INFORMATION TECHNOLOGY INSTITUTE

ITI Power bi developer track

Examination System

Feb 2024

Made by: -

- Shimaa Abdelaal
- Rahma Tarek
- Esraa Emad
- Marwa Samir

- # Introduction:

  The project's purpose is to design automated system that can perform online exams.

- # Database design:

Entities identifications: -

1- Students:

Represents data about students, like **st_id, st_fname, st_lname, st_age, st_address, st_email.**

2- Departments:

Represents data about departments, like **dept_id, dept_name, dept_location, mgr_hiredate.**

3- Instructors:

Represents data about instructors, like **ins_id, ins_name, salary.**

4- Courses:

Represents data about courses, like **crs_id, crs_name, crs_duration.**

5- Topics:

Represents data about topics, like **top_id, top_name.**

6- Skills:

Represents data about skills, like **sk_id, sk_name.**

7- Projects:

Represents data about projects, like **pro_id, pro_name, start_date, end_date.**

8- Company:

Represents data about company, like **company_id, company_name, location.**

9- Position:

Represents data about positions, like **position_id, position_name, salary.**

10- Training:

Represents data about training, like **training_id, training_name, start_date, end_date.**

11- Freelancing:

Represents data about freelancing, like **freelancing_id, freelancing_name.**

12- Certificates:

Represents data about certificates, like **cre_id, cre_name, issuer, issue_date.**

13- Exams:

Represents data about exams, like **exam_id, exam_name, exam_date.**

14- Exam_questions:

Represents data about exam_questions, like **question_id, question_text, question_type.**

15- Question_option:

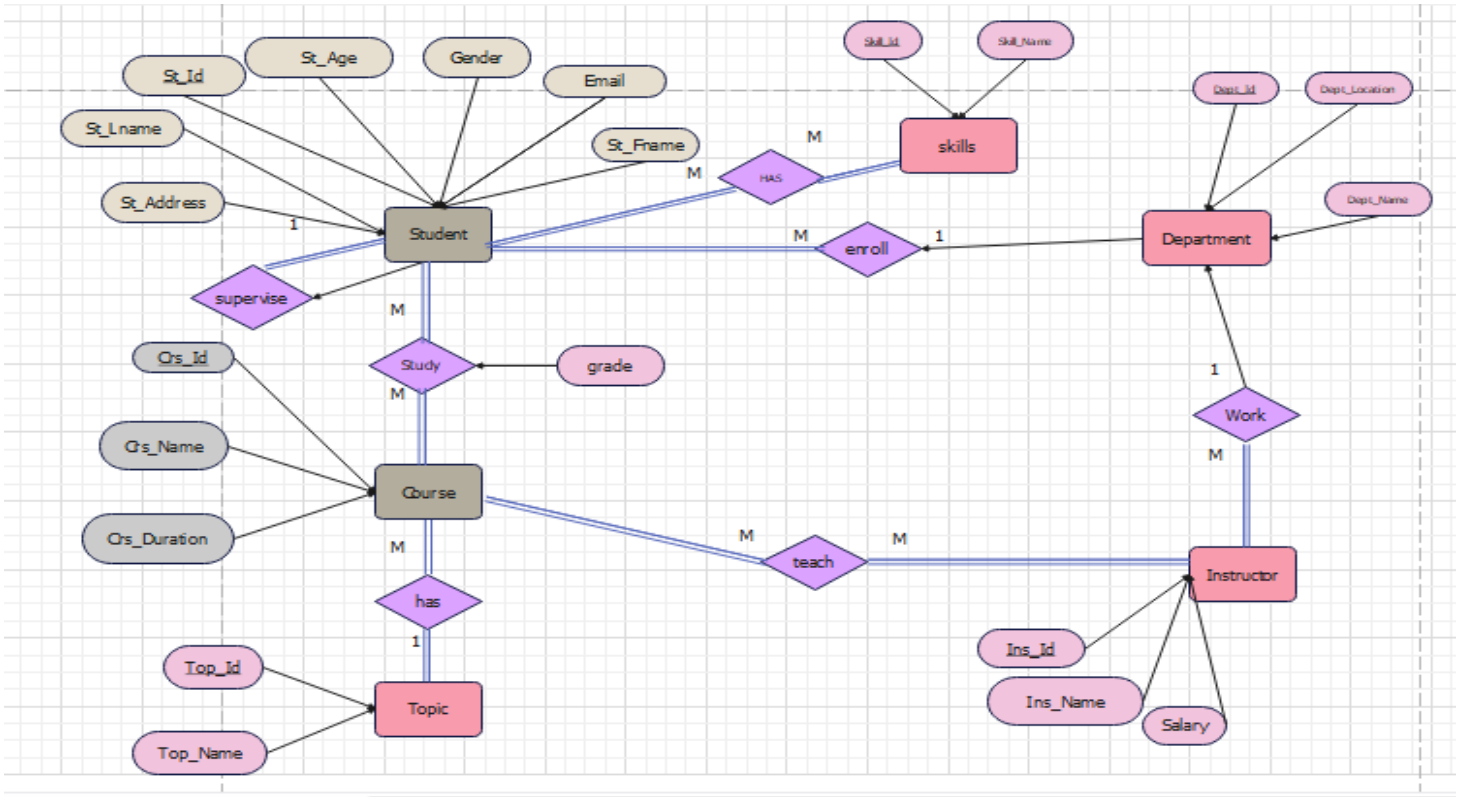
Represents data about question_options, like **option_id, option_text, is_currect.**

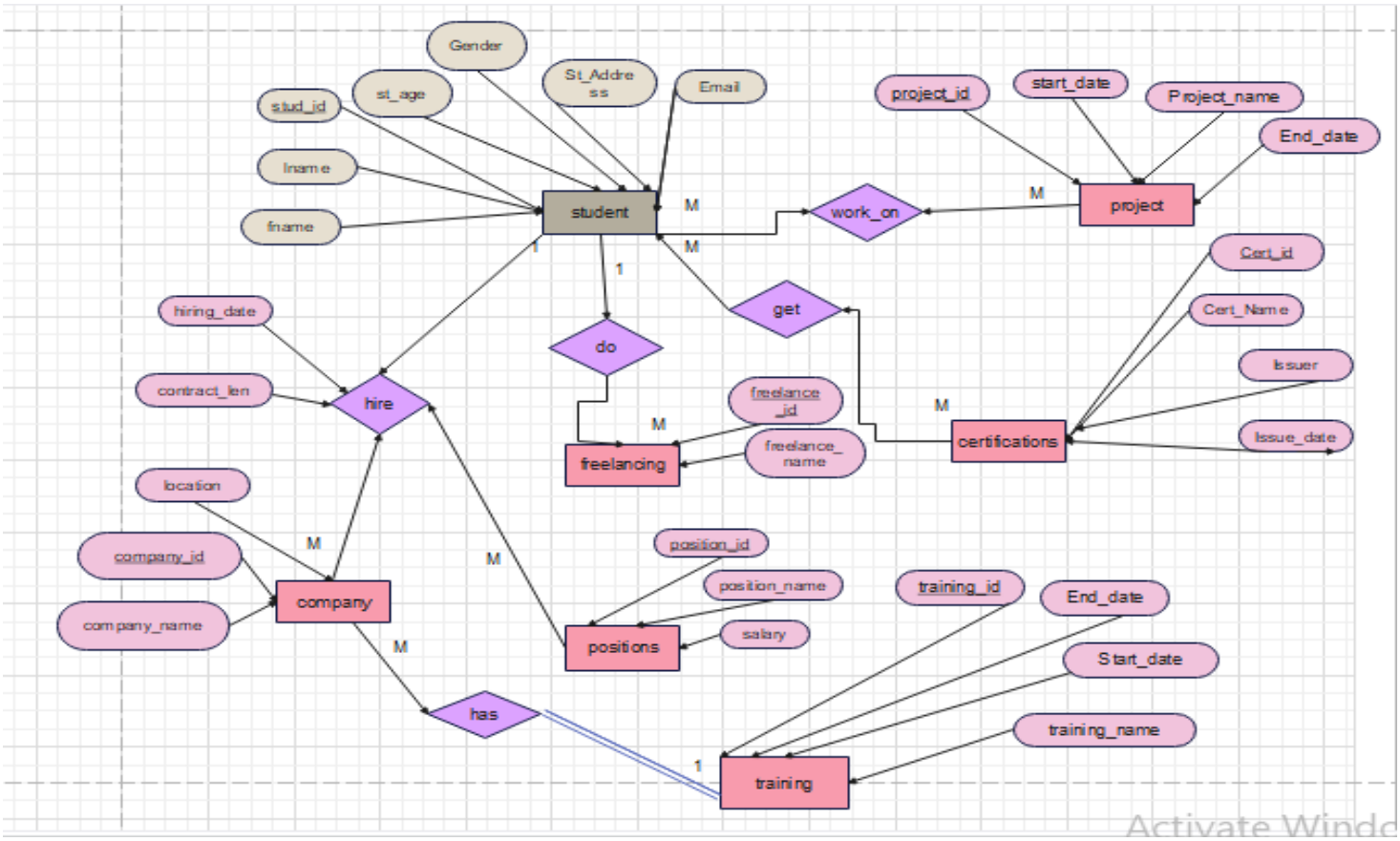
16- Exam_answers:

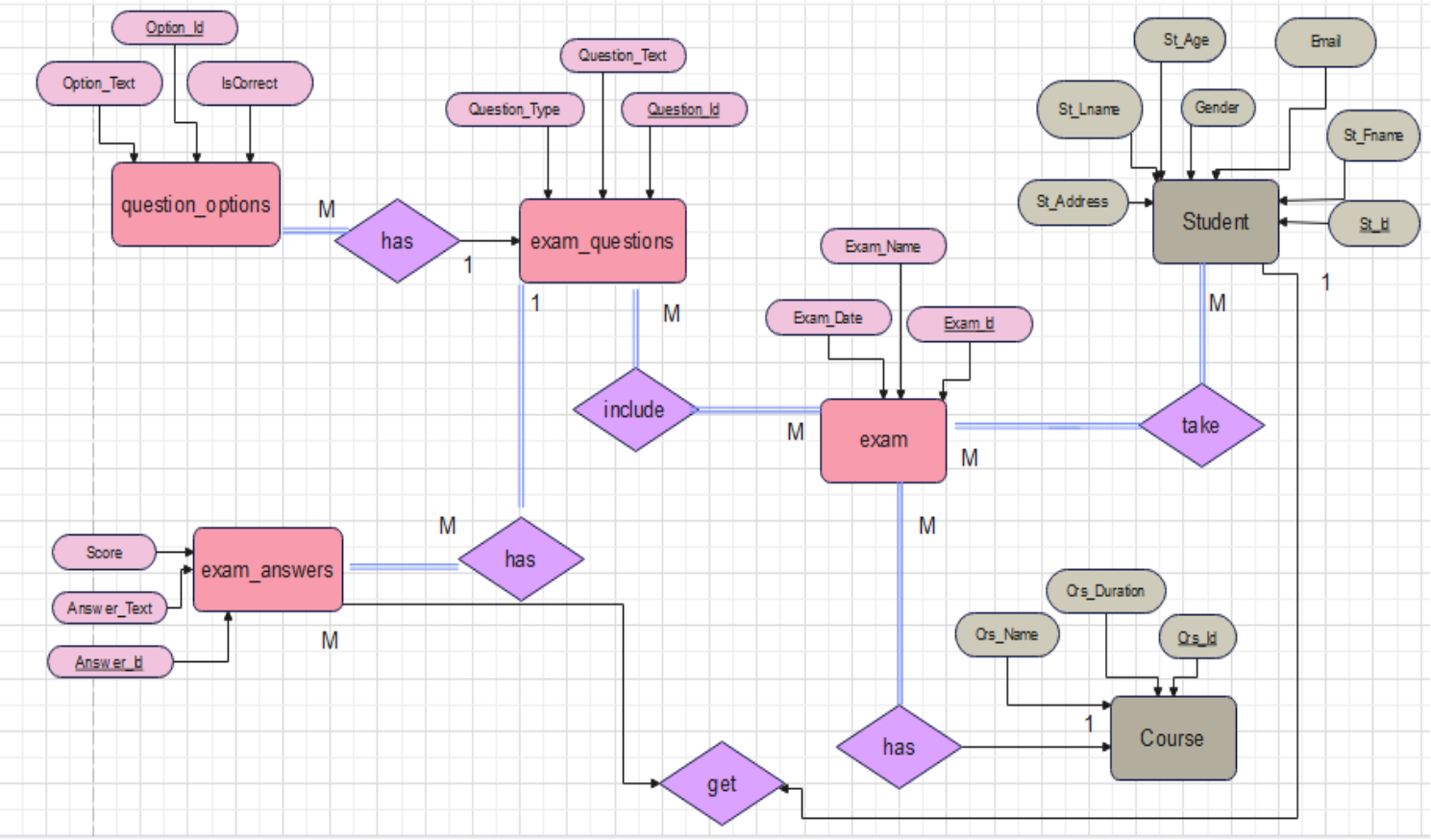
Represents data about exam_answers, like **answer_id, answer_text, score.**

ERD_diagram: -

This diagram explains the relationships, the cardinality ratio and participations between entities.


St_Id
St_Age
Gender
Email
St_Lname
St_Fname
St_Address
Student
HAS
skills
Skill_Id
Skill_Name
M
1
supervise
enroll
Department
Dept_Id
Dept_Location
Dept_Name
Study
grade
Crs_Id
Crs_Name
Crs_Duration
Course
Work
teach
Instructor
has
Top_Id
Top_Name
Topic
Ins_Id
Ins_Name
Salary


Gender
St_Addre ss
Email
stud_id
st_age
lname
fname
student
project_id
start_date
Project_name
End_date
project
work_on
Cert_id
Cert_Name
Issuer
Issue_date
certifications
get
do
hiring_date
contract_len
hire
freelance _id
freelance_ name
freelancing
location
company_id
company_name
company
position_id
position_name
salary
positions
training_id
End_date
Start_date
training_name
training
has
M
1
Activate Windo

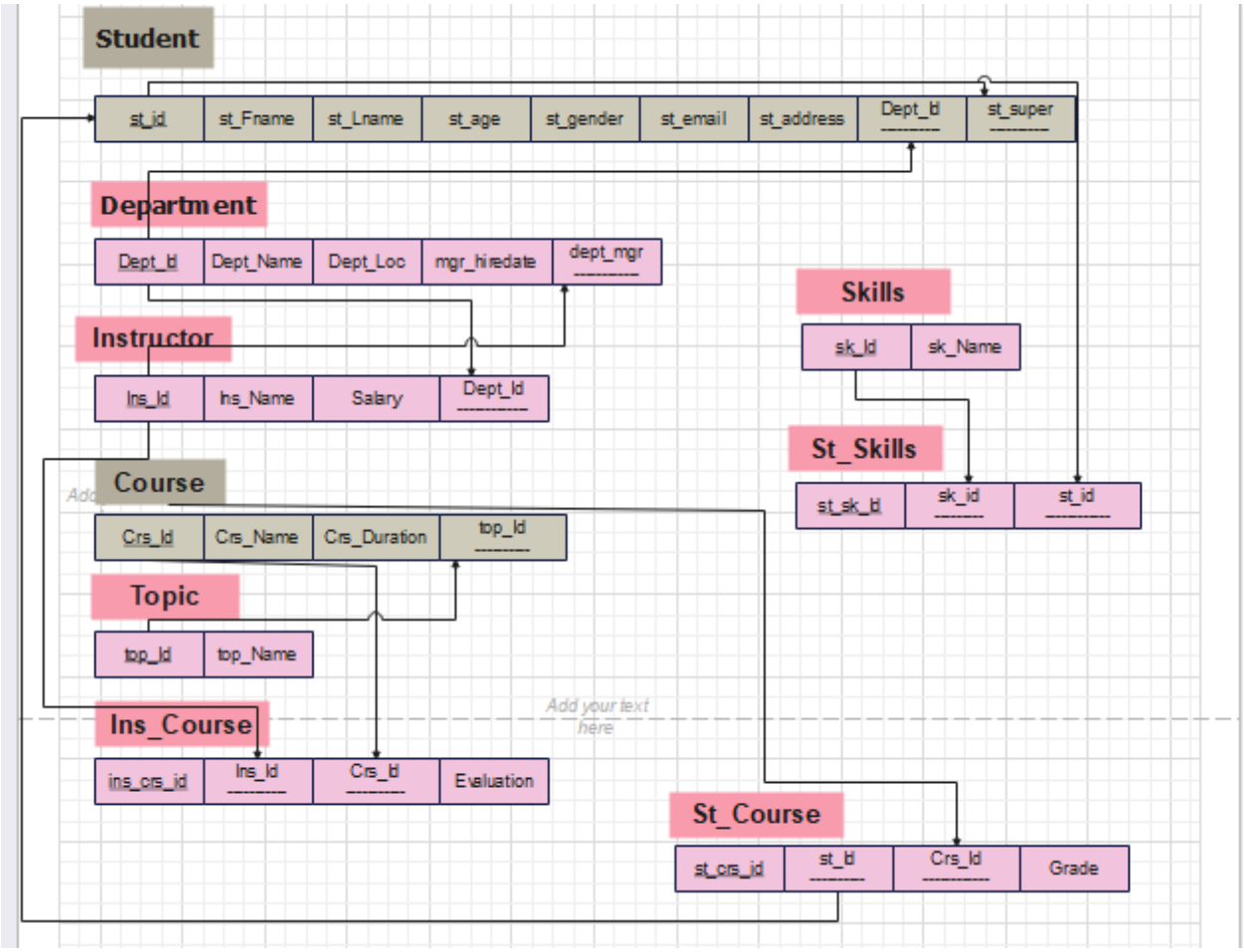


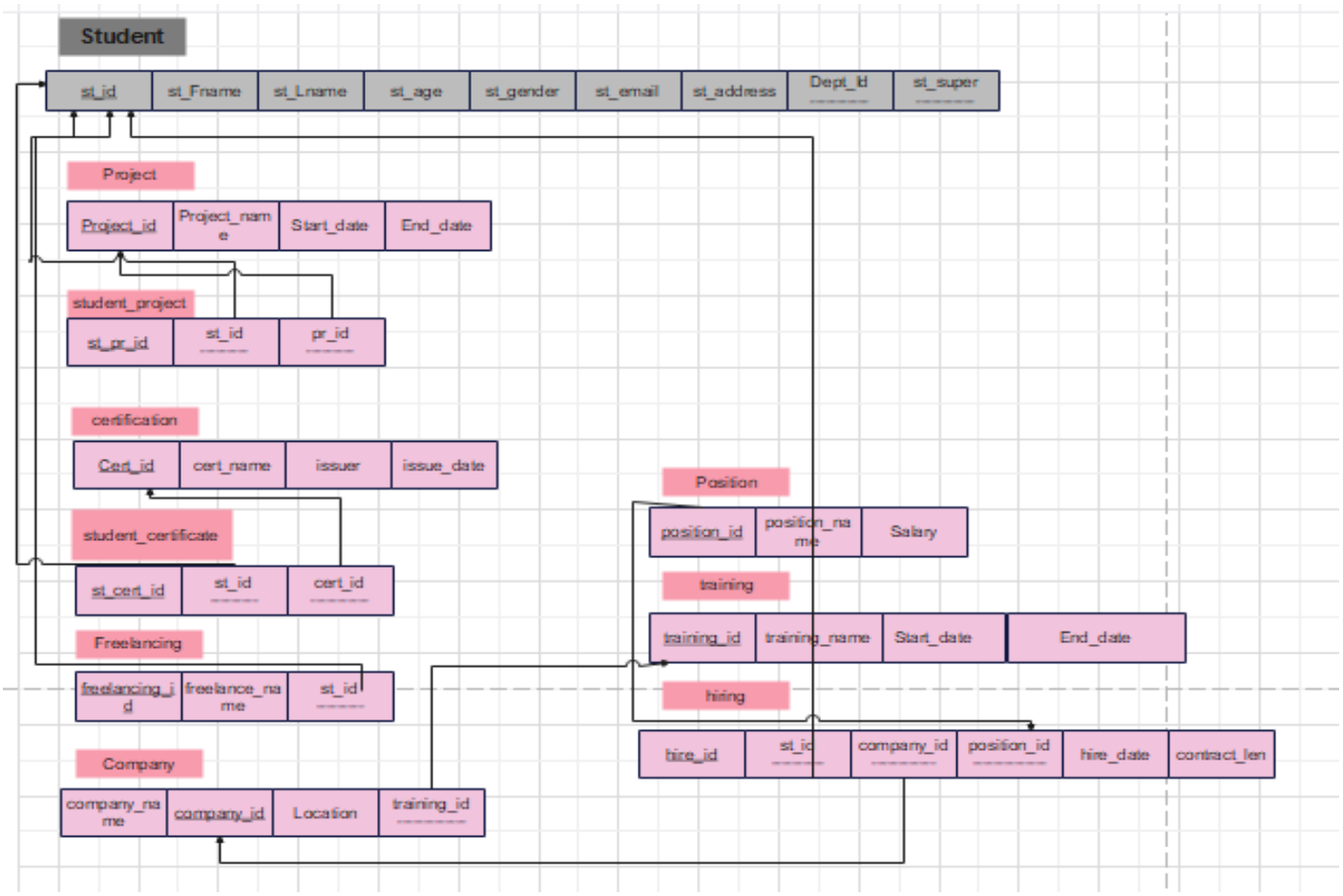
1- Each student must enroll in one department and each department may have many students.
2- Each instructor must work in many departments and each department may have many instructors.
3- Each student must study many courses and each course must being studied by many students with grade.
4- Each instructor must teach many courses and each course must being taught with many instructors.
5- Each course must have 1 topic but each topic must have many courses.
6- Each student must be under supervised by 1 supervisor but supervisor may supervise many students.
7- Each student must have many skills and each skill must be in many students.
8- Each student works on many projects and each project must have many students.
9- Each student may get many certificates and each certificate may be owned with many students.
10- Each student may do many freelancing jobs and each freelancing job may be done by many students.

