ITI Graduation Project

Team Members:

1. Omar Ashraf
2. Eslam Mohamed
3. Sohail Ali
4. Mokhtar Sameh

Online Examination System

Data Dictionary

ITI EXAM SYSTEM

Table of contents

Legend

- Primary key
- Primary key disabled
- User-defined primary key
- Unique key
- Unique key disabled
- User-defined unique key
- Active trigger
- Disabled trigger
- Many to one relationship
- User-defined many to one relationship
- One to many relationship
- User-defined one to many relationship
- Many to many relationship
- User-defined many to many relationship
- One to one relationship
- User-defined one to one relationship
- Input
- Output
- Input/Output
- Uses dependency
- User-defined uses dependency
- Used by dependency
- User-defined used by dependency

Online Examination System

1. Tables

1.1. Table: Branch

This Table displays all branches in the database

Columns

		Name	Data type	Description / Attributes
▤	🔑	Branch_ID	int	
▤		Branch_Name	varchar(100)	Nullable

Linked from

	Table	Join	Title / Name / Description
⊣	Branch_Department	**Branch**Branch_ID = Branch_Department**Branch**_ID	FK__Branch_De__Branc__03BB8E22

Unique keys

	Columns	Name / Description
🔑	Branch_ID	PK__Branch__12CEB0419B52F6D6

Used By

Name
▦ **Branch**
⊣ Branch_Department

1.2. Table: Branch_Department

The Table Displays The Branches and The Departments related to it

Columns

Name		Data type	Description / Attributes
	Branch_ID	int	**References**: Branch
	Department_ID	int	**References**: Department

Links to

Table	Join	Title / Name / Description
Branch	**Branch_Department**Branch_ID = BranchBranch_ID	FK__Branch_De__Branc__03BB8E22
Department	**Branch_Department**Department_ID = DepartmentDepartment_ID	FK__Branch_De__Depar__04AFB25B

Unique keys

Columns	Name / Description
Branch_ID, Department_ID	PK__Branch_D__139FD71C8353242E
Branch_ID, Department_ID	UQ_Branch_Department

Uses

Name
Branch_Department
Branch
Department

1.3. Table: Certificate

The Table Display The Certificates that are or should be taken by the students

Columns

Name	Data type	Description / Attributes
Certificate_ID	int	
Certificate_Name	varchar(100)	Nullable
Certificate_Hour	int	Nullable
Certificate_Website	varchar(100)	Nullable
Certificate_Date	date	Nullable

Linked from

Table	Join	Title / Name / Description
Student_Certificate	**Certificate**Certificate_ID = Student_Certificate**Student**_ID	FK__Student_C__Certi__0F2D40CE

Unique keys

Columns	Name / Description
Certificate_ID	PK__Certific__3AE412EA3CC0EEE7

Used By

Name
Certificate
Student_Certificate

1.4. Table: Choice

The Table displays The choices of The MCQ Questions in The Question Bank

Columns

Name	Data type	Description / Attributes
Choice_ID	int	
Choice_Text	varchar(MAX)	Nullable
Is_Correct	int	Nullable
Question_ID	int	Nullable References: Question

Links to

Table	Join	Title / Name / Description
Question	ChoiceQuestion_ID = QuestionQuestion_ID	FK_Choice_Question

Unique keys

Columns	Name / Description
Choice_ID	PK__Choice__D6C8DB1A31E61567
Choice_ID, Question_ID	UQ_Question_Choice

Uses

Name
Choice
Question

1.5. Table: Course

The Table displays the courses of The Students and the information related to it

Columns

Name	Data type	Description / Attributes
Course_ID	int	
Course_Name	varchar(100)	Nullable
Instructor_ID	int	Nullable References: Instructor
Exam_Duration	int	Nullable
Exam_Marks	int	Nullable

Links to

Table	Join	Title / Name / Description
Instructor	CourseInstructor_ID = InstructorInstructor_ID	FK_Instructor_Course

Linked from

Table	Join	Title / Name / Description
Question	CourseCourse_ID = QuestionCrs_ID	FK_Question_Course
Student_Course	CourseCourse_ID = Student_CourseCourse_ID	FK__Student_C__Cours__1E6F845E
Topic	CourseCourse_ID = TopicCourse_ID	FK_Topic_Course

Unique keys

Columns	Name / Description
Course_ID	PK__Course__37E005FBDB950454

Uses

Name
Course
Instructor

Used By

Name
Course
Question
Student_Course
Topic

1.6. Table: Department

The Table the Department Information of The Students.

Columns

		Name	Data type	Description / Attributes
		Department_ID	int	
		Department_Name	varchar(100)	Nullable
		Department_Description	varchar(MAX)	Nullable

Linked from

	Table	Join	Title / Name / Description
	Branch_Department	**Department**Department_ID = Branch_DepartmentDepartment_ID	FK__Branch_De__Depar__04AFB25B
	Student	**Department**Department_ID = StudentDepartment_ID	FK__Student__Departm__078C1F06

Unique keys

	Columns	Name / Description
	Department_ID	PK__Departme__151675D1944CA7C9

Used By

Name
Department
Branch_Department
Student

1.7. Table: exam_questions

The Table Displays The Questions that are or should be displayed in an exam.

Columns

Name	Data type	Description / Attributes
exam_id	int	**References**: generated_exams
question_id	int	**References**: Question

Links to

Table	Join	Title / Name / Description
generated_exams	**exam_questions**exam_id = generated_examsexam_id	FK__exam_ques__exam___23F3538A
Question	**exam_questions**question_id = QuestionQuestion_ID	FK__exam_ques__quest__24E777C3

Unique keys

Columns	Name / Description
exam_id, question_id	PK__exam_que__1E605ABD0AF710C2

Uses

Name
exam_questions
generated_exams
Question

1.8. Table: Freelancing

The Table displays the Freelancing jobs done by the students and the information related to it

Columns

		Name	Data type	Description / Attributes
		Job_ID	int	
		Job_Name	varchar(MAX)	Nullable
		Job_Website	varchar(100)	Nullable
		Job_Start_DATE	date	Nullable
		Job_Tools	varchar(100)	Nullable
		Feedback_Rating	int	Nullable
		Student_ID	int	Nullable References: Student

Links to

	Table	Join	Title / Name / Description
	Student	FreelancingStudent_ID = StudentStudent_ID	FK__Freelanci__Stude__15DA3E5D

Unique keys

	Columns	Name / Description
	Job_ID	PK__Freelanc__E76A7686E8E5A9C7

Uses

Name
Freelancing
Student

1.9. Table: generated_exams

The Table displays the exams generated by the system and the information related to it.

Columns

Name	Data type	Description / Attributes
exam_id	int	Identity / Auto increment
course_id	int	Nullable
total_mark	int	Nullable
Is_Quiz	bit	Nullable Default: 0
created_at	datetime	Nullable Default: getdate()
created_by	nvarchar(100)	Nullable Default: suser_sname()

Linked from

Table	Join	Title / Name / Description
exam_questions	generated_examsexam_id = exam_questionsexam_id	FK__exam_ques__exam___23F3538A

Unique keys

Columns	Name / Description
exam_id	PK__generate__9C8C7BE97D1DEE42

Used By

Name
generated_exams
exam_questions

1.10. Table: Hiring

The Table displays the companies that hired students and the information related to it

Columns

Name	Data type	Description / Attributes
Hiring_ID	int	
Hiring_Position	varchar(MAX)	Nullable
Hiring_Date	date	Nullable
Hiring_Company	varchar(MAX)	Nullable
Hiring_Location	varchar(100)	Nullable
Position_Type	varchar(100)	Nullable
Student_ID	int	Nullable References: Student

Links to

Table	Join	Title / Name / Description
Student	HiringStudent_ID = StudentStudent_ID	FK__Hiring__Student__12FDD1B2

Unique keys

Columns	Name / Description
Hiring_ID	PK__Hiring__888468A1D7957ABE

Uses

Name
Hiring
Student

1.11. Table: Instructor

The Table displays the information of the instructors teaching the courses.

Columns

Name	Data type	Description / Attributes
Instructor_ID	int	
First_Name	varchar(100)	Nullable
Last_Name	varchar(100)	Nullable
Birth_Date	date	Nullable
Gender	varchar(10)	Nullable
City	varchar(50)	Nullable
Email	varchar(100)	Nullable
Password	varchar(MAX)	

Linked from

Table	Join	Title / Name / Description
Course	**Instructor**Instructor_ID = CourseInstructor_ID	FK_Instructor_Course

Unique keys

Columns	Name / Description
Instructor_ID	PK__Instruct__DD4B9A8A1BFB6F08
Email	UQ_Instructor_Email

Used By

Name
Instructor
Course

1.12. Table: Question

The Table displays the questions that are in the question bank.

Columns

Name	Data type	Description / Attributes
Question_ID	int	
Question_Text	varchar(MAX)	Nullable
Question_Type	varchar(50)	Nullable
Question_Marks	bigint	Nullable
Crs_ID	int	Nullable References: Course

Links to

Table	Join	Title / Name / Description
Course	QuestionCrs_ID = CourseCourse_ID	FK_Question_Course

Linked from

Table	Join	Title / Name / Description
Choice	QuestionQuestion_ID = ChoiceQuestion_ID	FK_Choice_Question
exam_questions	QuestionQuestion_ID = exam_questionsquestion_id	FK__exam_ques__quest__24E777C3
Student_Answer	QuestionQuestion_ID = Student_AnswerQuestion_ID	FK__Student_A__Quest__2CBDA3B5

Unique keys

Columns	Name / Description
Question_ID	PK__Question__B0B2E4C6819C1DA9
Question_ID, Crs_ID	UQ_Question_Text_Per_Exam

Uses

Name
Question
Course

Used By

Name
Question
Choice
exam_questions
Student_Answer

1.13. Table: Student

The Table displays the information of the students that are in the ITI

Columns

Name	Data type	Description / Attributes
Student_ID	int	
First_Name	varchar(100)	Nullable
Last_Name	varchar(100)	Nullable
Birth_Date	date	Nullable
Gender	varchar(10)	Nullable
City	varchar(50)	Nullable
Email	varchar(100)	Nullable
Password	varchar(MAX)	
Department_ID	int	Nullable References: Department

Links to

Table	Join	Title / Name / Description
Department	StudentDepartment_ID = DepartmentDepartment_ID	FK__Student__Departm__078C1F06

Linked from

Table	Join	Title / Name / Description
Freelancing	StudentStudent_ID = FreelancingStudent_ID	FK__Freelanci__Stude__15DA3E5D
Hiring	StudentStudent_ID = HiringStudent_ID	FK__Hiring__Student__12FDD1B2
Student_Answer	StudentStudent_ID = Student_AnswerStudent_ID	FK__Student_A__Stude__2DB1C7EE
Student_Certificate	StudentStudent_ID = Student_CertificateCertificate_ID	FK__Student_C__Stude__10216507
Student_Course	StudentStudent_ID = Student_CourseStudent_ID	FK__Student_C__Stude__1D7B6025
Student_Phone	StudentStudent_ID = Student_PhoneStudent_ID	FK__Student_P__Stude__0A688BB1

Unique keys

Columns	Name / Description
Student_ID	PK__Student__A2F4E9AC82394512
Email	UQ_Student_Email

Uses

Name
Student
Department

Used By

Name
Student
Freelancing
Hiring
Student_Answer
Student_Certificate
Student_Course
Student_Phone

1.14. Table: Student_Answer

The Table displays the answers of the students that are used in exams.

Columns

Name	Data type	Description / Attributes
Student_Answer_ID	int	
Student_Answer_Text	varchar(MAX)	Nullable
Student_Answer_Mark	int	Nullable
Question_ID	int	Nullable References: Question
Student_ID	int	Nullable References: Student

Links to

Table	Join	Title / Name / Description
Question	Student_AnswerQuestion_ID = QuestionQuestion_ID	FK__Student_A__Quest__2CBDA3B5
Student	Student_AnswerStudent_ID = StudentStudent_ID	FK__Student_A__Stude__2DB1C7EE

Unique keys

Columns	Name / Description
Student_Answer_ID	PK__Student___1679E4AC19A4F501

Uses

Name
Student_Answer
Question
Student

1.15. Table: Student_Certificate

The Table displays the student ID and The Certificate ID

Columns

Name		Data type	Description / Attributes
Student_ID	int	**References**: Certificate
Certificate_ID	int	**References**: Student

Links to

Table	Join	Title / Name / Description
Certificate	**Student_Certificate**Student_ID = CertificateCertificate_ID	FK__Student_C__Certi__0F2D40CE
Student	**Student_Certificate**Certificate_ID = StudentStudent_ID	FK__Student_C__Stude__10216507

Unique keys

Columns	Name / Description
Student_ID, Certificate_ID	PK__Student___E0CB5C7040FE0741
Student_ID, Certificate_ID	UQ_Student_Certificate

Uses

Name
Student_Certificate
Certificate
Student

1.16. Table: Student_Course

The Table displays the Student ID, The Course ID and The Course Grade of The Student

Columns

Name			Data type	Description / Attributes
		Student_ID	int	**References**: Student
		Course_ID	int	**References**: Course
		Grade	int	**Nullable**

Links to

Table	Join	Title / Name / Description
Course	**Student_Course**Course_ID = CourseCourse_ID	FK__Student_C__Cours__1E6F845E
Student	**Student_Course**Student_ID = StudentStudent_ID	FK__Student_C__Stude__1D7B6025

Unique keys

Columns	Name / Description
Student_ID, Course_ID	PK__Student___018AE9F318E28916
Student_ID, Course_ID	UQ_Student_Course

Uses

Name
Student_Course
Course
Student

1.17. Table: Student_Phone

The Table Displays the Student ID and The Phone Number.

Columns

		Name	Data type	Description / Attributes
		Student_ID	int	**References**: Student
		Phone_Number	varchar(20)	

Links to

	Table	Join	Title / Name / Description
	Student	**Student_Phone**Student_ID = StudentStudent_ID	FK__Student_P__Stude__0A688BB1

Unique keys

	Columns	Name / Description
	Student_ID, Phone_Number	PK__Student___E38EDC666E37F51A

Uses

Name
Student_Phone
Student

1.18. Table: Topic

The Table displays the Topics in The Courses

Columns

Name	Data type	Description / Attributes
Topic_ID	int	
Topic_Name	varchar(100)	Nullable
Course_ID	int	Nullable References: Course

Links to

Table	Join	Title / Name / Description
Course	TopicCourse_ID = CourseCourse_ID	FK_Topic_Course

Unique keys

Columns	Name / Description
Topic_ID	PK__Topic__8DEAA425552FB490

Uses

Name
Topic
Course

2. Procedures

2.1. Procedure: DatabaseDrop

The Procedure performs database dropping

Script

```
CREATE PROCEDURE DatabaseDrop AS
	BEGIN
		-- Switch to the newly created or existing database
		EXEC('USE [Examination System]');

		-- Drop tables if they exist (starting from dependent to independent tables)
		IF OBJECT_ID('dbo.STUDENT_PHONE', 'U') IS NOT NULL DROP TABLE dbo.STUDENT_PHONE;
		IF OBJECT_ID('dbo.BRANCH_DEPARTMENT', 'U') IS NOT NULL DROP TABLE dbo.BRANCH_DEPARTMENT;
		IF OBJECT_ID('dbo.BRANCH', 'U') IS NOT NULL DROP TABLE dbo.BRANCH;
		IF OBJECT_ID('dbo.HIRING', 'U') IS NOT NULL DROP TABLE dbo.HIRING;
		IF OBJECT_ID('dbo.FREELANCING', 'U') IS NOT NULL DROP TABLE dbo.FREELANCING;
		IF OBJECT_ID('dbo.STUDENT_CERTIFICATE', 'U') IS NOT NULL DROP TABLE dbo.STUDENT_CERTIFICATE;
		IF OBJECT_ID('dbo.CERTIFICATE', 'U') IS NOT NULL DROP TABLE dbo.CERTIFICATE;
		IF OBJECT_ID('dbo.STUDENT_COURSE', 'U') IS NOT NULL DROP TABLE dbo.STUDENT_COURSE;
		IF OBJECT_ID('dbo.CHOICE', 'U') IS NOT NULL DROP TABLE dbo.CHOICE;
		IF OBJECT_ID('dbo.STUDENT_ANSWER', 'U') IS NOT NULL DROP TABLE dbo.STUDENT_ANSWER;
		IF OBJECT_ID('dbo.QUESTION', 'U') IS NOT NULL DROP TABLE dbo.QUESTION;
		IF OBJECT_ID('dbo.EXAM', 'U') IS NOT NULL DROP TABLE dbo.EXAM;
		IF OBJECT_ID('dbo.TOPIC', 'U') IS NOT NULL DROP TABLE dbo.TOPIC;
		IF OBJECT_ID('dbo.COURSE', 'U') IS NOT NULL DROP TABLE dbo.COURSE;
		IF OBJECT_ID('dbo.INSTRUCTOR', 'U') IS NOT NULL DROP TABLE dbo.INSTRUCTOR;
		IF OBJECT_ID('dbo.STUDENT', 'U') IS NOT NULL DROP TABLE dbo.STUDENT;
		IF OBJECT_ID('dbo.DEPARTMENT', 'U') IS NOT NULL DROP TABLE dbo.DEPARTMENT;

		PRINT 'All Tables Have Been Deleted Successfully.';
	END;
```

2.2. Procedure: DatabaseGeneration

The Procedure performs a Generation of Tables and Database.

