam degerlendirme puani 100''u gecemez! Islem iptal edildi.', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        PRINT '>>> TRIGGER 2 KONTROL ETTI: Puan toplami 100 siniri icinde.';
    END
END;
GO
```

```sql
USE ErasmusProjectDB;
GO

-- ============================================================
-- TRIGGER 3: Ders Türü Çakışma Kontrolü
-- Tablo: MANDATORY (Zorunlu Dersler)
-- Amaç: Bir ders 'Zorunlu' tablosuna eklenirken, 'Teknik Seçmeli'
-- tablosunda olup olmadığı kontrol edilir. Varsa engellenir.
-- ============================================================

CREATE OR ALTER TRIGGER trg_CheckCourseTypeConflict
ON MANDATORY
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @CourseCode NVARCHAR(50);
    -- Eklenen dersin kodunu al
    SELECT @CourseCode = CourseCode FROM inserted;

    -- Diğer tabloda (TECHNICAL_ELECTIVE) var mı diye bak
    IF EXISTS (SELECT * FROM TECHNICAL_ELECTIVE WHERE CourseCode = @CourseCode)
    BEGIN
        RAISERROR ('HATA: Bu ders zaten TEKNIK SECMELI olarak kayitli. Ayni anda ZORUNLU olamaz!', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        PRINT '>>> TRIGGER 3 ONAYLADI: Ders turu cakismasi yok.';
    END
END;
GO
```

CHECK CONSTRAINTS

```
USE ErasmusProjectDB;
GO

-- ============================================================
-- CONSTRAINT 1: AKTS Geçerlilik Kontrolü
-- Tablo: COURSE
-- Kural: ECTS değeri 0'dan büyük ve 30'a eşit veya küçük olmalıdır.
-- ============================================================

ALTER TABLE COURSE
ADD CONSTRAINT CHK_ECTS_Valid
CHECK (ECTS > 0 AND ECTS <= 30);

PRINT '>>> CONSTRAINT 1 EKLENDI: AKTS degerleri artik 1-30 arasinda olmak zorunda.';
GO
```

```
USE ErasmusProjectDB;
GO

-- ============================================================
-- CONSTRAINT 2: Pozitif Ders Saati Kontrolü
-- Tablo: COURSE
-- Kural: Teorik (T), Pratik (P) ve Lab (L) saatleri 0'dan küçük olamaz.
-- ============================================================

ALTER TABLE COURSE
ADD CONSTRAINT CHK_Hours_NonNegative
CHECK (T_Hour >= 0 AND P_Hour >= 0 AND L_Hour >= 0);

PRINT '>>> CONSTRAINT 2 EKLENDI: Ders saatleri (T, P, L) negatif girilemez.';
GO
```

```
USE ErasmusProjectDB;
GO

-- ============================================================
-- CONSTRAINT 3: Eğitim Dili (Language) Kontrolü
-- Tablo: COURSE
-- Kural: Ders eşdeğerliliği (Equivalency) kuralları gereği,
-- dersin dili sistemde tanımlı dillerden (Turkish veya English) biri olmalıdır.
-- ============================================================

ALTER TABLE COURSE
ADD CONSTRAINT CHK_Language_Valid
CHECK (Language IN ('Turkish', 'English'));

PRINT '>>> CONSTRAINT 3 EKLENDI: Egitim dili sadece Turkish veya English olabilir.';
GO
```

INSERT-UPDATE-DELETE

INSERT

```
INSERT INTO COURSE (
        CourseCode, Name, ECTS, Credit, Level, Language, ModeOfDelivery,
        Description, Purpose, InternshipStatus, T_Hour, P_Hour, L_Hour, OtherConsid,
        DeptID, SemID, CoordinatorID, Course_Type
    )
    VALUES (
        '501001052022',
        'PHYSICS-I',
        5,
        4,
        'Undergraduate',
        'English',
        'Face-to-Face',
        'Measurement and units, Vectors, Motion in 1D/2D/3D, Newton Laws, Work and Kinetic Energy, Potential Energy,
        'To develop students ability to learn the basic laws of physics, interpret basic physical problems and solve
        'None', 3, 2, 0, 'None',
        204,
        20241,
        3004,
        'Mandatory'
    );
END
```

