Microsoft Data Engineer Track

Project Idea 3:

Student Feedback Analysis and Improvement

Supervised by:

Mohamed Safi

Team Members:

Magdy Shabaan

Mohamed Hassan

Ahmed Sameh

Sherif Atef

Eman Abdelmohsen

Shrouk Mohamed

Abstract

This project outlines a comprehensive approach to analyzing and improving student feedback through advanced data management and machine learning techniques. In the initial phase, a SQL database is designed and populated with student feedback data, structured for efficient storage and retrieval. SQL queries are employed to extract meaningful insights from the feedback. A data warehouse is then implemented to aggregate the data, allowing for in-depth analysis across various dimensions, such as courses, instructors, and feedback categories.

Python scripts are developed to clean and preprocess the feedback data, preparing it for sentiment analysis. In the next phase, sentiment analysis models are created using machine learning algorithms to classify feedback into positive, neutral, or negative categories. The analysis is integrated into Azure Data Services for enhanced scalability and performance. Finally, the project employs MLOps techniques to manage, track, and deploy the sentiment analysis models via a dashboard, enabling real-time feedback monitoring and continuous improvement. The outcomes of the project showcase how data-driven insights can enhance the student learning experience by identifying areas for targeted improvements in teaching and course content.

Introduction

In the realm of education, continuous improvement based on student feedback is essential for enhancing the quality of teaching and learning experiences. This project aims to build a comprehensive system for analyzing and improving student feedback using modern data management techniques, including SQL databases, data warehousing, and machine learning-based sentiment analysis. By leveraging these technologies, educational institutions can gather deeper insights into student opinions, allowing them to refine course content, teaching methodologies, and overall student engagement.

The project is divided into four key phases, starting with the creation of a structured SQL database to store feedback data, followed by the implementation of a data warehouse to enable advanced analysis. Python-based data processing techniques are applied to clean and prepare the feedback for sentiment analysis. The feedback is then analyzed using machine learning models to classify responses into positive, neutral, or negative sentiments. The final stage involves deploying the sentiment analysis models using MLOps to ensure real-time insights and providing an interactive platform for institutions to visualize and act on the feedback effectively.

Project Phases

- SQL Database Setup and Data Collection.
- Data Warehouse and Python Data Processing
- Sentiment Analysis and Azure Integration.
- MLOps and Deployment

1. Data Collection and SQL Database Setup.

Data Collection

- Get real data from Faculty of Education of AinShams University.

Some Columns: Student_id, subject_title, subject_grade

Number of Records: 96046

student_id	subject_title	subject_Grade	subject_code	specialization	department	bylaws	with_written	Teacher_name
677	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	
679	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	
680	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	
682	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	
683	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	
684	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	
685	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	
686	subject 681	4	subject 681	أدبي	الجغرافيا		FALSE	

- Importing historical student data into the database.

Steps

Prepare student data (cleaning, structuring).

Import using SQL Server tools.

Verify data integrity (no duplicate student entries, consistent grade formats)

Database Design

Creating a structured SQL database to manage student-related data efficiently. These tables include :

program table: to store student program

Student table: to store student information

Instructor table: to store instructor information

Department table: to store department information

Subject table: to store courses information

Feedback Questions table: to categorize different questions of feedback.

Entity Relationship Diagram (ERD)

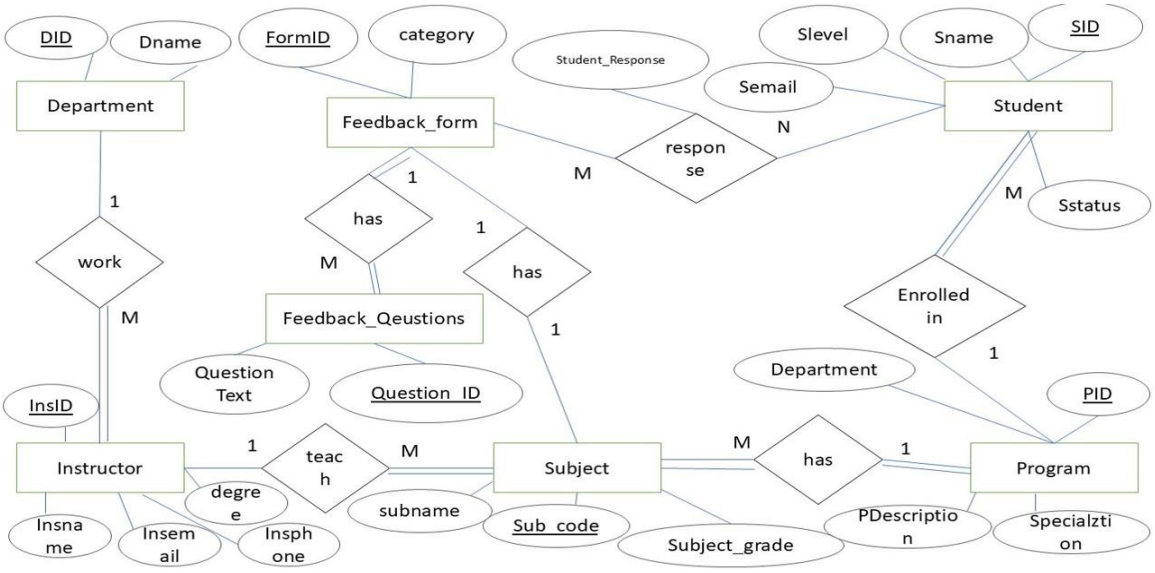
In any University, gathering feedback from students plays a crucial role in improving the quality of teaching and course content. The **Student Feedback System** presented in this Entity-Relationship Diagram (ERD) is designed to streamline the process of collecting, managing, and analyzing feedback from students on the courses they undertake.