Script

```
CREATE PROCEDURE DatabaseGeneration AS
          BEGIN
                    -- Switch to the newly created or existing database
                    EXEC('USE [Examination System]');

                    --Creating the Tables

                    -- Department
                    IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Department' AND schema_id = SCHEMA_ID('dbo'))
                    BEGIN
                              PRINT '========================================';
                              PRINT 'Creating [Department] table.';
                              PRINT '========================================';

                              CREATE TABLE dbo.Department (
                                        Department_ID INT PRIMARY KEY,
                                        Department_Name VARCHAR(100),
                                        Department_Description VARCHAR(MAX)
                              );

                              PRINT '========================================';
                              PRINT 'Table [Department] created.';
                              PRINT '========================================';
                    END

                    -- Branch
                    IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Branch' AND schema_id = SCHEMA_ID('dbo'))
                    BEGIN
                              PRINT '========================================';
                              PRINT 'Creating [Branch] table.';
                              PRINT '========================================';

                              CREATE TABLE dbo.Branch (
                                        Branch_ID INT PRIMARY KEY,
                                        Branch_Name VARCHAR(100)
                              );

                              PRINT '========================================';
                              PRINT 'Table [Branch] created.';
                              PRINT '========================================';
                    END

                    --Branch_Department
                    IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Branch_Department' AND schema_id =
SCHEMA_ID('dbo'))
                    BEGIN
                              PRINT '========================================';
                              PRINT 'Creating [Branch_Department] table.';
                              PRINT '========================================';

                              CREATE TABLE dbo.Branch_Department (
                                        Branch_ID INT,
                                        Department_ID INT,

                                        PRIMARY KEY(Branch_ID, Department_ID),
                                        FOREIGN KEY (Branch_ID) REFERENCES Branch(Branch_ID),
                                        FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID)
                              );

                              PRINT '========================================';
                              PRINT 'Table [Branch_Department] created.';
                              PRINT '========================================';
                    END

                    --Student
                    IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Student' AND schema_id = SCHEMA_ID('dbo'))
                    BEGIN
                              PRINT '========================================';
                              PRINT 'Creating [Student] table.';
                              PRINT '========================================';

                              CREATE TABLE dbo.Student (
                                        Student_ID INT PRIMARY KEY,
                                        First_Name VARCHAR(100),
                                        Last_Name VARCHAR(100),
                                        Birth_Date DATE,
                                        Gender VARCHAR(10),
                                        City VARCHAR(50),
                                        Email VARCHAR(100),
                                        Password VARCHAR(10),
                                        Department_ID INT,
```

```
                    FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID)
          );

          PRINT '========================================';
          PRINT 'Table [Student] created.';
          PRINT '========================================';
END

--Student_Phone
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Student_Phone' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
          PRINT '========================================';
          PRINT 'Creating [Student_Phone] table.';
          PRINT '========================================';

          CREATE TABLE dbo.Student_Phone (
                    Student_ID INT,
                    Phone_Number VARCHAR(20),

                    PRIMARY KEY(Student_ID, Phone_Number),
                    FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
          );

          PRINT '========================================';
          PRINT 'Table [Student_Phone] created.';
          PRINT '========================================';
END

--Certificate
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Certificate' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
          PRINT '========================================';
          PRINT 'Creating [Certificate] table.';
          PRINT '========================================';

          CREATE TABLE dbo.Certificate (
                    Certificate_ID INT PRIMARY KEY,
                    Certificate_Name VARCHAR(100),
                    Certificate_Hour INT,
                    Certificate_Website VARCHAR(100),
                    Certificate_Date DATE
          );

          PRINT '========================================';
          PRINT 'Table [Certificate] created.';
          PRINT '========================================';
END

--Student_Certificate
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Student_Certificate' AND schema_id =
SCHEMA_ID('dbo'))
BEGIN
          PRINT '========================================';
          PRINT 'Creating [Student_Certificate] table.';
          PRINT '========================================';

          CREATE TABLE dbo.Student_Certificate (
                    Certificate_ID INT,
                    Student_ID INT,

                    PRIMARY KEY(Certificate_ID, Student_ID),
                    FOREIGN KEY (Certificate_ID) REFERENCES Certificate(Certificate_ID),
                    FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
          );

          PRINT '========================================';
          PRINT 'Table [SLudent Certificate] created.';
          PRINT '========================================';
END

--Hiring
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Hiring' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
          PRINT '========================================';
          PRINT 'Creating [Hiring] table.';
          PRINT '========================================';

          CREATE TABLE dbo.Hiring (
                    Hiring_ID INT PRIMARY KEY,
                    Hiring_Position VARCHAR(MAX),
                    Hiring_Date DATE,
                    Hiring_Company VARCHAR(MAX),
                    Hiring_Location VARCHAR(100),
                    Position_Type VARCHAR(100),
                    Student_ID INT,

                    FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
          );

          PRINT '========================================';
          PRINT 'Table [Hiring] created.';
          PRINT '========================================';
```

```
END

--Freelancing
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Freelancing' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
        PRINT '========================================';
        PRINT 'Creating [Freelancing] table.';
        PRINT '========================================';

        CREATE TABLE dbo.Freelancing (
                Job_ID INT PRIMARY KEY,
                Job_Name VARCHAR(MAX),
                Job_Website VARCHAR(100),
                Job_Start_DATE DATE,
                Job_Tools VARCHAR(100),
                Feedback_Rating INT,
                Student_ID INT,

                FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
        );

        PRINT '========================================';
        PRINT 'Table [Freelancing] created.';
        PRINT '========================================';
END

--Instructor
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Instructor' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
        PRINT '========================================';
        PRINT 'Creating [Instructor] table.';
        PRINT '========================================';

        CREATE TABLE dbo.Instructor (
                Instructor_ID INT PRIMARY KEY,
                First_Name VARCHAR(100),
                Last_Name VARCHAR(100),
                Birth_Date DATE,
                Gender VARCHAR(10),
                City VARCHAR(50),
                Email VARCHAR(100),
                Password VARCHAR(10)
        );

        PRINT '========================================';
        PRINT 'Table [Instructor] created.';
        PRINT '========================================';
END

--Course
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Course' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
        PRINT '========================================';
        PRINT 'Creating [Course] table.';
        PRINT '========================================';

        CREATE TABLE dbo.Course (
                Course_ID INT PRIMARY KEY,
                Course_Name VARCHAR(100),
                Instructor_ID INT,

                FOREIGN KEY (Instructor_ID) REFERENCES Instructor(Instructor_ID)
        );

        PRINT '========================================';
        PRINT 'Table [Course] created.';
        PRINT '========================================';
END

--Student Course
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Student Course' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
        PRINT '========================================';
        PRINT 'Creating [Student Course] table.';
        PRINT '========================================';

        CREATE TABLE dbo.Student_Course (
                Student_ID INT,
                Course_ID INT,

                PRIMARY KEY(Student_ID, Course_ID),
                FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID),
                FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID)
        );

        PRINT '========================================';
        PRINT 'Table [Student_Course] created.';
        PRINT '========================================';
END

--Topic
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Topic' AND schema_id = SCHEMA_ID('dbo'))
```

```
BEGIN
            PRINT '========================================';
            PRINT 'Creating [Topic] table.';
            PRINT '========================================';

            CREATE TABLE dbo.Topic (
                        Topic_ID INT PRIMARY KEY,
                        Topic_Name VARCHAR(100),
                        Course_ID INT,

                        FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID)
            );

            PRINT '========================================';
            PRINT 'Table [Topic] created.';
            PRINT '========================================';
END

--Exam
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Exam' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
            PRINT '========================================';
            PRINT 'Creating [Exam] table.';
            PRINT '========================================';

            CREATE TABLE dbo.Exam (
                        Exam_ID INT PRIMARY KEY,
                        Exam_Name VARCHAR(100),
                        Exam_Duration INT,
                        Exam_Marks INT,
                        Course_ID INT,

                        FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID)
            );

            PRINT '========================================';
            PRINT 'Table [Exam] created.';
            PRINT '========================================';
END

--Question
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Question' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
            PRINT '========================================';
            PRINT 'Creating [Question] table.';
            PRINT '========================================';

            CREATE TABLE dbo.Question (
                        Question_ID INT PRIMARY KEY,
                        Question_Text VARCHAR(MAX),
                        Question_Type VARCHAR(3),
                        Question_Marks INT,
                        Exam_ID INT,

                        FOREIGN KEY (Exam_ID) REFERENCES Exam(Exam_ID)
            );

            PRINT '========================================';
            PRINT 'Table [Question] created.';
            PRINT '========================================';
END

--Choice
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Choice' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
            PRINT '========================================';
            PRINT 'Creating [Choice] table.';
            PRINT '========================================';

            CREATE TABLE dbo.Choice (
                        Choice_ID INT PRIMARY KEY,
                        Choice_Text VARCHAR(MAX),
                        Is_Correct INT,
                        Question_ID INT,

                        FOREIGN KEY (Question_ID) REFERENCES Question(Question_ID)
            );

            PRINT '========================================';
            PRINT 'Table [Choice] created.';
            PRINT '========================================';
END

--Student_Answer
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Student_Answer' AND schema_id = SCHEMA_ID('dbo'))
BEGIN
            PRINT '========================================';
            PRINT 'Creating [Student_Answer] table.';
            PRINT '========================================';

            CREATE TABLE dbo.Student_Answer (
                        Student_Answer_ID INT PRIMARY KEY,
```

```
                    Student_Answer_Text VARCHAR(MAX),
                    Student_Answer_Mark INT,
                    Question_ID INT,
                    Student_ID INT,

                    FOREIGN KEY (Question_ID) REFERENCES Question(Question_ID),
                    FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
                );

                PRINT '=======================================';
                PRINT 'Table [Student_Answer] created.';
                PRINT '=======================================';

                PRINT '---------------------------------------';
                PRINT'All Tables Created Successfully.';
                PRINT '---------------------------------------';
        END
END;
```

2.3. Procedure: DeleteBranch

The Procedure deletes the information of a branch from the database

Input/Output

	Name	Data type	Description
→@	branch_id	int	

Script

```
CREATE PROCEDURE DeleteBranch
    @branch_id INT
AS
BEGIN
    DELETE FROM branch
    WHERE Branch_ID = @branch_id;
END
```

2.4. Procedure: DeleteBranchDepartment

The Procedure deletes a branch id and department id from the database

Input/Output

	Name	Data type	Description
→@	branch_id	int	
→@	Department_id	int	

Script

```
CREATE PROCEDURE DeleteBranchDepartment
    @branch_id INT,
           @Department_id INT
AS
BEGIN
    DELETE FROM Branch_Department
    WHERE Branch_ID = @branch_id and Department_ID=@Department_id;
END
```

2.5. Procedure: DeleteCertificate

The Procedure deletes a Certificate data from the database

Input/Output

	Name	Data type	Description
→@	Certificate_ID	int	

Script

```
CREATE PROCEDURE DeleteCertificate
    @Certificate_ID INT
AS
BEGIN
    DELETE FROM Certificate
    WHERE Certificate_ID = @Certificate_ID;
END
```

2.6. Procedure: DeleteChoice

The Procedure deletes a choice from the database

Input/Output

	Name	Data type	Description
→@	Choice_ID	int	

Script

```
CREATE PROCEDURE DeleteChoice
    @Choice_ID INT
AS
BEGIN
    DELETE FROM Choice
    WHERE Choice_ID = @Choice_ID;
END
```

2.7. Procedure: DeleteCourse

The Procedure deletes a Course from database

Input/Output

	Name	Data type	Description
→@	Course_ID	int	

Script

```
CREATE PROCEDURE DeleteCourse
    @Course_ID INT
AS
BEGIN
    DELETE FROM Course
    WHERE Course_ID = @Course_ID;
END
```

2.8. Procedure: DeleteDepartment

The Procedure deletes a Department from the database

Input/Output

	Name	Data type	Description
→@	Department_ID	int	

Script

```
CREATE PROCEDURE DeleteDepartment
    @Department_ID INT
AS
BEGIN
    DELETE FROM Department
    WHERE Department_ID = @Department_ID;
END
```

2.9. Procedure: DeleteExam

The Procedure deletes an Exam from the database

Input/Output

	Name	Data type	Description
→@	Exam_ID	int	

Script

```
CREATE PROCEDURE DeleteExam
    @Exam_ID INT
AS
BEGIN
    DELETE FROM Exam
    WHERE Exam_ID = @Exam_ID;
END
```

2.10. Procedure: DeleteFreelancing

The Procedure deletes a Freelancing Job from the database

Input/Output

	Name	Data type	Description
→@	Job_ID	int	

Script

```
CREATE PROCEDURE DeleteFreelancing
    @Job_ID INT
AS
BEGIN
    DELETE FROM Freelancing
    WHERE Job_ID = @Job_ID;
END
```

2.11. Procedure: DeleteHiring

The Procedure deletes a Company hiring students from the database.

Input/Output

	Name	Data type	Description
→@	Hiring_ID	int	

Script

```
CREATE PROCEDURE DeleteHiring
    @Hiring_ID INT
AS
BEGIN
    DELETE FROM Hiring
    WHERE Hiring_ID = @Hiring_ID;
END
```

2.12. Procedure: DeleteInstructor

The Procedure deletes an instructor from the database

Input/Output

	Name	Data type	Description
→@	Instructor_ID	int	

Script

```
CREATE PROCEDURE DeleteInstructor
    @Instructor_ID INT
AS
BEGIN
    DELETE FROM Instructor
    WHERE Instructor_ID = @Instructor_ID;
END
```

2.13. Procedure: DeleteQuestion

The Procedure deletes a Question from database

Input/Output

	Name	Data type	Description
→@	Question_ID	int	

Script

```
CREATE PROCEDURE DeleteQuestion
    @Question_ID INT
AS
BEGIN
    DELETE FROM Question
    WHERE Question_ID = @Question_ID;
END
------
```

2.14. Procedure: DeleteStudent

The Procedure deletes a Student from the database

Input/Output

	Name	Data type	Description
→@	Student_ID	int	

Script

```
CREATE PROCEDURE DeleteStudent
    @Student_ID INT
AS
BEGIN
    DELETE FROM Student
    WHERE Student_ID = @Student_ID;
END
```

2.15. Procedure: DeleteStudentAnswer

The Procedure deletes an Answer of a Student in an Exam

Input/Output

	Name	Data type	Description
→@	Student_Answer_ID	int	

Script

```
CREATE PROCEDURE DeleteStudentAnswer
    @Student_Answer_ID INT
AS
BEGIN
    DELETE FROM Student_Answer
    WHERE Student_Answer_ID = @Student_Answer_ID;
END
```

2.16. Procedure: DeleteStudentCertificate

The Procedure deletes a Certificate taken by student from the database

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	Certificate_ID	int	

Script

```
CREATE PROCEDURE DeleteStudentCertificate
    @Student_ID INT,
           @Certificate_ID INT
AS
BEGIN
    DELETE FROM Student_Certificate
    WHERE Student_ID = @Student_ID and Certificate_ID=@Certificate_id;
END
```

2.17. Procedure: DeleteStudentCourse

The Procedure deletes a Student ID and Course ID from the database

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	Course_ID	int	

Script

```
CREATE PROCEDURE DeleteStudentCourse
    @Student_ID INT,
           @Course_ID INT
AS
BEGIN
    DELETE FROM Student_Course
    WHERE Student_ID = @Student_ID and Course_ID=@Course_ID;
END
```

2.18. Procedure: DeleteStudentPhone

The Procedure deletes the phone numbers of a Student

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	Phone_Number	varchar(20)	

Script

```
CREATE PROCEDURE DeleteStudentPhone
    @Student_ID INT,
           @Phone_Number varchar(20)
AS
BEGIN
    DELETE FROM Student_Phone
    WHERE Student_ID = @Student_ID and Phone_Number=@Phone_Number;
END
```

2.19. Procedure: DeleteTopic

The Procedure deletes a Topic from the database

Input/Output

	Name	Data type	Description
→@	Topic_ID	int	

Script

```
CREATE PROCEDURE DeleteTopic
    @Topic_ID INT
AS
BEGIN
    DELETE FROM Topic
    WHERE Topic_ID = @Topic_ID;
END
```

2.20. Procedure: GenerateExam

The Procedure generates an Exam with a specified number of MCQ Quesions and TF Questions with Marks

Input/Output

	Name	Data type	Description
→@	course_name	varchar(100)	
→@	num_mcq	int	
→@	num_tf	int	
→@	total_marks	int	

Script

```
CREATE PROCEDURE [dbo].[GenerateExam]
    @course_name VARCHAR(100),
    @num_mcq INT,
    @num_tf INT,
    @total_marks INT = NULL
AS
BEGIN
          -- Temporary table to store selected question IDs and marks
    CREATE TABLE selected_questions (
        question_id INT PRIMARY KEY,
        mark FLOAT
    );

    -- Insert random MCQs
    INSERT INTO selected_questions (question_id, mark)
    SELECT TOP (@num_mcq) question_id, question_marks
    FROM question q , course c
    WHERE c.course_name = @course_name AND c.course_id = q.Crs_ID
                        AND q.question_type = 'MCQ'
    ORDER BY NEWID(); -- Random order

    -- Insert random TFs
    INSERT INTO selected_questions (question_id, mark)
    SELECT TOP (@num_tf) question_id, question_marks
    FROM question q , course c
    WHERE c.course_name = @course_name AND c.course_id = q.Crs_ID
                        AND question_type = 'TF'
    ORDER BY NEWID();

          DECLARE @actual_total_mark INT;
          set @actual_total_mark = (SELECT c.Exam_Marks from course c
                                                                  where c.course_name = @course_name);

          IF @total_marks IS NULL
          BEGIN
                    SET @total_marks = @actual_total_mark
          END

    -- Check total mark
    DECLARE @sum_marks FLOAT;
    SELECT @sum_marks = SUM(mark) FROM selected_questions;

          IF @total_marks != @sum_marks
          BEGIN
                    DECLARE @question_mark FLOAT;
                    SET @question_mark = CAST(@total_marks as FLOAT) / (@num_mcq + @num_tf);
                    UPDATE selected_questions
                    SET mark = @question_mark
                    SET @sum_marks = (SELECT SUM(mark) FROM selected_questions);
          END


          DECLARE @course_id INT;
          SELECT @course_id = course_id FROM  course
          WHERE course_name = @course_name;

          DECLARE @exam_id INT;
    IF ROUND(@sum_marks,0) = @total_marks
    BEGIN
                    INSERT INTO generated_exams (course_id, total_mark)
        VALUES (@course_id, @sum_marks);

        SET @exam_id = SCOPE_IDENTITY();

                    INSERT INTO exam_questions (exam_id, question_id)
        SELECT @exam_id, question_id FROM selected_questions;

                    SELECT @exam_id AS exam_id, @sum_marks as total_mark;

        SELECT q.Question_ID, q.Question_Text, q.Question_Type,
                                        COALESCE(ROUND(@question_mark,2), q.Question_Marks) as Question_Marks
        FROM question q
        JOIN selected_questions sq ON q.question_id = sq.question_id;
    END
    ELSE
    BEGIN
        RAISERROR('Unable to generate exam with exact total marks. Try changing question counts or total mark.', 16, 1);
    END

    DROP TABLE selected_questions;
END;
```

2.21. Procedure: GenerateQuiz

The Procedure generates an Exam with a specified number of MCQ Quesions and TF Questions with Marks

Input/Output

	Name	Data type	Description
→@	course_name	varchar(100)	
→@	num_mcq	int	
→@	num_tf	int	
→@	total_marks	int	

