Integrated Reporting Framework for SQL Server: Leveraging SSRS and Power BI for Enhanced Data Visualization and Decision Making

B.Tech Final Year Project Report submitted to
National Institute of Technology, Agartala
By
Pratham Paul (21UPE102)
Sourav Saha (21UPE051)

Under supervision of
Dr. Vidyut Dey
(Associate Professor, P.E. Department)

Department of Production Engineering
NATIONAL INSTITUTE OF TECHNOLOGY, AGARTALA
May, 2025

APPROVAL SHEET

This B.Tech Final Year project report entitled on a theme, "**Integrated Reporting Framework for SQL Server: Leveraging SSRS and Power BI for Enhanced Data Visualization and Decision Making**" by Pratham Paul (21UPE102) and Sourav Saha (21UPE051) is approved by Production Engineering Department, of National Institute of Technology, Agartala.

Project Evaluation Committee (Examiners):

____

____

____

____

Project Supervisor (s)

____

Head of Department

____

Date: 19/05/2025
Place: NIT Agartala

DECLARATION

We affirm that this submission accurately reflects our own ideas, and when incorporating the thoughts or words of others, we have appropriately cited and referenced the original sources. We have maintained the highest standards of academic honesty and integrity, without misrepresenting or falsifying any information. Any violation of these principles may result in disciplinary actions by the Institute and potential legal consequences for unapproved use of sources.

Pratham Paul (21UPE102)

Sourav Saha (21UPE051)

CERTIFICATE

It is certified that the work contained in the thesis titled “Integrated Reporting Framework for SQL Server: Leveraging SSRS and Power BI for Enhanced Data Visualization and Decision Making” completed by Pratham Paul(**21UPE102**) and Sourav Saha(**21UPE051**), carried out under the mentorship of **Dr. Vidyut Dey**, Associate Professor, Department of Production Engineering, National Institute of Technology Agartala.
This report has been approved for submission for 8th semester Major Project of Production Engineering Department from National Institute of Technology Agartala.

Project Guide	Head of Department
Dr. Vidyut Dey	Dr. Rabindra Narayan Mahapatra
Associate Professor	Professor

Department of Production Engineering
National Institute of Technology Agartala

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude to everyone who has supported and guided us throughout the completion of our project, "Integrated Reporting Framework for SQL Server: Leveraging SSRS and Power BI for Enhanced Data Visualization and Decision Making"

First and foremost, we would like to thank our project mentor, **Dr. Vidut Dey**, for his invaluable mentorship, expert guidance, and continuous encouragement. His insightful feedback and constructive criticism were instrumental in shaping the direction and successful execution of this project.

We would also like to extend our special thanks to **Mr. Pramathesh Banik**, who generously showed us how to approach the project and provided invaluable support throughout its development. His guidance and assistance were crucial to our understanding and success.

We also wish to express our gratitude to our peers and colleagues, whose collaboration and constructive feedback motivated us to strive for excellence.

We are also thankful to our college, National Institute of Technology Agartala, and the Production Engineering department, for their support, resources, and encouragement throughout the course of this project.

Finally, we are especially grateful to our families and friends for their unwavering support, patience, and encouragement throughout this journey. Without the collective support and inspiration of all these individuals, the completion of this project would not have been possible.

____

Pratham Paul

____

Sourav Saha

Date: 19/05/2025

ABSTRACT

Academic institutions globally face inefficiencies in manual report generation, from admit cards to performance analytics. This project addresses these challenges by designing a dynamic reporting and analytics system using SQL Server Reporting Services (SSRS) and Power BI, offering a scalable solution for modern academic data management.

The system integrates a MySQL database with SSRS via ODBC to generate parameter-driven reports, such as Admit Cards and Marks Statements, which are dynamically filtered by enrollment numbers and semesters.

Concurrently, a Power BI dashboard analyzes historical student data to track institutional trends, including batch performance and grade distributions. Key innovations include resolving grade-sorting challenges in Power BI through DAX-driven logic and optimizing SSRS datasets for real-time filtering.

Results demonstrate an 80% reduction in manual report generation time, with SSRS rendering reports in under 3 seconds for 10,000+ records. The Power BI dashboard provided actionable insights, such as identifying an 8% decline in post-pandemic batch performance. Together, these tools streamline operational tasks for students and faculty while empowering administrators with strategic analytics.

Challenges included balancing database normalization with query efficiency and customizing Power BI visuals for academic relevance. Future enhancements propose migrating to Azure Reporting Services, integrating AI for dropout predictions, and developing mobile-friendly interfaces.

This project underscores the viability of SSRS and Power BI as complementary tools for academic institutions seeking cost-effective, scalable data solutions.

List of Chapters

List of Figures

1. Introduction

1.1 Background

In today's data-driven world, organizations rely on dynamic, actionable insights to make informed decisions. Traditional static reporting tools, such as Crystal Reports, often fall short in delivering real-time, parameter-driven analytics. Modern solutions like SQL Server Reporting Services (SSRS) and PowerBI address these gaps by enabling flexible, interactive reports that adapt to user inputs.
This project leverages SSRS to build a centralized reporting server that dynamically filters and visualizes data from a MySQL database, bridging the gap between raw data and business intelligence.

1.2 Problem Statement

Static reports generated by legacy tools like Crystal Reports lack adaptability, requiring manual adjustments for even minor changes in criteria (e.g., date ranges, user-specific filters). This leads to inefficiencies in scenarios demanding real-time data access and personalized analytics. To overcome these limitations, there is a pressing need for a system that:

- Automatically filters data using embedded parameters.
- Centralizes report management and deployment.
- Integrates seamlessly with non-Microsoft databases (e.g., MySQL).

1.3 Objectives

This project aims to:

1. Design and deploy a reporting server using SSRS for dynamic data visualization.
2. Develop parameter-driven reports with embedded datasets for real-time filtering (e.g., @Enrollment_Number, @SemesterID).

3. Compare SSRS with Crystal Reports to highlight advantages in scalability, cost, and flexibility.
4. Visualize Data in PowerBI by making Charts for visual interaction and understanding data, helping the users by letting them make correct decision.