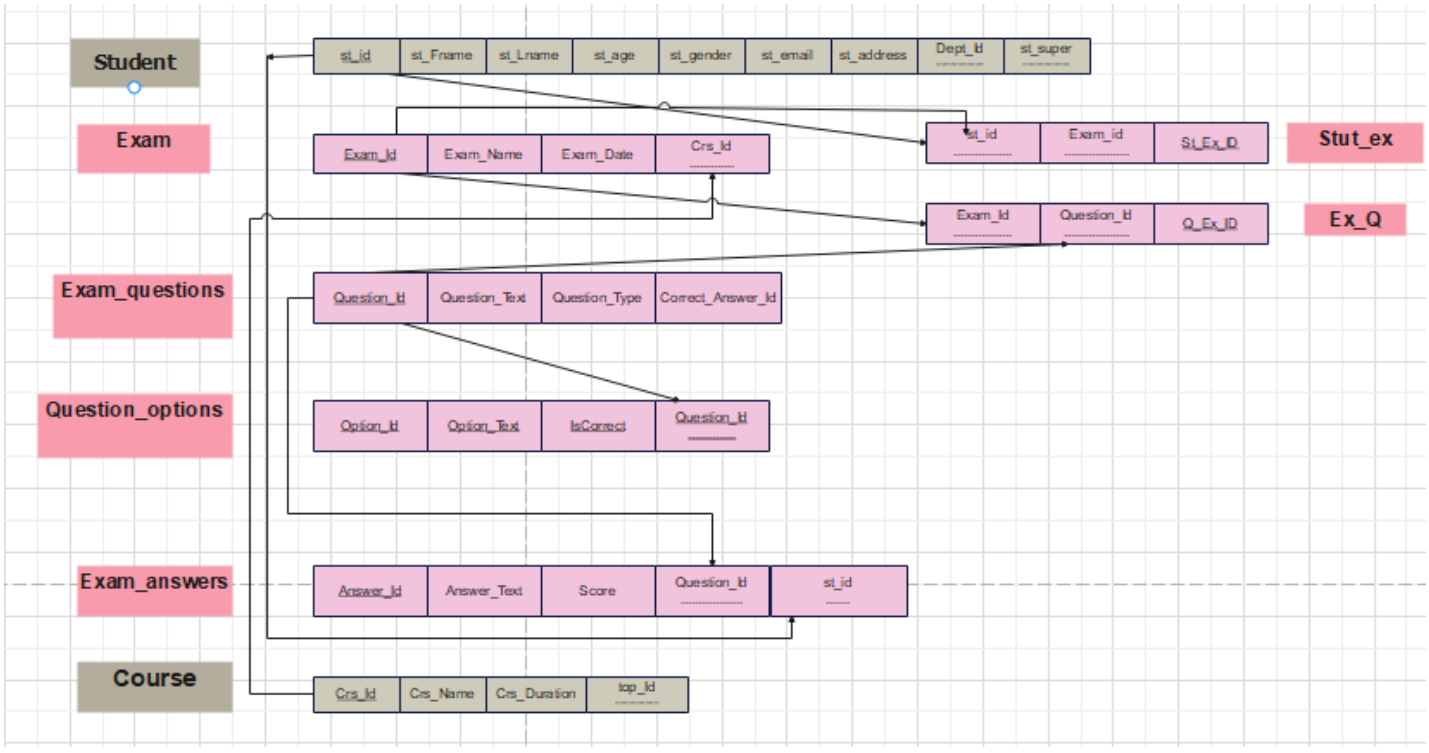
11- Each company hiring with many positions with hiring date and contract length and many students may apply.
12- Each company may have 1 training program for every position and each training program must be in many companies.
13- Each student must take many exams and each exam must be taken with many students.
14- Each exam must be for 1 course and each course may have many exams.
15- Each exam must have many questions and each question must be in many exams.
16- Each exam may have many options and each option must be in 1 question.
17- Each question must have many answers and each answer must have 1 question.
18- Each student may have many answers but each answer to 1 student.

Mapping: -

convert the conceptual design to logical design, and represent the relationships.




Student
st_id
st_Fname
st_Lname
st_age
st_gender
st_email
st_address
Dept_Id
st_super
Project
Project_id
Project_name
Start_date
End_date
student_project
st_pr_id
st_id
pr_id
certification
Cert_id
cert_name
issuer
issue_date
student_certificate
st_cert_id
st_id
cert_id
Freelancing
freelancing_id
freelance_name
st_id
Company
company_name
company_id
Location
training_id
Position
position_id
position_name
Salary
training
training_id
training_name
Start_date
End_date
hiring
hire_id
st_id
company_id
position_id
hire_date
contract_len


Student
st_id
st_Fname
st_Lname
st_age
st_gender
st_email
st_address
Dept_Id
st_super
Exam
Exam_Id
Exam_Name
Exam_Date
Crs_Id
st_id
Exam_id
St_Ex_ID
Stut_ex
Exam_Id
Question_Id
Q_Ex_ID
Ex_Q
Exam_questions
Question_Id
Question_Text
Question_Type
Correct_Answer_Id
Question_options
Option_Id
Option_Text
IsCorrect
Question_Id
Exam_answers
Answer_Id
Answer_Text
Score
Question_Id
st_id
Course
Crs_Id
Crs_Name
Crs_Duration
top_Id

Physical Model:

The creation of the tables using SQL queries.

1- Student:

```
CREATE TABLE [dbo].[Student](
    [St_Id] int IDENTITY(1,1) NOT NULL,
    [St_Fname] varchar(100),
    [St_Lname] varchar(100),
    [St_Address] varchar(100),
    [St_Age] int,
    [Email] varchar(100),
    [Dept_Id] int,
    [St_super] int,
    [Gender] varchar(50),
    CONSTRAINT [FK_Student_Department] FOREIGN KEY([Dept_Id]) REFERENCES
[dbo].[Department] ([Dept_Id]),
    CONSTRAINT [FK_Student_Student] FOREIGN KEY([St_super]) REFERENCES [dbo].[Student]
([St_Id]))
```

2- Department:

```
CREATE TABLE [dbo].[Department](
    [Dept_Id] int IDENTITY(1,1) NOT NULL,
    [Dept_Name] varchar(100),
    [Dept_Location] varchar(100),
    [Dept_Manager] int,
    [Manager_hiredate] date,
    CONSTRAINT [FK_Department_Instructor] FOREIGN KEY([Dept_Manager]) REFERENCES
[dbo].[Instructor] ([Ins_Id]))
```

3- Instructor:

```
CREATE TABLE [dbo].[Instructor](
    [Ins_Id] int IDENTITY(1,1) NOT NULL,
    [Ins_Name] varchar(100),
    [Salary] int,
    [Dept_Id] int,
    CONSTRAINT [FK_Instructor_Department] FOREIGN KEY([Dept_Id]) REFERENCES
[dbo].[Department] ([Dept_Id]))
```

4- Course:

```
CREATE TABLE [dbo].[Course](
    [Crs_Id] int IDENTITY(1,1) NOT NULL,
    [Crs_Name] varchar(100),
    [Crs_Duration] int,
    [Top_Id] int,
    CONSTRAINT [FK_Course_Topic] FOREIGN KEY([Top_Id]) REFERENCES [dbo].[Topic]
([Top_Id]))
```

5- Topic:

```
CREATE TABLE [dbo].[Topic](
    [Top_Id] int IDENTITY(1,1) NOT NULL,
    [Top_Name] varchar(100))
```

6- Skills:

```
CREATE TABLE [dbo].[Skills](
    [Skill_Id] int IDENTITY(1,1) NOT NULL,
    [Skill_Name] varchar(100))
```

7- Student_skills:

```
CREATE TABLE [dbo].[Student_skills](
    [Sk_St_id] int IDENTITY(1,1) NOT NULL,
    [Sk_Id] int,
    [St_Id] int,
    CONSTRAINT [FK_Student_skills_Skills] FOREIGN KEY([Sk_Id]) REFERENCES [dbo].[Skills]
([Skill_Id]),
    CONSTRAINT [FK_Student_skills_Student] FOREIGN KEY([St_Id]) REFERENCES
[dbo].[Student] ([St_Id]))
```

8- Student_course:

```
CREATE TABLE [dbo].[Student_course](
    [Crs_Id] int,
    [St_Id] int,
    [Grade] int,
    [crs_st_id] int IDENTITY(1,1) NOT NULL,
    CONSTRAINT [FK_Student_course_Course] FOREIGN KEY([Crs_Id]) REFERENCES
[dbo].[Course] ([Crs_Id]),
    CONSTRAINT [FK_Student_course_Student] FOREIGN KEY([St_Id]) REFERENCES
[dbo].[Student] ([St_Id]))
```

9- Instructor_course:

```
CREATE TABLE [dbo].[Instructor_course](
    [Ins_Crs_ID] int IDENTITY(1,1) NOT NULL,
    [Ins_Id] int,
    [Crs_Id] int,
    [Evaluation] varchar(100) NULL,
    CONSTRAINT [FK_Instructor_course_Instructor] FOREIGN KEY([Ins_Id]) REFERENCES
[dbo].[Instructor] ([Ins_Id]),
    CONSTRAINT [FK_Instructor_course_Course] FOREIGN KEY([Crs_Id]) REFERENCES
[dbo].[Course] ([Crs_Id]))
```

10- Certificates:

```
CREATE TABLE [dbo].[Certificates](
    [Cert_Id] int IDENTITY(1,1) NOT NULL,
    [Cert_Name] varchar(100),
    [Issuer] varchar(100),
    [Issue_Date] date)
```

11- Student_Certificates:

```
CREATE TABLE [dbo].[Student_Certificates](
    [St_Cer_ID] int IDENTITY(1,1) NOT NULL,
    [St_Id] int,
    [Cert_Id] int,
    CONSTRAINT [FK_Student_Certificates_Student] FOREIGN KEY([St_Id]) REFERENCES
[dbo].[Student] ([St_Id]),
    CONSTRAINT [FK_Student_Certificates_Certificates] FOREIGN KEY([Cert_Id]) REFERENCES
[dbo].[Certificates] ([Cert_Id]))
```

12- Freelancing:

```
CREATE TABLE [dbo].[Freelancing](
    [Freelance_Id] int IDENTITY(1,1) NOT NULL,
    [Freelance_Name] varchar(100) NULL,
    [St_Id] int NULL,
    CONSTRAINT [FK_Freelancing_Student] FOREIGN KEY([St_Id]) REFERENCES [dbo].[Student]
([St_Id]))
```

13- Companies:

```
CREATE TABLE [dbo].[Companies](
    [Company_Id] int IDENTITY(1,1) NOT NULL,
    [Company_Name] varchar(100),
    [Location] varchar(100),
    [Training_ID] int,
    CONSTRAINT [FK_Companies_Training] FOREIGN KEY([Training_ID]) REFERENCES
[dbo].[Training] ([Train_ID]))
```

14- Positions:

```
CREATE TABLE [dbo].[Positions](
    [Position_Id] int IDENTITY(1,1) NOT NULL,
    [Position_Name] varchar(100),
    [Salary] int)
```

15- Hiring:

```
CREATE TABLE [dbo].[Hiring](
    [Hiring_Id] int IDENTITY(1,1) NOT NULL,
    [Company_Id] int,
    [Position_Id] int,
    [St_Id] int,
    [Hire_Date] date,
    [ContractLength] int,
    CONSTRAINT [FK_Hiring_Companies] FOREIGN KEY([Company_Id]) REFERENCES
[dbo].[Companies] ([Company_Id]),
    CONSTRAINT [FK_Hiring_Positions] FOREIGN KEY([Position_Id]) REFERENCES
[dbo].[Positions] ([Position_Id]),
    CONSTRAINT [FK_Hiring_Student] FOREIGN KEY([St_Id]) REFERENCES [dbo].[Student]
([St_Id]))
```

16- Training:

```
CREATE TABLE [dbo].[Training](
    [Train_ID] int IDENTITY(1,1) NOT NULL,
    [Train_Name] varchar(100),
    [Start_Date] date,
    [End_Date] date)
```

17- Projects:

```
CREATE TABLE [dbo].[Projects](
    [Project_Id] int IDENTITY(1,1) NOT NULL,
    [Project_Name] varchar(100),
    [Start_Date] date,
    [End_Date] date)
```

18- Student_Project:

```
CREATE TABLE [dbo].[Student_Project](
    [St_Pro_Id] int IDENTITY(1,1) NOT NULL,
    [St_Id] int,
    [Pro_Id] int,
    CONSTRAINT [FK_Student_Project_Project] FOREIGN KEY([Pro_Id]) REFERENCES
[dbo].[Projects] ([Project_Id]),
    CONSTRAINT [FK_Student_Project_Student] FOREIGN KEY([St_Id]) REFERENCES
[dbo].[Student] ([St_Id]))
```

19- Exam:

```
CREATE TABLE [dbo].[Exam](
    [Exam_Id] int IDENTITY(1,1) NOT NULL,
    [Exam_Name] varchar](100),
    [Exam_Date] date,
    [Crs_Id] int,
       CONSTRAINT [FK_Exam_Course] FOREIGN KEY(Crs_Id]) REFERENCES [dbo].[Course]
([Crs_Id]))
```

20- St_Exam:

```
CREATE TABLE [dbo].[St_Exam](
    [St_Ex_Id] int IDENTITY(1,1) NOT NULL,
    [St_Id] int,
    [Ex_Id] int,
    CONSTRAINT [FK_Student_Exam_Student] FOREIGN KEY([St_Id]) REFERENCES [dbo].[Student]
([St_Id]),
    CONSTRAINT [FK_Student_Exam_Exam] FOREIGN KEY([Ex_Id]) REFERENCES [dbo].[Exam]
([Exam_Id]))
```

21- Exam_answers:

```
CREATE TABLE [dbo].[Exam_answers](
    [Answer_Id] int IDENTITY(1,1) NOT NULL,
    [Answer_Text] varchar(300),
    [Question_Id] int,
    [Score] int,
    [St_Id] int,
    CONSTRAINT [FK_Exam_answers_Exam_questions] FOREIGN KEY([Question_Id]) REFERENCES
[dbo].[Exam_questions] ([Question_Id]),
    CONSTRAINT [FK_Exam_answers_Student] FOREIGN KEY([St_Id]) REFERENCES [dbo].[Student]
([St_Id]))
```

22- Exam_questions:

```
CREATE TABLE [dbo].[Exam_questions](
    [Question_Id] int IDENTITY(1,1) NOT NULL,
    [Question_Text] varchar(200),
    [Question_Type] varchar(50),
    [Correct_Answer_Id] int,
    [Category_Name] varchar(100),
    CONSTRAINT [FK_Exam_questions_Exam_answers] FOREIGN KEY([Correct_Answer_Id])
REFERENCES [dbo].[Exam_answers] ([Answer_Id]))
```

23- Ques_Exam:

```
CREATE TABLE [dbo].[Ques_Exam](
    [Q_Ex_Id] int IDENTITY(1,1) NOT NULL,
    [Ex_Id] int,
    [Q_Id] int,
    CONSTRAINT [FK_Ques_Exam_Exam] FOREIGN KEY([Ex_Id]) REFERENCES [dbo].[Exam]
([Exam_Id]),
    CONSTRAINT [FK_Ques_Exam_Exam_questions] FOREIGN KEY([Q_Id]) REFERENCES
[dbo].[Exam_questions] ([Question_Id]))
```

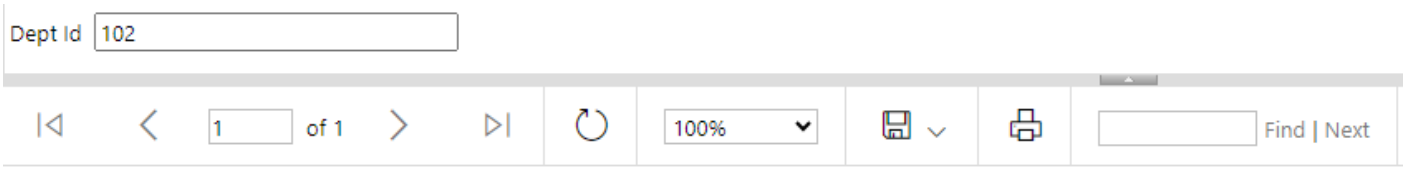
24- Ques_Exam:

```
CREATE TABLE [dbo].[Question_options](
    [Option_Id] int IDENTITY(1,1) NOT NULL,
    [Option_Text] varchar(max),
    [IsCorrect] int,
    [Question_Id] int,
    [Category_Name] varchar(100),
    CONSTRAINT [FK_Question_options_Exam_questions] FOREIGN KEY([Question_Id])
REFERENCES [dbo].[Exam_questions] ([Question_Id]))
```