```
INSERT INTO WEEKLY_PLAN (CourseCode, WeekNo, T_Topic, TeachingMethod) VALUES
('501001052022', 1, 'Introduction, Measurement and Vectors (Introduction, length, mass and time standards)', 'Lectur
('501001052022', 2, 'Motion in one dimension (Average velocity, acceleration)', 'Lecture'),
('501001052022', 3, 'Vectors (Scalar and vectors quantities, addition)', 'Lecture'),
('501001052022', 4, 'Motion in two dimensions (Projectile motion, circular motion)', 'Lecture'),
('501001052022', 5, 'Laws of motion (Force, Newton Laws, weight)', 'Lecture'),
('501001052022', 6, 'Applications of Newton Laws (Friction, Drag force)', 'Lecture'),
('501001052022', 7, 'Work and Kinetic Energy', 'Lecture'),
('501001052022', 8, 'Potential Energy and Conservation of Energy', 'Lecture'),
('501001052022', 9, 'Static Equilibrium (Center of mass)', 'Lecture'),
('501001052022', 10, 'Linear Momentum and Collisions', 'Lecture'),
('501001052022', 11, 'Rotation of a rigid body about a fixed axis', 'Lecture'),
('501001052022', 12, 'Calculation of moment of inertia, Torque', 'Lecture'),
('501001052022', 13, 'Rolling motion, Angular momentum', 'Lecture'),
('501001052022', 14, 'Final Review', 'Lecture');

INSERT INTO RESOURCE (CourseCode, Name, Type) VALUES
('501001052022', 'Fundamentals of Physics, 9th Edition - Halliday & Resnick', 'Book'),
('501001052022', 'University Physics with Modern Physics - Young & Freedman', 'Book'),
('501001052022', 'Physics for Scientists and Engineers - Serway & Jewett', 'Book');
```

```
INSERT INTO ASSESSMENT (CourseCode, ActivityName, [Count], Contribution) VALUES
('501001052022', 'Midterm Exam', 1, 35),
('501001052022', 'Laboratory', 1, 15),
('501001052022', 'Final Exam', 1, 50);

INSERT INTO ECTS_WORKLOAD (CourseCode, ActivityName, [Count], Duration, TotalWorkload) VALUES
('501001052022', 'Midterm Exam', 1, 1, 1),
('501001052022', 'Study for Midterm', 1, 20, 20),
('501001052022', 'Self Study', 8, 1, 8),
('501001052022', 'Attending Lectures', 14, 3, 42),
('501001052022', 'Final Exam', 1, 2, 2),
('501001052022', 'Study for Final', 1, 30, 30),
('501001052022', 'Laboratory', 8, 2, 16),
('501001052022', 'Homework', 14, 1, 14),
('501001052022', 'Problem Solving', 14, 1, 14),
('501001052022', 'Report Preparation', 8, 1, 8);
```

UPDATE

```sql
UPDATE WEEKLY_PLAN
SET T_Topic = 'Introduction to Physics: Units, Dimensions, and Vector Analysis'
WHERE CourseCode = '501001052022' AND WeekNo = 1;

UPDATE INSTRUCTOR
SET OfficeNo = 'E-405'
WHERE InstructorID = 3004;
```

DELETE

```sql
DELETE FROM RESOURCE
WHERE CourseCode = '501001052022'
AND Name = 'University Physics with Modern Physics - Young & Freedman';

DELETE FROM RESOURCE
WHERE CourseCode = '501001052022'
AND Name = 'Physics for Scientists and Engineers - Serway & Jewett';
```

SELECT STATEMENTS

UNIVERSITY – COURSE REPORT