This system provides a structured way to link feedback to various academic entities such as students, courses, instructors, departments, and specializations. By establishing clear relationships between these entities, the feedback system enables detailed evaluation and improvement of course delivery, instructional methods, and overall student satisfaction.

The ERD illustrates a comprehensive feedback collection mechanism that includes multiple components such as structured and open-ended feedback, suggestions for improvement, and performance evaluations. By integrating feedback with the courses, instructors, and departments, the system allows for a thorough analysis of academic performance at multiple levels, from individual courses to entire specializations.

This student feedback system not only supports course improvement but also helps academic institutions align their offerings with the needs and expectations of students, thereby fostering an environment of continuous academic enhancement.

Here's an explanation of each component in the ERD as shown in figure 1



Figure

Key Entities:

1. **Student**
   - **Attributes: SID (Student ID), Sname (Student Name), Slevel (Student Level), Semail (Student Email), Sstatus (Student Status) A student can respond to multiple feedback forms and can enroll in multiple departments.**
2. **Department**

- **Attributes: DID (Department ID), Dname (Department Name)**
- **A department can offer multiple programs and subjects, and it employs multiple instructors.**

3. **Program**
    a. **Attributes: PID (Program ID), PDescription (Program Description), Specialization**
    b. **Each program belongs to one department and can include many subjects.**
4. **Subject**
    a. **Attributes: Sub_code (Subject Code), Subname (Subject Name), Subject_grade**
    b. **Each subject is part of a program and is taught by multiple instructors.**
5. **Instructor**
    a. **Attributes: InsID (Instructor ID), Insname (Instructor Name), Insemail (Instructor Email), Insphone (Instructor Phone), degree**
    b. **An instructor can teach multiple subjects in different departments.**
6. **Feedback_Form**
    a. **Attributes: FormID (Form ID), Category**
    b. **A feedback form has many feedback questions and can be responded to by multiple students.**
7. **Feedback_Questions**
    a. **Attributes: Question_ID, Question_Text**
    b. **Each feedback question is associated with a feedback form.**
8. **Student_Response**
    a. **Represents the relationship between students and their feedback form responses.**

Relationships:

1. **Student → Feedback_Form (M**
    - **A student can respond to multiple feedback forms, and each feedback form can be responded to by many students. This is managed via the Student_Response entity.**
2. **Feedback_Form → Feedback_Questions (1**
    - **A feedback form contains multiple questions.**
3. **Instructor → Department (M:1)**
    - **Multiple instructors can work in one department.**
4. **Instructor → Subject (M:1)**
    - **An instructor can teach multiple subjects.**
5. **Subject → Program (M:1)**
    - **Multiple subjects belong to one program.**
6. **Program → Department (M:1)**
    - **Each program belongs to a specific department.**
7. **Student → Department (M)**
    - **A student can enroll in multiple departments, and a department can have many students.**

Mapping (ERD to Relational Model):

1. **Student Table**
    - **SID (Primary Key)**
    - **Sname**
    - **Slevel**
    - **Semail**
    - **Sstatus**

2. **Department Table**
    - **DID (Primary Key)**
    - **Dname**
3. **Program Table**
    - **PID (Primary Key)**
    - **PDescription**
    - **Specialization**
    - **DID (Foreign Key referencing Department)**
4. **Subject Table**
    - **Sub_code (Primary Key)**
    - **Subname**
    - **Subject_grade**
    - **PID (Foreign Key referencing Program)**
5. **Instructor Table**
    - **InsID (Primary Key)**
    - **Insname**
    - **Insemail**
    - **Insphone**
    - **degree**
    - **DID (Foreign Key referencing Department)**
6. **Feedback_Form Table**
    - **FormID (Primary Key)**
    - **Category**
7. **Feedback_Questions Table**

- **Question_ID (Primary Key)**
- **Question_Text**
- **FormID (Foreign Key referencing Feedback_Form)**