Script

```
CREATE PROCEDURE [dbo].[GenerateQuiz]
    @course_name VARCHAR(100),
    @num_mcq INT,
    @num_tf INT,
    @total_marks INT
AS
BEGIN

          -- Temporary table to store selected question IDs and marks
    CREATE TABLE selected_questions (
        question_id INT PRIMARY KEY,
        mark FLOAT
    );

    -- Insert random MCQs
    INSERT INTO selected_questions (question_id, mark)
    SELECT TOP (@num_mcq) question_id, question_marks
    FROM question q , course c
    WHERE c.course_name = @course_name AND c.course_id = q.Crs_ID
                      AND q.question_type = 'MCQ'
    ORDER BY NEWID(); -- Random order

    -- Insert random TFs
    INSERT INTO selected_questions (question_id, mark)
    SELECT TOP (@num_tf) question_id, question_marks
    FROM question q , course c
    WHERE c.course_name = @course_name AND c.course_id = q.Crs_ID
                      AND question_type = 'TF'
    ORDER BY NEWID();

    -- Check total mark
    DECLARE @sum_marks FLOAT;
    SELECT @sum_marks = SUM(mark) FROM selected_questions;


          DECLARE @course_id INT;
          SELECT @course_id = course_id FROM  course
          WHERE course_name = @course_name;

          DECLARE @exam_id INT;
    IF @sum_marks = @total_marks
    BEGIN
                      INSERT INTO generated_exams (course_id, total_mark, Is_Quiz)
        VALUES (@course_id, @sum_marks, 1);

        SET @exam_id = SCOPE_IDENTITY();

                      INSERT INTO exam_questions (exam_id, question_id)
        SELECT @exam_id, question_id FROM selected_questions;

                      SELECT @exam_id AS exam_id, @sum_marks as total_mark;

        SELECT q.Question_ID, q.Question_Text, q.Question_Type,q.Question_Marks
        FROM question q
        JOIN selected_questions sq ON q.question_id = sq.question_id;
    END
    ELSE
    BEGIN
        RAISERROR('Unable to generate quiz with exact total marks. Try changing question counts or total mark.', 16, 1);
    END

    DROP TABLE selected_questions;
END;
```

2.22. Procedure: GetCourseTopics

The Procedure displays Topics taught in Courses

Input/Output

	Name	Data type	Description
→@	CourseID	int	

Script

```
create procedure GetCourseTopics
    @CourseID int
as
begin
    select
        t.Topic_ID,
        t.Topic_Name
    from
        Topic t
    where
        t.Course_ID = @CourseID;
end;
```

2.23. Procedure: GetExamAnswers

The Procedure displays Answers of Exams

Input/Output

	Name	Data type	Description
→@	exam_id	int	

Script

```
CREATE PROCEDURE GetExamAnswers
    @exam_id INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        --q.question_id,
        q.question_text,
        --c.choice_id,
        c.choice_text as answer
        --c.is_correct
    FROM exam_questions eq
    JOIN question q ON eq.question_id = q.question_id
    JOIN choice c ON q.question_id = c.question_id
    WHERE eq.exam_id = @exam_id AND c.Is_Correct=1
    ORDER BY q.question_id, c.choice_id;
END;
```

2.24. Procedure: GetExamQuestionsAndChoices

The Procedure displays the questions and choices of an Exam

Input/Output

	Name	Data type	Description
→@	ExamID	int	

Script

```
CREATE procedure [dbo].[GetExamQuestionsAndChoices]
    @ExamID int
as
begin
    select
        q.Question_ID,
        q.Question_Text,
        c.Choice_ID,
        c.Choice_Text,
        c.Is_Correct
    from
        Question q
    inner join
        Choice c on q.Question_ID = c.Question_ID
    where
        q.Crs_ID = @ExamID
    order by
        q.Question_ID, c.Choice_ID;
end;
```

2.25. Procedure: GetInstructorCoursesWithStudentCount

The Procedure gets the number of Students and the courses taught by every instructor

Input/Output

	Name	Data type	Description
→@	InstructorID	int	

Script

```
create procedure GetInstructorCoursesWithStudentCount
    @InstructorID INT
as
begin
    select
        c.Course_Name,
        count(sc.Student_ID) as NumberOfStudents
    from
        Course c
        inner join Student_Course sc on c.Course_ID = sc.Course_ID
    where
        c.Instructor_ID = @InstructorID
    group by
        c.Course_Name;
end;
```

2.26. Procedure: GetStudentAnswersForExam

The Procedure gets the answers of a student answered in an exam

Input/Output

	Name	Data type	Description
→@	ExamID	int	
→@	StudentID	int	

Script

```
CREATE procedure [dbo].[GetStudentAnswersForExam]
    @ExamID int,
    @StudentID int
as
begin
    select
        q.Question_ID,
        q.Question_Text,
        sa.Student_Answer_Text
    from
        Question q
    left join
        Student_Answer sa ON q.Question_ID = sa.Question_ID AND sa.Student_ID = @StudentID
    where
        q.Crs_ID = @ExamID
    order by
        q.Question_ID;
end;
```

2.27. Procedure: GetStudentExamCorrection

The Procedure displays the Exam with the correct answers

Input/Output

	Name	Data type	Description
→@	student_id	int	
→@	exam_id	int	

Script

```
CREATE PROCEDURE GetStudentExamCorrection
    @student_id INT,
    @exam_id INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        q.question_id,
        q.question_text,
        sa.student_answer_text,
        corr.choice_text AS correct_choice,
        CASE
            WHEN sa.student_answer_text = corr.choice_text THEN q.question_marks
            ELSE 0
        END AS corrected_mark
    FROM exam_questions eq
    JOIN question q ON eq.question_id = q.question_id
    LEFT JOIN student_answer sa
        ON sa.question_id = q.question_id
    OUTER APPLY (
        SELECT choice_text
        FROM choice
        WHERE question_id = q.question_id AND is_correct = 1
    ) AS corr
    WHERE eq.exam_id = @exam_id AND sa.student_id = @student_id;

    -- Summary Report
    DECLARE @total_marks_exam INT, @total_obtained FLOAT, @course_name VARCHAR(100), @student_name VARCHAR(100);

    SELECT @total_marks_exam = e.exam_marks,
           @course_name = c.course_name
    FROM exam e
    JOIN course c ON e.course_id = c.course_id
    WHERE e.exam_id = @exam_id;

    SELECT @student_name = Concat(first_name,' ',Last_Name)
    FROM student
    WHERE student_id = @student_id;

    SELECT @total_obtained = SUM(
        CASE
            WHEN sa.student_answer_text = corr.choice_text THEN q.question_marks
            ELSE 0
        END
    )
    FROM exam_questions eq
    JOIN question q ON eq.question_id = q.question_id
    LEFT JOIN student_answer sa
        ON sa.question_id = q.question_id AND sa.student_id = @student_id
    OUTER APPLY (
        SELECT TOP 1 choice_text
        FROM choice
        WHERE question_id = q.question_id AND is_correct = 1
    ) AS corr
    WHERE eq.exam_id = @exam_id;

    -- Final Summary Output
    SELECT CONCAT(@student_name, ' got ', @total_obtained, ' out of ', @total_marks_exam, ' in ', @course_name) AS Result;
END;
```

2.28. Procedure: GetStudentGrade

The Procedure displays the Student Grades of Courses.

Input/Output

	Name	Data type	Description
→@	StudentID	int	

Script

```
create Procedure GetStudentGrade (@StudentID int)
as
begin

           select CONCAT(First_Name, ' ', Last_Name) as Student_Name,
                        Course_Name,
                        Grade
           from Student S inner join Student_Course SC
           on S.Student_ID = SC.Student_ID
           inner join Course C
           on C.Course_ID = SC.Course_ID
           where S.Student_ID = @StudentID

end
```

2.29. Procedure: InsertBranch

The Procedure adds a new branch with its information.

Input/Output

	Name	Data type	Description
→@	Branch_ID	int	
→@	Branch_Name	varchar(50)	

Script

```
CREATE PROCEDURE InsertBranch
    @Branch_ID int,
    @Branch_Name VARCHAR(50)
AS
BEGIN
    INSERT INTO Branch (Branch_ID,Branch_Name)
    VALUES (@Branch_ID, @Branch_Name);
END
```

2.30. Procedure: InsertBranchDepartment

The Procedure adds a new branch id and a new department id.

Input/Output

	Name	Data type	Description
→@	Branch_ID	int	
→@	Dept_ID	int	

Script

```
CREATE PROCEDURE InsertBranchDepartment
    @Branch_ID int,
    @Dept_ID int
AS
BEGIN
    INSERT INTO Branch_Department (Branch_ID,Department_ID)
    VALUES (@Branch_ID, @Dept_ID);
END
```

2.31. Procedure: InsertCertificate

The Procedure adds a new Certificate with its information

Input/Output

	Name	Data type	Description
→@	Certificate_ID	int	
→@	Certificate_Name	varchar(100)	
→@	Certificate_Hour	int	
→@	Certificate_Website	varchar(100)	
→@	Certificate_Date	date	

Script

```
CREATE PROCEDURE InsertCertificate
    @Certificate_ID int,
    @Certificate_Name varchar(100),
           @Certificate_Hour int,
           @Certificate_Website varchar(100),
           @Certificate_Date date
AS
BEGIN
    INSERT INTO Certificate(Certificate_ID,Certificate_Name,Certificate_Hour,Certificate_Website,Certificate_Date)
    VALUES (@Certificate_ID, @Certificate_Name,@Certificate_Hour,@Certificate_Website,@Certificate_Date);
END
```

2.32. Procedure: InsertChoice

The Procedure inserts a Choice of an MCQ Question.

Input/Output

	Name	Data type	Description
→@	Choice_ID	int	
→@	ChoiceText	varchar(MAX)	
→@	Is_Correct	int	
→@	Question_ID	int	

Script

```
CREATE PROCEDURE InsertChoice
    @Choice_ID int,
    @ChoiceText varchar(max),
           @Is_Correct int,
           @Question_ID int
AS
BEGIN
    INSERT INTO Choice(Choice_ID,Choice_Text,Is_Correct,Question_ID)
    VALUES (@Choice_ID, @ChoiceText,@Is_Correct,@Question_ID);
END
```

2.33. Procedure: InsertCourse

The Procedure inserts a Course with the information to The Database

Input/Output

	Name	Data type	Description
→@	Course_ID	int	
→@	Course_Name	varchar(100)	
→@	Instructor_ID	int	
→@	Exam_Duration	int	
→@	Exam_Marks	int	

Script

```
CREATE PROCEDURE [dbo].[InsertCourse]
    @Course_ID int,
    @Course_Name varchar(100),
           @Instructor_ID int,
           @Exam_Duration int,
           @Exam_Marks int
AS
BEGIN
    INSERT INTO Course(Course_ID,Course_Name,Instructor_ID,Exam_Duration,Exam_Marks)
    VALUES (@Course_ID,@Course_Name,@Instructor_ID,@Exam_Duration,@Exam_Marks);
END;
```

2.34. Procedure: InsertDepartment

The Procedure inserts a new Department with its information to the database

Input/Output

	Name	Data type	Description
→@	Department_ID	int	
→@	Department_Name	varchar(100)	
→@	Department_Description	varchar(MAX)	

Script

```
CREATE PROCEDURE InsertDepartment
    @Department_ID int,
    @Department_Name varchar(100),
           @Department_Description varchar(max)
AS
BEGIN
    INSERT INTO Department(Department_ID,Department_Name,Department_Description)
    VALUES (@Department_ID,@Department_Name,@Department_Description);
END
```

2.35. Procedure: InsertExam

The Procedure inserts a new Exam with its information to the database

Input/Output

	Name	Data type	Description
→@	Exam_ID	int	
→@	Exam_Name	varchar(100)	
→@	Exam_Duration	int	
→@	Exam_Marks	int	
→@	Course_ID	int	

Script

```
CREATE PROCEDURE InsertExam
    @Exam_ID int,
    @Exam_Name varchar(100),
           @Exam_Duration int,
           @Exam_Marks int,
           @Course_ID int
AS
BEGIN
    INSERT INTO Exam(Exam_ID,Exam_Name,Exam_Duration,Exam_Marks,Course_ID)
    VALUES (@Exam_ID,@Exam_Name,@Exam_Duration,@Exam_Marks,@Course_ID);
END
```

2.36. Procedure: InsertHiring

The Procedure adds a Company that hires students with its information

Input/Output

	Name	Data type	Description
→@	Hiring_ID	int	
→@	Hiring_Position	varchar(MAX)	
→@	Hiring_Date	date	
→@	Hiring_Company	varchar(MAX)	
→@	Hiring_Location	varchar(100)	
→@	Position_Type	varchar(100)	
→@	Student_ID	int	

Script

```
CREATE PROCEDURE InsertHiring
    @Hiring_ID int,
    @Hiring_Position varchar(max),
           @Hiring_Date date,
           @Hiring_Company varchar(max),
           @Hiring_Location varchar(100),
           @Position_Type varchar(100),
           @Student_ID int
AS
BEGIN
    INSERT INTO Hiring(Hiring_ID,Hiring_Position,Hiring_Date,Hiring_Company,Hiring_Location,Position_Type,Student_ID)
    VALUES (@Hiring_ID,@Hiring_Position,@Hiring_Date,@Hiring_Company,@Hiring_Location,@Position_Type,@Student_ID);
END
```

2.37. Procedure: InsertInstructor

The Procedure adds an Instructor that teaches students with his/her information

Input/Output

	Name	Data type	Description
→@	Instructor_ID	int	
→@	First_Name	varchar(MAX)	
→@	Last_Name	varchar(MAX)	
→@	Birth_Date	date	
→@	Gender	varchar(10)	
→@	City	varchar(50)	
→@	Email	varchar(100)	
→@	Password	varchar(MAX)	

Script

```
CREATE PROCEDURE InsertInstructor
    @Instructor_ID int,
    @First_Name varchar(max),
           @Last_Name varchar(max),
           @Birth_Date date,
           @Gender varchar(10),
           @City varchar(50),
           @Email varchar(100),
           @Password varchar(max)
AS
BEGIN
    INSERT INTO Instructor(Instructor_ID,First_Name,Last_Name,Birth_Date,Gender,City,Email,Password)
    VALUES (@Instructor_ID,@First_Name,@Last_Name,@Birth_Date,@Gender,@City,@Email,@Password);
END
```

2.38. Procedure: InsertJob

The Procedure adds a Freelance job that is done by students with its information

Input/Output

	Name	Data type	Description
→@	Job_ID	int	
→@	Job_Name	varchar(MAX)	
→@	Job_Website	varchar(100)	
→@	Job_Start_DATE	date	
→@	Job_Tools	varchar(100)	
→@	Feedback_Rating	int	
→@	Student_ID	int	

Script

```
CREATE PROCEDURE InsertJob
    @Job_ID int,
    @Job_Name varchar(max),
           @Job_Website varchar(100),
           @Job_Start_DATE date,
           @Job_Tools varchar(100),
           @Feedback_Rating int,
           @Student_ID int
AS
BEGIN
    INSERT INTO Freelancing(Job_ID,Job_Name,Job_Website,Job_Start_DATE,Job_Tools,Feedback_Rating,Student_ID)
    VALUES (@Job_ID,@Job_Name,@Job_Website,@Job_Start_DATE,@Job_Tools,@Feedback_Rating,@Student_ID);
END
```

2.39. Procedure: InsertQuestion

The Procedure adds a Question that can be used in exams.

Input/Output

	Name	Data type	Description
→@	Question_ID	int	
→@	Question_Text	varchar(MAX)	
→@	Question_Type	varchar(50)	
→@	Question_Mark	bigint	
→@	Crs_ID	int	

Script

```
CREATE PROCEDURE [dbo].[InsertQuestion]
     @Question_ID int,
    @Question_Text varchar(max),
           @Question_Type varchar(50),
           @Question_Mark bigint,
           @Crs_ID int
AS
BEGIN
    INSERT INTO Question(Question_ID,Question_Text,Question_Type,Question_Marks,Crs_ID)
    VALUES (@Question_ID,@Question_Text,@Question_Type,@Question_Mark,@Crs_ID);
END;
```

2.40. Procedure: InsertStudent

The Procedure adds a new student with his/her information

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	First_Name	varchar(100)	
→@	Last_Name	varchar(100)	
→@	Birth_Date	date	
→@	Gender	varchar(10)	
→@	City	varchar(50)	
→@	Email	varchar(100)	
→@	Password	varchar(MAX)	
→@	Department_ID	int	

Script

```
CREATE PROCEDURE InsertStudent
    @Student_ID int,
    @First_Name varchar(100),
           @Last_Name varchar(100),
           @Birth_Date date,
           @Gender varchar(10),
           @City varchar(50),
           @Email varchar(100),
           @Password varchar(max),
           @Department_ID int
AS
BEGIN
    INSERT INTO Student(Student_ID,First_Name,Last_Name,Birth_Date,Gender,City,Email,Password,Department_ID)
    VALUES (@Student_ID,@First_Name,@Last_Name,@Birth_Date,@Gender,@City,@Email,@Password,@Department_ID);
END
```

2.41. Procedure: InsertStudentAnswer

The Procedure adds an answer of a student that is used in exams

Input/Output

	Name	Data type	Description
→@	Student_Answer_ID	int	
→@	Student_Answer_Text	varchar(MAX)	
→@	Student_Answer_Mark	int	
→@	Question_ID	int	
→@	Student_ID	int	

Script

```
Create PROCEDURE InsertStudentAnswer
    @Student_Answer_ID int,
    @Student_Answer_Text varchar(max),
           @Student_Answer_Mark int,
           @Question_ID int,
           @Student_ID int
AS
BEGIN
    INSERT INTO Student_Answer(Student_Answer_ID,Student_Answer_Text,Student_Answer_Mark,Question_ID,Student_ID)
    VALUES (@Student_Answer_ID,@Student_Answer_Text,@Student_Answer_Mark,@Question_ID,@Student_ID);
END
```

2.42. Procedure: InsertStudentCertificate

The Procedure adds a new certificate obtained by a student.