Reports using SSRS:

1- Report that returns the students information according to Department No parameter.

```
    CREATE PROCEDURE StudentInfo
        @DeptId INT
    AS
    BEGIN
        SET NOCOUNT ON;
        SELECT S.St_Id, S.St_Fname, S.St_Lname, S.St_Address, S.St_Age, S.Email, S.Gender
        FROM Student S
        INNER JOIN Department D ON S.Dept_Id = D.Dept_Id
        WHERE D.Dept_Id = @DeptId;
    END;
StudentInfo   @DeptId = 102
```

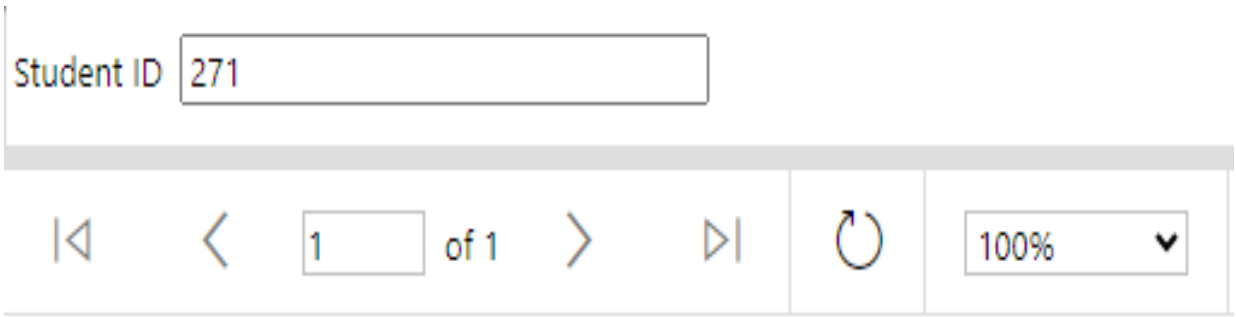


Student_Info

St Id	St Fname	St Lname	St Address	St Age	Email	Gender
125	Timothy	Lopez	80336 Miller Knoll Apt. 695 Turnerland, WY 88056	25	jeffery11@example.com	Female
205	Nicholas	Zavala	449 Michael Fork Apt. 308 Traceyburgh, IA 29139	24	mark53@example.net	Male
211	Melissa	Dixon	75950 Gonzales Via Suite 513 Amberview, FL 44952	23	amandagarcia@example.com	Male
212	Michael	Ortiz	71669 Faith Glen Lake Ericberg, AZ 70506	23	michaelpage@example.net	Male
260	Ashley	Lawson	56765 Myers Forest Apt. 389 Monicahaven, KS 71406	23	charleshernandez@example.net	Female
262	Michael	Haynes	292 Samantha Rapids Suite 456 Lake Ricardo, SD 88951	24	bdunn@example.net	Male
303	Nicholas	Herrera	99763 Samantha Gateway Apt. 561 Diazbury, GU 75643	25	obrienevan@example.net	Female
306	Billy	Garcia	1107 Smith Squares Lake Nicholas, PR 24871	24	mmiller@example.com	Female
334	Virginia	Mills	4925 Howard Extensions Jamesmouth, WI 42562	24	hrobles@example.com	Female
343	William	Shannon	1703 Williams Light Suite 192 Jamesport, MP 21287	24	jasonmartinez@example.com	Female

2- Report that takes the student ID and returns the grades of the student in all courses.

```sql
CREATE PROCEDURE GetStudentGrades
    @StudentID INT
AS
BEGIN
    SET NOCOUNT ON;
    SELECT
        C.Crs_Name AS CourseName,
        SC.Grade
    FROM
        Student_course SC
    INNER JOIN
        Course C ON SC.Crs_Id = C.Crs_Id
    WHERE
        SC.St_Id = @StudentID;
END;

EXEC GetStudentGrades @StudentID = 271;
```

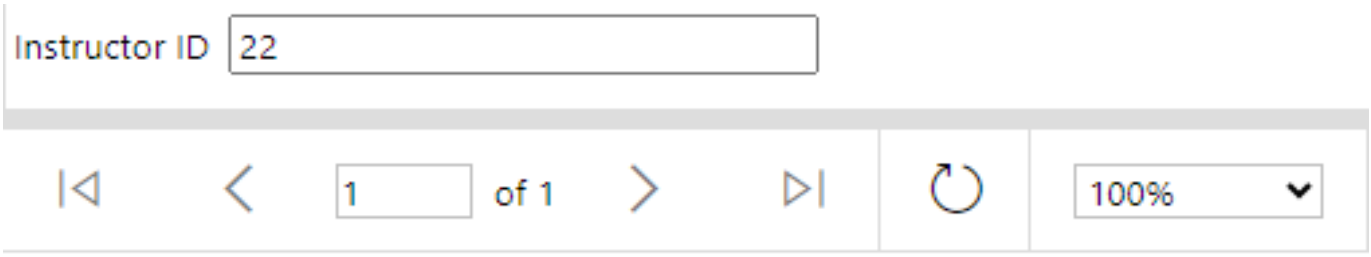


Student_Grades

Course Name	Grade
Data Structures and Algorithms	80
Web Development	28
Software Testing	59

3- Report that takes the instructor ID and returns the name of the courses that he teaches and the number of students per course.

```
CREATE PROCEDURE GetrCoursesAndStudentsNum
    @InstructorID INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        C.Crs_Name AS CourseName,
        COUNT(SC.St_Id) AS NumberOfStudents
    FROM
        Instructor_course IC
    INNER JOIN
        Course C ON IC.Crs_Id = C.Crs_Id
    LEFT JOIN
        Student_course SC ON IC.Crs_Id = SC.Crs_Id
    WHERE
        IC.Ins_Id = @InstructorID
    GROUP BY
        C.Crs_Name;
END;

GetrCoursesAndStudentsNum  @InstructorID=22
```

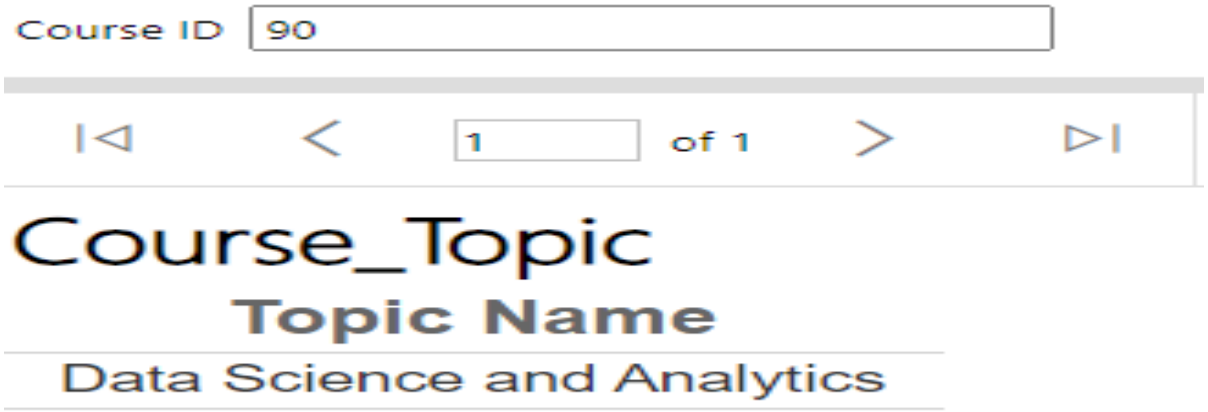


Courses & Students No.

Course Name	Number Of Students
Artificial Intelligence	22
Blockchain Technology	12
Cybersecurity	16
Data Structures and Algorithms	18
Internet of Things (IoT)	22
Introduction to Programming	30

4- Report that takes course ID and returns its topics

```
CREATE PROCEDURE GetCourseTopics
    @CourseID INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        T.Top_Name AS TopicName
    FROM
        Course C
    INNER JOIN
        Topic T ON C.Top_Id = T.Top_Id
    WHERE
        C.Crs_Id = @CourseID;
END;

EXEC GetCourseTopics @CourseID = 90;
```



5- Report that takes exam number and returns the Questions in it and chocies [freeform report]

```
create PROCEDURE GetExamQuestionsAndChoices
    @ExamNumber INT
AS
BEGIN
    SET NOCOUNT ON;

    WITH CTE AS (
        SELECT
            CASE
                WHEN ROW_NUMBER() OVER (PARTITION BY EQ.Question_Text ORDER BY (SELECT
NULL)) = 1 THEN EQ.Question_Text
                ELSE ''
            END AS Question_Text,
            QO.Option_Text,
            QO.IsCorrect
        FROM
            Exam E
        INNER JOIN
            Ques_Exam QE ON E.Exam_Id = QE.Ex_Id
        INNER JOIN
            Exam_questions EQ ON QE.Q_Id = EQ.Question_Id
        LEFT JOIN
            Question_options QO ON EQ.Question_Id = QO.Question_Id
        WHERE
            E.Exam_Id = @ExamNumber
    )
    SELECT
        Question_Text,
        Option_Text,
        IsCorrect
    FROM CTE;
END;
EXEC GetExamQuestionsAndChoices @ExamNumber =383;
```

Exam Number 383

Actuators are responsible for sensing the environment in robotics.

True

False

Artificial Intelligence is not a crucial component in robotics.

True

False

Biomimicry in robotics involves mimicking the behavior of living organisms.

True

False

6- Report that takes exam number and the student ID then returns the Questions in this exam with the student answers.

```
  CREATE PROCEDURE GetExamQuestionsAndAnswers
    @ExamId INT,
    @StudentId INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT EQ.Question_Text,
           EA.Answer_Text AS Student_Answer,
           EA.Score
    FROM Exam E
    INNER JOIN Ques_Exam QE ON E.Exam_Id = QE.Ex_Id
    INNER JOIN Exam_questions EQ ON QE.Q_Id = EQ.Question_Id
    LEFT JOIN Exam_answers EA ON EQ.Question_Id = EA.Question_Id
    LEFT JOIN Student S ON EA.St_Id = S.St_Id
    LEFT JOIN St_Exam SE ON E.Exam_Id = SE.Ex_Id AND S.St_Id = SE.St_Id
    WHERE E.Exam_Id = @ExamId
      AND S.St_Id = @StudentId;
END;

EXEC GetExamQuestionsAndAnswers @ExamId = 373, @StudentId = 106;
```

Exam Id 373 Student Id 106

1 of 1 100%

Exam Questions & Answers

Question Text	Student Answer	Score
Raster graphics are resolution-independent.	False	1
Vector graphics use mathematical equations to represent images.	False	0
Anti-aliasing is a technique used to reduce jagged edges in computer graphics.	True	1
Ray tracing is a rendering technique used for real-time graphics in video games.	False	1
What is the purpose of the Z-buffer in 3D computer graphics?	To store the depth information of each pixel	1
Which rendering technique is commonly used for creating realistic reflections and shadows?	Ray Tracing	0
What is the primary function of a GPU (Graphics Processing Unit) in computer graphics?	To store and display images on the screen	0
Which file format is commonly used for storing 3D models?	FBX	0
What is the difference between orthographic and perspective projections in computer graphics?	Orthographic projection is used for 3D models, while perspective projection is used for 2D drawings	0
		Total score = 4

Data warehousing:

Creation of dimension and fact tables.

1- Dim_Exam:

This table holds information about exams. It includes details such as exam ID (both surrogate and business key), exam name, date, and related question and answer details.

```
CREATE TABLE [dbo].[Dim_Exam](
    [exam_id_sk] int IDENTITY(1,1) NOT NULL,
    [exam_id_bk] int,
    [exam_name] varchar(100),
    [exam_date] date,
    [question_id_bk] int,
    [question_text] varchar(200),
    [question_type] varchar(50),
    [correct_answer_id] int,
    [answer_id_bk] int,
    [answer_text] varchar(300),
    [score] int,
    [option_id_bk] int,
    [option_text] varchar(300),
    [is_correct] int,
    [start_date] datetime,
    [end_date] datetime,
    [is_current] tinyint,
      CONSTRAINT [FK_Dim_Exam_Fact_table] FOREIGN KEY([exam_id_sk]) REFERENCES
  [dbo].[Fact_table] ([exam_id_fk]))
```

2- Dim_Hiring:

This table stores data related to hiring processes. It includes hiring ID (surrogate and business key), hire date, contract length, company details, training information, position details, salary, and temporal information.

```
CREATE TABLE [dbo].[Dim_Hiring](
    [hiring_id_sk] int IDENTITY(1,1) NOT NULL,
    [hiring_id_bk] int,
    [hire_date] date,
    [contract_length] int,
    [company_id_bk] int,
    [company_name] varchar(100),
    [location] varchar(100),
    [training_id_bk] int,
    [training_name] varchar(100),
    [training_startdate] date,
    [training_enddate] date,
    [position_id_bk] int,
    [position_name] varchar(100),
    [salary] int,
    [start_date] datetime,
    [end_date] datetime,
    [is_current] tinyint,
      CONSTRAINT [FK_Dim_Hiring_Fact_table] FOREIGN KEY ([hiring_id_sk]) REFERENCES
  [dbo].[Fact_table] ([hiring_id_fk]))
```

3- Dim_Student:

This table contains information about students. It includes student ID (surrogate and business key), name, address, age, email, department details, certification information, freelancing details, skill and project information, temporal data, etc.

```
CREATE TABLE [dbo].[Dim_Student](
    [st_id_sk] int IDENTITY(1,1) NOT NULL,
    [st_id_bk] int,
    [st_fname] varchar(100),
    [st_lname] varchar(100),
    [st_address] varchar(100),
    [st_age] int,
    [email] varchar(100),
    [st_super_bk] int,
    [gender] varchar(50),
    [dept_id_bk] int,
    [dept_name] varchar(100),
    [dept_location] varchar(100),
    [dept_mgr] int,
    [mgr_hiredate] date,
    [cert_id_bk] int,
    [cert_name] varchar(100),
    [issuer] varchar(100),
    [issue_date] date,
    [freelancing_id_bk] int,
    [freelancing_name] varchar(100),
    [skill_id] int,
    [skill_name] varchar(100),
    [project_id] int,
    [project_name] varchar(100),
    [pro_startdate] date,
    [pro_enddate] date,
    [start_date] datetime,
    [end_date] datetime,
    [is_current] tinyint,
     CONSTRAINT [FK_Dim_Student_Fact_table] FOREIGN KEY ([st_id_sk]) REFERENCES
 [dbo].[Fact_table] ([student_id_fk]))
```

4- DimDate:

This table is a time dimension table. It includes various attributes related to dates such as day, month, quarter, year, and some additional attributes like holiday text.

```
CREATE TABLE [dbo].[DimDate](
    [DateSK] int NOT NULL,
    [Date] date NOT NULL,
    [Day] char(2) NOT NULL,
    [DaySuffix] varchar(4) NOT NULL,
    [DayOfWeek] varchar(9) NOT NULL,
    [DOWInMonth] tinyint NOT NULL,
    [DayOfYear] int NOT NULL,
    [WeekOfYear] tinyint NOT NULL,
    [WeekOfMonth] tinyint NOT NULL,
    [Month] char(2) NOT NULL,
    [MonthName] varchar(9) NOT NULL,
```

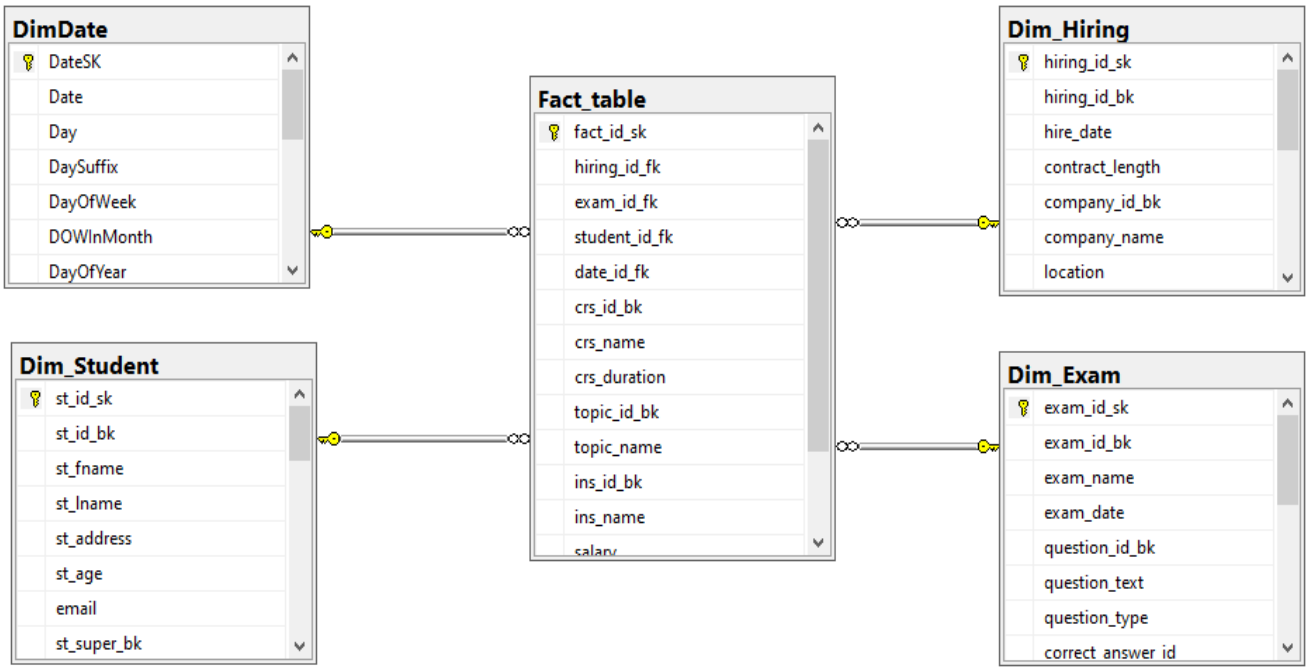
```
    [Quarter] tinyint NOT NULL,
    [QuarterName] varchar(6) NOT NULL,
    [Year] char(4) NOT NULL,
    [StandardDate] varchar(10) NULL,
    [HolidayText] varchar(50) NULL,
     CONSTRAINT [FK_DimDate_Fact_table] FOREIGN KEY ([DateSK]) REFERENCES
[dbo].[Fact_table] ([date_id_fk]))
```

5- Fact_table:

This table represents a fact table that likely holds the central metrics or measurements of interest. It includes foreign keys to the dimension tables (Dim_Exam, Dim_Hiring, Dim_Student, DimDate) and additional attributes like course details, instructor details, salary, grade, and creation timestamp.

```
CREATE TABLE [dbo].[Fact_table](
    [fact_id_sk] int IDENTITY(1,1) NOT NULL,
    [hiring_id_fk] int,
    [exam_id_fk] int,
    [student_id_fk] int,
    [date_id_fk] int,
    [crs_id_bk] int,
    [crs_name] varchar(100),
    [crs_duration] int,
    [topic_id_bk] int,
    [topic_name] varchar(100),
    [ins_id_bk] int,
    [ins_name] varchar(100),
    [salary] int,
    [grade] int,
    [created_at] datetime,
    CONSTRAINT [FK_Fact_table_Dim_Exam] FOREIGN KEY ([exam_id_fk]) REFERENCES
[dbo].[Dim_Exam] ([exam_id_sk]),
    CONSTRAINT [FK_Fact_table_Dim_Hiring] FOREIGN KEY ([hiring_id_fk]) REFERENCES
[dbo].[Dim_Hiring] ([hiring_id_sk]),
    CONSTRAINT [FK_Fact_table_Dim_Student] FOREIGN KEY ([student_id_fk]) REFERENCES
[dbo].[Dim_Student] ([st_id_sk]),
      CONSTRAINT [FK_Fact_table_DimDate] FOREIGN KEY ([date_id_fk]) REFERENCES
  [dbo].[DimDate] ([DateSK]))
```

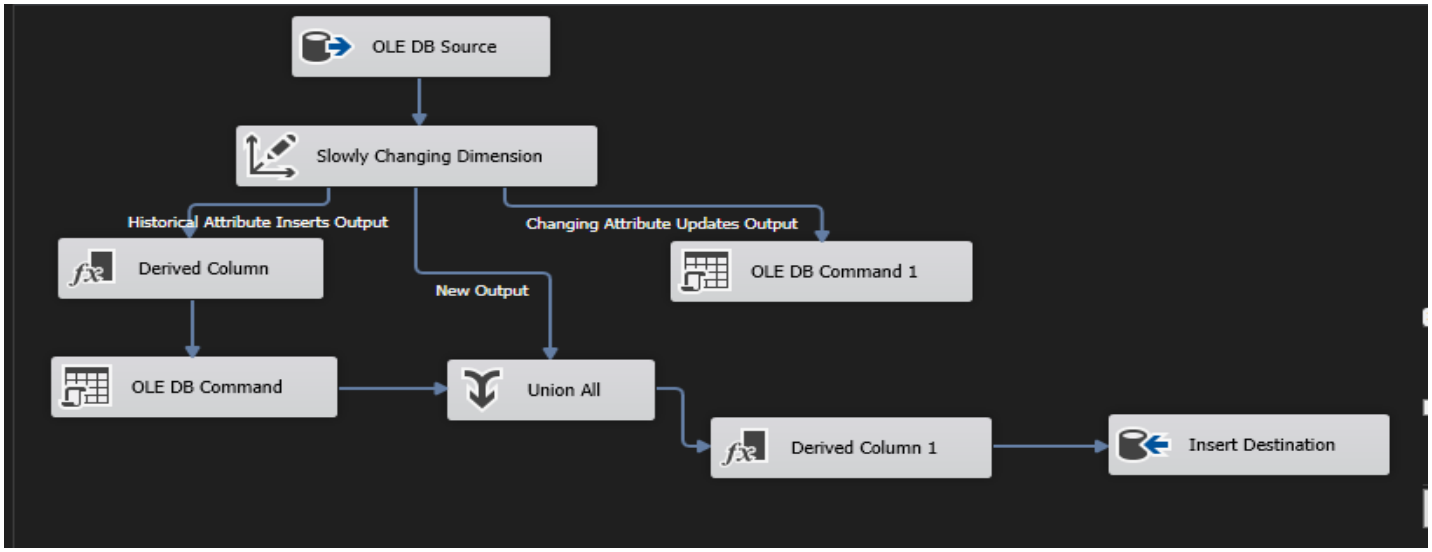
DWH_Diagram:



ETL using SSIS:

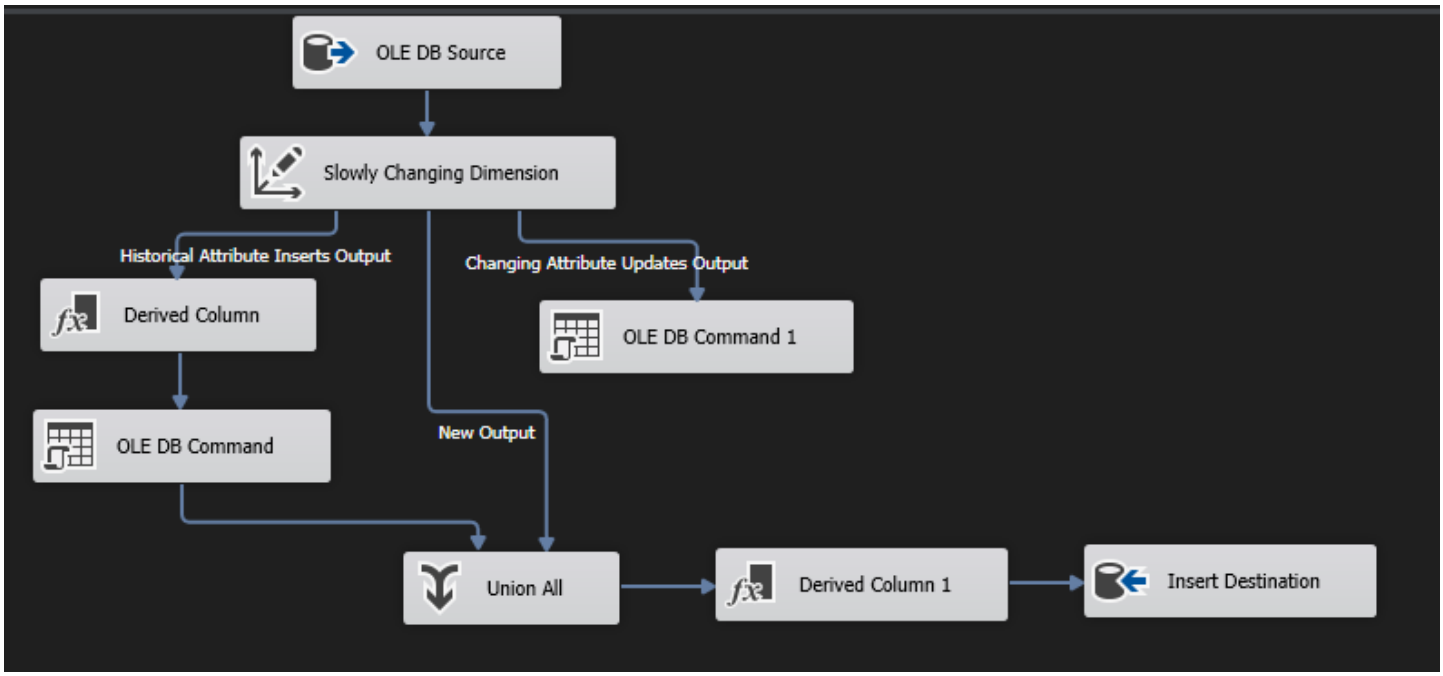
1- Dim_Exam:

```
select e.Exam_Id, e.Exam_Name, e.Exam_Date, eq.Question_Id, eq.Question_Text,
eq.Question_Type, eq.Correct_Answer_Id,
qo.Option_Id, qo.Option_Text, qo.IsCorrect, ea.Answer_Id, ea.Answer_Text, ea.Score
from exam e left join Ques_Exam qe
on e.Exam_Id = qe.Ex_Id
left join Exam_questions eq
on qe.Q_Id = eq.Question_Id
left join Question_options qo
on qo.Question_Id = eq.Question_Id
left join Exam_answers ea
on ea.Question_Id = eq.Question_Id
```



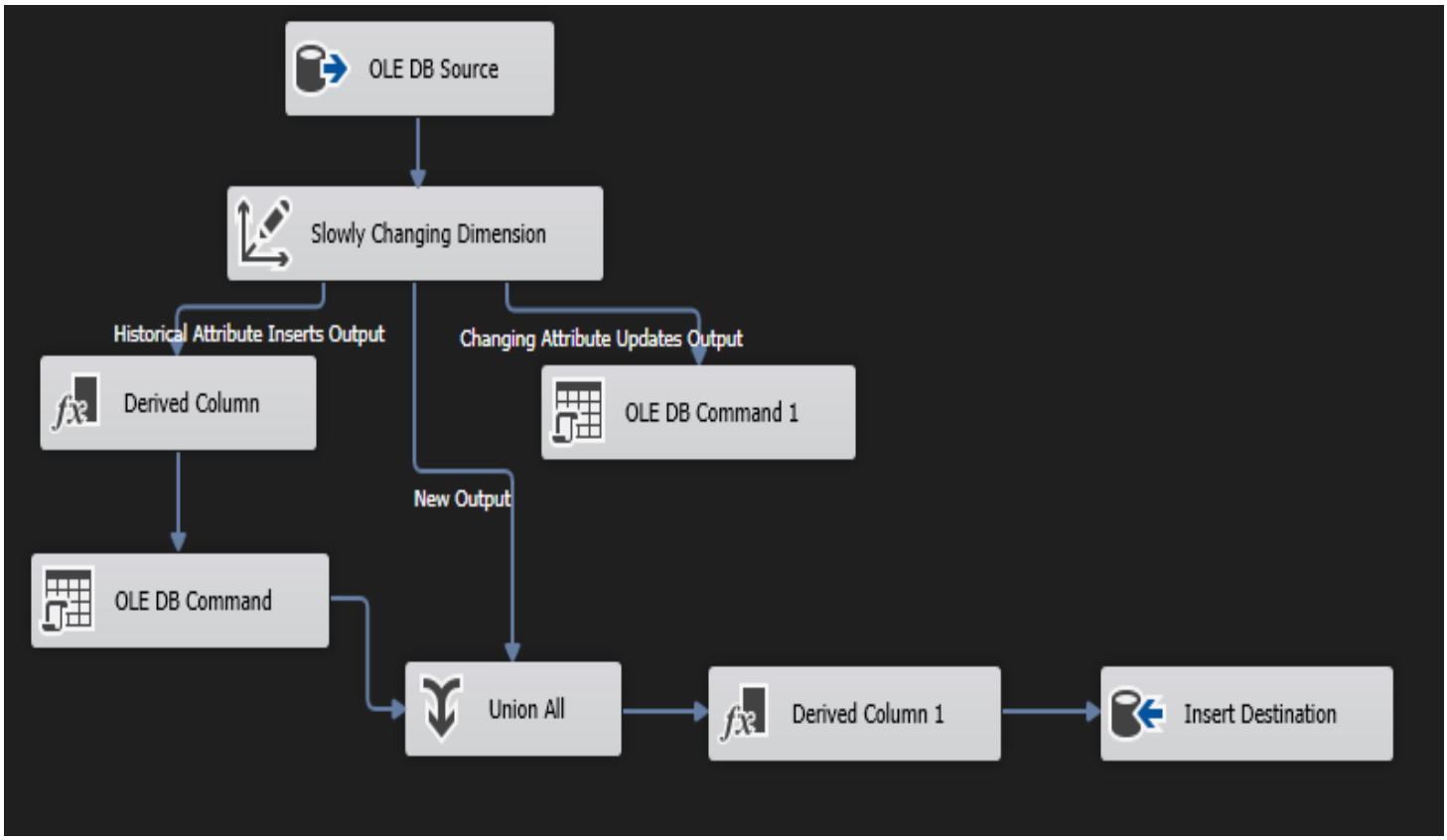
2- Dim_Hiring:

```
select c.Company_Id, c.Company_Name, c.Location, t.Train_ID, t.Train_Name, t.Start_Date,
t.End_Date, h.Hiring_Id, h.Hire_Date, h.ContractLength, p.Position_Id, p.Position_Name,
p.Salary
from Companies c left join Training t
on t.Train_ID = c.Training_ID
left join Hiring h
on h.Company_Id = c.Company_Id
left join Positions p
on p.Position_Id = h.Position_Id
```



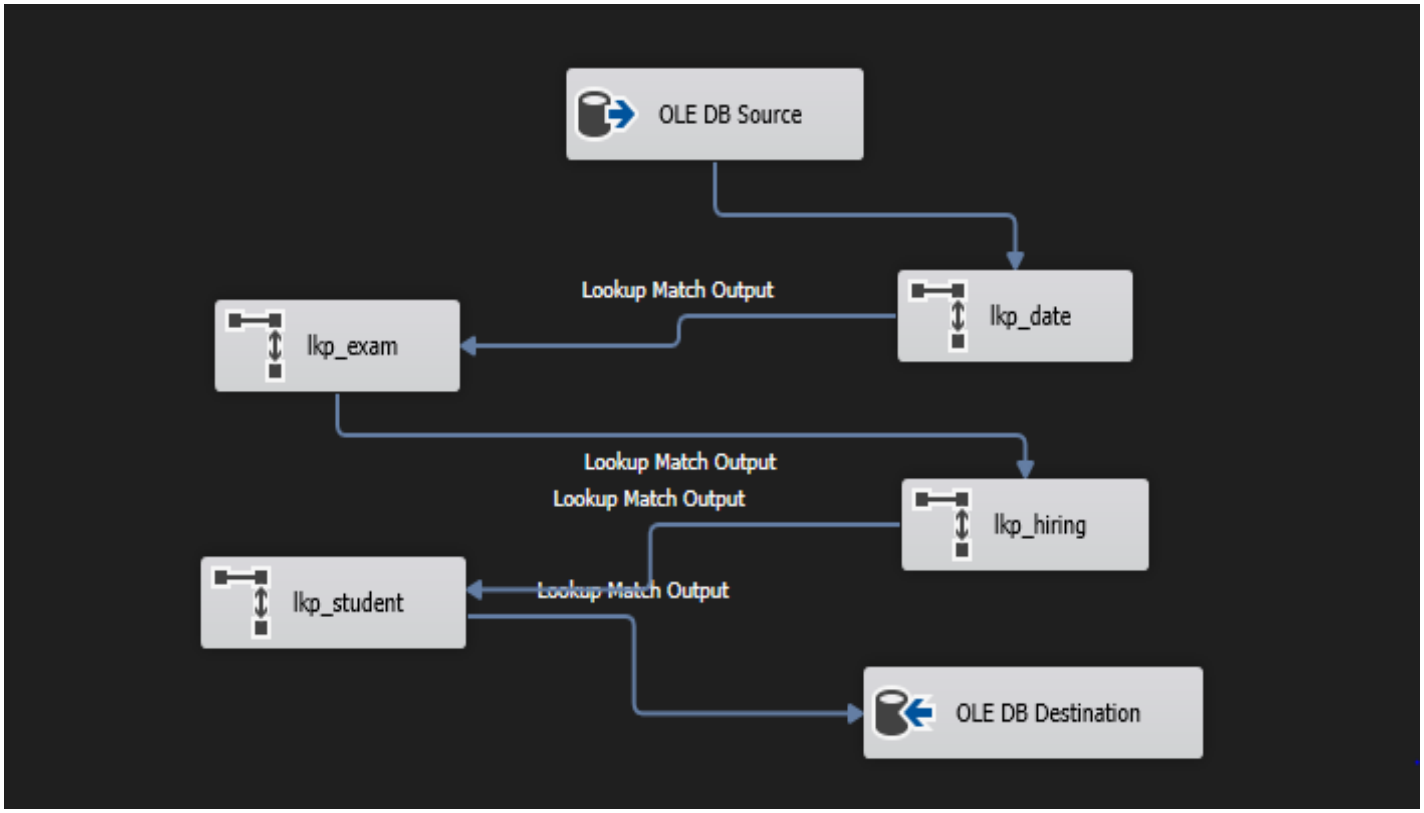
3- Dim_Student:

```
select s.St_Id, s.St_Fname, s.St_Lname, s.St_Address, s.Gender, s.St_Age,s.Email,
s.St_super, c.Cert_Id, c.Cert_Name, c.Issue_Date, c.Issuer,
d.Dept_Id, d.Dept_Name, d.Dept_Location, d.Dept_Manager, d.Manager_hiredate,
sk.Skill_Id, sk.Skill_Name, f.Freelance_Id, f.Freelance_Name,
p.Project_Id, p.Project_Name, p.Start_Date, p.End_Date
from student s left join Student_Certificates sc
on s.St_Id = sc.St_Id
left join Certificates c
on c.Cert_Id = sc.Cert_Id
left join Department d
on d.Dept_Id = s.Dept_Id
left join Student_skills ss
on ss.St_Id = s.St_Id
left join Skills sk
on sk.Skill_Id = ss.Sk_Id
left join Freelancing f
on f.St_Id = s.St_Id
left join Student_Project sp
on sp.St_Id = s.St_Id
left join Projects p
on sp.Pro_Id = p.Project_Id
```



4- Fact_table:

```
select s.St_Id, h.Hiring_Id, e.Exam_Id, c.Crs_Id, c.Crs_Name, c.Crs_Duration, c.Top_Id,
t.Top_Name, i.Ins_Id,
i.Ins_Name, i.Salary, sc.Grade, d.Manager_hiredate, ce.Issue_Date, e.Exam_Date,
h.Hire_Date
from Student s inner join Student_course sc
on s.St_Id = sc.St_Id
inner join Course c
on c.Crs_Id = sc.Crs_Id
inner join Instructor_course ic
on ic.Crs_Id = c.Crs_Id
inner join Instructor i
on i.Ins_Id = ic.Ins_Id
inner join Topic t
on t.Top_Id = c.Top_Id
inner join Department d
on s.Dept_Id = d.Dept_Id
inner join Hiring h
on h.St_Id = s.St_Id
inner join Exam e
on e.Crs_Id = c.Crs_Id
inner join Student_Certificates ss
on ss.St_Id = s.St_Id
inner join Certificates ce
on ce.Cert_Id = ss.Cert_Id
```



Select stored procedures:

1- Certificates_table:

```
create proc GetCertificateData as
select * from Certificates
```

2- Companies_table:

```
create proc GetCompaniesData as
select * from Companies
```

3- Course_table:

```
create proc GetcourseData as
select * from Course
```

4- Department_table:

```
create proc GetDepartmentData as
select * from Department
```

5- Exam_table:

```
create proc GetExamData as
select * from Exam
```

6- Exam_answers_table:

```
create proc GetExamAswersData as
select * from Exam_answers
```

7- Exam_questions_table:

```
create proc GetExamQuestionData as
select * from Exam_questions
```

8- Freelancing_table:

```
create proc GetFreelacingData as
select * from Freelancing
```

9- Hiring _table:

```
create proc GetExamHringData as
select * from Hiring
```

10- Instructor_table:

```
create proc GetInstructorData as
select * from Instructor
```

11- Instructor_course_table:

```
create proc GetInstructorCourseData as
select * from Instructor_course
```

12- Positions_table:

```
create proc GetPositionData as
select * from Positions
```

13- Projects_table:

```
create proc GetProjectsData  as
select * from Projects
```

14- Ques_Exam_table:

```
create proc GetQues_ExamData  as
select * from Ques_Exam
```

15- Question_options_table:

```
create proc GetQuestion_optionsData  as
select * from Question_options
```

16- Skills_table:

```
create proc GetSkillsData  as
select * from Skills
```

17- St_Exam_table:

```
create proc GetStExamData  as
select * from St_Exam
```

18- Student_table:

```
create proc GetStudentData  as
select * from Student
```

19- Student_Certificates_table:

```
create proc GetStudentCertificatesData  as
select * from Student_Certificates
```

20- Student_course_table:

```
create proc GetStudentCourseData  as
select * from Student_course
```

21- Student_Project_table:

```
create proc GetStudenProjectstData  as
select * from Student_Project
```

22- Student_skills_table:

```
create proc GetStudentSkillsData  as
select * from Student_skills
```