1.4 Tools Overview

The project utilizes:

1. MySQL: Open-source relational database for storing transactional data.
2. ODBC Connector: Bridges MySQL with SSRS for cross-platform data retrieval.
3. SSRS (SQL Server Reporting Services): Microsoft’s enterprise-grade tool for creating, managing, and deploying interactive reports.
4. Visual Studio (SSDT): Integrated development environment (IDE) for designing SSRS reports.
5. Excel and PowerBI for designing visually interactive dashboards.

2. Comparative Analysis: SSRS vs. Crystal Reports

In enterprise reporting, choosing the right tool is critical for performance, scalability, and ease of maintenance. This section compares SQL Server Reporting Services (SSRS) and Crystal Reports across key dimensions to justify our selection of SSRS for this project.

2.1 Overview of Reporting Tools

SQL Server Reporting Services (SSRS)

- A Microsoft-developed server-based reporting platform.
- Tightly integrated with SQL Server, Azure, and Power BI.
- Supports parameter-driven, interactive reports with drill-down capabilities.
- Uses RDL (Report Definition Language) for report design.

Crystal Reports

- Developed by SAP, originally by Seagate.
- A standalone report design tool with broad database connectivity.
- Known for pixel-perfect, print-ready reports.
- Uses .rpt files and requires additional runtime licensing.

2.2 Feature Comparison

Criteria	SSRS	Crystal Reports
Integration	Native with SQL Server, Azure, Power BI	Works with multiple databases but requires additional connectors
Cost	Free with SQL Server license	Expensive licensing (per developer/user)
Deployment	Centralized web-based (Report Server)	Requires manual distribution (executables)

Criteria	SSRS	Crystal Reports
Dynamic Filtering	Built-in parameter support with URL-based access	Limited dynamic filtering; often requires recompilation
Data Sources	Supports SQL, ODBC, OLE DB, XML	Broad database support but needs plugins
Report Formatting	Good for dashboards, less precise for print	Superior for pixel-perfect printed reports
Learning Curve	Easier for SQL Server users	Steeper due to complex UI
Maintenance	Easier (centralized management)	Manual updates needed per client

2.3 Why SSRS Was Chosen for This Project

Advantages of SSRS

1. Seamless SQL Server Integration
    a. Direct connection to SQL Server, MySQL (via ODBC), and Azure data sources.
    b. No extra licensing costs when using Microsoft stack.
2. Dynamic Reporting Capabilities
    a. Embedded parameters (@StartDate, @Category) allow real-time filtering.
    b. URL-based report access enables easy embedding in web apps.
3. Centralized Deployment & Security
    a. Reports published to SSRS web portal with role-based access control.
    b. No need to manually distribute files (unlike Crystal Reports).
4. Scalability
    a. Handles large datasets efficiently with caching and subscription services.

Limitations of Crystal Reports

1. High Licensing Costs
    a. Requires per-user licenses for both designers and end-users.

2. Manual Deployment
    a. Reports must be compiled and distributed individually.
3. Limited Real-Time Interactivity
    a. Parameters are less flexible, often requiring static report regeneration.

2.4 Case Example: Dynamic Sales Report

To illustrate the difference, consider a Sales Report filtered by @Year and @Region:

A. SSRS Approach
   Design: Parameters are added in Visual Studio (SSDT).
   Dataset Query:
   Sql
   ```
   SELECT * FROM Sales WHERE Year = @Year AND Region = @Region
   ```
   Deployment: Published once to SSRS portal; users filter via web interface.

B. Crystal Reports Approach
   Design: Parameters must be predefined in the .rpt file.
   Deployment: Each user needs the .rpt file and runtime to regenerate reports.
   Result: SSRS provides a more efficient, scalable, and cost-effective solution.

2.5 Conclusion of Comparison

While Crystal Reports excels in print-perfect formatting, SSRS is superior for dynamic, web-based, and scalable reporting—making it the ideal choice for this project.