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	Certificate_ID	int	

Script

```
CREATE PROCEDURE InsertStudentCertificate
    @Student_ID int,
    @Certificate_ID int
AS
BEGIN
    INSERT INTO Student_Certificate (Student_ID,Certificate_ID)
    VALUES (@Student_ID, @Certificate_ID);
END
```

2.43. Procedure: InsertStudentCourse

The Procedure adds a Student ID, Course ID and grade of a student

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	Course_ID	int	

Script

```
CREATE PROCEDURE InsertStudentCourse
    @Student_ID int,
    @Course_ID int
AS
BEGIN
    INSERT INTO Student_Course (Student_ID,Course_ID)
    VALUES (@Student_ID, @Course_ID);
END
```

2.44. Procedure: InsertStudentPhone

The Procedure adds a student id and his phone numbers

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	Phone_Number	varchar(20)	

Script

```
CREATE PROCEDURE InsertStudentPhone
    @Student_ID int,
    @Phone_Number varchar(20)
AS
BEGIN
    INSERT INTO Student_Phone (Student_ID,Phone_Number)
    VALUES (@Student_ID, @Phone_Number);
END
```

2.45. Procedure: InsertTopic

The Procedure adds a new topic that is used in courses

Input/Output

	Name	Data type	Description
→@	Topic_ID	int	
→@	Topic_Name	varchar(100)	
→@	Course_ID	int	

Script

```
CREATE PROCEDURE InsertTopic
    @Topic_ID int,
    @Topic_Name VARCHAR(100),
           @Course_ID int
AS
BEGIN
    INSERT INTO Topic (Topic_ID,Topic_Name,Course_ID)
    VALUES (@Topic_ID,@Topic_Name,@Course_ID);
END
```

2.46. Procedure: SelectAllBranches

The Procedure displays all the branches in The System

Script

```
create procedure SelectAllBranches
as
begin
select * from Branch
end
```

2.47. Procedure: SelectAllBranchesDepartments

The Procedure displays all the branches id and departments id related to it

Script

```
create procedure SelectAllBranchesDepartments
as
begin
select * from Branch_Department
end
```

2.48. Procedure: SelectAllCertificates

The Procedure displays all certificates

Script

```
create procedure SelectAllCertificates
as
begin
select * from Certificate
end
```

2.49. Procedure: SelectAllChoices

The Procedure displays all choices in the database

Script

```
create procedure SelectAllChoices
as
begin
select * from Choice
end
```

2.50. Procedure: SelectAllCourses

The Procedure displays all courses in the database

Script

```
create procedure SelectAllCourses
as
begin
select * from Course
end
```

2.51. Procedure: SelectAllDepartment

The Procedure displays all departments in the database

Script

```
create procedure SelectAllDepartment
as
begin
select * from Department
end
```

2.52. Procedure: SelectAllExam

The Procedure displays all Exams in the database

Script

```
create procedure SelectAllExam
as
begin
select * from Exam
end
```

2.53. Procedure: SelectAllFreelancing

The Procedure displays all Freelancing Jobs in the database

Script

```
--------> SelectAllFreelancing
create procedure SelectAllFreelancing
as
begin
select * from Freelancing
end
```

2.54. Procedure: SelectAllHiring

The Procedure displays all Companies hiring in the database

Script

```
create procedure SelectAllHiring
as
begin
select * from Hiring
end
```

2.55. Procedure: SelectAllInstructor

The Procedure displays all Instructors in the database

Script

```
create procedure SelectAllInstructor
as
begin
select * from Instructor
end
```

2.56. Procedure: SelectAllQuestions

The Procedure displays all Questions in the database

Script

```
create procedure SelectAllQuestions
as
begin
select * from Question
end
```

2.57. Procedure: SelectAllStudent

The Procedure displays all Students in the database

Script

```
create procedure SelectAllStudent
as
begin
select * from Student
end
```

2.58. Procedure: SelectAllStudentAnswer

The Procedure displays all Answers of The Students in the database

Script

```
create procedure SelectAllStudentAnswer
as
begin
select * from Student_Answer
end
```

2.59. Procedure: SelectAllStudentCertificate

The Procedure displays all Students and Certificates related to them in the database

Script

```
create procedure SelectAllStudentCertificate
as
begin
select * from Student_Certificate
end
```

2.60. Procedure: SelectAllStudentCourse

The Procedure displays all Students and their courses in the database

Script

```
create procedure SelectAllStudentCourse
as
begin
select * from Student_Course
end
```

2.61. Procedure: SelectAllStudentPhone

The Procedure displays all students ids and their phones in the databases

Script

```
create procedure SelectAllStudentPhone
as
begin
select * from Student_Phone
end
```

2.62. Procedure: SelectAllTopics

The Procedure displays all topics in the databases

Script

```
create procedure SelectAllTopics
as
begin
select * from Topic
end
```

2.63. Procedure: SelectBranchbyID

The Procedure displays all Branch chosen by ID

Input/Output

	Name	Data type	Description
→@	Bid	int	

Script

```
Create procedure SelectBranchbyID
    @Bid int
as
begin
    select Branch_ID, Branch_Name
    from Branch
    where Branch_ID = @Bid;
end
```

2.64. Procedure: SelectCertificatebyID

The Procedure displays the certificate chosen by ID

Input/Output

	Name	Data type	Description
→@	Cid	int	

Script

```
create procedure SelectCertificatebyID
    @Cid int
as
begin
    select *
    from Certificate
    where Certificate_ID = @Cid;
end
```

2.65. Procedure: SelectChoicebyID

The Procedure displays all choice chosen by id

Input/Output

	Name	Data type	Description
→@	Chid	int	

Script

```
create procedure SelectChoicebyID
    @Chid int
as
begin
    select *
    from Choice
    where Choice_ID = @Chid;
end
```

2.66. Procedure: SelectCourseByID

The Procedure displays the course chosen by id

Input/Output

	Name	Data type	Description
→@	Courseid	int	

Script

```
create procedure SelectCourseByID
    @Courseid int
as
begin
    select *
    from Course
    where Course_ID = @Courseid;
end
```

2.67. Procedure: SelectDepartmentByID

The Procedure displays the department chosen by ID

Input/Output

	Name	Data type	Description
→@	Departmentid	int	

Script

```
create procedure SelectDepartmentByID
    @Departmentid int
as
begin
    select *
    from Department
    where Department_ID = @Departmentid;
end
```

2.68. Procedure: SelectExamByID

The Procedure displays the Exam chosen by ID

Input/Output

	Name	Data type	Description
→@	Exam_Id	int	

Script

```
create procedure SelectExamByID
    @Exam_Id int
as
begin
    select *
    from Exam
    where Exam_ID = @Exam_id;
end
```

2.69. Procedure: SelectHiringByID

The Procedure displays the company chosen by ID

Input/Output

	Name	Data type	Description
→@	Hiring_Id	int	

Script

```
create procedure SelectHiringByID
    @Hiring_Id int
as
begin
    select *
    from Hiring
    where Hiring_ID = @Hiring_ID;
end
```

2.70. Procedure: SelectInstructorByID

The Procedure displays the instructor chosen by ID

Input/Output

	Name	Data type	Description
→@	Instructor_ID	int	

Script

```
create procedure SelectInstructorByID
    @Instructor_ID int
as
begin
    select *
    from Instructor
    where Instructor_ID = @Instructor_ID;
end
```

2.71. Procedure: SelectJobByID

The Procedure displays the job chosen by ID

Input/Output

	Name	Data type	Description
→@	Job_Id	int	

Script

```
create procedure SelectJobByID
    @Job_Id int
as
begin
    select *
    from Freelancing
    where Job_ID = @Job_ID;
end
```

2.72. Procedure: SelectQuestionByID

The Procedure displays the question chosen by ID

Input/Output

	Name	Data type	Description
→@	Question_ID	int	

Script

```
-------> Select Question by ID
create procedure SelectQuestionByID
    @Question_ID int
as
begin
    select *
    from Question
    where Question_ID = @Question_ID;
end
```

2.73. Procedure: SelectStudentAnswerByID

The Procedure displays all student answer chosen by ID

Input/Output

	Name	Data type	Description
→@	StudentAnswer_ID	int	

Script

```
create procedure SelectStudentAnswerByID
    @StudentAnswer_ID int
as
begin
    select *
    from Student_Answer
    where Student_Answer_ID = @StudentAnswer_ID;
end
```

2.74. Procedure: SelectStudentByID

The Procedure displays the student chosen by ID

Input/Output

	Name	Data type	Description
→@	Student_ID	int	

Script

```
create procedure SelectStudentByID
    @Student_ID int
as
begin
    select *
    from Student
    where Student_ID = @Student_ID;
end
```

2.75. Procedure: SelectTopicbyID

The Procedure displays the topic chosen by ID

Input/Output

	Name	Data type	Description
→@	Topic_ID	int	

Script

```
Create procedure SelectTopicbyID
    @Topic_ID int
as
begin
    select Topic_ID, Topic_Name,Course_ID
    from Topic
    where Topic_ID = @Topic_ID;
end
```

2.76. Procedure: StudentAccordingtoDepartment

The Procedure displays the student according to department ID

Input/Output

	Name	Data type	Description
→@	DeptNo	int	

Script

```
create procedure StudentAccordingtoDepartment @DeptNo int
as
begin
          select *
          from student
          where @DeptNo=Student.Department_ID
end
```

2.77. Procedure: UpdateBranch

The Procedure updates new information to Branch

Input/Output

	Name	Data type	Description
→@	branch_id	int	
→@	branch_name	varchar(100)	

Script

```
create procedure UpdateBranch
    @branch_id int,
    @branch_name varchar(100)
as
begin
    update branch
    set Branch_Name = @branch_name
    where Branch_ID = @branch_id;
end
```

2.78. Procedure: UpdateCertificate

The Procedure updates new information to Certificate

Input/Output

	Name	Data type	Description
→@	Certificate_ID	int	
→@	Certificate_Name	varchar(100)	
→@	Certificate_Hour	int	
→@	Certificate_Website	varchar(100)	
→@	Certificate_Date	date	

Script

```
create procedure UpdateCertificate
    @Certificate_ID int,
    @Certificate_Name varchar(100),
           @Certificate_Hour int,
           @Certificate_Website varchar(100),
           @Certificate_Date date
as
begin
    update Certificate
    set Certificate_Name = @Certificate_Name,
           Certificate_Hour=@Certificate_Hour,
           Certificate_Website=@Certificate_Website,
           Certificate_Date=@Certificate_Date
    where Certificate_ID = @Certificate_ID;
end
```

2.79. Procedure: UpdateChoice

The Procedure updates new information to Choice

Input/Output

	Name	Data type	Description
→@	Choice_ID	int	
→@	Choice_Text	varchar(MAX)	
→@	Is_Correct	int	
→@	Question_ID	int	

Script

```
create procedure UpdateChoice
    @Choice_ID int,
    @Choice_Text varchar(max),
           @Is_Correct int,
           @Question_ID int
as
begin
    update Choice
    set
           Choice_Text=@Choice_Text,
           Is_Correct=@Is_Correct,
           Question_ID=@Question_ID
    where Choice_ID = @Choice_ID;
end
```

2.80. Procedure: UpdateCourse

The Procedure updates new information to Course

Input/Output

	Name	Data type	Description
→@	Course_ID	int	
→@	Course_Name	varchar(100)	
→@	Instructor_ID	int	
→@	Exam_Duration	int	
→@	Exam_Marks	int	

Script

```
CREATE procedure [dbo].[UpdateCourse]
    @Course_ID int,
    @Course_Name varchar(100),
           @Instructor_ID int,
           @Exam_Duration int,
           @Exam_Marks int
as
begin
    update Course
    set
           Course_Name=@Course_Name,
           Instructor_ID=@Instructor_ID,
           Exam_Duration=@Exam_Duration,
           Exam_Marks=@Exam_Marks
    where Course_ID = @Course_ID;
end;
```

2.81. Procedure: UpdateDepartment

The Procedure updates new information to Department

Input/Output

	Name	Data type	Description
→@	Department_ID	int	
→@	Department_Name	varchar(100)	
→@	Department_Description	varchar(MAX)	

Script

```
create procedure UpdateDepartment
    @Department_ID int,
    @Department_Name varchar(100),
           @Department_Description varchar(max)
as
begin
    update Department
    set
           Department_Name=@Department_Name,
           Department_Description=@Department_Description
    where Department_ID = @Department_ID;
end
```

2.82. Procedure: UpdateExam

The Procedure updates new information to Exam

Input/Output

	Name	Data type	Description
→@	Exam_ID	int	
→@	Exam_Name	varchar(100)	
→@	Exam_Duration	int	
→@	Exam_Marks	int	
→@	Course_ID	int	

Script

```
create procedure UpdateExam
    @Exam_ID int,
    @Exam_Name varchar(100),
           @Exam_Duration int,
           @Exam_Marks int,
           @Course_ID int
as
begin
    update Exam
    set
           Exam_Name=@Exam_Name,
           Exam_Duration=@Exam_Duration,
           Exam_Marks=@Exam_Marks,
           Course_ID=@Course_ID
    where Exam_ID = @Exam_ID;
end
```

2.83. Procedure: UpdateFreelancing

The Procedure updates new information to Freelancing

Input/Output

	Name	Data type	Description
→@	Job_ID	int	
→@	Job_Name	varchar(MAX)	
→@	Job_Website	varchar(100)	
→@	Job_Start	date	
→@	Job_Tools	varchar(100)	
→@	Feedback_Rating	int	
→@	Student_ID	int	

Script

```
create procedure UpdateFreelancing
    @Job_ID int,
    @Job_Name varchar(max),
           @Job_Website varchar(100),
           @Job_Start date,
           @Job_Tools varchar(100),
           @Feedback_Rating int,
           @Student_ID int
as
begin
    update Freelancing
    set
           Job_Name=@Job_Name,
           Job_Website=@Job_Website,
           Job_Start_DATE=@Job_Start,
           Job_Tools=@Job_Tools,
           Feedback_Rating=@Feedback_Rating
    where Student_ID = @Student_ID;
end
```

2.84. Procedure: UpdateHiring

The Procedure updates new information to Hiring

Input/Output

	Name	Data type	Description
→@	Hiring_ID	int	
→@	Hiring_Position	varchar(MAX)	
→@	Hiring_Date	date	
→@	Hiring_Company	varchar(MAX)	
→@	Hiring_Location	varchar(100)	
→@	Position_Type	varchar(100)	
→@	Student_ID	int	

Script

```
create procedure UpdateHiring
    @Hiring_ID int,
    @Hiring_Position varchar(max),
           @Hiring_Date date,
           @Hiring_Company varchar(max),
           @Hiring_Location varchar(100),
           @Position_Type varchar(100),
           @Student_ID int
as
begin
    update Hiring
    set
           Hiring_Position=@Hiring_Position,
           Hiring_Date=@Hiring_Date,
           Hiring_Company=@Hiring_Company,
           Hiring_Location=@Hiring_Location,
           Position_Type=@Position_Type,
           Student_ID=@Student_ID
    where Hiring_ID = @Hiring_ID;
end
```

2.85. Procedure: UpdateInstructor

The Procedure updates new information to Instructor

Input/Output

	Name	Data type	Description
→@	Instructor_ID	int	
→@	First_Name	varchar(MAX)	
→@	Last_Name	varchar(MAX)	
→@	Birth_Date	date	
→@	Gender	varchar(10)	
→@	City	varchar(50)	
→@	Email	varchar(100)	
→@	Password	varchar(MAX)	

Script

```
create procedure UpdateInstructor
    @Instructor_ID int,
    @First_Name varchar(max),
           @Last_Name varchar(max),
           @Birth_Date date,
           @Gender varchar(10),
           @City varchar(50),
           @Email varchar(100),
           @Password varchar(max)
as
begin
    update Instructor
    set
           First_Name=@First_Name,
           Last_Name=@Last_Name,
           Birth_Date=@Birth_Date,
           Gender=@Gender,
           City=@City,
           Email=@Email,
           Password=@Password
    where Instructor_ID = @Instructor_ID;
end
```

2.86. Procedure: UpdateQuestion

The Procedure updates new information to Question

Input/Output

	Name	Data type	Description
→@	Question_ID	int	
→@	Question_Text	varchar(MAX)	
→@	Question_Type	varchar(50)	
→@	Question_Mark	bigint	
→@	Crs_ID	int	

Script

```
CREATE procedure [dbo].[UpdateQuestion]
    @Question_ID int,
    @Question_Text varchar(max),
           @Question_Type varchar(50),
           @Question_Mark bigint,
           @Crs_ID int
as
begin
    update Question
    set
           Question_Text=@Question_Text,
           Question_Type=@Question_Type,
           Question_Marks=@Question_Mark,
           Crs_ID=@Crs_ID

    where Question_ID = @Question_ID;
end;
```

2.87. Procedure: UpdateStudent

The Procedure updates new information to Student

Input/Output

	Name	Data type	Description
→@	Student_ID	int	
→@	First_Name	varchar(100)	
→@	Last_Name	varchar(100)	
→@	Birth_Date	date	
→@	Gender	varchar(10)	
→@	City	varchar(50)	
→@	Email	varchar(100)	
→@	Password	varchar(MAX)	
→@	Department_ID	int	

Script

```
create procedure UpdateStudent
    @Student_ID int,
    @First_Name varchar(100),
           @Last_Name varchar(100),
           @Birth_Date date,
           @Gender varchar(10),
           @City varchar(50),
           @Email varchar(100),
           @Password varchar(max),
           @Department_ID int
as
begin
    update Student
    set
           First_Name=@First_Name,
           Last_Name=@Last_Name,
           Birth_Date=@Birth_Date,
           Gender=@Gender,
           City=@City,
           Email=@Email,
           Password=@Password,
           Department_ID=@Department_ID
    where Student_ID = @Student_ID;
end
```

2.88. Procedure: UpdateStudentAnswer

The Procedure updates new information to Student Answer

Input/Output

	Name	Data type	Description
→@	Student_Answer_ID	int	
→@	Student_Answer_Text	varchar(MAX)	
→@	Student_Answer_Mark	int	
→@	Question_ID	int	
→@	Student_ID	int	

Script

```
create procedure UpdateStudentAnswer
    @Student_Answer_ID int,
    @Student_Answer_Text varchar(max),
           @Student_Answer_Mark int,
           @Question_ID int,
           @Student_ID int
as
begin
    update Student_Answer
    set
           Student_Answer_Text=@Student_Answer_Text,
           Student_Answer_Mark=@Student_Answer_Mark,
           Question_ID=@Question_ID,
           Student_ID=@Student_ID
    where Student_Answer_ID=@Student_Answer_ID;
end
```

2.89. Procedure: UpdateTopic

The Procedure updates new information to Topic

Input/Output

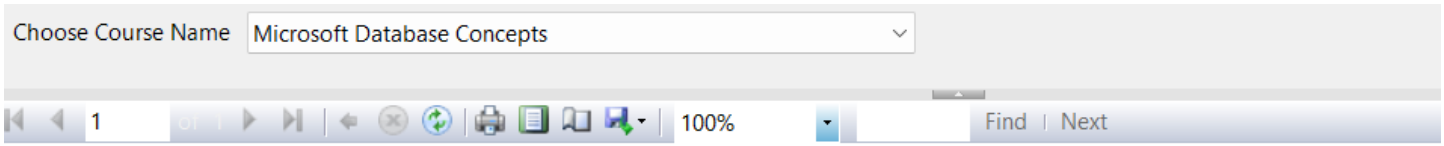
	Name	Data type	Description
→@	Topic_ID	int	
→@	Topic_Name	varchar(100)	
→@	Course_ID	int	

Script

```
Create procedure UpdateTopic
    @Topic_ID int,
    @Topic_Name varchar(100),
           @Course_ID int
as
begin
    update Topic
    set Topic_Name = @Topic_Name,
                     Course_ID = @Course_ID
    where Topic_ID = @Topic_ID;
end
```