```
USE ErasmusProjectDB;
GO

-- ============================================================
-- UNIVERSITY & COURSE HIERARCHY REPORT
-- Amaç: Üniversite/Fakülte/Bölüm isimlerini tekrar etmeden (Grouping Visual),
-- altındaki dersleri listelemek.
-- Teknik: JOIN + WINDOW FUNCTION (LAG)
-- ============================================================
```

```
SELECT
    -- 1. ÜNİVERSİTE: Bir önceki satırla aynıysa boş göster
    CASE
        WHEN U.Name = LAG(U.Name) OVER(ORDER BY U.Name, F.Name, D.Name, C.Name)
        THEN ''
        ELSE U.Name
    END AS [University],

    -- 2. FAKÜLTE: Bir öncekiyle aynıysa ve üniversite değişmediyse boş göster
    CASE
        WHEN F.Name = LAG(F.Name) OVER(ORDER BY U.Name, F.Name, D.Name, C.Name)
             AND U.Name = LAG(U.Name) OVER(ORDER BY U.Name, F.Name, D.Name, C.Name)
        THEN ''
        ELSE F.Name
    END AS [Faculty],

    -- 3. BÖLÜM: Bir öncekiyle aynıysa boş göster
    CASE
        WHEN D.Name = LAG(D.Name) OVER(ORDER BY U.Name, F.Name, D.Name, C.Name)
             AND F.Name = LAG(F.Name) OVER(ORDER BY U.Name, F.Name, D.Name, C.Name)
        THEN ''
        ELSE D.Name
    END AS [Department],

    -- 4. DERS BİLGİLERİ
    C.Name AS [Course Name],
    C.CourseCode AS [Code],
    C.ECTS AS [ECTS],
    C.Language AS [Language]

FROM UNIVERSITY U
JOIN FACULTY F ON U.UniversityID = F.UniversityID
JOIN DEPARTMENT D ON F.FacultyID = D.FacultyID
JOIN COURSE C ON D.DeptID = C.DeptID
ORDER BY U.Name, F.Name, D.Name, C.Name;
```

QUERY RESULT

	University	Faculty	Department	Course Name	Code	ECTS	Language
1	Bahçeşehir University	Faculty of Engineering and Natural Sciences	Software Engineering	Information Security	COP4456	6	English
2				Physics II	PHY1002	7	English
3				Probability and Statistics	MAT3026	6	English
4				Software Architecture	SEN3006	7	English
5	Ege University	Faculty of Engineering	Computer Engineering	COMPLEX NETWORK APPLICATIONS	501004232023	5	English
6				COMPUTER NETWORKS	501003452022	5	English
7				INTRODUCTION TO DATABASES	501003222023	6	English
8				Logic Design	501002232023	5	English
9				PHYSICS-I	501001052022	5	English
10	TOBB University of Economics and Technology	Faculty of Engineering	Artificial Intelligence Engineering	Bilgisayar Programlama II	BIL211	8	Turkish
11				Computer Programming I	BIL113	8	Turkish
12				Data Mining	BIL476	6	Turkish
13				Veritabani Sistemleri	BIL372	8	Turkish

SEARCH BY ECTS

```sql
USE ErasmusProjectDB;
GO

-- ============================================================
-- STORED PROCEDURE: AKTS'ye Göre Ders Arama Motoru
-- Amaç: Kullanıcıdan sadece bir sayı alır ve o kredideki dersleri listeler.
-- Kullanımı: EXEC sp_SearchCourseByECTS int;
-- ============================================================

CREATE OR ALTER PROCEDURE sp_SearchCourseByECTS
    @TargetECTS INT -- (Kullanıcının gireceği değer parametre olarak tanımlandı)
AS
BEGIN
    SET NOCOUNT ON; -- (Gereksiz "x rows affected" mesajını gizler, performansı artırır)

    PRINT '>>> ARANAN KRITER: ' + CAST(@TargetECTS AS NVARCHAR(5)) + ' AKTS DEGERINE SAHIP DERSLER GETIRILIYOR...';