8. **Student_Response Table**
    - **SID (Foreign Key referencing Student)**
    - **FormID (Foreign Key referencing Feedback_Form)**
    - **Composite Primary Key: (SID, FormID)**

Relationship Tables:

1. **Student_Department Table**
    - **SID (Foreign Key referencing Student)**
    - **DID (Foreign Key referencing Department)**
    - **Composite Primary Key: (SID, DID)**
2. **Instructor_Subject Table**
    - **InsID (Foreign Key referencing Instructor)**
    - **Sub_code (Foreign Key referencing Subject)**
    - **Composite Primary Key: (InsID, Sub_code)**

This ERD organizes a feedback system where students can submit feedback on subjects and instructors through feedback forms with multiple questions. It efficiently manages the relationships between students, instructors, subjects, departments, and programs.

Schemas

• **Students Schema**
Tables: `Student`

• **Human Resources Schema**
Tables: `Department`, `Instructor`

• **Courses Schema**
Tables: `Course`, `Specialization`

• **Student Courses Interaction Schema**
Tables: `Stud_Crs`, `Feedback_Stud_Crs`

Implementing ERD on MS SQL Server

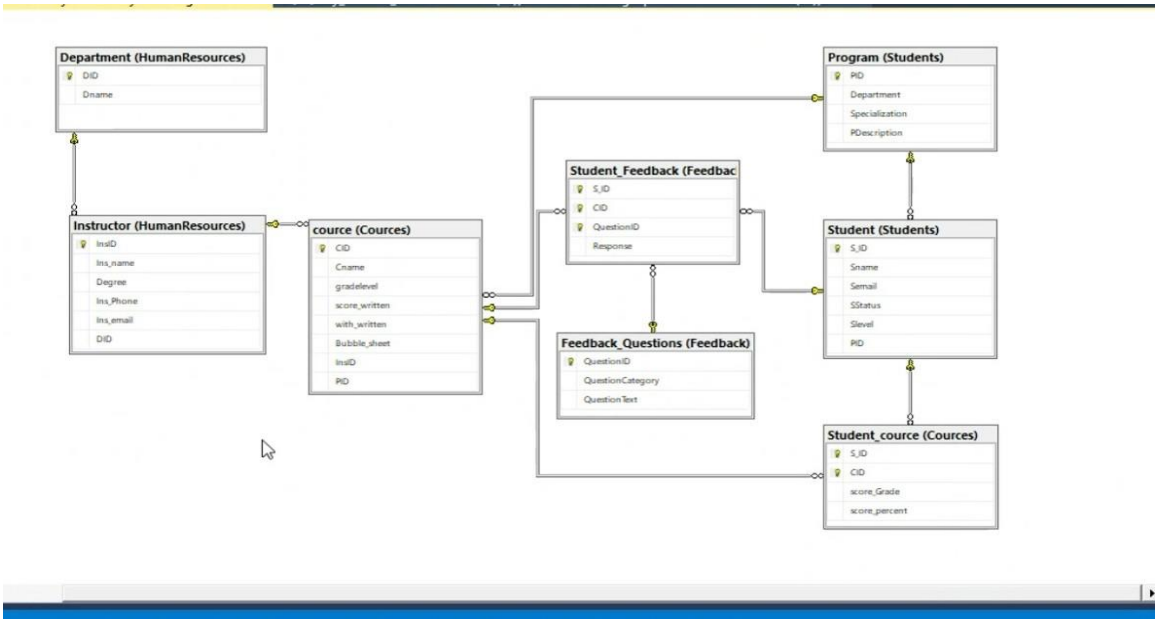


Figure: 2

Examples of SQL queries to extract and analyze student data.

Extract all courses a student is enrolled in:

```
SELECT Courses.CourseName
FROM students
JOIN Courses ON students.CourseCode = Courses.CourseCode
WHERE StudentID = 1001;
```

Summarize student grades by course

```
SELECT S_ID, AVG(Courses.score_Grade) AS AverageGrade
FROM Courses.Student_course
GROUP BY CID;
```

Find top-performing students:

```
SELECT S_ID, AVG(Courses.score_Grade) AS AverageGrade
FROM Courses.Student_course
GROUP BY S_ID
ORDER BY AverageGrade DESC;
```