3 SSRS Reports

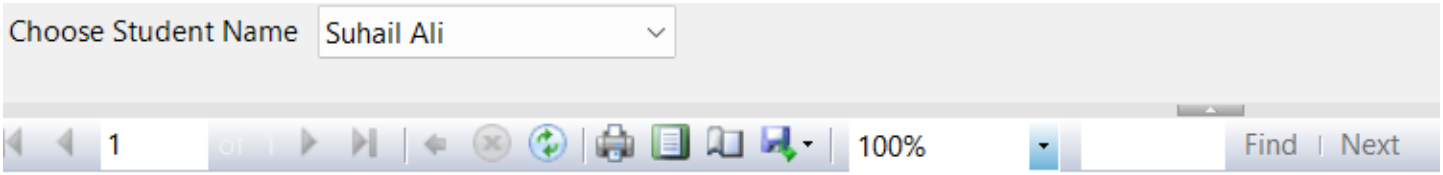
- 3.1 Get Course Topics



All Topics in Microsoft Database Concepts Course

Topic ID	Topic Name
1	Database Fundamentals
2	Advanced SQL Server
3	SQL Server
4	ERD

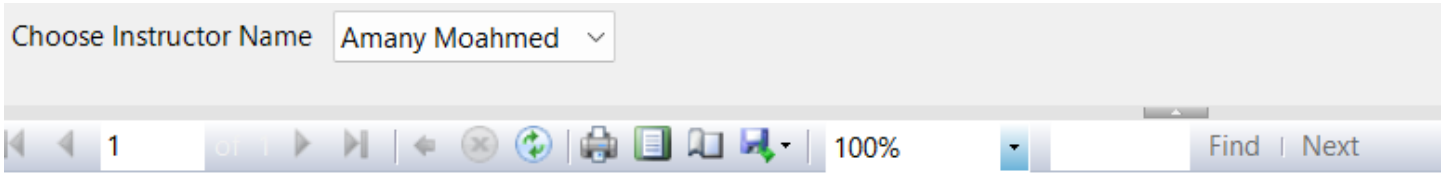
- 3.2 Get Student Grades



Suhail Ali Grades

Student Name	Course Name	Grade
Suhail Ali	Microsoft Database Concepts	31
Suhail Ali	Analytics Courses	45
Suhail Ali	Visualization Courses	28
Suhail Ali	Software Development Fundamentals	55
Suhail Ali	Workshop Sessions	35

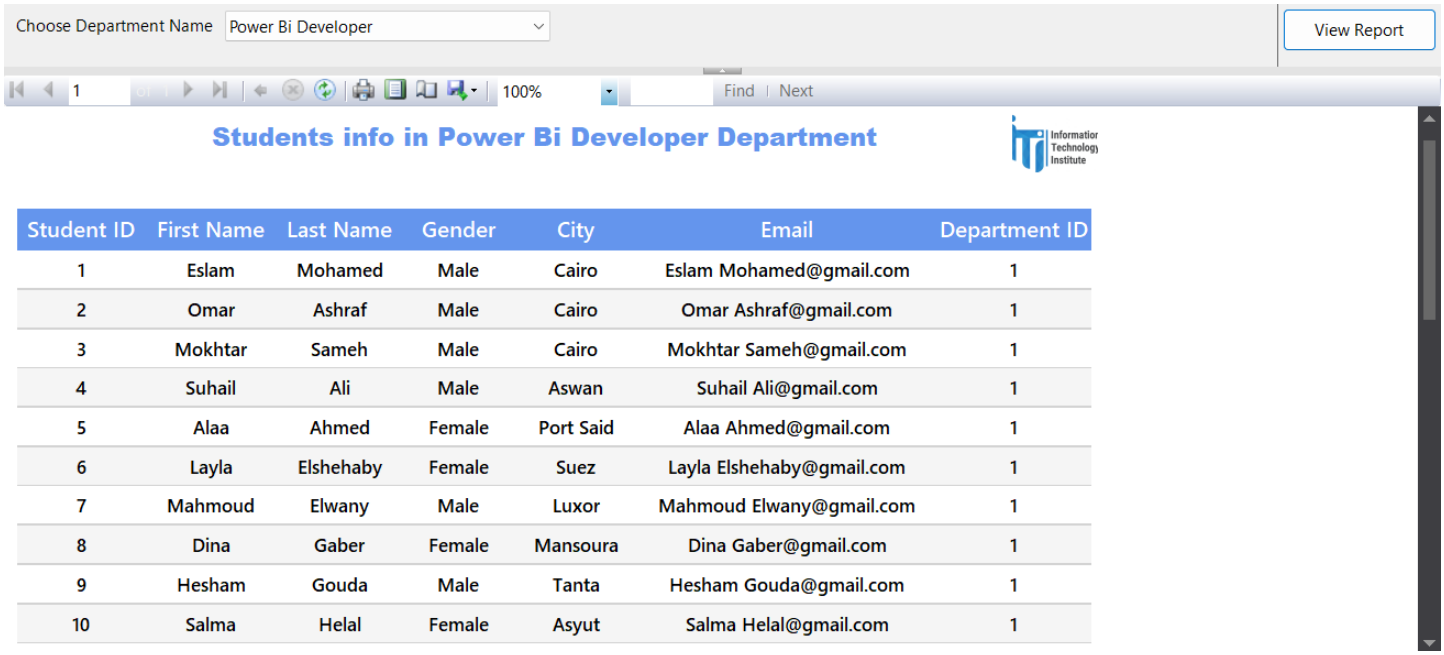
- 3.3 Course Name and Number of Students by Instructor



Courses taught by Amany Moahmed

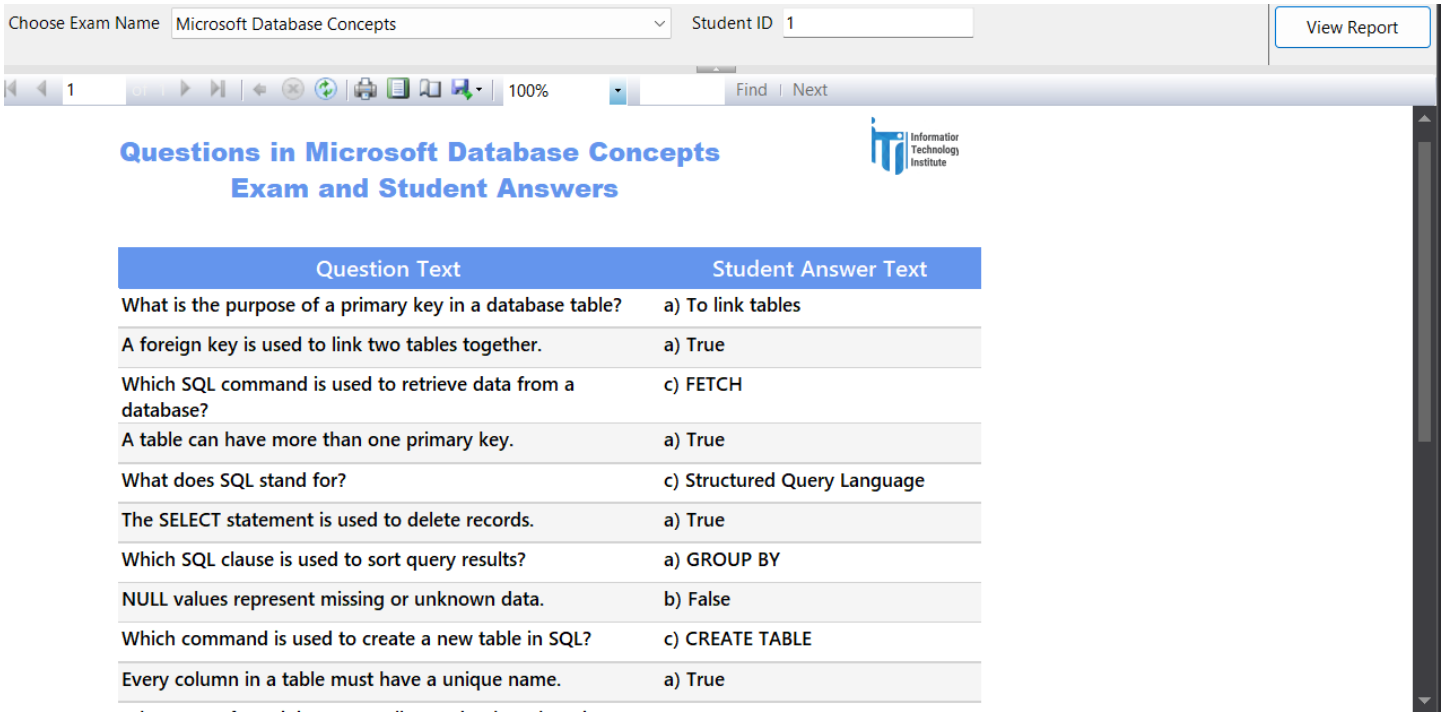
Course Name	Number Of Students
Supporting Courses	23
User Experience Courses	37
User Interface Courses	37

- 3.4 Students info according to Department



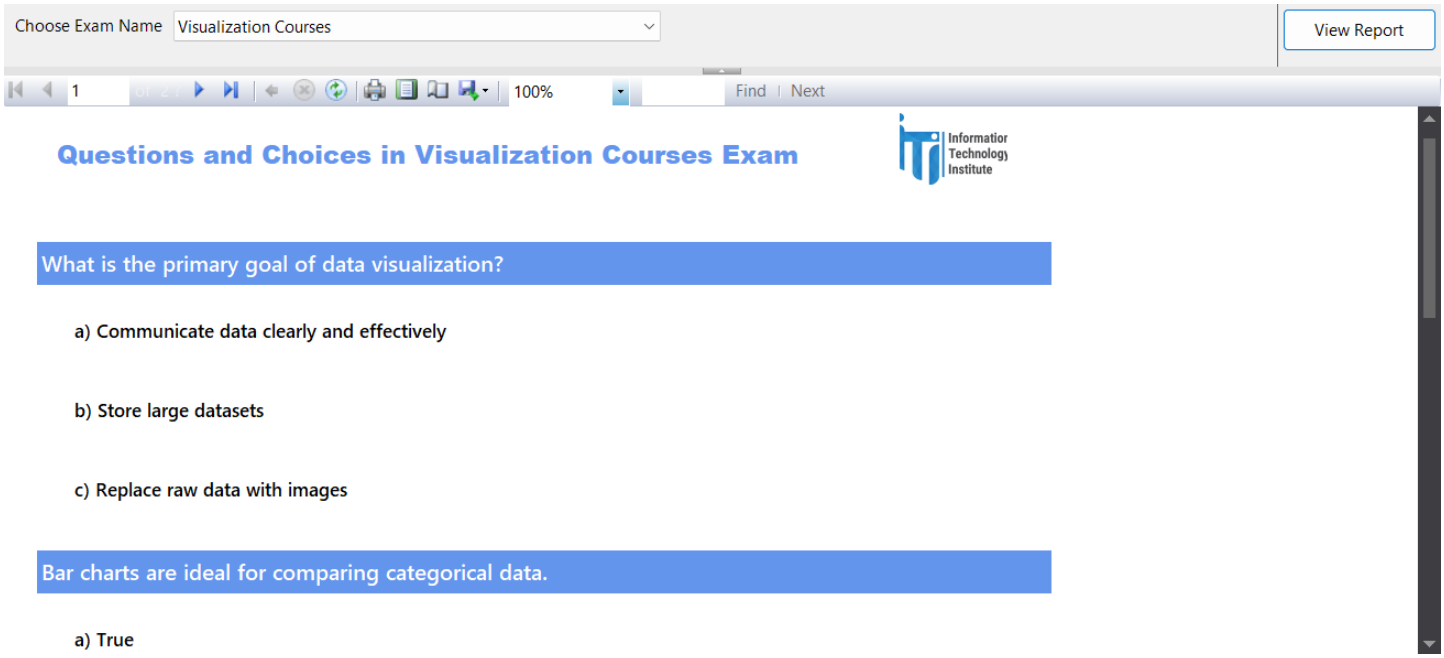
Student ID	First Name	Last Name	Gender	City	Email	Department ID
1	Eslam	Mohamed	Male	Cairo	Eslam Mohamed@gmail.com	1
2	Omar	Ashraf	Male	Cairo	Omar Ashraf@gmail.com	1
3	Mokhtar	Sameh	Male	Cairo	Mokhtar Sameh@gmail.com	1
4	Suhail	Ali	Male	Aswan	Suhail Ali@gmail.com	1
5	Alaa	Ahmed	Female	Port Said	Alaa Ahmed@gmail.com	1
6	Layla	Elshehaby	Female	Suez	Layla Elshehaby@gmail.com	1
7	Mahmoud	Elwany	Male	Luxor	Mahmoud Elwany@gmail.com	1
8	Dina	Gaber	Female	Mansoura	Dina Gaber@gmail.com	1
9	Hesham	Gouda	Male	Tanta	Hesham Gouda@gmail.com	1
10	Salma	Helal	Female	Asyut	Salma Helal@gmail.com	1

- 3.5 Exam Question and Student answers



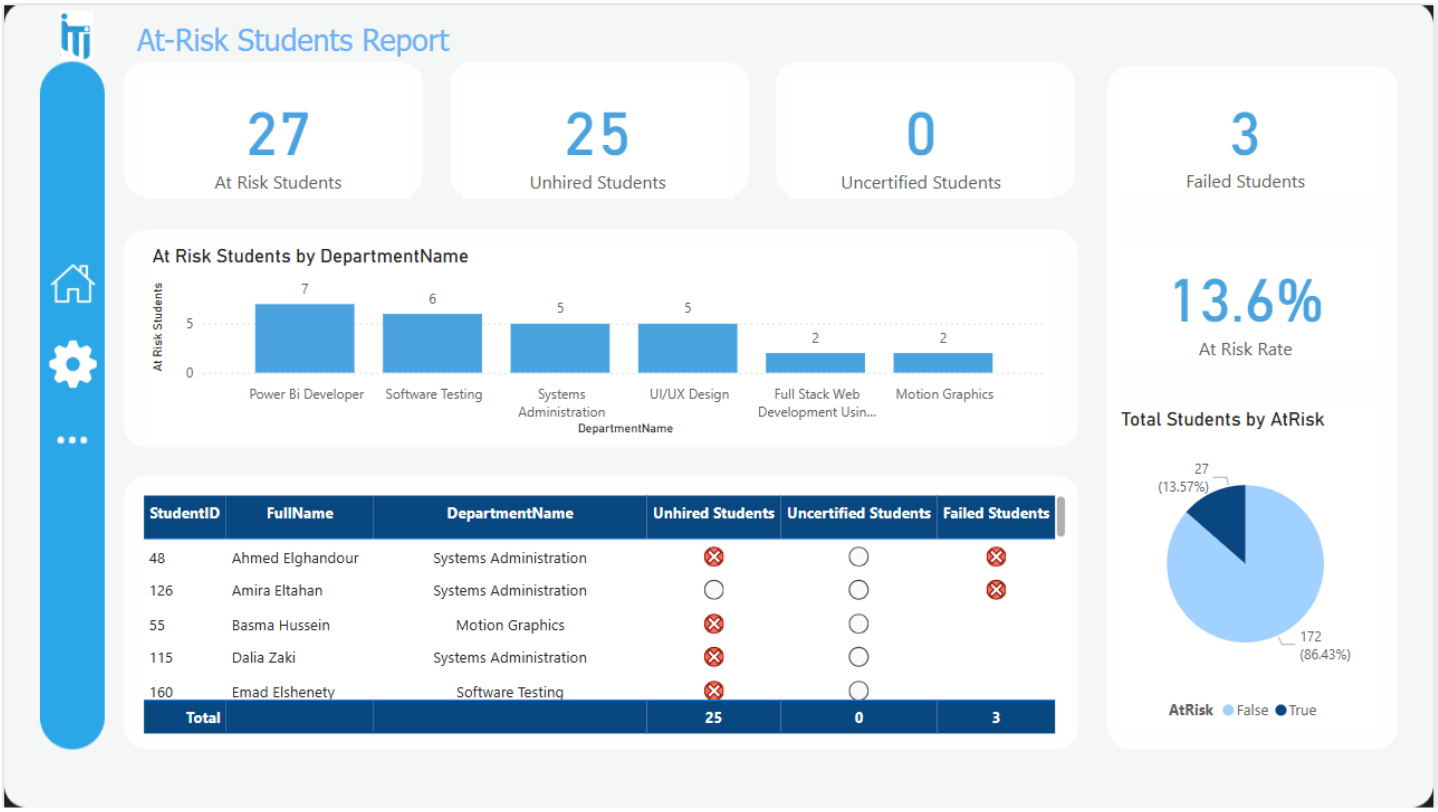
Question Text	Student Answer Text
What is the purpose of a primary key in a database table?	a) To link tables
A foreign key is used to link two tables together.	a) True
Which SQL command is used to retrieve data from a database?	c) FETCH
A table can have more than one primary key.	a) True
What does SQL stand for?	c) Structured Query Language
The SELECT statement is used to delete records.	a) True
Which SQL clause is used to sort query results?	a) GROUP BY
NULL values represent missing or unknown data.	b) False
Which command is used to create a new table in SQL?	c) CREATE TABLE
Every column in a table must have a unique name.	a) True