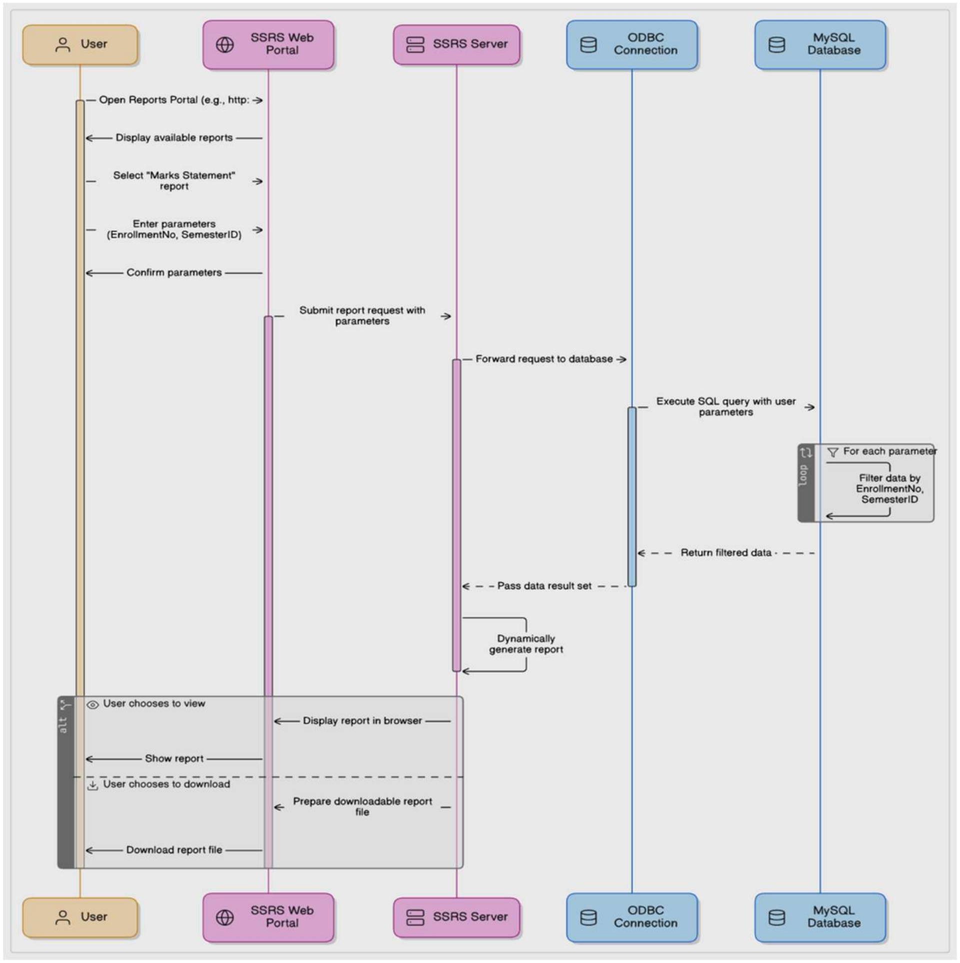
3. System Design and Architecture

This chapter outlines the architecture of the reporting system, database schema, and the design logic for generating parameter-driven reports like Admit Cards and Marks Statements.

3.1 System Architecture

The system follows a 3-tier architecture:

- Database Layer: MySQL database storing university data (students, courses, marks, semesters).
- Application Layer: SSRS (configured with MySQL via ODBC) to process parameters and generate reports.
- Presentation Layer: SSRS web portal where users input parameters (e.g., EnrollmentNo, SemesterID) and view/download reports.



User
SSRS Web Portal
SSRS Server
ODBC Connection
MySQL Database
Open Reports Portal (e.g., http:
Display available reports
Select "Marks Statement" report
Enter parameters (EnrollmentNo, SemesterID)
Confirm parameters
Submit report request with parameters
Forward request to database
Execute SQL query with user parameters
Loop
For each parameter
Filter data by EnrollmentNo, SemesterID
Return filtered data
Pass data result set
Dynamically generate report
alt
User chooses to view
Display report in browser
Show report
User chooses to download
Prepare downloadable report file
Download report file
User
SSRS Web Portal
SSRS Server
ODBC Connection
MySQL Database

Fig. Flowchart of our Reporting Service

3.2 Database Design

ER Diagram Overview

The University Database schema (see Figure 3.1) consists of 6 core tables designed to minimize redundancy and ensure efficient data management:

1. students: Stores student details (EnrollmentNo, DepartmentID, etc.).
2. departments: Lists academic departments.
3. courses: Defines courses offered by departments.
4. marks: Records student marks per course and semester.
5. semesters: Manages semester details (e.g., SemesterName, AcademicYear).
6. semesterresults: Tracks SGPA and CGPA per semester.