23- Topic_table:

```
create proc GetTopicData  as
select * from Topic
```

24- Training_table:

```
create proc GettraningData  as
select * from Training
```

Insert stored procedures:

1- Certificates_table:

```
CREATE PROCEDURE InsertCertificate
    @Cert_Name VARCHAR(100),
    @Issuer VARCHAR(100),
    @Issue_Date DATE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Certificates] ([Cert_Name], [Issuer], [Issue_Date])
    VALUES (@Cert_Name, @Issuer, @Issue_Date);

    SELECT SCOPE_IDENTITY() AS Cert_Id;
END;
```

2- Companies_table:

```
create PROCEDURE InsertCompany
    @Company_Name VARCHAR(100),
    @Location VARCHAR(100),
   @Training_ID INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Companies] ([Company_Name], [Location],[Training_ID])
    VALUES (@Company_Name, @Location, @Training_ID);
END;
```

3- Course_table:

```
create PROCEDURE InsertCourse
    @Crs_Name VARCHAR(100),
    @Crs_Duration INT,
    @Top_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Course] ([Crs_Name], [Crs_Duration], [Top_Id])
    VALUES (@Crs_Name, @Crs_Duration, @Top_Id);
END;
```

4- Department_table:

```
CREATE PROCEDURE InsertDepartment
    @Dept_Name VARCHAR(100),
    @Dept_Location VARCHAR(100),
    @Dept_Manager INT,
    @Manager_hiredate DATE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Department] ([Dept_Name], [Dept_Location], [Dept_Manager],
[Manager_hiredate])
```

```
    VALUES (@Dept_Name, @Dept_Location, @Dept_Manager, @Manager_hiredate);
END;
```

5- Exam_table:

```
CREATE PROCEDURE InsertExam
    @Exam_Name VARCHAR(100),
    @Exam_Date DATE,
    @Crs_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Exam] ([Exam_Name], [Exam_Date], [Crs_Id])
    VALUES (@Exam_Name, @Exam_Date, @Crs_Id);
END;
```

6- Exam_answers_table:

```
Create PROCEDURE InsertExamAnswer
    @Answer_Text VARCHAR(100),
   @Question_Id INT,
   @Score INT,
   @st_id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Exam_answers] ([Answer_Text],[Question_Id] ,[Score],[St_Id])
    VALUES (@Answer_Text,@Question_Id, @Score,  @st_id );
END;
```

7- Exam_questions_table:

```
CREATE PROCEDURE InsertExamQuestion
    @Question_Text VARCHAR(100),
    @Question_Type VARCHAR(50),
       @Correct_ans int,
       @Category varchar(100)

AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Exam_questions] ( [Question_Text],
[Question_Type],[Correct_Answer_Id],[Category_Name])
    VALUES ( @Question_Text, @Question_Type,@Correct_ans ,@Category);
END;
```

8- Freelancing_table:

```
CREATE PROCEDURE InsertFreelancing
    @Freelance_Name VARCHAR(100),
    @St_Id INT
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO [dbo].[Freelancing] ([Freelance_Name],[St_Id])
    VALUES (@Freelance_Name,@St_Id);
END;
```

9- Hiring _table:

```
CREATE PROCEDURE InsertHiring
    @Company_Id INT,
    @Position_Id INT,
    @St_Id INT,
    @Hire_Date DATE,
    @ContractLength INT
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO [dbo].[Hiring] ([Company_Id], [Position_Id], [St_Id], [Hire_Date],
[ContractLength])
    VALUES (@Company_Id, @Position_Id, @St_Id, @Hire_Date, @ContractLength);
END;
```

10- Instructor_table:

```
CREATE PROCEDURE InsertInstructor
    @Ins_Name VARCHAR(100),
    @Salary INT,
    @Dept_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Instructor] ([Ins_Name], [Salary], [Dept_Id])
    VALUES (@Ins_Name, @Salary, @Dept_Id);
END;
```

11- Instructor_course_table:

```
CREATE PROCEDURE InsertInstructorCourse
    @Ins_Id INT,
    @Crs_Id INT,
    @Evaluation VARCHAR(100)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Instructor_course] ([Ins_Id], [Crs_Id], [Evaluation])
    VALUES (@Ins_Id, @Crs_Id, @Evaluation);
END;
```

12- Positions_table:

```
CREATE PROCEDURE InsertPosition
    @Position_Name VARCHAR(100),
    @Salary INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Positions] ([Position_Name], [Salary])
    VALUES (@Position_Name, @Salary);
END;
```

13- Projects_table:

```
CREATE PROCEDURE InsertProject
    @Project_Name VARCHAR(100),
    @Start_Date DATE,
    @End_Date DATE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Projects] ([Project_Name], [Start_Date], [End_Date])
    VALUES (@Project_Name, @Start_Date, @End_Date);
END;
```

14- Ques_Exam_table:

```
CREATE PROCEDURE InsertQuesExam
    @Ex_Id INT,
    @Q_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Ques_Exam] ([Ex_Id], [Q_Id])
    VALUES (@Ex_Id, @Q_Id);
END;
```

15- Question_options_table:

```
CREATE PROCEDURE InsertQuestionOption
    @Option_Text VARCHAR(10),
    @IsCorrect INT,
    @Question_Id INT

AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Question_options] ([Option_Text], [IsCorrect], [Question_Id])
    VALUES (@Option_Text, @IsCorrect, @Question_Id);
END;
```

16- Skills_table:

```
CREATE PROCEDURE InsertSkill
    @Skill_Name VARCHAR(100)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Skills] ([Skill_Name])
    VALUES (@Skill_Name);
END;
```

17- St_Exam_table:

```
CREATE PROCEDURE InsertStExam
    @St_Id INT,
    @Ex_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[St_Exam] ([St_Id], [Ex_Id])
    VALUES (@St_Id, @Ex_Id);
END;
```

18- Student_table:

```
CREATE PROCEDURE InsertStudent
    @St_Fname VARCHAR(100),
    @St_Lname VARCHAR(100),
    @St_Address VARCHAR(100),
    @St_Age INT,
    @Email VARCHAR(100),
    @Dept_Id INT,
    @St_super INT,
    @Gender VARCHAR(50)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Student] ([St_Fname], [St_Lname], [St_Address], [St_Age],
[Email], [Dept_Id], [St_super], [Gender])
    VALUES (@St_Fname, @St_Lname, @St_Address, @St_Age, @Email, @Dept_Id, @St_super,
@Gender);
END;
```

19- Student_Certificates_table:

```
CREATE PROCEDURE InsertStudentCertificate
    @St_Id INT,
    @Cert_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Student_Certificates] ([St_Id], [Cert_Id])
    VALUES (@St_Id, @Cert_Id);
END;
```

20- Student_course_table:

```
CREATE PROCEDURE InsertStudentCourse
    @Crs_Id INT,
    @St_Id INT,
    @Grade INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Student_course] ([Crs_Id], [St_Id], [Grade])
    VALUES (@Crs_Id, @St_Id, @Grade);
END;
```

21- Student_Project_table:

```
CREATE PROCEDURE InsertStudentProject
    @St_Id INT,
    @Pro_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Student_Project] ([St_Id], [Pro_Id])
    VALUES (@St_Id, @Pro_Id);
END;
```

22- Student_skills_table:

```
CREATE PROCEDURE InsertStudentSkill
    @Sk_Id INT,
    @St_Id INT
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Student_skills] ([Sk_Id], [St_Id])
    VALUES (@Sk_Id, @St_Id);
END;
```

23- Topic_table:

```
CREATE PROCEDURE InsertTopic
    @Top_Name VARCHAR(100)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Topic] ([Top_Name])
    VALUES (@Top_Name);
END;
```

24- Training_table:

```
CREATE PROCEDURE InsertTraining
    @Train_Name VARCHAR(100),
    @Start_Date DATE,
    @End_Date DATE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Training] ([Train_Name], [Start_Date], [End_Date])
    VALUES (@Train_Name, @Start_Date, @End_Date);
END;
```

Update stored procedures:

1- Certificates_table:

```
CREATE PROCEDURE UpdateCertificate
    @CertID INT,
    @CertName varchar(100),
    @IssueDate DATE,
    @Issuer varchar(100)
AS
BEGIN
    UPDATE certificates
    SET
        cert_name = @CertName,
        issue_date = @IssueDate,
        issuer = @Issuer
    WHERE
        cert_id = @CertID;
END;
```

2- Companies_table:

```
create PROCEDURE UpdateCompany
    @Company_Id int,
    @Company_Name varchar(100),
    @Location varchar(100)

AS
BEGIN
    UPDATE companies
    SET
        Company_Name=  @Company_Name,
        Location =  @Location

    WHERE
        Company_Id = @Company_Id;
END;
```

3- Course_table:

```
CREATE PROCEDURE UpdateCourse
    @Crs_Id INT,
    @Crs_Name VARCHAR(100),
    @Crs_Duration INT

AS
BEGIN
    UPDATE Course
    SET
        Crs_Name = @Crs_Name,
        Crs_Duration = @Crs_Duration

    WHERE
        Crs_Id = @Crs_Id;
END;
```

4- Department_table:

```
CREATE PROCEDURE UpdateDepartment
    @Dept_Id INT,
    @Dept_Name VARCHAR(100),
    @Dept_Location VARCHAR(100),
    @Dept_Manager INT,
    @Manager_hiredate DATE
AS
BEGIN
    UPDATE Department
    SET
        Dept_Name = @Dept_Name,
        Dept_Location = @Dept_Location,
        Dept_Manager = @Dept_Manager,
        Manager_hiredate = @Manager_hiredate
    WHERE
        Dept_Id = @Dept_Id;
END;
```

5- Exam_table:

```
CREATE PROCEDURE UpdateExam
    @Exam_Id INT,
    @Exam_Name VARCHAR(100),
    @Exam_Date DATE

AS
BEGIN
    UPDATE Exam
    SET
        Exam_Name = @Exam_Name,
        Exam_Date = @Exam_Date

    WHERE
        Exam_Id = @Exam_Id;
END;
```

6- Freelancing_table:

```
CREATE PROCEDURE UpdateFreelancing
    @Freelance_Id INT,
    @Freelance_Name VARCHAR(100),
    @St_Id INT
AS
BEGIN
    UPDATE Freelancing
    SET
        Freelance_Name = @Freelance_Name,
        St_Id = @St_Id
    WHERE
        Freelance_Id = @Freelance_Id;
END;
```

7- Hiring _table:

```
CREATE PROCEDURE UpdateHiring
    @Hiring_Id INT,

    @Position_Id INT,
    @St_Id INT,
    @Hire_Date DATE

AS
BEGIN
    UPDATE Hiring
    SET

        Position_Id = @Position_Id,
        St_Id = @St_Id,
        Hire_Date = @Hire_Date

    WHERE
        Hiring_Id = @Hiring_Id;
END;
```

8- Instructor_table:

```
CREATE PROCEDURE UpdateInstructor
    @Ins_Id INT,
    @Ins_Name VARCHAR(100),
    @Salary INT

AS
BEGIN
    UPDATE Instructor
    SET
        Ins_Name = @Ins_Name,
        Salary = @Salary

    WHERE
        Ins_Id = @Ins_Id;
END;
```

9- Positions_table:

```
CREATE PROCEDURE UpdatePosition
    @Position_Id INT,
    @Position_Name VARCHAR(100),
    @Salary INT
AS
BEGIN
    UPDATE Positions
    SET
        Position_Name = @Position_Name,
        Salary = @Salary
    WHERE
        Position_Id = @Position_Id;
END;
```

10- Projects_table:

```
CREATE PROCEDURE UpdateProject
    @Project_Id INT,
    @Project_Name VARCHAR(100),
    @Start_Date DATE,
    @End_Date DATE
AS
BEGIN
    UPDATE Projects
    SET
        Project_Name = @Project_Name,
        Start_Date = @Start_Date,
        End_Date = @End_Date
    WHERE
        Project_Id = @Project_Id;
END;
```

11- Skills_table:

```
CREATE PROCEDURE UpdateSkills
    @Skill_Id INT,
    @Skill_Name VARCHAR(100)
AS
BEGIN
    UPDATE Skills
    SET Skill_Name = @Skill_Name
    WHERE Skill_Id = @Skill_Id;
END;
```

12- Student_table:

```
CREATE PROCEDURE UpdateStudent
    @St_Id INT,

    @St_Lname VARCHAR(100),
    @St_Address VARCHAR(100),
    @St_Age INT,
    @Email VARCHAR(100),
    @Dept_Id INT,

    @Gender VARCHAR(50)
AS
BEGIN
    UPDATE Student
    SET
        St_Lname = @St_Lname,
        St_Address = @St_Address,
        St_Age = @St_Age,
        Email = @Email,
        Dept_Id = @Dept_Id,

        Gender = @Gender
    WHERE St_Id = @St_Id;
END;
```

13- Topic_table:

```
CREATE PROCEDURE UpdateTopic
    @Top_Id INT,
    @Top_Name VARCHAR(100)
AS
BEGIN
    UPDATE Topic
    SET
        Top_Name = @Top_Name
    WHERE Top_Id = @Top_Id;
END;
```

14- Training_table:

```
CREATE PROCEDURE UpdateTraining
    @Train_ID INT,
    @Train_Name VARCHAR(100),
    @Start_Date DATE,
    @End_Date DATE
AS
BEGIN
    UPDATE Training
    SET
        Train_Name = @Train_Name,
        Start_Date = @Start_Date,
        End_Date = @End_Date
    WHERE Train_ID = @Train_ID;
END;
```

Delete stored procedures:

1- Certificates_table:

```
CREATE PROCEDURE DeleteCertificate
    @CertificateId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Student_Certificates table
    DELETE FROM dbo.Student_Certificates WHERE Cert_Id = @CertificateId;

    -- Delete records from Certificates table
    DELETE FROM dbo.Certificates WHERE Cert_Id = @CertificateId;
END;
```

2- Companies_table:

```
CREATE PROCEDURE DeleteCompany
    @CompanyId INT
AS
BEGIN
    SET NOCOUNT ON;
    -- Delete records from Hiring table
    DELETE FROM dbo.Hiring WHERE Company_Id = @CompanyId;
    -- Delete records from Companies table
    DELETE FROM dbo.Companies WHERE Company_Id = @CompanyId;
END
```

3- Course_table:

```
CREATE PROCEDURE DeleteCourse
    @CourseId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Instructor_course table
    DELETE FROM dbo.Instructor_course WHERE Crs_Id = @CourseId;

    -- Delete records from Student_course table
    DELETE FROM dbo.Student_course WHERE Crs_Id = @CourseId;

    -- Delete records from Exam table
    DELETE FROM dbo.Exam WHERE Crs_Id = @CourseId;

    -- Delete records from Course table
    DELETE FROM dbo.Course WHERE Crs_Id = @CourseId;
END
```

4- Department_table:

```
CREATE PROCEDURE DeleteDepartment
    @DepartmentId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Instructor table
    DELETE FROM dbo.Instructor WHERE Dept_Id = @DepartmentId;

    -- Delete records from Student table
    DELETE FROM dbo.Student WHERE Dept_Id = @DepartmentId;

    -- Delete records from Department table
    DELETE FROM dbo.Department WHERE Dept_Id = @DepartmentId;
END
```

5- Exam_table:

```
CREATE PROCEDURE DeleteExam
    @ExamId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Ques_Exam table
    DELETE FROM dbo.Ques_Exam WHERE Ex_Id = @ExamId;

    -- Delete records from Exam table
    DELETE FROM dbo.Exam WHERE Exam_Id = @ExamId;
END
```

6- Exam_questions_table:

```
CREATE PROCEDURE DeleteExamQuestions
    @QuestionId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Question_options table
    DELETE FROM dbo.Question_options WHERE Question_Id = @QuestionId;

    -- Delete records from Ques_Exam table
    DELETE FROM dbo.Ques_Exam WHERE Q_Id = @QuestionId;

    -- Delete records from Exam_questions table
    DELETE FROM dbo.Exam_questions WHERE Question_Id = @QuestionId;
END
```

7- Freelancing_table:

```
CREATE PROCEDURE DeleteFreelancing
    @FreelanceId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Freelancing table
    DELETE FROM dbo.Freelancing WHERE Freelance_Id = @FreelanceId;
END
```

8- Instructor_table:

```
CREATE PROCEDURE DeleteInstructor
    @InstructorId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Instructor_course table
    DELETE FROM dbo.Instructor_course WHERE Ins_Id = @InstructorId;

    -- Delete records from Instructor table
    DELETE FROM dbo.Instructor WHERE Ins_Id = @InstructorId;
END
```

9- Positions_table:

```
CREATE PROCEDURE DeletePosition
    @PositionId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Hiring table
    DELETE FROM dbo.Hiring WHERE Position_Id = @PositionId;

    -- Delete records from Instructor_course table
    DELETE FROM dbo.Instructor_course WHERE Evaluation = 'Position' AND Crs_Id =
@PositionId;
```

```
    -- Delete records from Positions table
    DELETE FROM dbo.Positions WHERE Position_Id = @PositionId;
END
```

10- Projects_table:

```
CREATE PROCEDURE DeleteProject
    @ProjectId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Student_Project table
    DELETE FROM dbo.Student_Project WHERE Pro_Id = @ProjectId;

    -- Delete records from Projects table
    DELETE FROM dbo.Projects WHERE Project_Id = @ProjectId;
END
```

11- Question_options_table:

```
CREATE PROCEDURE DeleteQuestionAndOptions
    @QuestionId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Question_options table
    DELETE FROM dbo.Question_options WHERE Question_Id = @QuestionId;

    -- Delete records from Ques_Exam table
    DELETE FROM dbo.Ques_Exam WHERE Q_Id = @QuestionId;

    -- Delete records from Exam_answers table
    DELETE FROM dbo.Exam_answers WHERE Question_Id = @QuestionId;

    -- Delete records from Exam_questions table
    DELETE FROM dbo.Exam_questions WHERE Question_Id = @QuestionId;
END
```

12- Skills_table:

```
CREATE PROCEDURE DeleteSkill
    @SkillId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Delete records from Student_skills table
    DELETE FROM dbo.Student_skills WHERE Sk_Id = @SkillId;

    -- Delete records from Instructor_course table
    DELETE FROM dbo.Instructor_course WHERE Evaluation = 'Skill' AND Crs_Id = @SkillId;

    -- Delete records from Skills table
    DELETE FROM dbo.Skills WHERE Skill_Id = @SkillId;
END
```

13- Student_table:

```
create PROCEDURE DeleteStudent
    @StudentId INT
AS
BEGIN
    SET NOCOUNT ON;
    -- Remove self-references
    UPDATE dbo.Student
    SET St_super = NULL
    WHERE St_super = @StudentId;
    -- Delete records from St_Exam table
    DELETE FROM dbo.St_Exam WHERE St_Id = @StudentId;
    -- Delete records from Student_course table
    DELETE FROM dbo.Student_course WHERE St_Id = @StudentId;
    -- Delete records from Hiring table
    DELETE FROM dbo.Hiring WHERE St_Id = @StudentId;
    -- Delete records from Freelancing table
    DELETE FROM dbo.Freelancing WHERE St_Id = @StudentId;
    -- Delete records from Student_Project table
    DELETE FROM dbo.Student_Project WHERE St_Id = @StudentId;
    -- Delete records from Student_skills table
    DELETE FROM dbo.Student_skills WHERE St_Id = @StudentId;
    -- Delete records from Student_Certificates table
    DELETE FROM dbo.Student_Certificates WHERE St_Id = @StudentId;
    -- Delete records from Student table
    DELETE FROM dbo.Student WHERE St_Id = @StudentId;
END
```