- 3.5 Exam Question and Choices

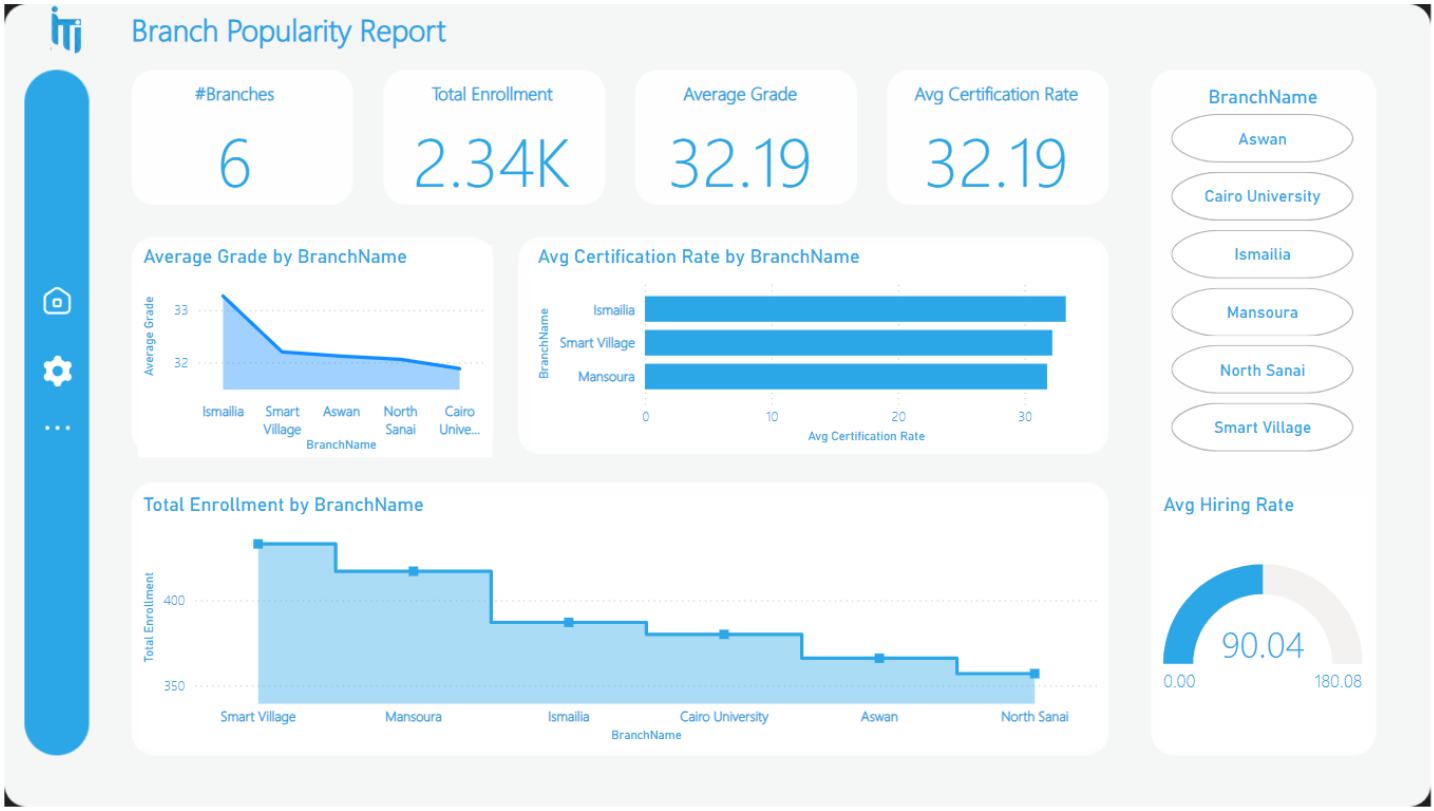


4 Power BI Dashboards

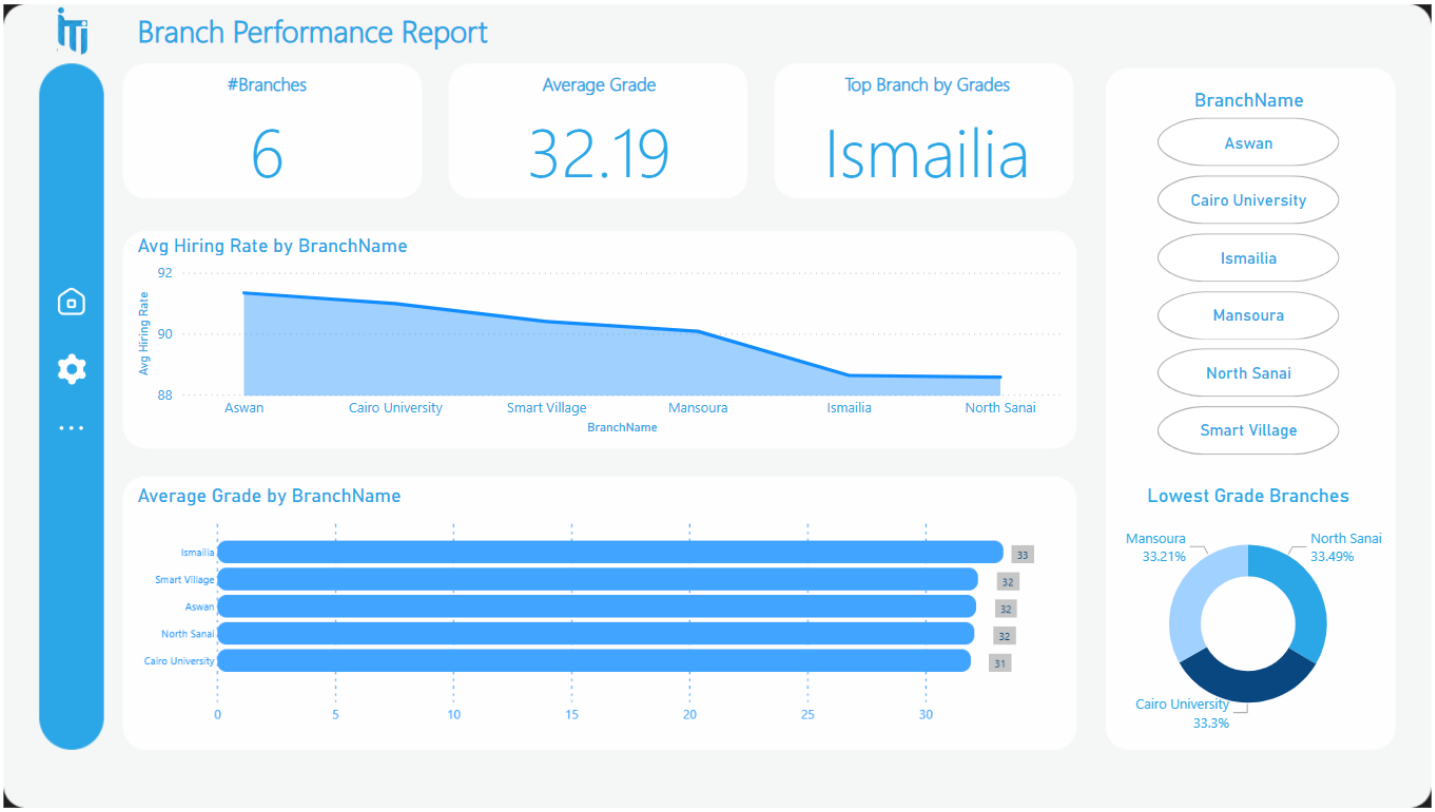
- 4.1 At-Risk Students Report



- 4.2 Branch Popularity Report



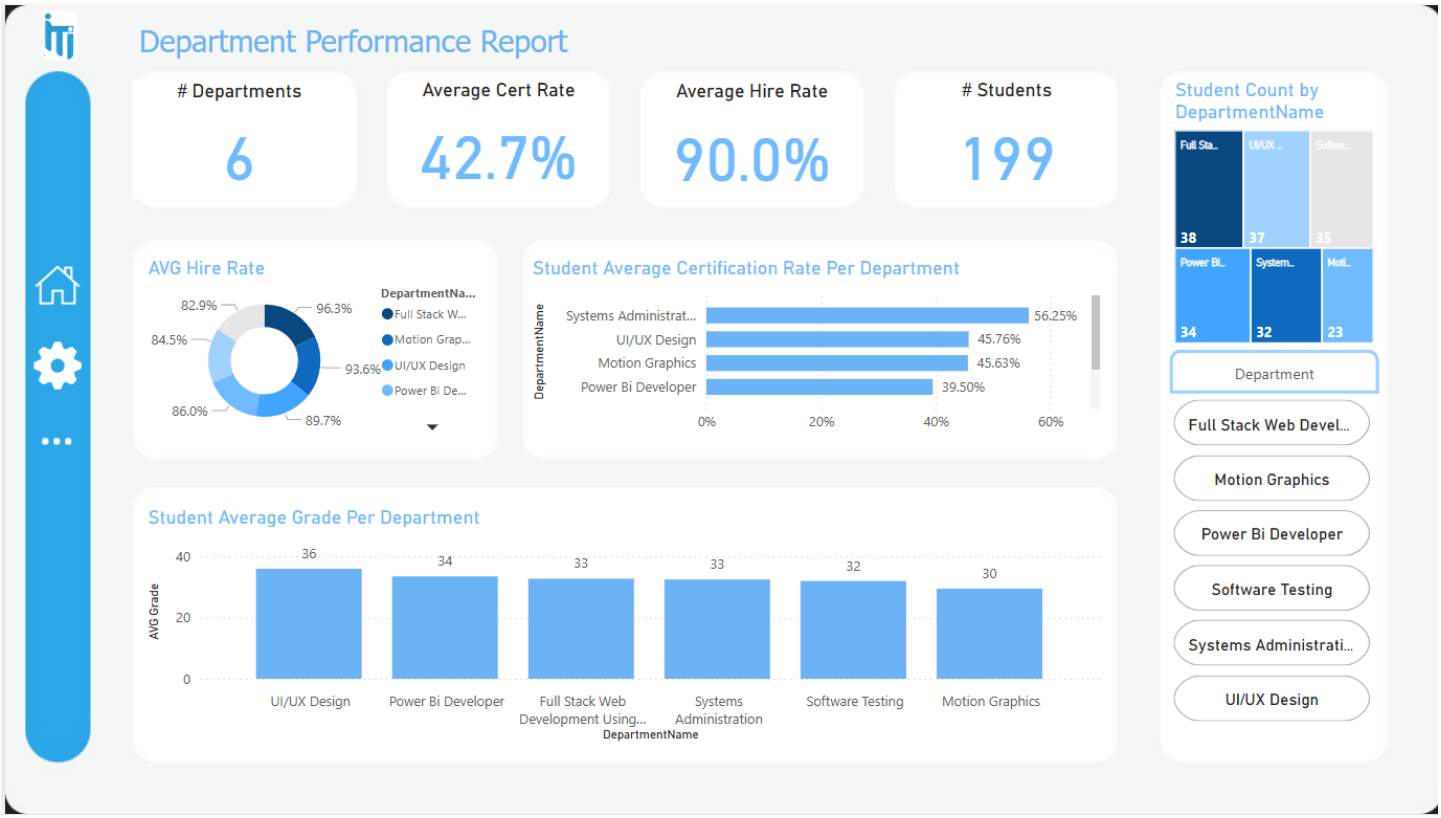
- 4.3 Branch Performance Report



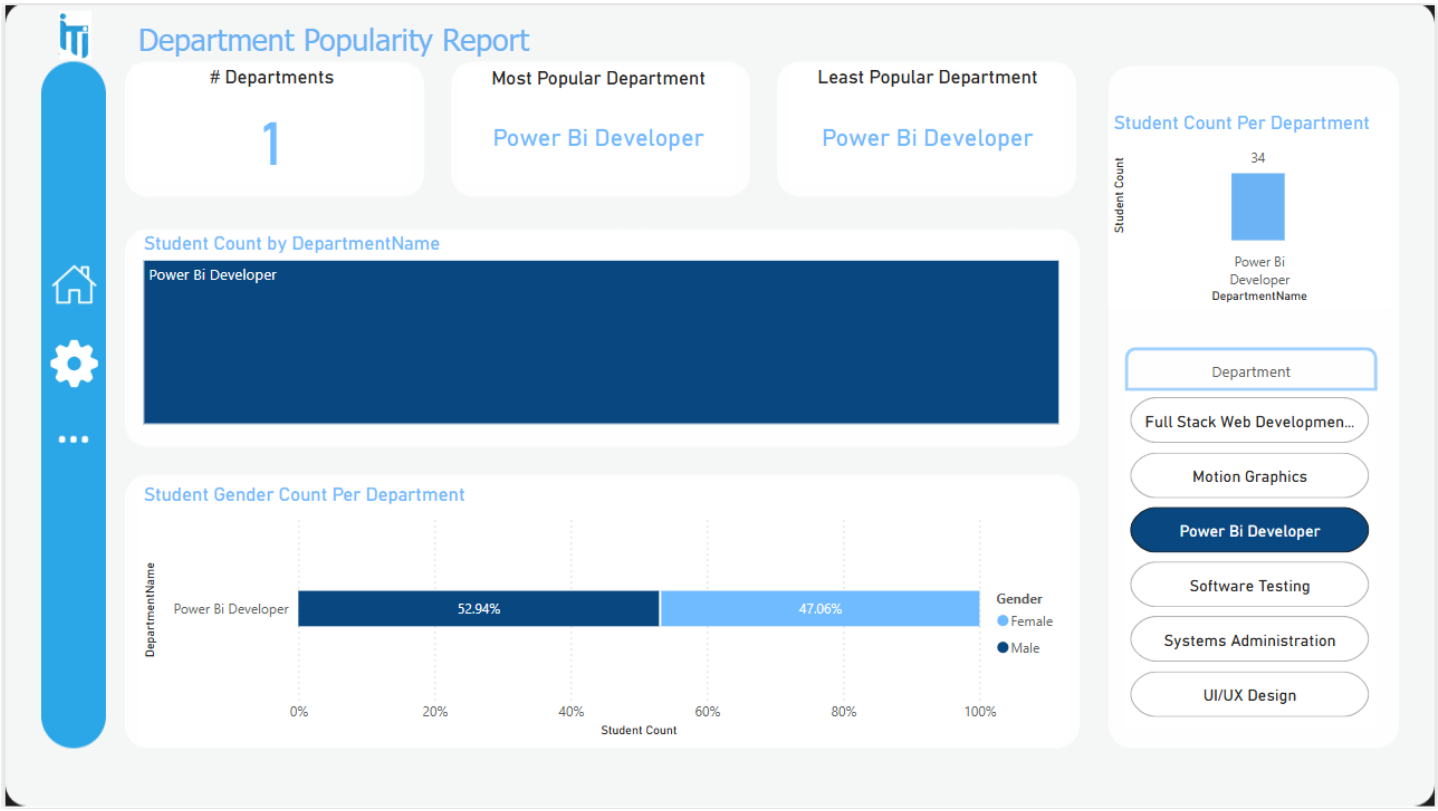
- 4.4 Course Progress Report

- 4.5 Course Progress Report

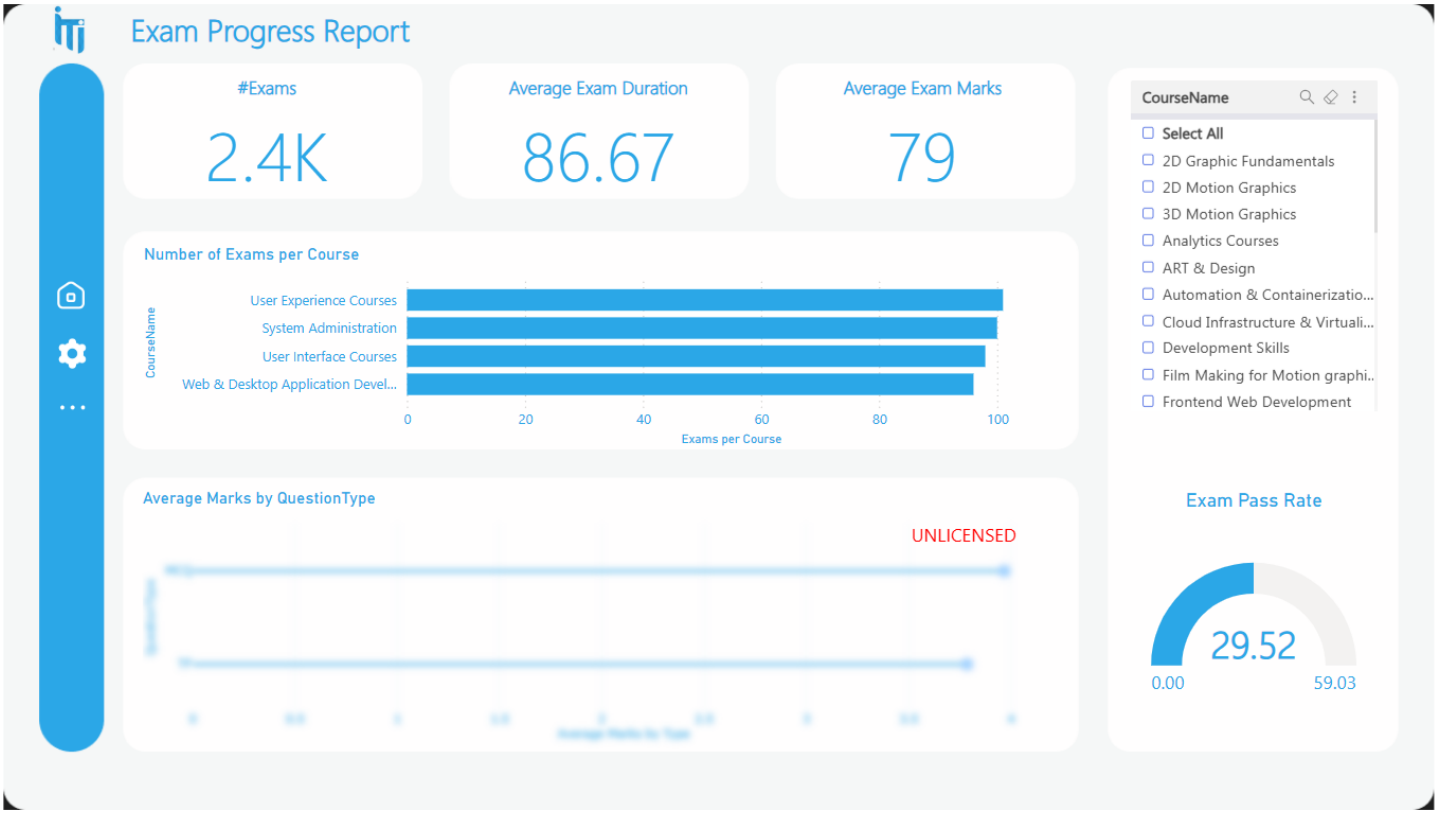
- 4.6 Department Performance Report



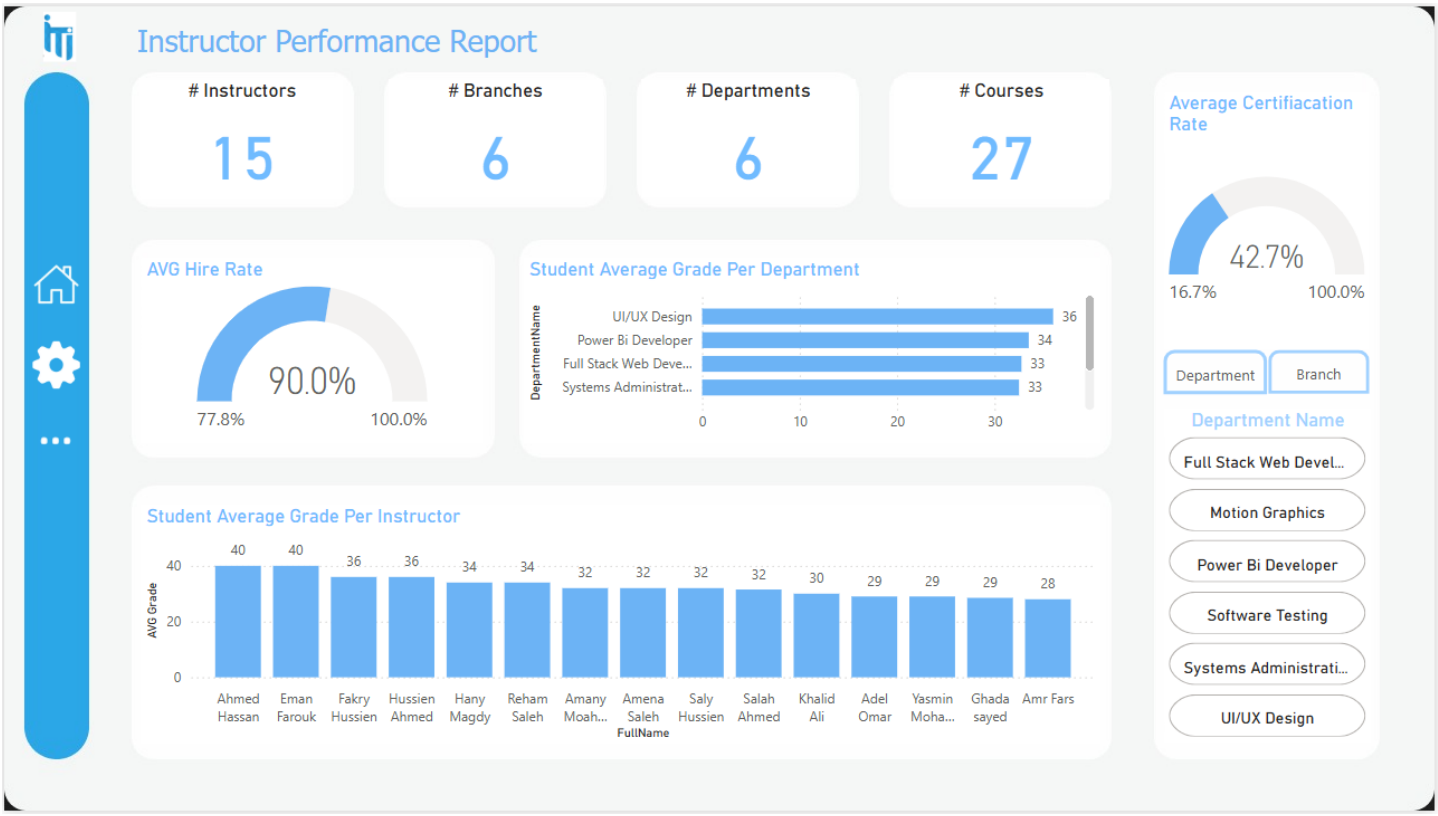