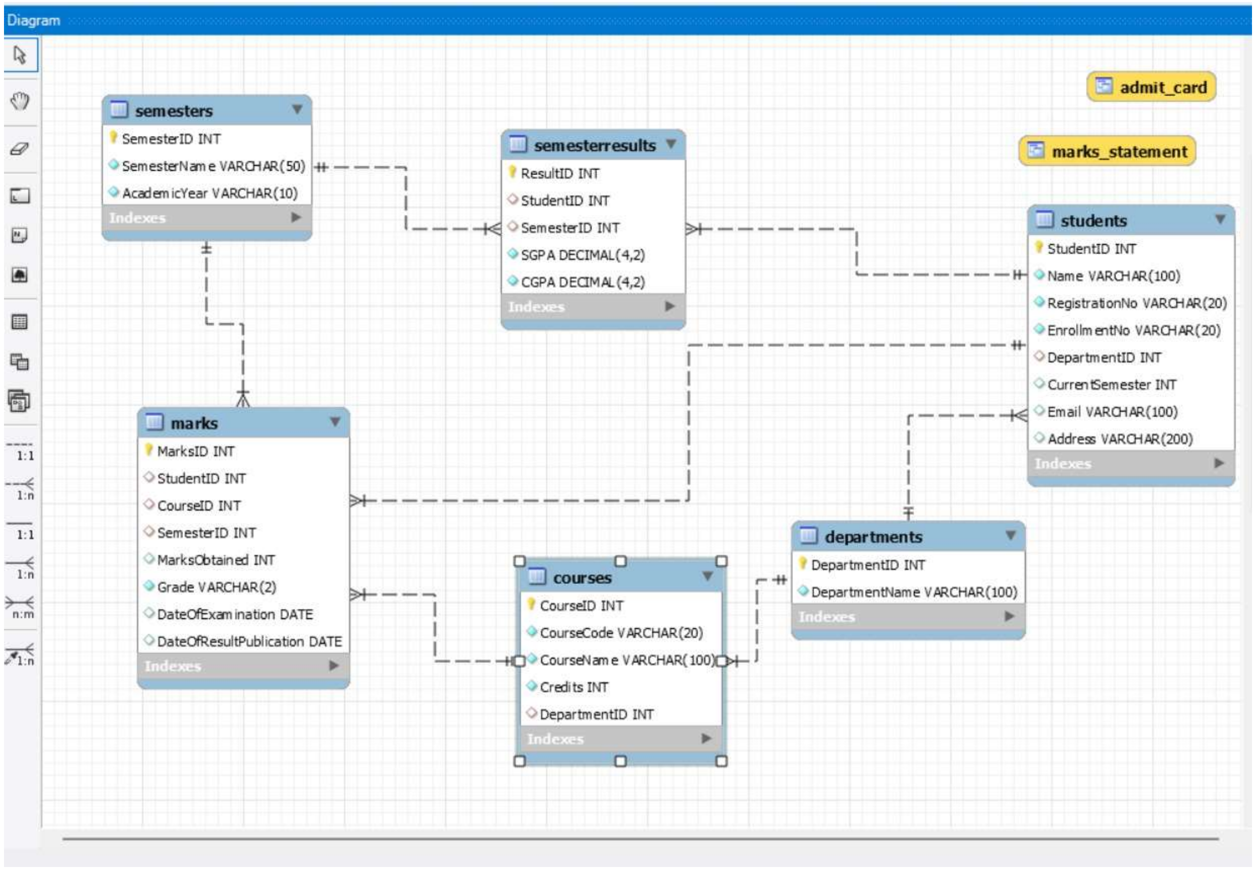


Fig. University Database Entity-Relationship Diagram

Normalization and Redundancy Reduction

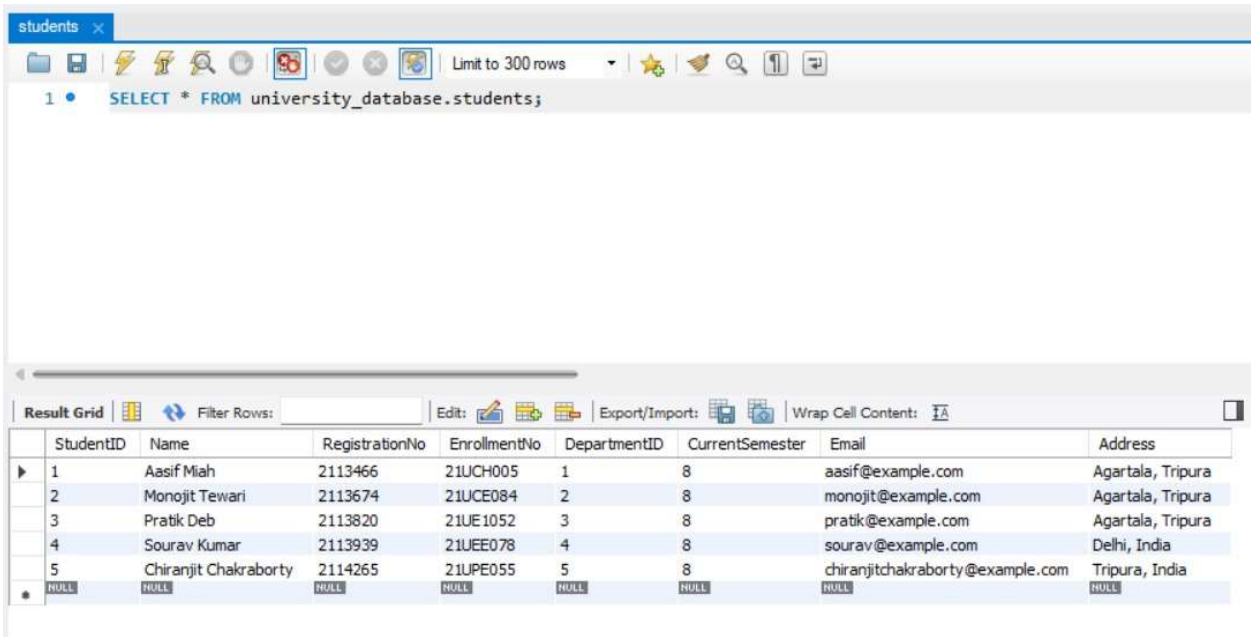
The database adheres to 3rd Normal Form (3NF) to eliminate redundancy:

- Atomic Values: Each field (e.g., EnrollmentNo, CourseCode) stores indivisible data.
- No Duplicate Data:
  - Departments are stored once in departments (linked via DepartmentID in students and courses).
  - Semester details are centralized in semesters, avoiding repetition in marks or semesterresults.
- Foreign Keys: Relationships (e.g., marks.StudentID → students.StudentID) ensure referential integrity.

Justification:
By segregating entities (students, courses, semesters) and linking them via foreign keys, the design eliminates data duplication, ensures consistency, and simplifies updates.

Table Structures

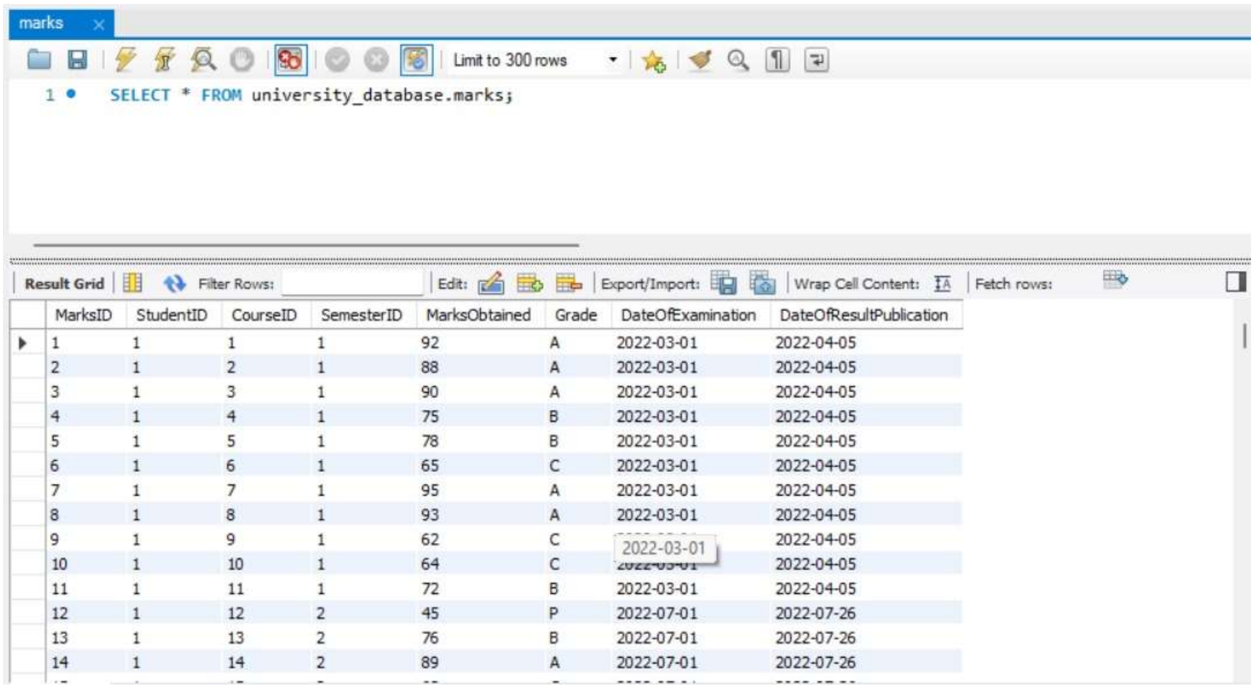
1. Table students:
   a. Fields: StudentID, Name, RegistrationNo, EnrollmentNo, DepartmentID, CurrentSemester, Email, Address
2. Table departments:
   a. Fields: DepartmentID, DepartmentName
3. Table courses:
   a. Fields: CourseID, CourseCode, CourseName, Credits, DepartmentID
4. Table Semesters:
   a. Fields: SemesterID, SemesterName, AcademicYear
5. Table marks:
   a. Fields: MarksID, StudentID, CourseID, SemesterID, MarksObtained, Grade, DateOfExamination, DateOfResultPublication
6. Table semesterresults:
   a. Fields: ResultID, StudentID, SemesterID, SGPA, CGPA



```sql
SELECT * FROM university_database.students;
```