14- Topic_table:

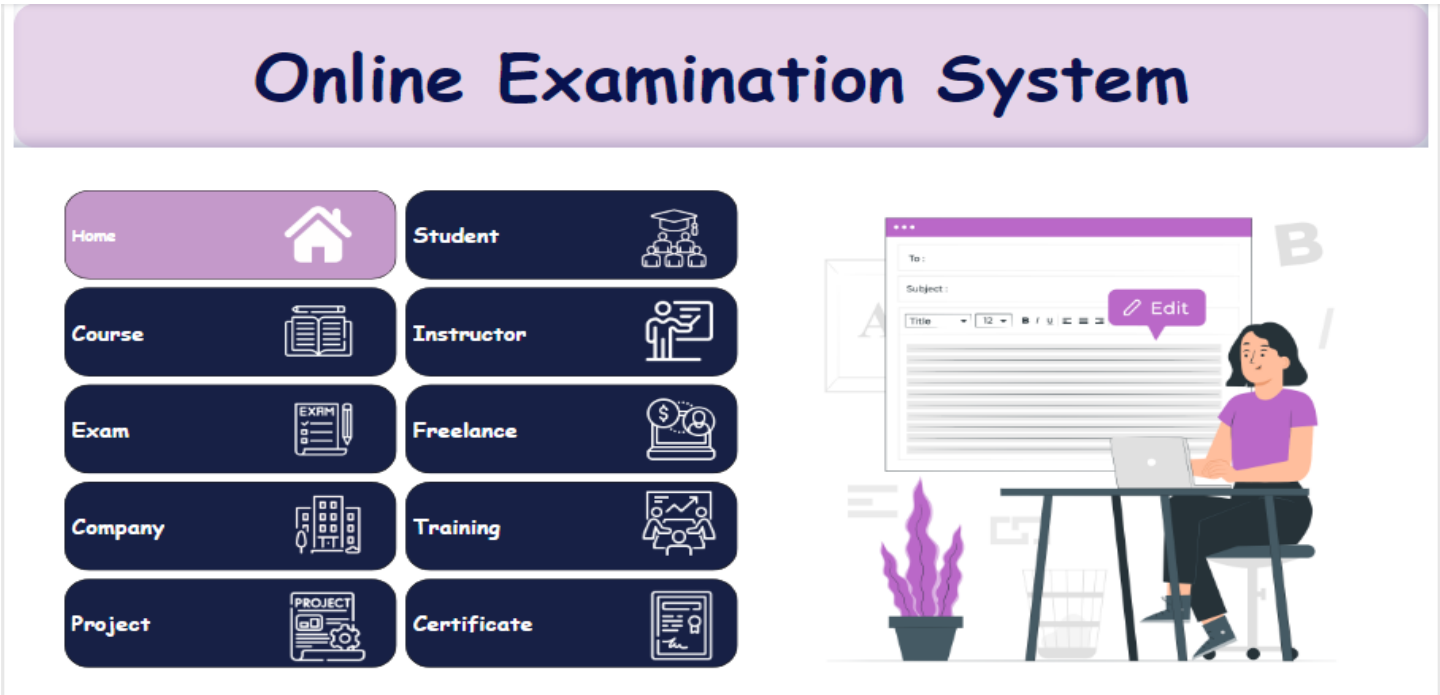
```
CREATE PROCEDURE DeleteTopic
    @TopicId INT
AS
BEGIN
    SET NOCOUNT ON;
    -- Delete records from Course table
    DELETE FROM dbo.Course WHERE Top_Id = @TopicId;
    -- Delete records from Topic table
    DELETE FROM dbo.Topic WHERE Top_Id = @TopicId;
END
```

15- Training_table:

```
CREATE PROCEDURE DeleteTraining
    @TrainingId INT
AS
BEGIN
    SET NOCOUNT ON;
    -- Delete records from Companies table
    DELETE FROM dbo.Companies WHERE Training_ID = @TrainingId;

    -- Delete records from Training table
    DELETE FROM dbo.Training WHERE Train_ID = @TrainingId;
END
```

Power bi Dashboards:

1- Overview:

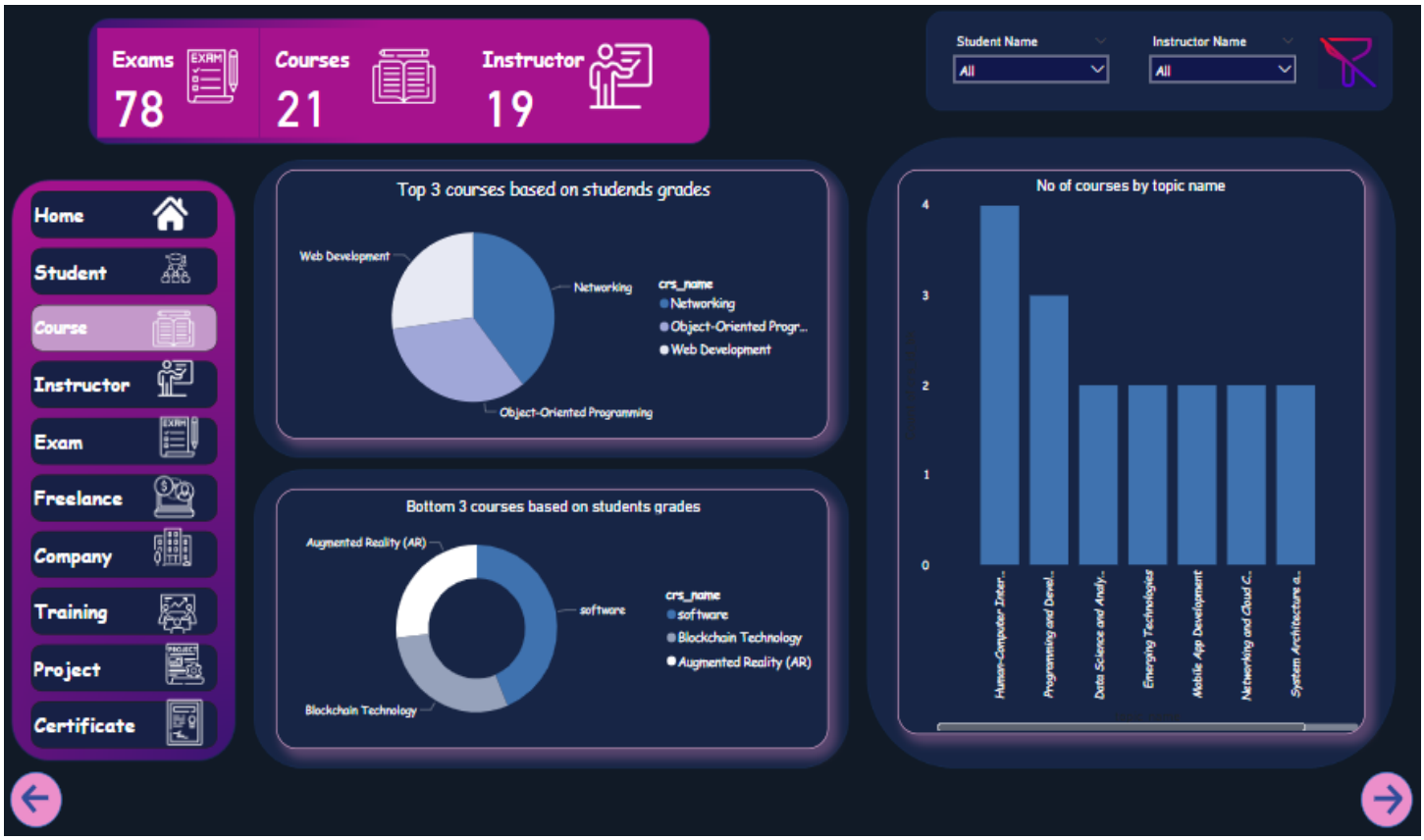


2- About Student:

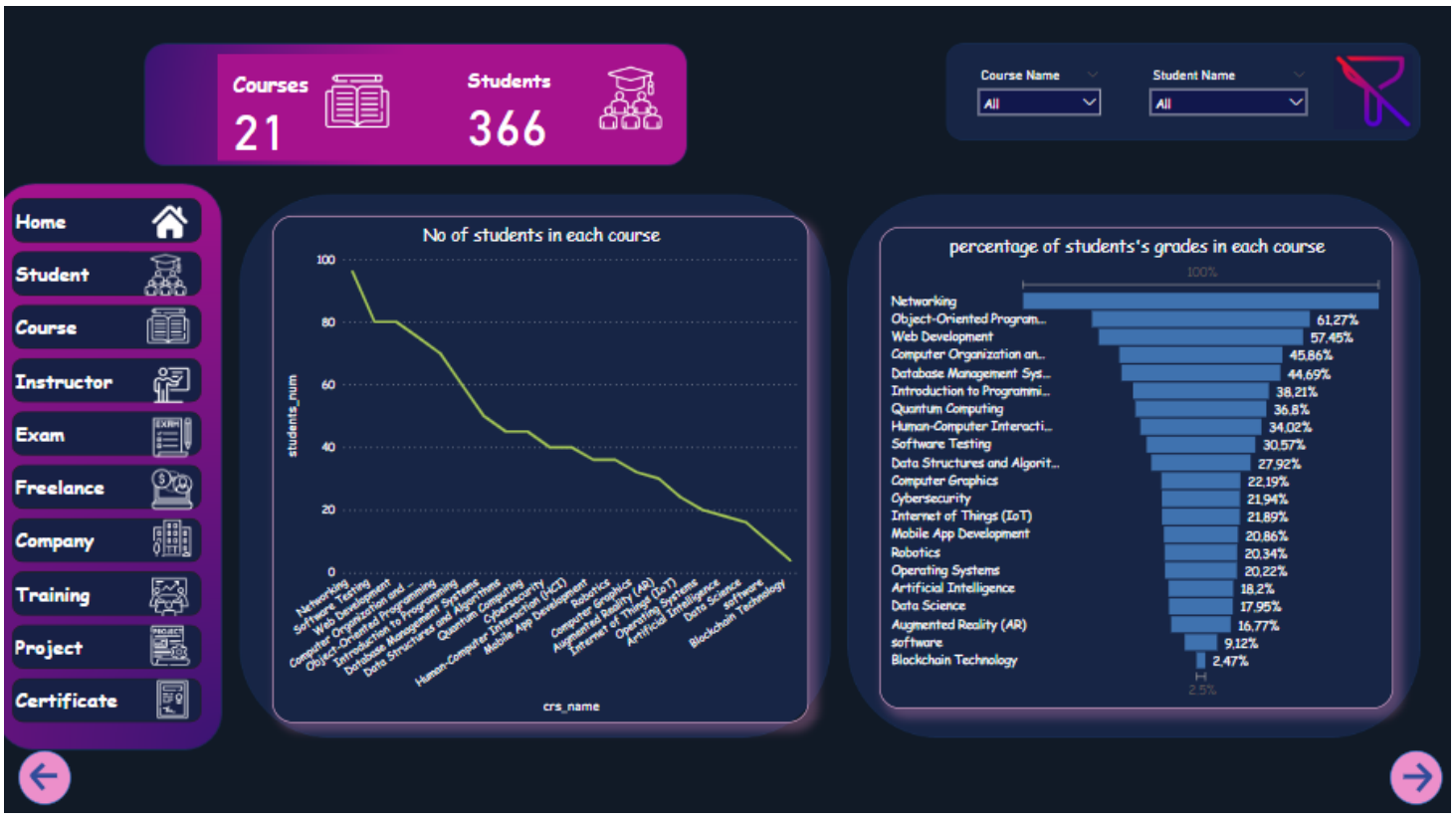

Students
366
Address
All
Course Name
All
Home
Student
Course
Instructor
Exam
Freelance
Company
Training
Project
Certificate
Student info
st_id_sk st_fname st_lname gender cert_name
1 Brianna Howell Male Certified ScrumMaster (CS
2 Brianna Howell Male Certified ScrumMaster (CS
3 Brianna Howell Male Microsoft Certified: Azure
4 Brianna Howell Male Microsoft Certified: Azure
5 Marc Kent Female Certified Information Syst
6 Andrew Clark Male
7 Michael Munoz Male
8 Michael Munoz Male
9 Tina Chandler Male
10 Christopher Davis Male Certified Pharmacy Technic
11 Christopher Davis Male Certified Pharmacy Technic
12 Christopher Davis Male Certified Pharmacy Technic
13 Jenna Wilson Female Certified ScrumMaster (CS
14 Holly Bell Female
15 Holly Bell Female
16 Cheryl Jenkins Male
17 Cheryl Jenkins Male
18 Lauren Carter Male
19 Douglas Kelley Male Certified ScrumMaster (CS
20 Willie Murphy Male Cisco Certified Network As
21 Jeffrey Hernandez Female
Students Location
UNITED STATES
MEXICO
Gulf of Mexico
Nassau
Havana

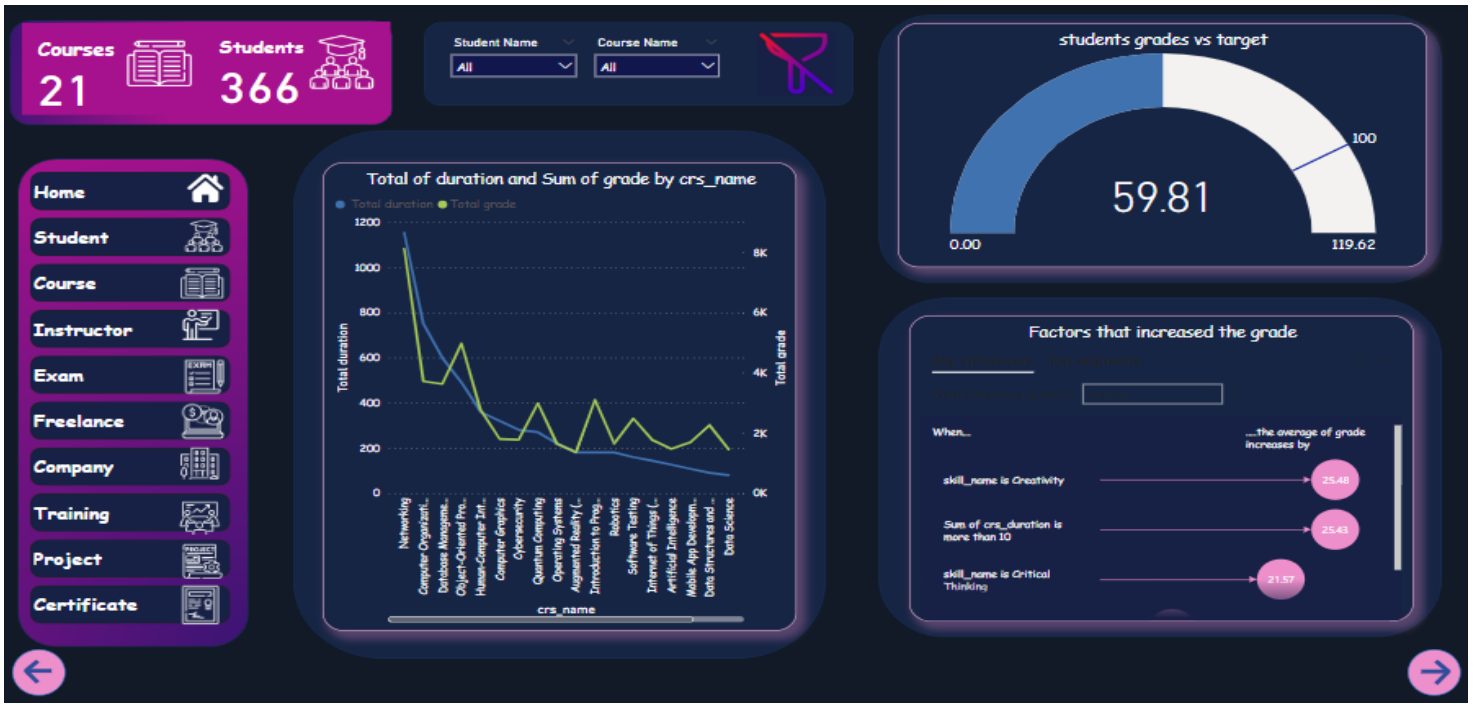

Students
366
Skills
10
Student Name
All
1
4
Home
Student
Course
Instructor
Exam
Freelance
Company
Training
Project
Certificate
No of students per grade and skills
No of students
Sum of grade
No of students
Sum of grade
skill_name
Presentation
Leadership
Critical Thinking
Adaptability
Technical
(Blank)
Creativity
Teamwork
Time Management
Emotional Intelligence
Difference between number of students in 2023 and 2024
No of students
Year
Jan 2023
Jul 2023
Jan 2024
Jul 2024

3- About Courses:

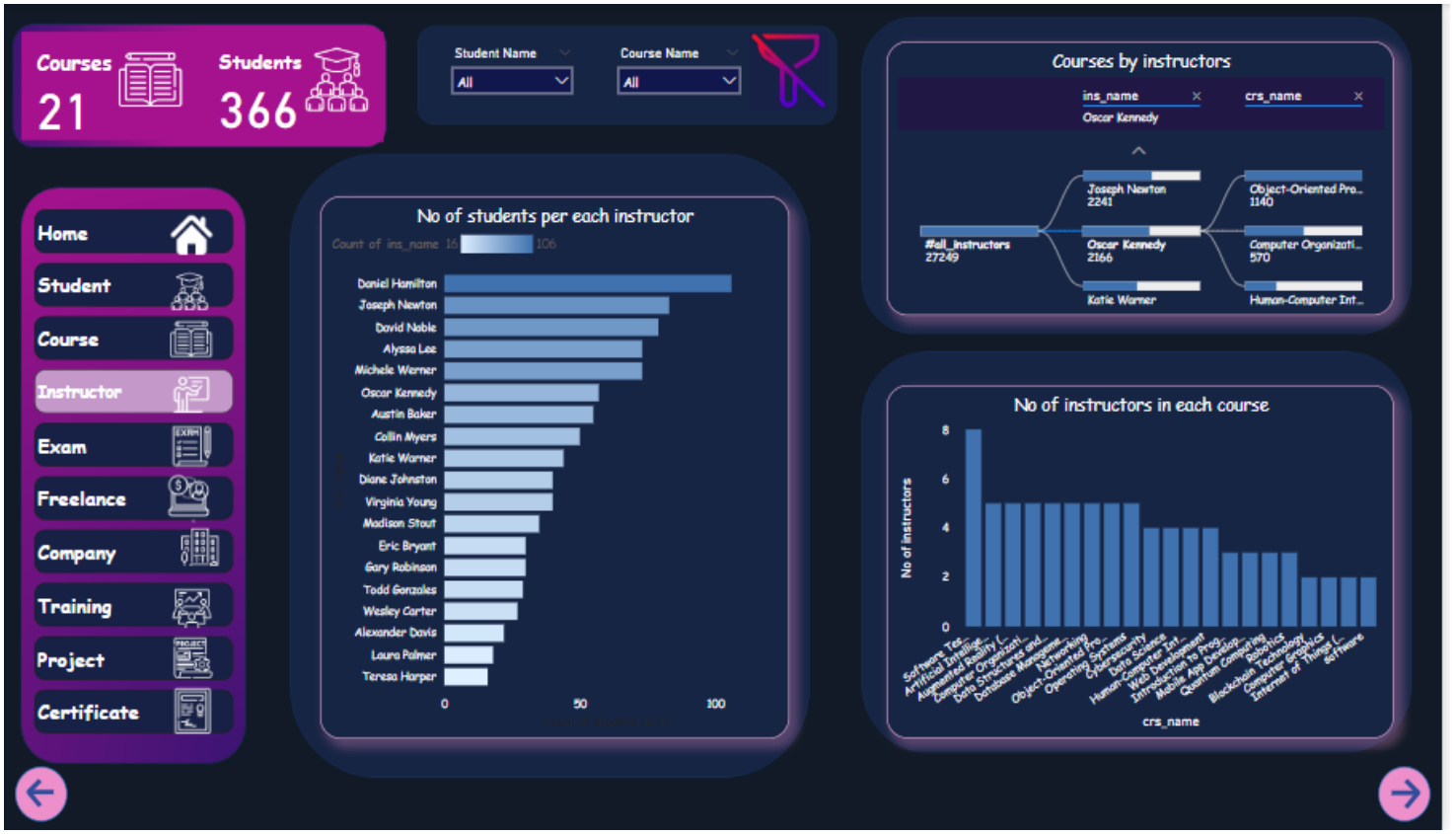



Courses
21
Topics
8
Course Name
All
Home
Student
Course
Instructor
Exam
Freelance
Company
Training
Project
Certificate
The courses per each Topic
topic_name
System Architecture...
crs_name
topics id
10359
Programming and Dev...
1764
Networking and Cloud...
1680
Software Quality and...
1536
System Architecture...
900
Emerging Technologies
825
Mobile App Developm...
640
Data Science and Ana...
544
Database Managemen...
500
Human-Computer Int...
400
Course duration
Artificial Intelligence
126
total duration
Augmented Reality (AR)
180
total duration
Blockchain Technology
28
total duration
Computer Graphics
320
total duration
Computer Organization and Architecture
750
total duration
Cybersecurity
280
total duration
Data Science
80