- 4.7 Department Popularity Report



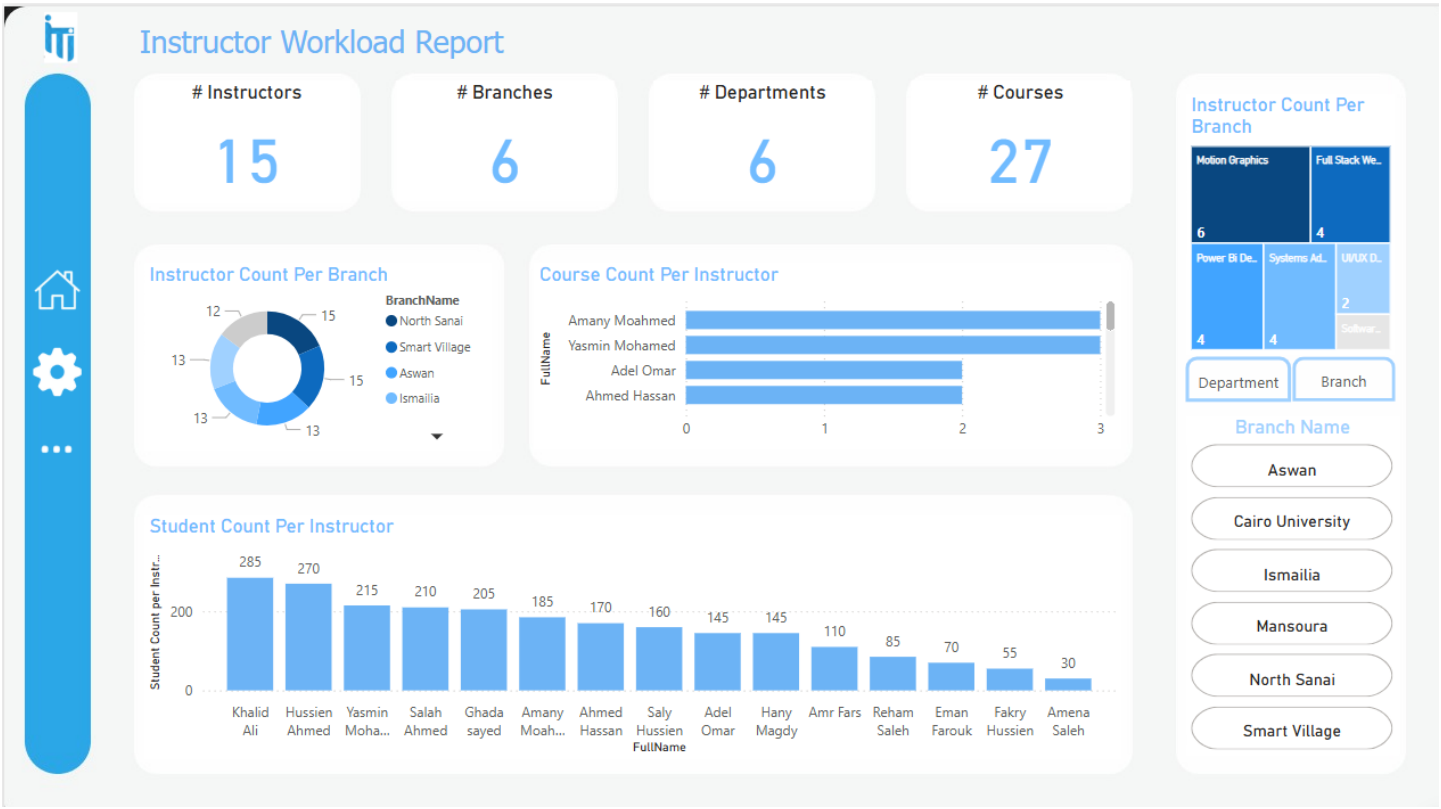
- 4.8 Exam Progress Report



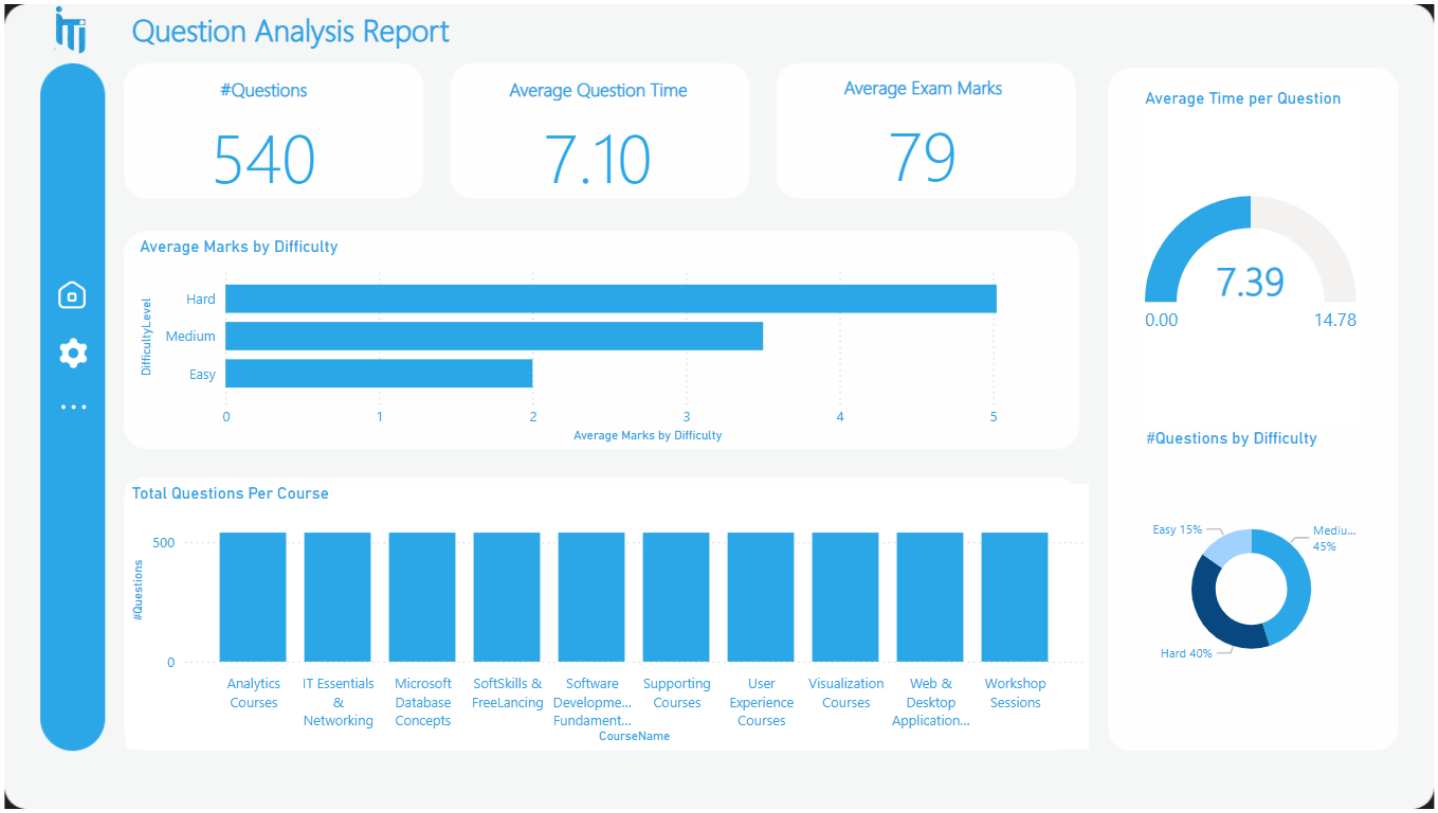
- 4.9 Instructor Performance Report



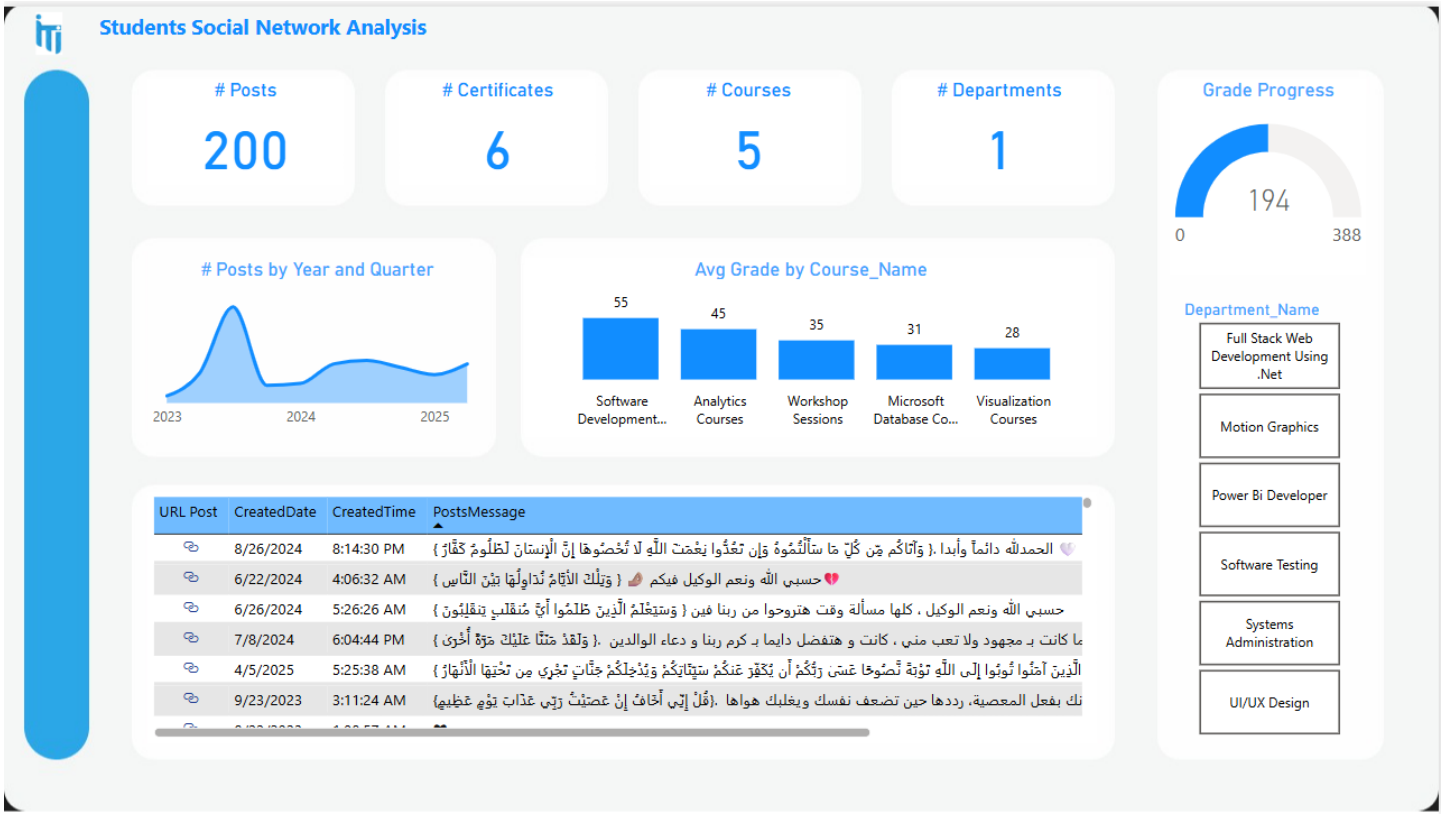
- 4.10 Instructor Workload Report



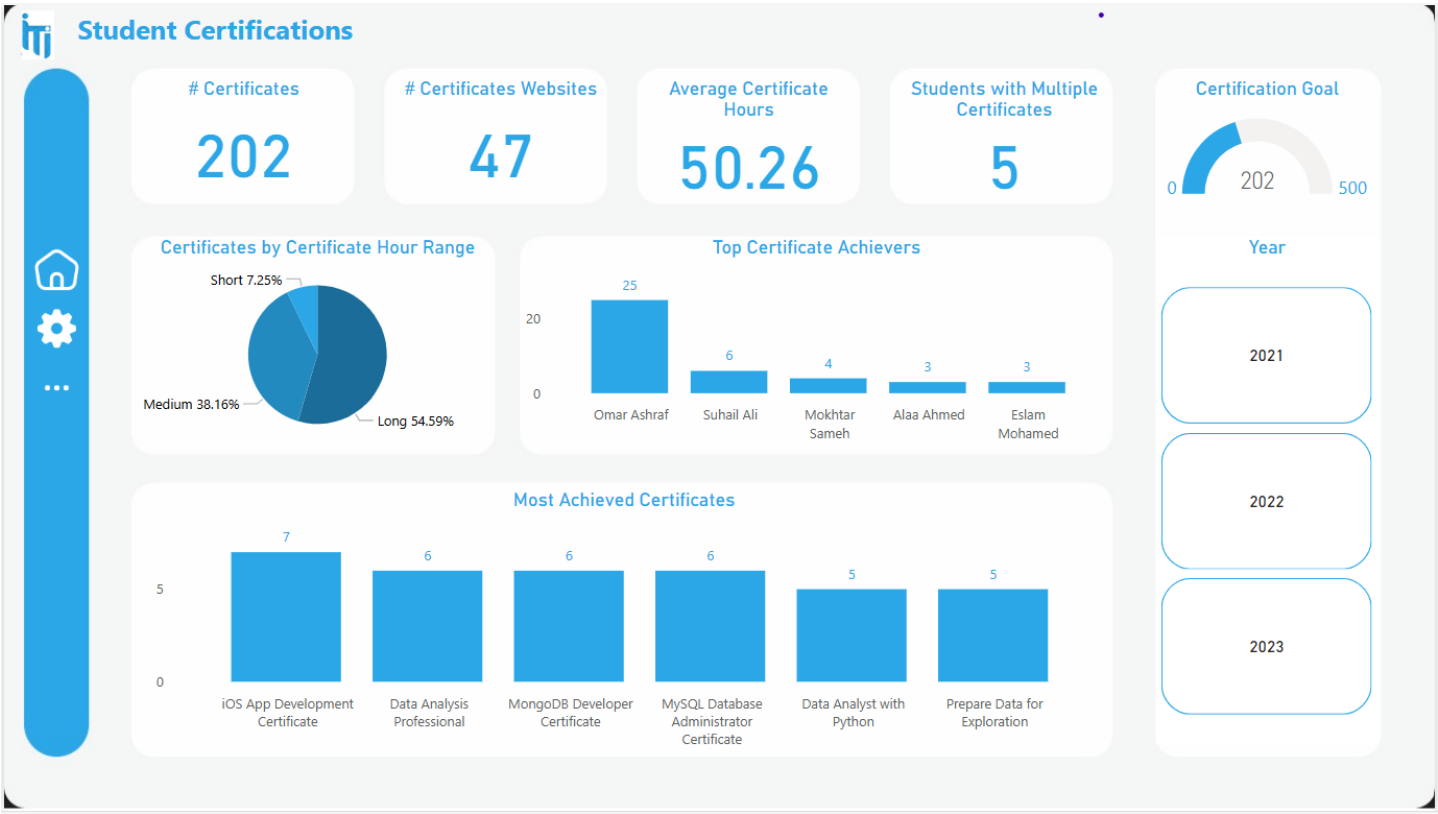
- 4.11 Question Analysis Report



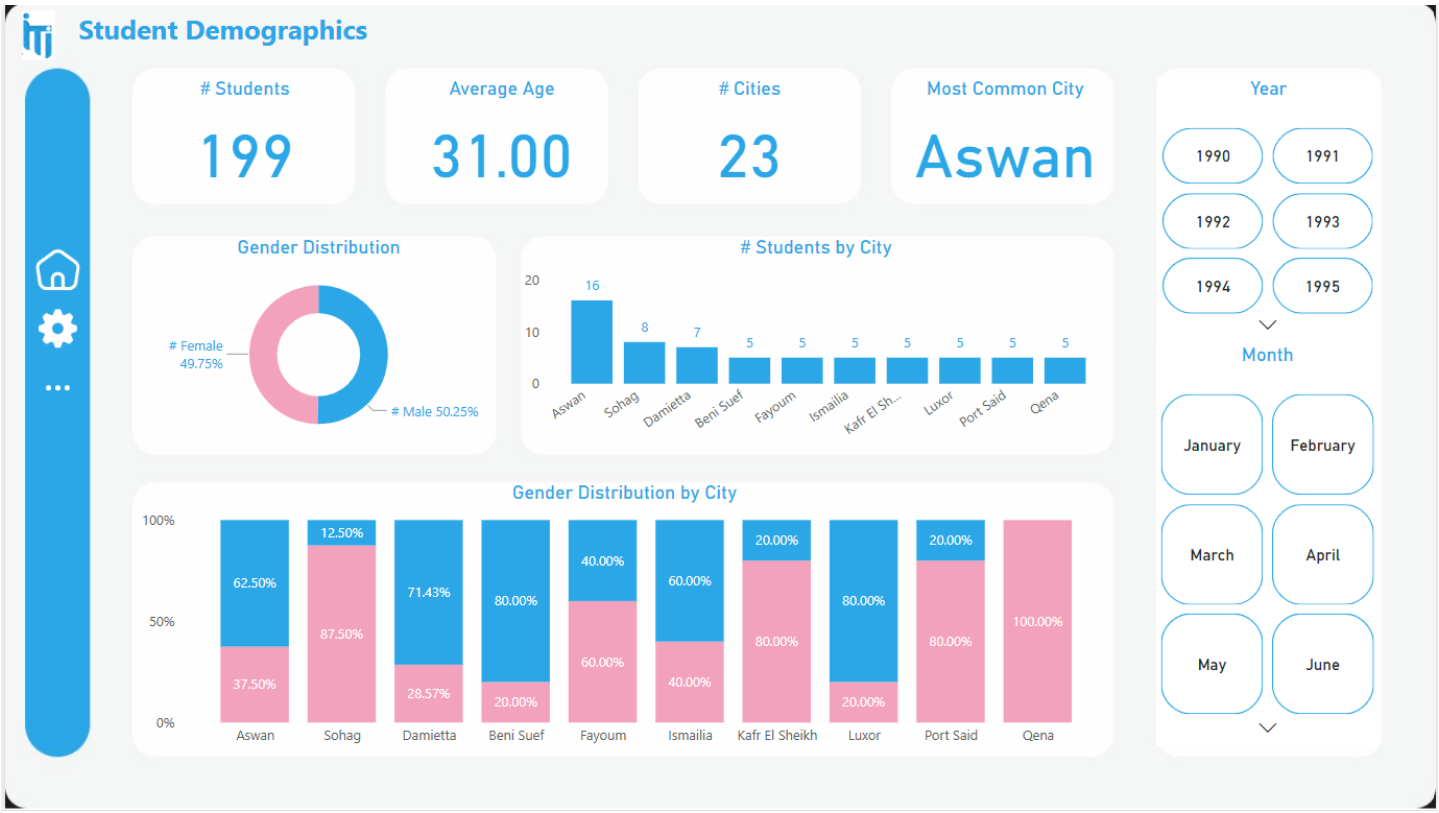
- 4.12 Students Social Network Analysis



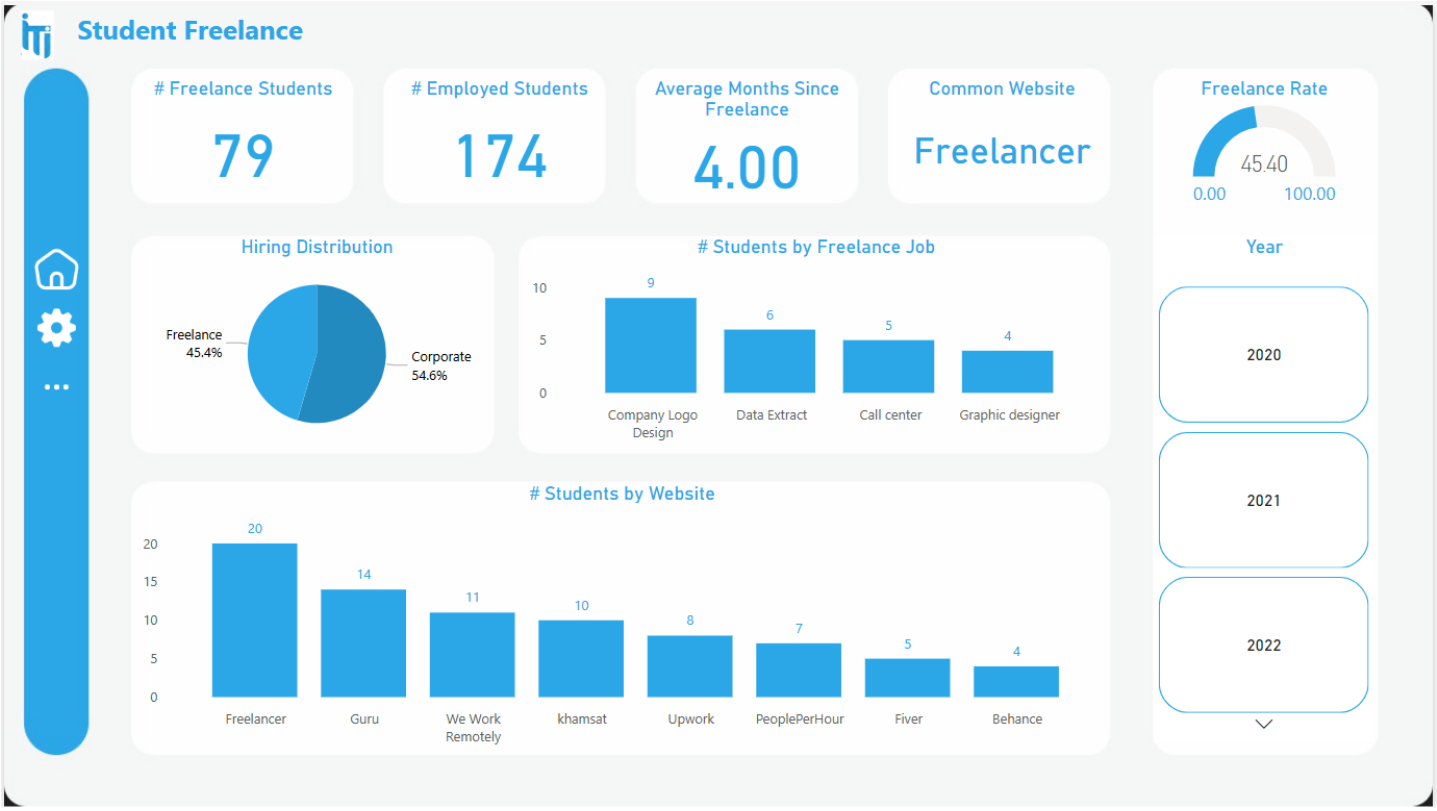