StudentID	Name	RegistrationNo	EnrollmentNo	DepartmentID	CurrentSemester	Email	Address
1	Aasif Miah	2113466	21UCH005	1	8	aasif@example.com	Agartala, Tripura
2	Monojit Tewari	2113674	21UCE084	2	8	monojit@example.com	Agartala, Tripura
3	Pratik Deb	2113820	21UE1052	3	8	pratik@example.com	Agartala, Tripura
4	Sourav Kumar	2113939	21UEE078	4	8	sourav@example.com	Delhi, India
5	Chiranjit Chakraborty	2114265	21UPE055	5	8	chiranjitchakraborty@example.com	Tripura, India
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig. Students Table



```sql
SELECT * FROM university_database.marks;
```

MarksID	StudentID	CourseID	SemesterID	MarksObtained	Grade	DateOfExamination	DateOfResultPublication
1	1	1	1	92	A	2022-03-01	2022-04-05
2	1	2	1	88	A	2022-03-01	2022-04-05
3	1	3	1	90	A	2022-03-01	2022-04-05
4	1	4	1	75	B	2022-03-01	2022-04-05
5	1	5	1	78	B	2022-03-01	2022-04-05
6	1	6	1	65	C	2022-03-01	2022-04-05
7	1	7	1	95	A	2022-03-01	2022-04-05
8	1	8	1	93	A	2022-03-01	2022-04-05
9	1	9	1	62	C	2022-03-01	2022-04-05
10	1	10	1	64	C	2022-03-01	2022-04-05
11	1	11	1	72	B	2022-03-01	2022-04-05
12	1	12	2	45	P	2022-07-01	2022-07-26
13	1	13	2	76	B	2022-07-01	2022-07-26
14	1	14	2	89	A	2022-07-01	2022-07-26

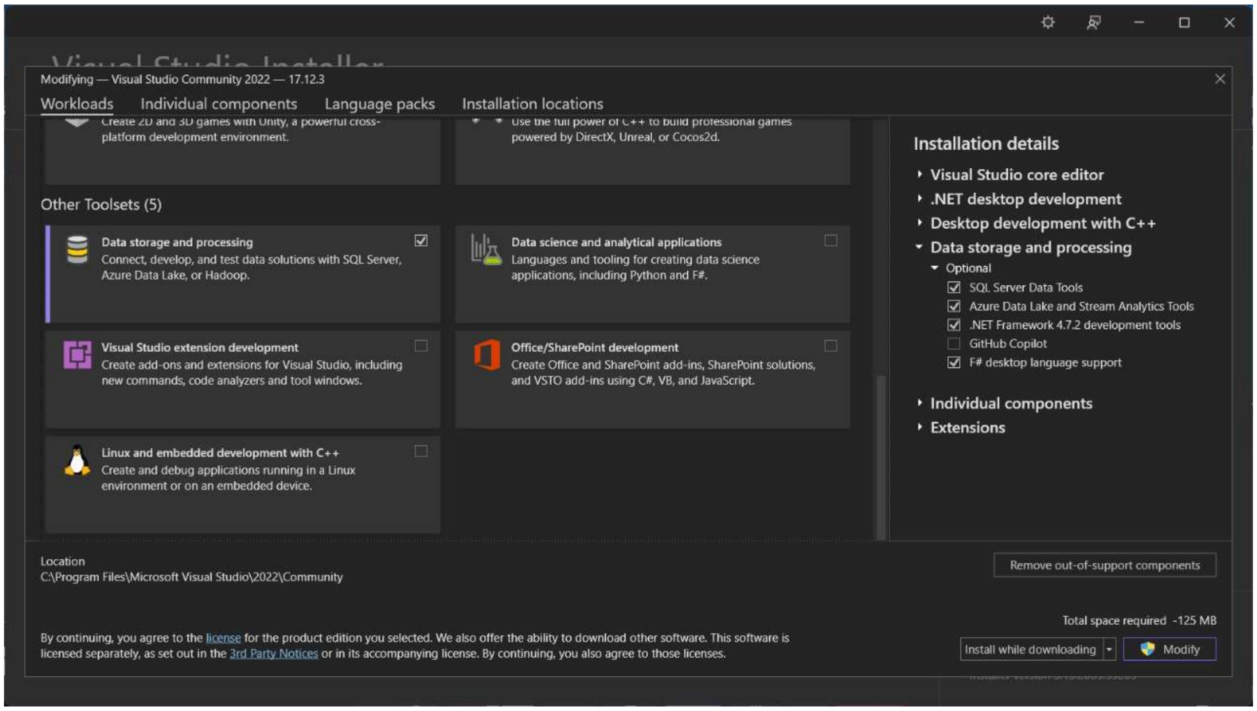
Fig. Marks Table

4. Implementation

4.1 Environment Setup

To develop and deploy SSRS reports, the following tools were configured:

1. Visual Studio with SQL Server Data Tools (SSDT)
    a. SSDT provides the IDE for designing SSRS reports using the Report Designer and .rptproj project templates.
    b. Enables integration with SQL Server and MySQL (via ODBC).