2. Data Warehouse and Python Data Processing

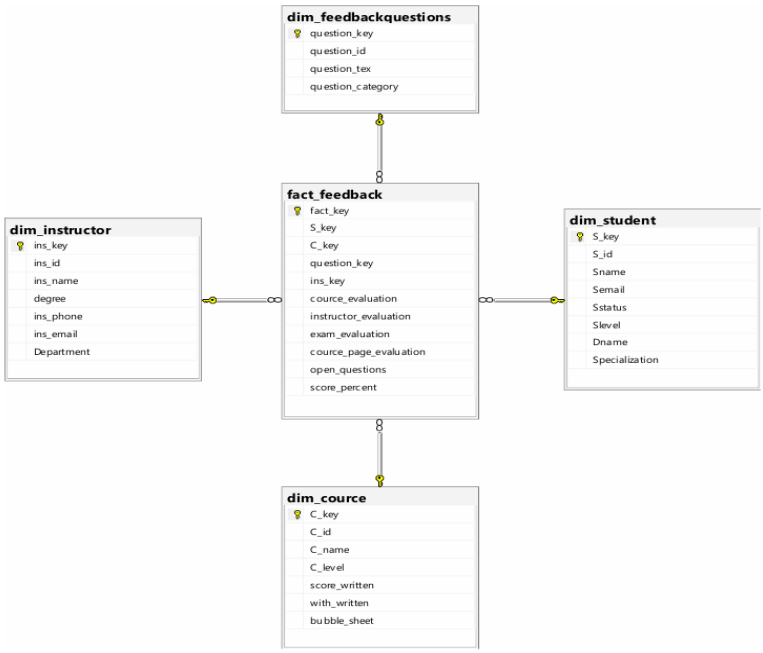
Steps

- Data Warehouse Implementation
  implement a data warehouse for aggregating feedback data
- Data Integration
  Load historical and current feedback data into the data warehouse
- Data Processing Using python
  Use Python to clean and preprocess the feedback data.

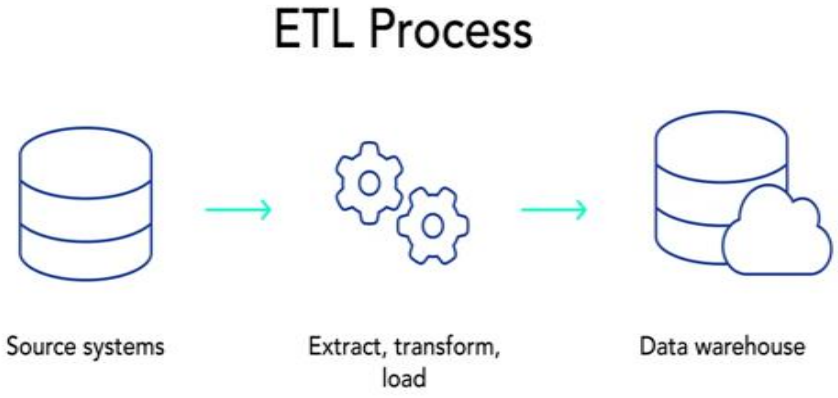
Data Warehouse

- Fact table
  Feedback
- Dimension Tables
  Student, Courses, Instructor, Feedback questions

Implement DataWarehouse on Ms SQL server



Data Integration



Goal: Load data into the data warehouse.

Steps:

- Extract feedback data from source system
- Transform the data (formatting, cleaning miss data).
- Load the cleaned data into the warehouse.

Data preprocessing Using Python

Use Python to clean and preprocess the feedback data.

- **Steps**:
  Data Cleaning: Remove duplicates, handle missing values, and normalize text data.
  Text Preprocessing:
  - Tokenize feedback text.
  - Remove stopwords and punctuation.
  - Apply stemming or lemmatization.

  Data preprocessing Using Python

- **Sentiment Analysis**:
  Use NLTK's sentiment analysis tools to categorize feedback as positive, negative, or neutral.

2. Sentiment Analysis

Naïve based usage

1. Data Input & Preprocessing

Load Data

```
# Load the Excel file
file_path = 'E:/1.0.0.0.1 DEPI/Final Project/Docs/sherif/Document1.xlsx'
df = pd.read_excel(file_path)
```

- Load the Excel file containing the responses.

Clean Text

- Remove unnecessary characters and apply text preprocessing.

Cleaning Process:

- Remove Non-Arabic Characters:
  Use regex to retain only Arabic characters and spaces.
- **Remove Stop-words:**
  Filter out common Arabic stop-words that do not add meaningful information.

```
stemmer = SnowballStemmer("arabic")

corpus = []
for i in range(len(df)):
    s = df['ConcatenatedResponse'][i]

    # Step 1: Replace specific characters
    s = re.sub('[ - ^]', ' ', s)  # Removed | and ـ from the pattern
    # Step 2: Custom split
    s = custom_split(s)
    # Step 3: Remove stopwords
    s = [word for word in s if word not in stopwords.words('arabic')]
    # Step 4: Join words back into a string
    s = ' '.join(s)
    # Step 5: Stem the words
    s = stemmer.stem(s)

    # Append to corpus
    corpus.append(s)
```

3. MLOps and Deployment

Implement a simple web page that predicts the sentiment of the given text on local host using VS Code and Flask (Python)