- 4.13 Students Certification Analysis



- 4.14 Students Demographics Analysis

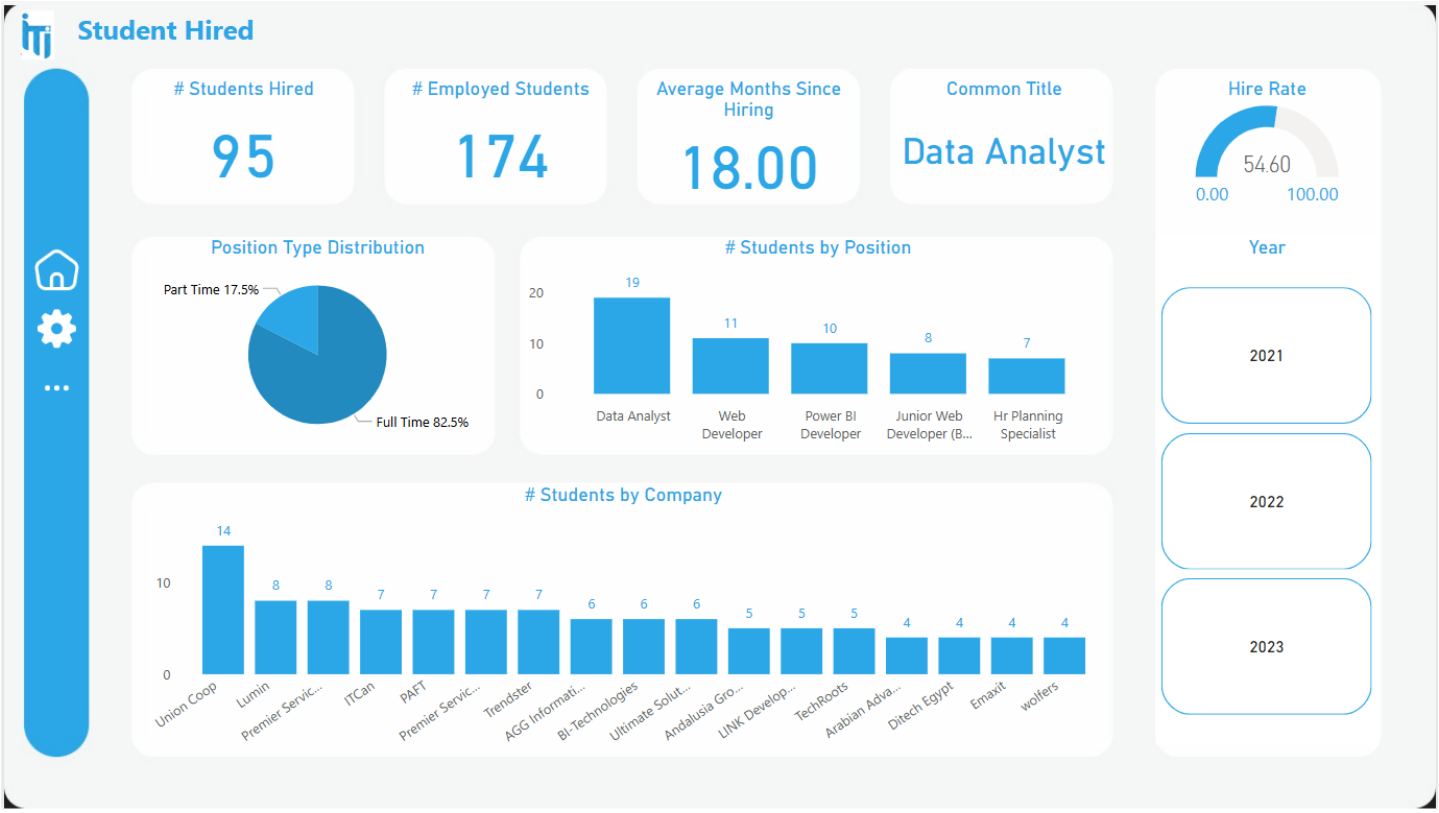


- 4.15 Students Freelance Analysis



- 4.16 Students Grades Analysis

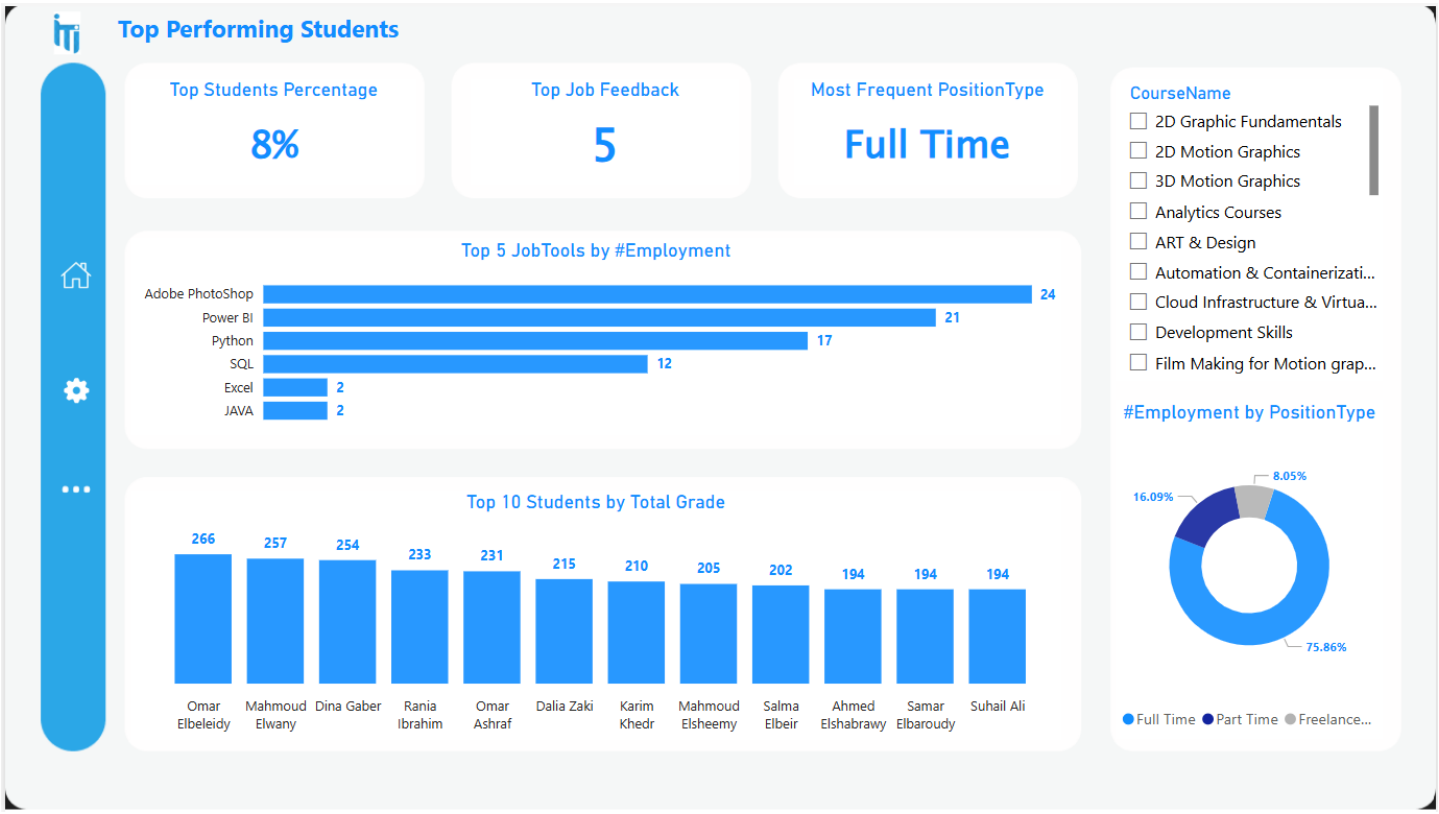
- 4.17 Students Hiring Analysis



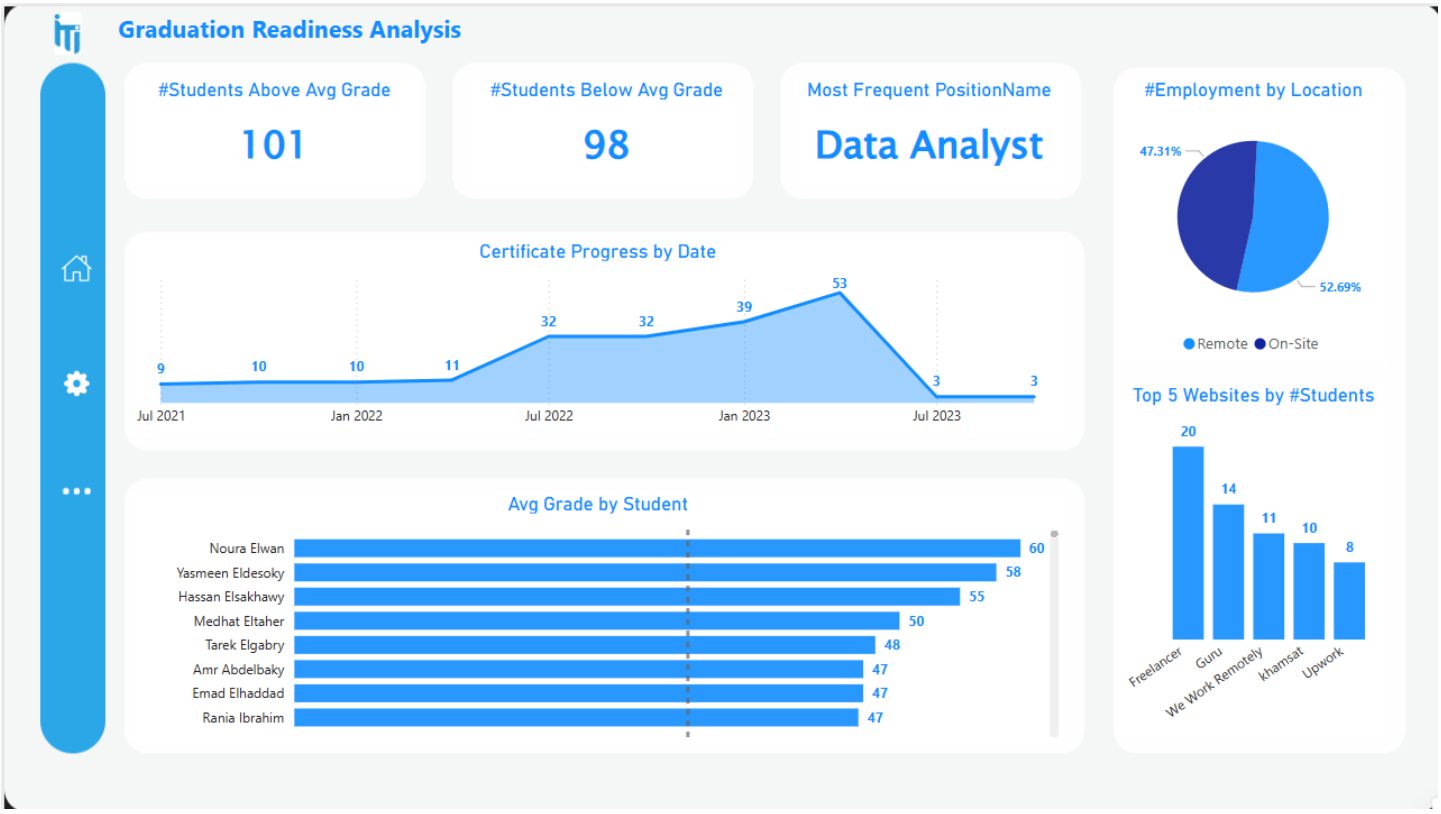
- 4.18 Students Retakes Analysis

- 4.19 Topic Coverage Report

- 4.20 Top Performing Students



- 4.21 Graduation Readiness Analysis



5 Desktop Application

- 5.1 Login