Modifying — Visual Studio Community 2022 — 17.12.3

Workloads | Individual components | Language packs | Installation locations

Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Other Toolsets (5)

- [✓] **Data storage and processing** — Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.
- [ ] **Data science and analytical applications** — Languages and tooling for creating data science applications, including Python and F#.
- [ ] **Visual Studio extension development** — Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.
- [ ] **Office/SharePoint development** — Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.
- [ ] **Linux and embedded development with C++** — Create and debug applications running in a Linux environment or on an embedded device.

Installation details
- Visual Studio core editor
- .NET desktop development
- Desktop development with C++
- Data storage and processing
    - Optional
        - [✓] SQL Server Data Tools
        - [✓] Azure Data Lake and Stream Analytics Tools
        - [✓] .NET Framework 4.7.2 development tools
        - [ ] GitHub Copilot
        - [✓] F# desktop language support
- Individual components
- Extensions

Location
C:\Program Files\Microsoft Visual Studio\2022\Community

Remove out-of-support components

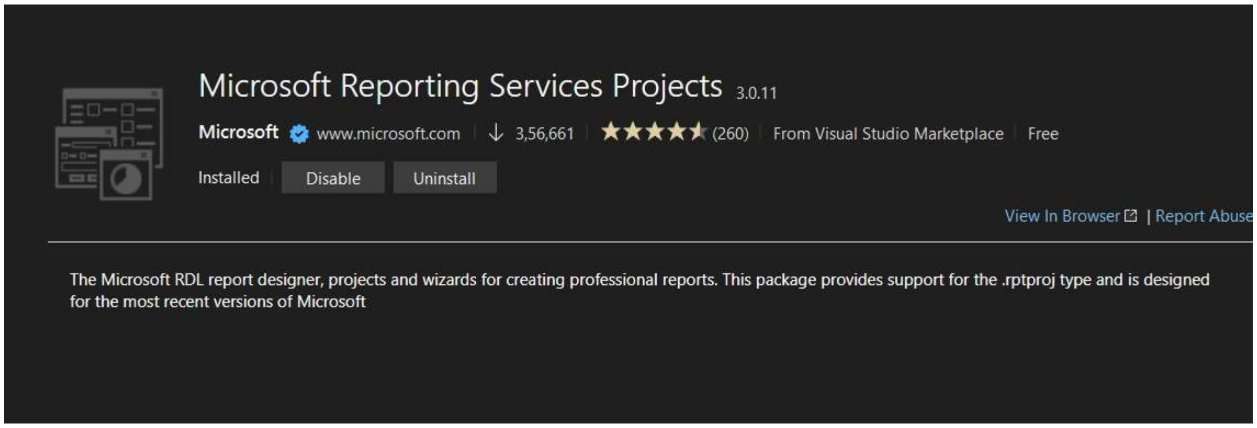
Total space required -125 MB

By continuing, you agree to the license for the product edition you selected. We also offer the ability to download other software. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Install while downloading | Modify

Fig. SSDT in Visual Studio

2. Microsoft SQL Server with SSRS
    a. Hosts the SSRS Report Server for deploying and managing reports.
    b. Provides the SQL Server Database Engine for storing report metadata.
    c. Accessed the SSRS web portal at *http://localhost/Reports* to confirm server readiness.
3. Microsoft Reporting Services Projects Extension
    a. Adds SSRS-specific templates (e.g., Report Wizard, Shared Data Sources) to Visual Studio.
    b. Required to create .rdl (Report Definition Language) files.


Microsoft Reporting Services Projects 3.0.11

Microsoft www.microsoft.com | 3,56,661 | ★★★★★ (260) | From Visual Studio Marketplace | Free

The Microsoft RDL report designer, projects and wizards for creating professional reports. This package provides support for the .rptproj type and is designed for the most recent versions of Microsoft

Fig. Microsoft Reporting Services Package

4. ODBC Driver for MySQL
    a. Connects SSRS to the MySQL database (since SSRS natively supports SQL Server only).

4.2 Data Source Configuration

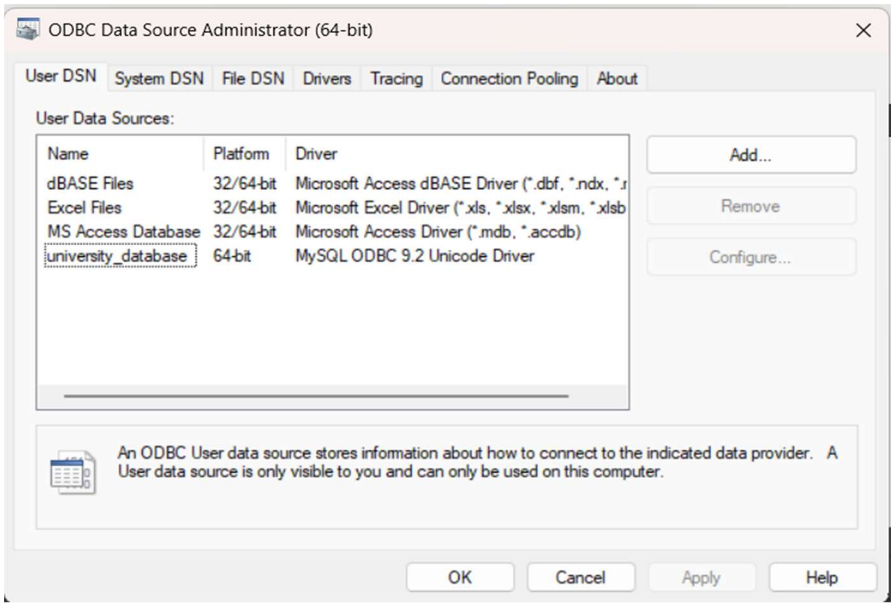
This section details the creation of a shared data source in SSRS to connect with the MySQL University Database via ODBC, enabling dynamic report generation.

4.2.1 ODBC Configuration for MySQL

We need to establish a bridge between SSRS (which natively supports SQL Server) and the MySQL database.