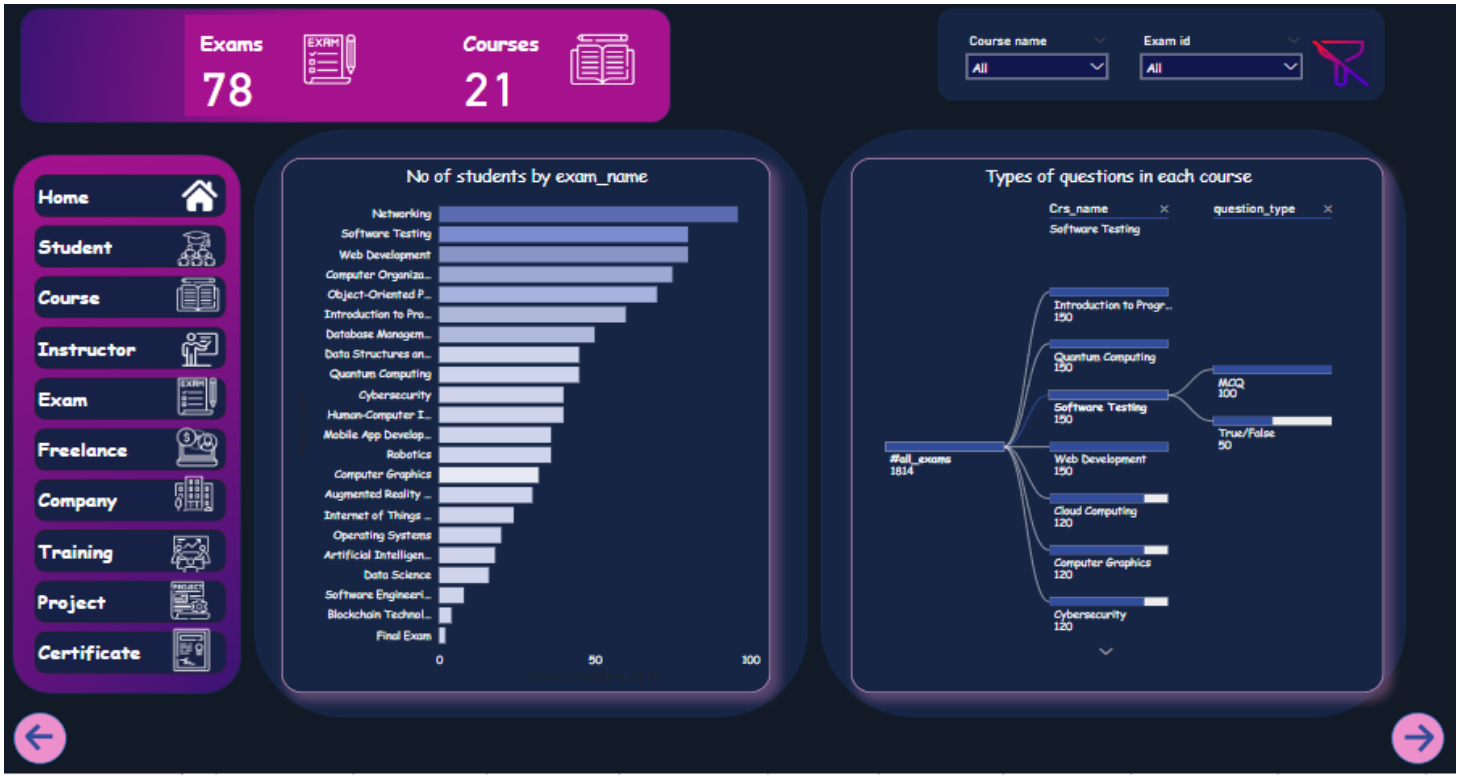

Courses
21
Students
366
Course Name
All
Student Name
All
Home
Student
Course
Instructor
Exam
Freelance
Company
Training
Project
Certificate
No of students in each course
students_num
100
80
60
40
20
0
crs_name
Networking
Software Testing
Web Development
Computer Organization and ...
Object-Oriented Programming
Introduction to Programming
Database Management Systems
Data Structures and Algorithms
Quantum Computing
Cybersecurity
Human-Computer Interaction (HCI)
Mobile App Development
Robotics
Computer Graphics
Augmented Reality (AR)
Internet of Things (IoT)
Operating Systems
Artificial Intelligence
Data Science
software
Blockchain Technology
percentage of students's grades in each course
100%
Networking
Object-Oriented Program...
61.27%
Web Development
57.45%
Computer Organization an...
45.86%
Database Management Sys...
44.69%
Introduction to Programmi...
38.21%
Quantum Computing
36.8%
Human-Computer Interacti...
34.02%
Software Testing
30.57%
Data Structures and Algorit...
27.92%
Computer Graphics
22.19%
Cybersecurity
21.94%
Internet of Things (IoT)
21.89%
Mobile App Development
20.86%
Robotics
20.34%
Operating Systems
20.22%
Artificial Intelligence
18.2%
Data Science
17.95%
Augmented Reality (AR)
16.77%
software
9.12%
Blockchain Technology
2.47%
2.5%



4- About Instructors:

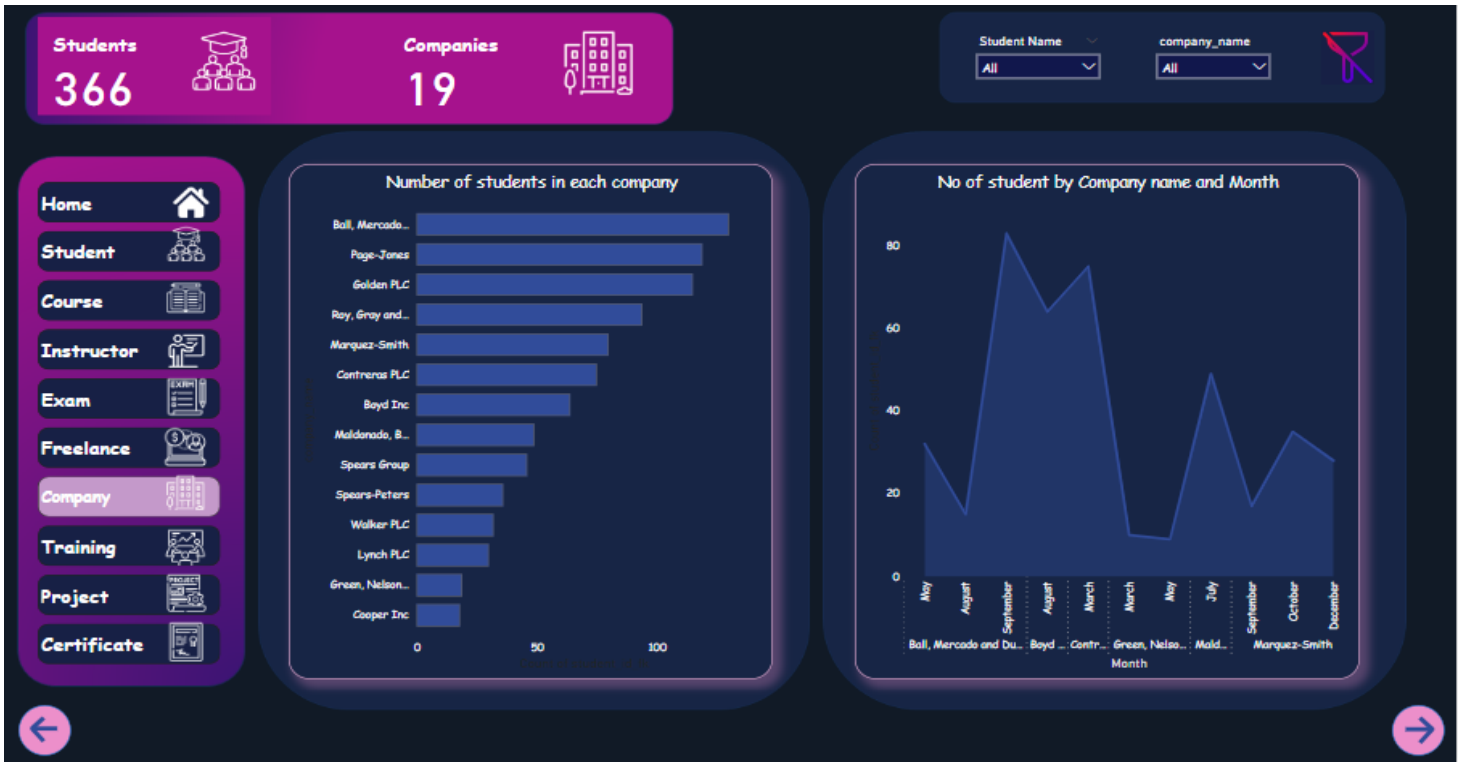


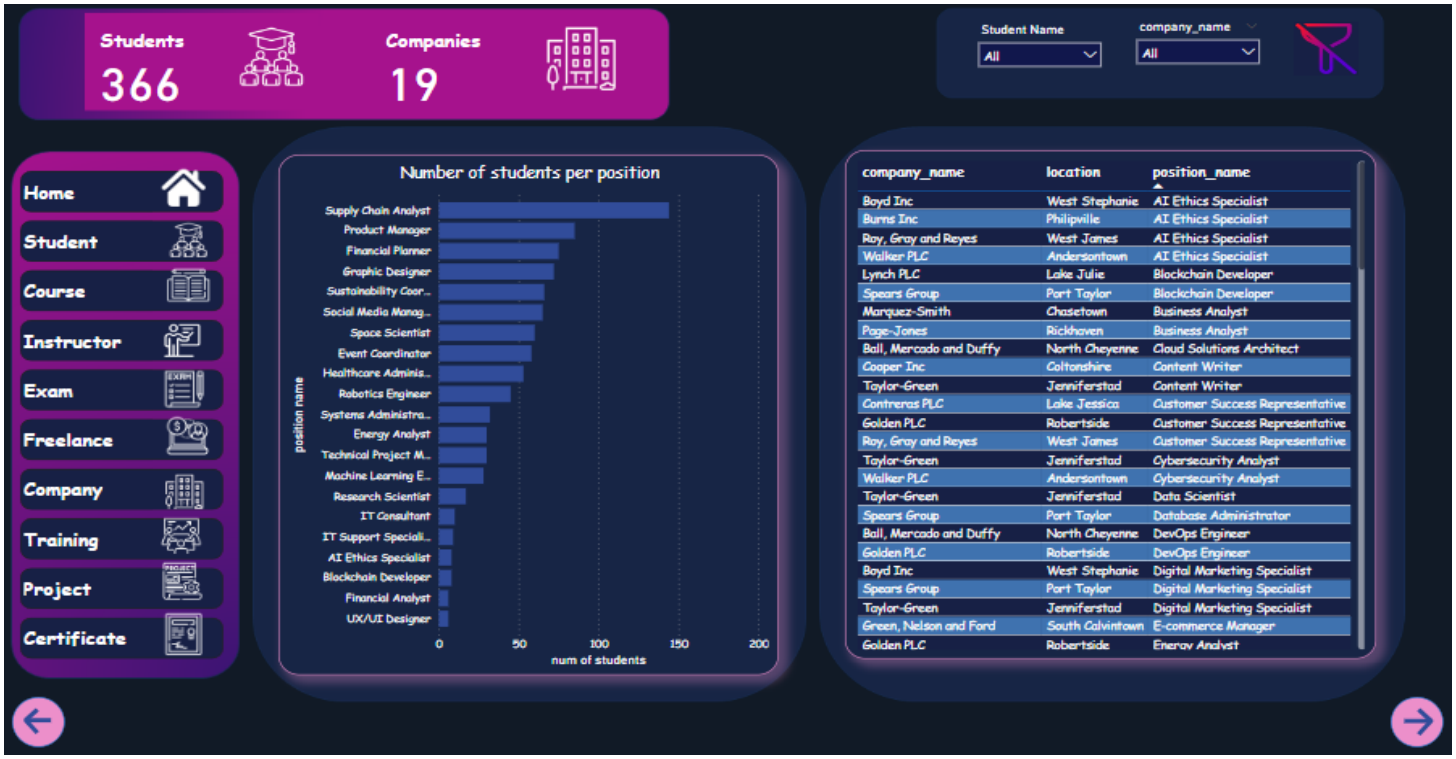
5- About Exams:

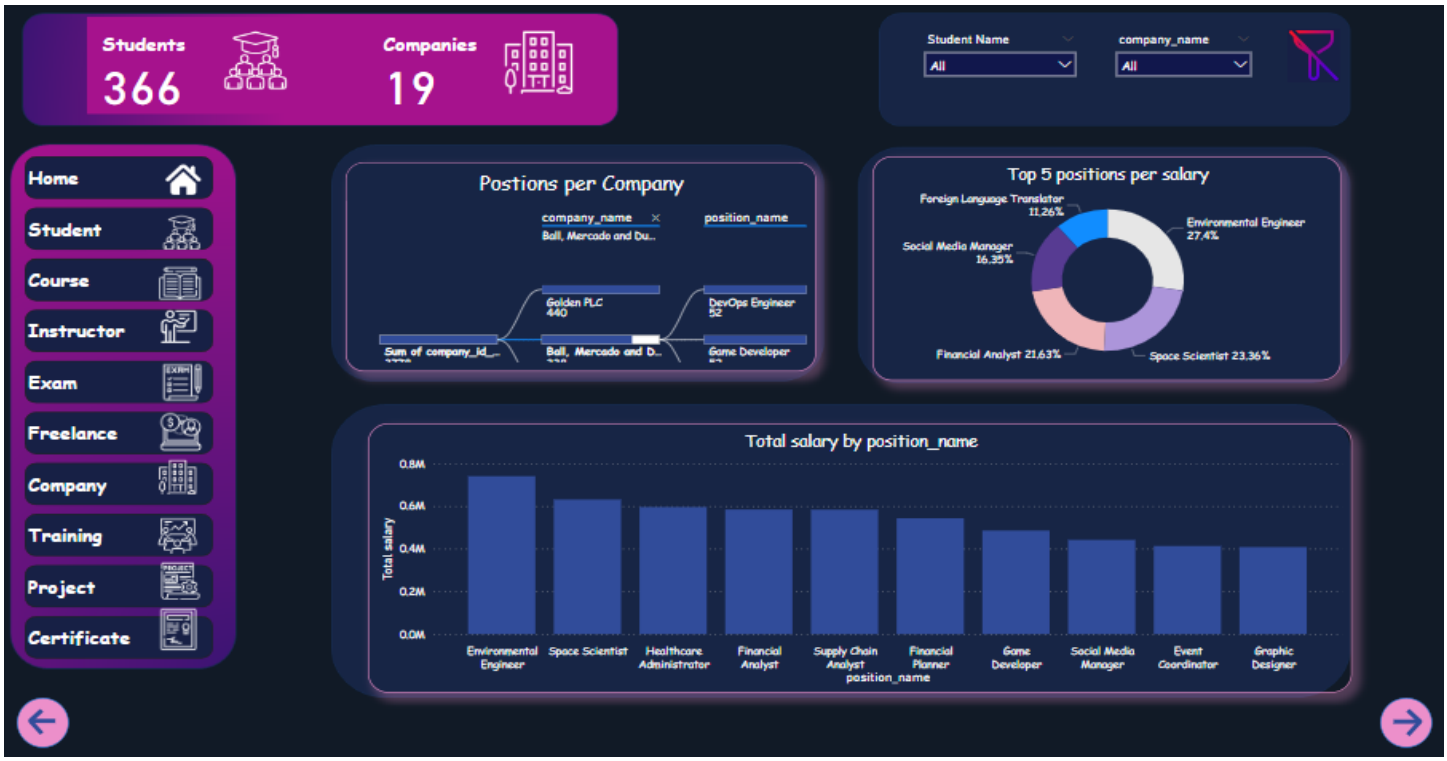


6- About Freelancing:

7- About Companies:





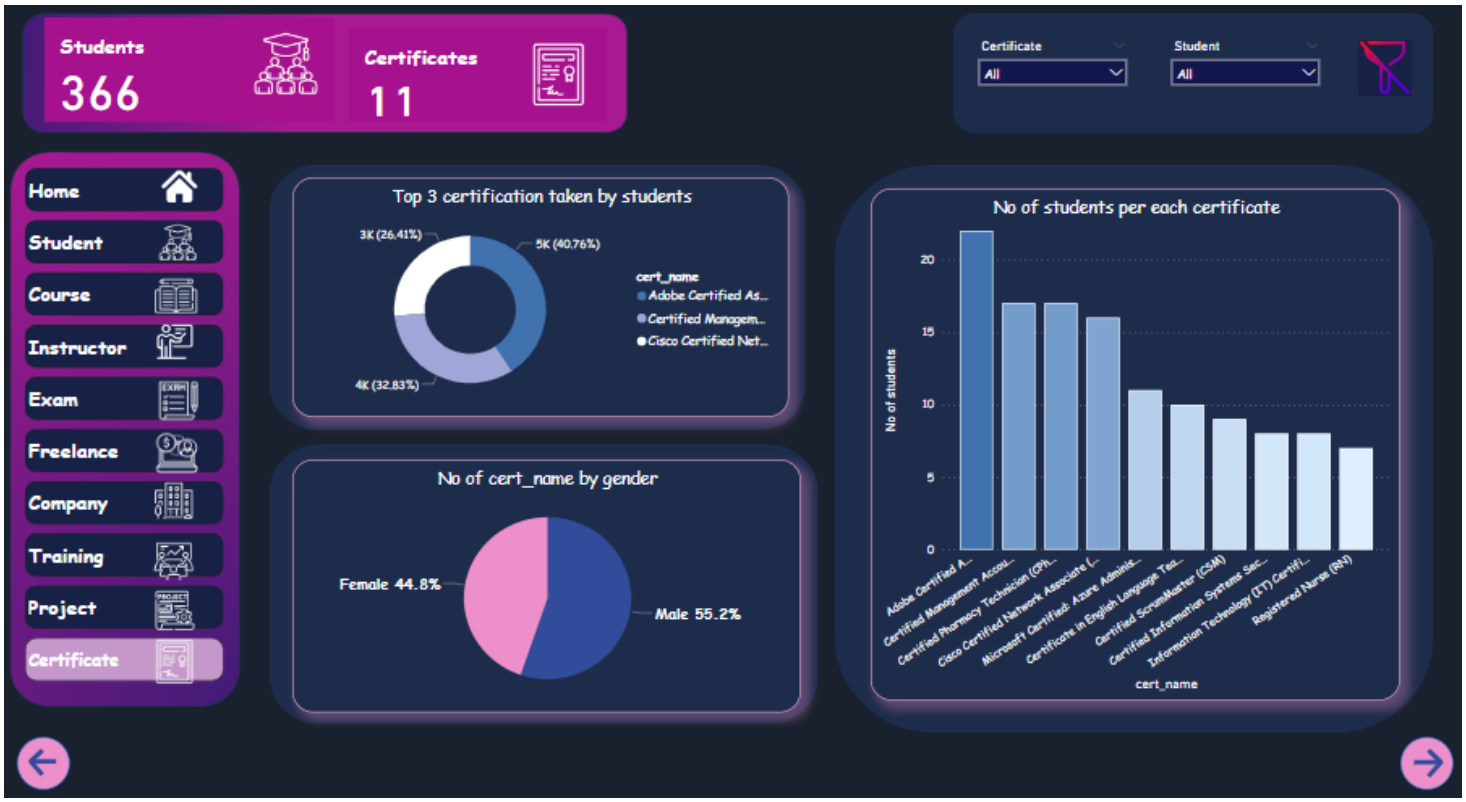

Students
366
Companies
19
Student Name
All
company_name
All
Home
Student
Course
Instructor
Exam
Freelance
Company
Training
Project
Certificate
Postions per Company
company_name
Ball, Mercado and Du...
position_name
Golden PLC
440
DevOps Engineer
92
Sum of company_id_...
Ball, Mercado and D...
Game Developer
Top 5 positions per salary
Foreign Language Translator
11.26%
Environmental Engineer
27.4%
Social Media Manager
16.35%
Financial Analyst 21.63%
Space Scientist 23.36%
Total salary by position_name
0.8M
0.6M
0.4M
0.2M
0.0M
Total salary
Environmental Engineer
Space Scientist
Healthcare Administrator
Financial Analyst
Supply Chain Analyst
Financial Planner
Game Developer
Social Media Manager
Event Coordinator
Graphic Designer
position_name


Students
366
Companies
19
Student Name
All
company_name
All
Home
Student
Course
Instructor
Exam
Freelance
Company
Training
Project
Certificate
No of companies per contract length group
Total companies
1000
500
0
1-4 years
5-8 years
9-12 years
contract_length (groups)
No of contract_length (groups) by position_name
position_name
Space Scientist
Environmental Engin...
Financial Analyst
Foreign Language Tr...
Social Media Manager
AI Ethics Specialist
Financial Planner
Game Developer
Graphic Designer
0
2
4
6
Count of contract_length (groups)
company location
EUROPE
Atlantic Ocean
AFRICA
SOUTH AMERICA
Indian Ocean

8- About Training:

9- About Projects:

10- About Certificates:



Web Application:

1- Stored procedure for Exam generation:

```
create PROCEDURE GenerateBubbleSheet
    @ExamId INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Declare variables
    DECLARE @ExamName VARCHAR(100);

    -- Get exam name
    SELECT @ExamName = Exam_Name
    FROM Exam
    WHERE Exam_Id = @ExamId;

    -- Print exam name
    PRINT 'Exam: ' + @ExamName;

    -- Declare a temporary table to store the results
    CREATE TABLE #BubbleSheet (
        QuestionText VARCHAR(200),
        OptionText VARCHAR(MAX)
    );
```

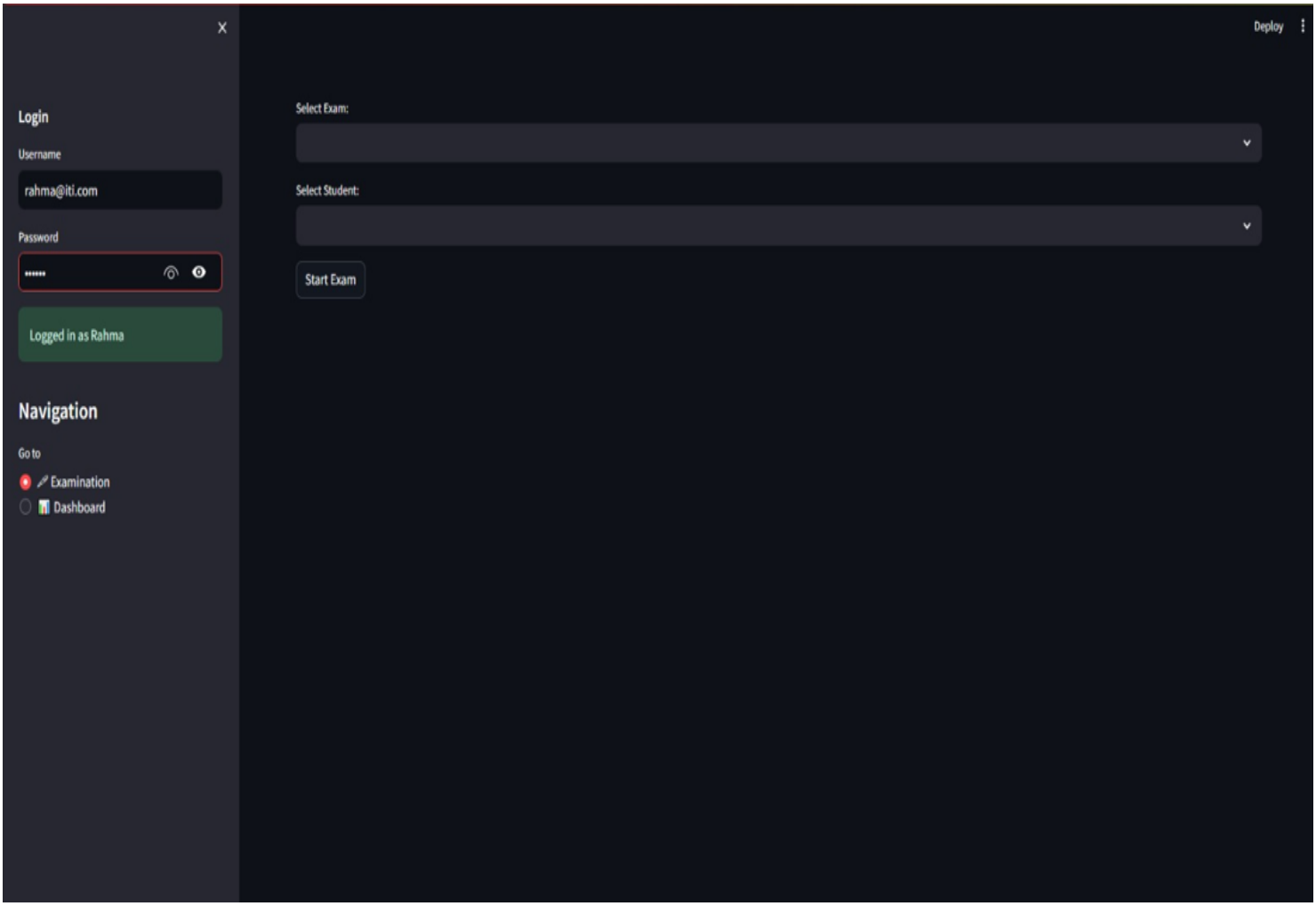
```
    -- Insert questions and options into the temporary table
    INSERT INTO #BubbleSheet (QuestionText, OptionText)
    SELECT EQ.Question_Text, QO.Option_Text
    FROM Exam_questions EQ
    INNER JOIN Ques_Exam QE ON EQ.Question_Id = QE.Q_Id
    INNER JOIN Question_options QO ON EQ.Question_Id = QO.Question_Id
    WHERE QE.Ex_Id = @ExamId;

    -- Select the results from the temporary table
    SELECT * FROM #BubbleSheet;

    -- Drop the temporary table
    DROP TABLE #BubbleSheet;
END
```

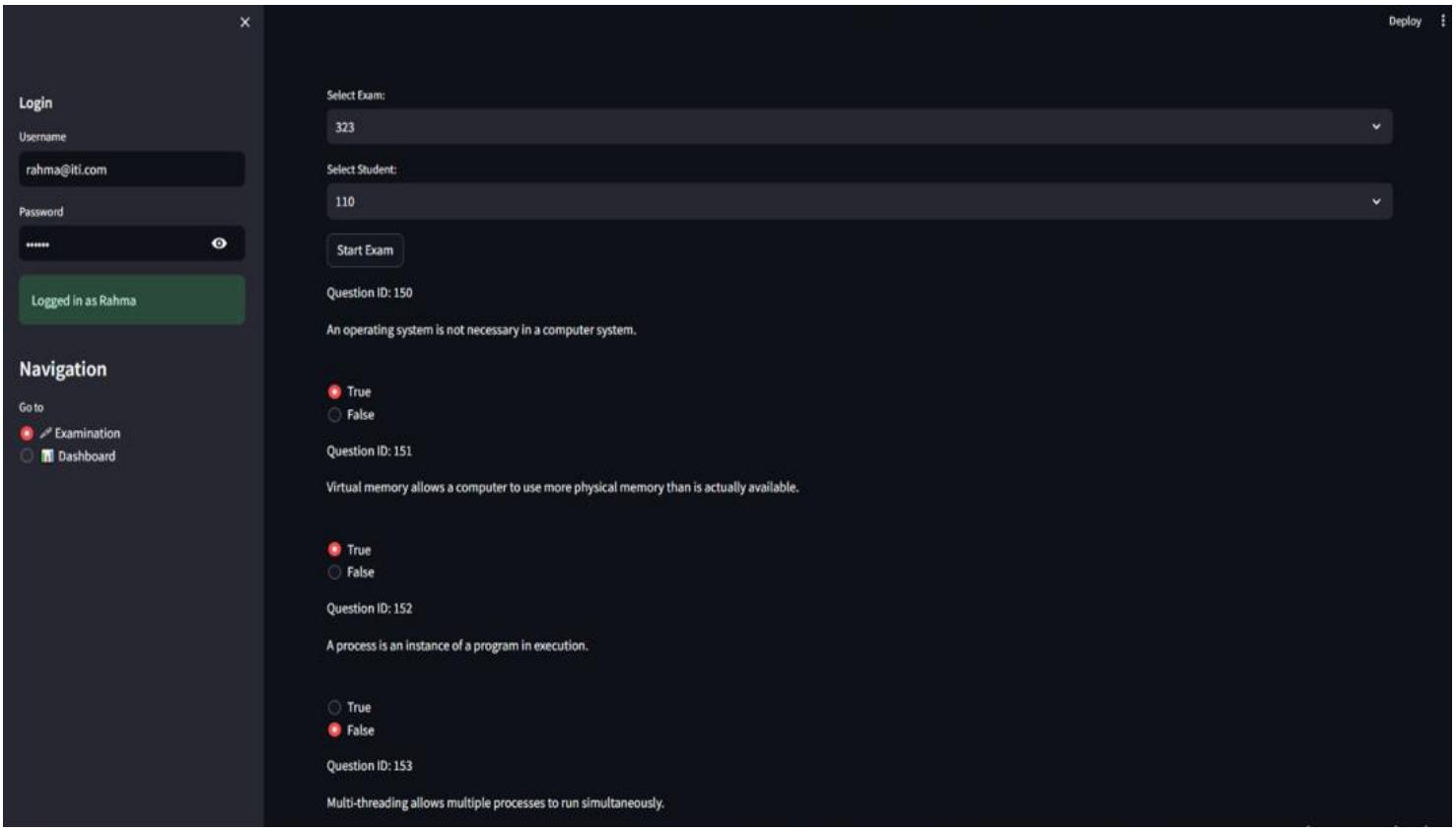
2- Stored procedure for Exam Answers:

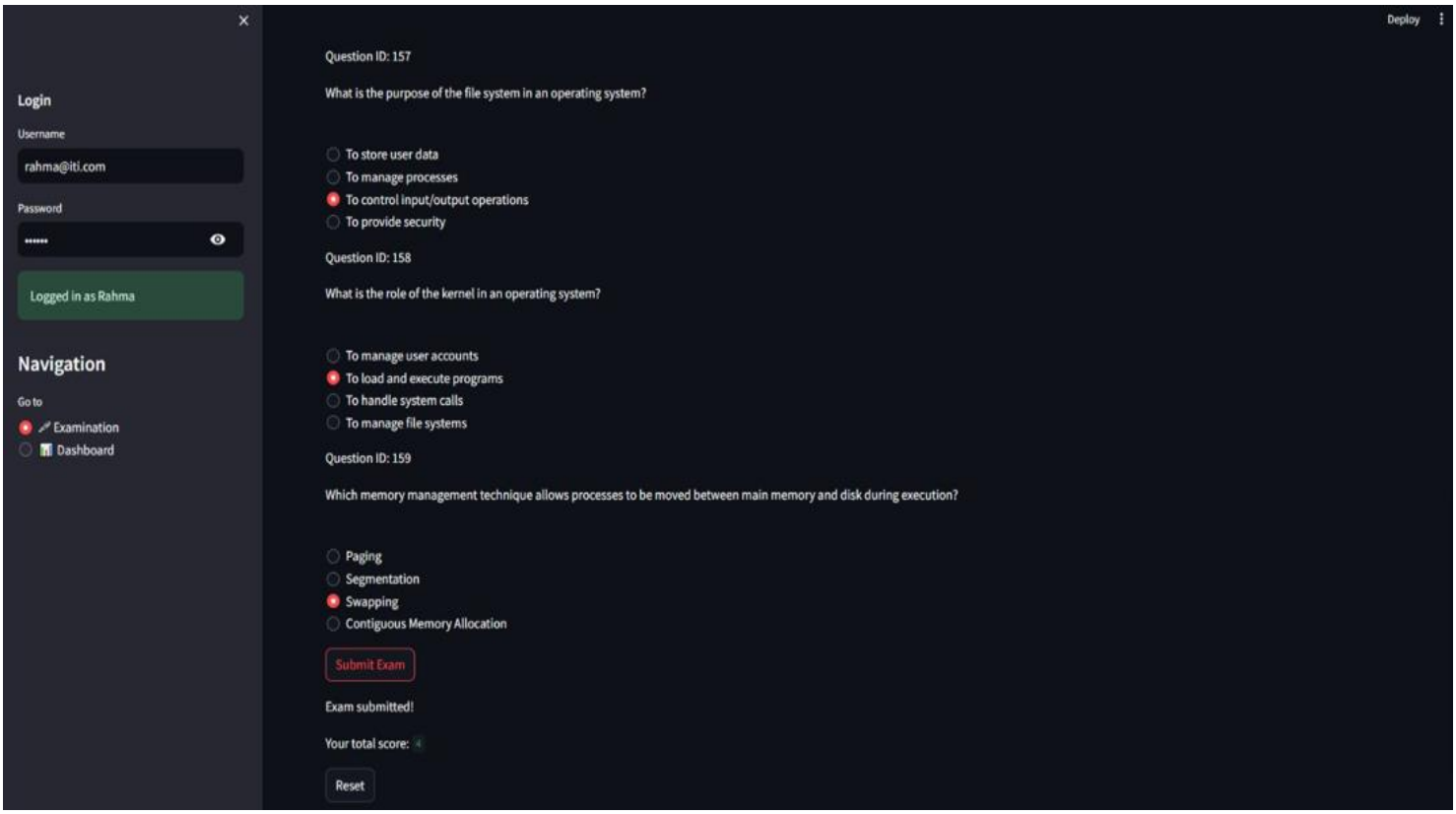
```
CREATE PROCEDURE InsertExamAnswer
    @Answer_Text VARCHAR(300),
    @Question_Id INT,
    @St_Id INT
AS
BEGIN
    DECLARE @Score INT;

    -- Check if the provided answer matches a correct option for the given question
    SELECT @Score = CASE WHEN EXISTS (
                                SELECT 1
                                FROM Question_options
                                WHERE Question_Id = @Question_Id
                                AND Option_Text = @Answer_Text
                                AND IsCorrect = 1
                            )
                            THEN 1
                            ELSE 0
                    END;

    INSERT INTO Exam_answers (Answer_Text, Question_Id, Score, St_Id)
    VALUES (@Answer_Text, @Question_Id, @Score, @St_Id);
END;

DECLARE @Answer_Text VARCHAR(300) = 'False';
DECLARE @Question_Id INT = 140; -- Provide the appropriate Question_Id
DECLARE @St_Id INT = 108; -- Provide the Student_Id
```

3- Stored procedure for Exam corrections:

```sql
create PROCEDURE CompareAnswers
    @Question_Id INT,
    @Selected_Answer VARCHAR(MAX)
AS
BEGIN
    DECLARE @Score INT = 0;

    -- Check if the selected answer is correct for the given question
    SELECT
        CASE WHEN EXISTS (
                SELECT 1
                FROM Question_options
                WHERE Question_Id = @Question_Id
                AND Option_Text = @Selected_Answer
                AND IsCorrect = 1
            )
            THEN 1
            ELSE 0
        END AS Score;
END;
```




Deploy
Login
Username
rahma@iti.com
Password
Logged in as Rahma
Navigation
Go to
Examination
Dashboard
Select Exam:
323
Select Student:
110
Start Exam
Question ID: 150
An operating system is not necessary in a computer system.
True
False
Question ID: 151
Virtual memory allows a computer to use more physical memory than is actually available.
True
False
Question ID: 152
A process is an instance of a program in execution.
True
False
Question ID: 153
Multi-threading allows multiple processes to run simultaneously.


Deploy
Login
Username
rahma@iti.com
Password
Logged in as Rahma
Navigation
Go to
Examination
Dashboard
Question ID: 157
What is the purpose of the file system in an operating system?
To store user data
To manage processes
To control input/output operations
To provide security
Question ID: 158
What is the role of the kernel in an operating system?
To manage user accounts
To load and execute programs
To handle system calls
To manage file systems
Question ID: 159
Which memory management technique allows processes to be moved between main memory and disk during execution?
Paging
Segmentation
Swapping
Contiguous Memory Allocation
Submit Exam
Exam submitted!
Your total score: 4
Reset