    SELECT
        U.Name AS [University (Host)],
        D.Name AS [Department],
        C.CourseCode AS [Code],
        C.Name AS [Course Name],
        C.ECTS AS [ECTS],
        C.Language AS [Lang],
        C.ModeOfDelivery AS [Mode],
        LEFT(C.Description, 50) + '...' AS [Short Description]
    FROM COURSE C
    JOIN DEPARTMENT D ON C.DeptID = D.DeptID
    JOIN FACULTY F ON D.FacultyID = F.FacultyID
    JOIN UNIVERSITY U ON F.UniversityID = U.UniversityID
    WHERE C.ECTS = @TargetECTS
    ORDER BY U.Name, C.Name;
END;
GO
```

*Execute the procedure

```sql
EXEC sp_SearchCourseByECTS 6;
```

QUERY RESULT

	University (Host)	Department	Code	Course Name	ECTS	Lang	Mode	Short Description
1	Bahçeşehir University	Software Engineering	COP4456	Information Security	6	English	Face-to-Face	Security components, principles of cyber security,...
2	Bahçeşehir University	Software Engineering	MAT3026	Probability and Statistics	6	English	Face-to-Face	Counting, probability rules, random variables, dis...
3	Ege University	Computer Engineering	501003222023	INTRODUCTION TO DATABASES	6	English	Face-to-Face	Relational Data Model, SQL, Conceptual Modeling, ...
4	TOBB University of Economics and Technology	Artificial Intelligence Engineering	BIL476	Data Mining	6	Turkish	Face-to-Face	Basic Data Mining Concepts, Data, Data Warehous...

EQUIVALENCE CHECK

```
USE ErasmusProjectDB;
GO

-- ============================================================
-- ERASMUS DERS EŞDEĞERLİLİK RAPORU
-- Format: Yan Yana Karşılaştırma (Side-by-Side Comparison)
-- ============================================================

DECLARE @SourceCode NVARCHAR(20) = '501002232023'; -- Örn: Ege (Logic Design)
DECLARE @TargetCode NVARCHAR(20) = 'BIL113';       -- Örn: TOBB (Prog I)

-- 1. BAŞLIK: HANGİ DERSLERİ KARŞILAŞTIRIYORUZ?
-- ============================================================
PRINT '>>> DERS KARSILASTIRMA RAPORU ';
```

```
SELECT
    U.Name AS University,
    D.Name AS Department,
    F.Name AS Faculty,
    C.Name AS Course_Name,
    C.CourseCode,
    C.ECTS AS Official_ECTS,
    SUM(W.TotalWorkload) AS Calculated_Total_Hours,
    (SUM(W.TotalWorkload) / 30) AS Estimated_ECTS, -- (1 AKTS ~ 30 Saat)
    CASE
        WHEN C.CourseCode = @SourceCode THEN 'SOURCE'
        ELSE 'TARGET'
    END AS Status
FROM
    COURSE C
JOIN
    DEPARTMENT D ON C.DeptID = D.DeptID
JOIN
    FACULTY F ON D.FacultyID = F.FacultyID
JOIN
    UNIVERSITY U ON F.UniversityID = U.UniversityID
JOIN
    ECTS_WORKLOAD W ON C.CourseCode = W.CourseCode
WHERE
    C.CourseCode IN (@SourceCode, @TargetCode)
GROUP BY
    U.Name, D.Name, F.Name, C.Name, C.CourseCode, C.ECTS;
```

```sql
-- ============================================================
-- TABLO 2: STRUCTURAL COMPARISON (YAPISAL DENKLİK)
-- Amaç: Dersin yapısal iskeletini (AKTS, Kredi, Dil vb.) kıyaslamak.
-- ============================================================
PRINT ' ';
PRINT '>>> 2. STRUCTURAL COMPARISON (Yapisal Karsilastirma)';
```