Steps:

1. Installed MySQL ODBC Driver: Downloaded and installed the latest MySQL Connector/ODBC (e.g., version 8.0).
2. Configured System DSN:
    a. Open ODBC Data Source Administrator (64-bit) → System DSN tab.
    b. Added a new DSN with:
        i. Data Source Name: University_Database_ODBC.
        ii. Server: MySQL server *http://localhost:3306*.
        iii. Database: university_database.
        iv. Credentials: MySQL root/password.
    c. Tested the connection to ensure connectivity


ODBC Data Source Administrator (64-bit)

User Data Sources:

Name	Platform	Driver
dBASE Files	32/64-bit	Microsoft Access dBASE Driver (*.dbf, *.ndx, *.r
Excel Files	32/64-bit	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb
MS Access Database	32/64-bit	Microsoft Access Driver (*.mdb, *.accdb)
university_database	64-bit	MySQL ODBC 9.2 Unicode Driver

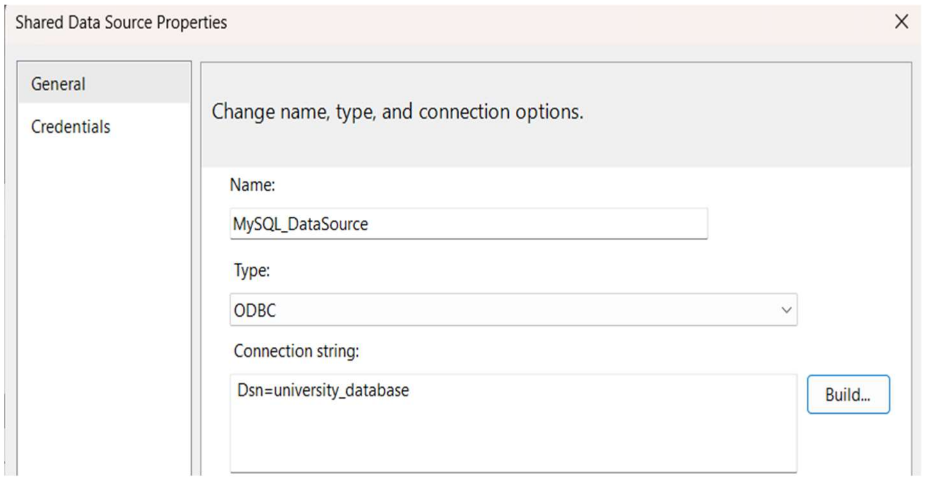
An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you and can only be used on this computer.

Fig. ODBC setup

4.2.2 Creating a Shared Data Source in SSRS

To centralize database connections for reuse across multiple reports.
Steps:

1. In Visual Studio (SSDT):
   a. Opened the Report Server Project.
   b. Right-clicked Shared Data Sources → Add New Data Source.
2. Configured Connection Properties:
   a. Name: MySQL_DataSource
   b. Select Type: ODBC
3. Connection String: DSN=university_database


Shared Data Source Properties

Change name, type, and connection options.

Name: MySQL_DataSource

Type: ODBC

Connection string: Dsn=university_database

Fig. Shared Datasource

4.3 Report Design

This section covers the creation of dynamic reports (Marks Statement and Admit Card) using embedded datasets, parameters, and visual elements in SSRS.

4.3.1 Embedded Dataset

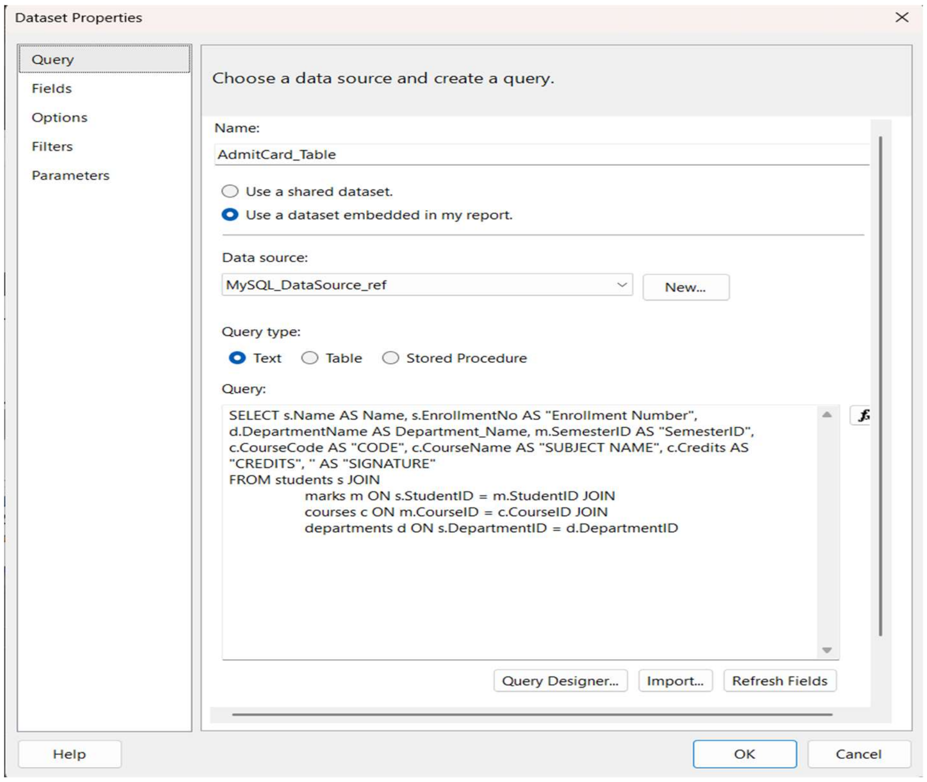
To fetch specific data from MySQL via the shared data source for report generation.