```sql
SELECT
    Attributes AS [Criteria],
    MAX(CASE WHEN CourseCode = @SourceCode THEN Value END) AS [Source Course],
    MAX(CASE WHEN CourseCode = @TargetCode THEN Value END) AS [Target Course],
    CASE
        WHEN MAX(CASE WHEN CourseCode = @SourceCode THEN Value END) = MAX(CASE WHEN CourseCode = @TargetCode THEN Va
        THEN 'MATCH'
        ELSE 'DIFFERENT'
    END AS [Equivalence Check]
FROM (
    -- 1. ECTS (En önemli kriter en başta)
    SELECT CourseCode, 'ECTS' AS Attributes, CAST(ECTS AS NVARCHAR(50)) AS Value
    FROM COURSE WHERE CourseCode IN (@SourceCode, @TargetCode)

    UNION ALL

    -- 2. Local Credit
    SELECT CourseCode, 'Local Credit', CAST(Credit AS NVARCHAR(50))
    FROM COURSE WHERE CourseCode IN (@SourceCode, @TargetCode)

    UNION ALL

    -- 3. Language
    SELECT CourseCode, 'Language', Language
    FROM COURSE WHERE CourseCode IN (@SourceCode, @TargetCode)

    UNION ALL
```

```sql
    -- 4. Mode of Delivery
    SELECT CourseCode, 'Mode of Delivery', ModeOfDelivery
    FROM COURSE WHERE CourseCode IN (@SourceCode, @TargetCode)


) AS BaseData
GROUP BY Attributes
ORDER BY Attributes ASC; -- Numaralandırdığımız için (1, 2, 3...) sırasıyla gelir.
```

```sql
-- ============================================================
-- TABLO 3: İÇERİK ÖZETİ (YAN YANA)
-- Amaç: Dersin amacı ve tanımı benziyor mu?
-- ============================================================
PRINT ' ';
PRINT '>>> 3. ICERIK VE AMAC KARSILASTIRMASI';

SELECT
    'Course Purpose' AS [Section],
    (SELECT Purpose FROM COURSE WHERE CourseCode = @SourceCode) AS [Source_Course],
    (SELECT Purpose FROM COURSE WHERE CourseCode = @TargetCode) AS [Target_Course]
UNION ALL
SELECT
    'Course Description',
    (SELECT Description FROM COURSE WHERE CourseCode = @SourceCode),
    (SELECT Description FROM COURSE WHERE CourseCode = @TargetCode);
```

```sql
-- ============================================================
-- TABLO 4: LEARNING OUTCOMES COMPARISON (KAZANIM KIYASLAMASI)
-- Amaç: Öğrenme çıktılarını yan yana, madde numarası olmadan listelemek.
-- ============================================================
PRINT ' ';
PRINT '>>> 4. LEARNING OUTCOMES COMPARISON';

WITH SourceOutcomes AS (
    SELECT Description, ROW_NUMBER() OVER(ORDER BY Description) AS RowNo
    FROM LEARNING_OUTCOME WHERE CourseCode = @SourceCode
),
TargetOutcomes AS (
    SELECT Description, ROW_NUMBER() OVER(ORDER BY Description) AS RowNo
    FROM LEARNING_OUTCOME WHERE CourseCode = @TargetCode
)
SELECT
    ISNULL(S.Description, '') AS [Source Course Learning Outcomes], -- NULL ise boşluk bas
    ISNULL(T.Description, '') AS [Target Course Learning Outcomes]
FROM SourceOutcomes S
FULL OUTER JOIN TargetOutcomes T ON S.RowNo = T.RowNo;
```

QUERY RESULT

	University	Department	Faculty	Course_Name	CourseCode	Official_ECTS	Calculated_Total_Hours	Estimated_ECTS	Status
1	Ege University	Computer Engineering	Faculty of Engineering	Logic Design	501002232023	5	150	5	SOURCE
2	TOBB University of Economics and Technology	Artificial Intelligence Engineering	Faculty of Engineering	Computer Programming I	BIL113	8	248	8	TARGET

	Criteria	Source Course	Target Course	Equivalence Check
1	ECTS	5	8	DIFFERENT
2	Language	English	Turkish	DIFFERENT
3	Local Credit	4	4	MATCH
4	Mode of Delivery	Face-to-Face	Face-to-Face	MATCH

	Section	Source_Course	Target_Course
1	Course Purpose	To enable students to analyze, design and implem...	To provide basic logic of programming using Java la...
2	Course Description	Digital Systems, Combinational Logic Circuits, Seq...	Covers basic properties of computer programming ...

	Source Course Learning Outcomes	Target Course Learning Outcomes
1	To analyze sequential circuits.	Learns and applies programming logic.
2	To define combinational circuit components.	Learns class and method fundamentals.
3	To define sequential circuit components.	Learns data types, variables, assignment stateme...
4	To design combinational circuits and solve relat...	Learns flow control, decision making and loops.
5	To design sequential circuits.	Learns recursion.
6	To perform binary operations.	
7	To simplify functions via algebraic and Karnaugh...	

Erasmus Course Equivalence Report: Design and Logic Analysis

1. Design Philosophy: The Funnel Approach

While designing this report, the decision-making workflow of an Erasmus Coordinator or Equivalence Commission was simulated. Instead of examining two courses in a single complex table, a 4-stage hierarchical analysis funnel progressing from "Macro-to-Micro" was created. This structure reduces cognitive load and ensures the most accurate decision is made using the elimination method.

2. Logical Rationales of the Stages

STAGE 1: Identity & Workload Verification

- **Analysis Type:** Quantitative Validation.
- **Design Logic:** At the beginning of the report, not only the course name or code but also a "Real Workload" analysis is presented.
- **Rationale:** Paper-based official ECTS can sometimes be misleading. At this stage, the Calculated Total Hours derived from the ECTS_WORKLOAD table (lecture hours, exam duration, homework load, etc.) are compared with the official ECTS. If there is an inconsistency between the official credit and the effort spent by the student, the equivalence process is questioned at the very beginning. This step guarantees Data Integrity.

STAGE 2: Structural and Technical Filtering (Structural Gatekeeper)

- **Analysis Type:** Technical Compliance (Go/No-Go Decision).
- **Design Logic:** Before diving into course content, "Technical Prerequisites" are checked. By placing Source and Target columns side-by-side, Credit, Language, and Mode of Delivery are compared.
- **Rationale:** No matter how similar the content is, a course with mismatched language (e.g., English vs. Turkish) or insufficient credits cannot be deemed equivalent. This stage is a Filter Gate that prevents time loss. The dynamic [Equivalence Check] column provides instant status information to the decision-maker.

STAGE 3: Scope and Semantic Context

- **Analysis Type:** Qualitative Preview.
- **Design Logic:** The "Spirit" and "Purpose" of courses that pass the technical requirements are examined at this stage. Course Purpose and Description fields are compared.
- **Rationale:** Looking directly at learning outcomes item-by-item can sometimes cause missing the big picture. This stage reveals whether the course is theoretical or applied and if the general visions align. It is critical for understanding the Course Context.

STAGE 4: Competency-Based Deep Analysis (Competency-Based Matching)

- **Analysis Type:** Academic Equivalence.
- **Design Logic:** This is the most detailed and final stage of the report. Learning Outcomes are stripped of item numbers and listed in corresponding rows.
- **Rationale:** According to the Bologna Process and MÜDEK/ABET standards, equivalence is granted based on "Acquired Competencies," not topic headings. The fact that one side of the table is full while the other is empty (NULL) indicates a Curriculum Gap. The final equivalence decision is made based on the matching rate in this table.

3. Conclusion

This 4-stage SQL report is not a simple data listing operation; it is a Decision Support System. By guiding the user from general data to technical details, and from there to academic competencies, it ensures an error-free equivalence process is conducted.