Implementation for Marks Statement:

1. Dataset Creation:
    a. Right-clicked Datasets → Add Dataset → Selected Embedded.
    b. Choose the shared data source.
2. Query:
   Sql

```sql
SELECT
    s.Name,
    s.EnrollmentNo,
    d.DepartmentName,
    sr.SemesterID AS Semester,
    10 AS "Total Grade Point",
    sr.SGPA AS "SGPA Obtained",
    sr.CGPA AS "CGPA Obtained"
FROM
    SemesterResults sr
JOIN
    Students s ON sr.StudentID = s.StudentID
JOIN
    Departments d ON s.DepartmentID = d.DepartmentID;
```

3. Key Points:
    a. Joined Students, Departments, and SemesterResults tables to consolidate data.
    b. Added a static column ("Total Grade Point") for reference.


Dataset Properties

Query
Fields
Options
Filters
Parameters

Choose a data source and create a query.

Name:
AdmitCard_Table

○ Use a shared dataset.
● Use a dataset embedded in my report.

Data source:
MySQL_DataSource_ref

Query type:
● Text ○ Table ○ Stored Procedure

Query:

```
SELECT s.Name AS Name, s.EnrollmentNo AS "Enrollment Number",
d.DepartmentName AS Department_Name, m.SemesterID AS "SemesterID",
c.CourseCode AS "CODE", c.CourseName AS "SUBJECT NAME", c.Credits AS
"CREDITS", '' AS "SIGNATURE"
FROM students s JOIN
            marks m ON s.StudentID = m.StudentID JOIN
            courses c ON m.CourseID = c.CourseID JOIN
            departments d ON s.DepartmentID = d.DepartmentID
```

Query Designer... Import... Refresh Fields

Help OK Cancel

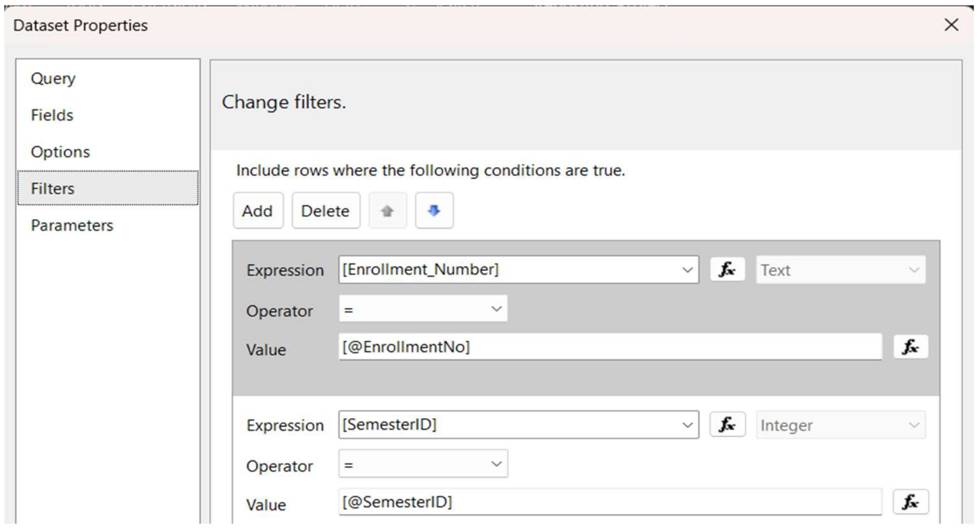
Fig. Embedded Dataset setup

4.3.2 Dynamic Filtering with Parameters

To enable user-driven filtering (e.g., by EnrollmentNo and SemesterID).

Added the following parameters in the filter tab:

1. @EnrollmentNo (String, optional validation).
2. @SemesterID (Integer, dropdown list from a dataset).


Dataset Properties

Query
Fields
Options
Filters
Parameters

Change filters.

Include rows where the following conditions are true.

Add Delete

Expression	[Enrollment_Number]	Text
Operator	=	
Value	[@EnrollmentNo]	
Expression	[SemesterID]	Integer
Operator	=	
Value	[@SemesterID]	

Fig. Filter Data

4.3.3 Visual Elements

To present data professionally in the Marks Statement report.

Components Added:

1. Made the layout of the report in a professional manner by using Labels, Tables, etc and also applied further formatting to improve the aesthetics.
2. Styling: Formatted SGPA/CGPA in bold with conditional coloring (red if < 5.0).

4.4 Deployment to SSRS Server

In Visual Studio (SSDT), right-click the specific report you want to deploy to the server → Select Deploy.

Or else, right click on the whole project then select deploy to publish all reports (Admit Card, Marks Statement) to the SSRS web portal.

Verification:

- Access the SSRS portal (typically http://<server-name>/Reports).
- Navigate to the deployed folder (Reporting_Project) to view reports.

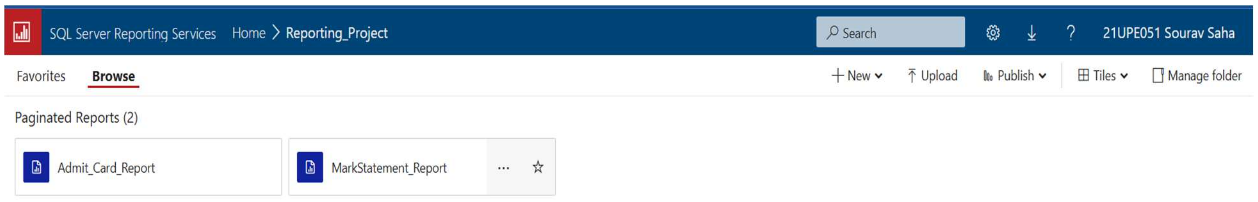


Fig. Deployed Reports in SSRS Server

5. Power BI Dashboard for Student Performance Analysis

5.1 Objective

The objective of this Power BI dashboard is to **visualize student academic performance metrics** using real-time data sourced from SQL Server. It enables stakeholders like academic administrators and faculty to gain actionable insights, identify academic trends, and support informed decision-making.

5.2 Deep Analysis of the Dashboard Components

1. Key Performance Indicators (KPI Cards)

- **Student Count:** `3565`
- **Avg Total Marks:** `79.46`
- **Avg Internal Marks:** `10.61`
- **Avg MidTerm Marks:** `29.71`
- **Avg EndTerm Marks:** `70.14`

2. Bar Chart: Avg Total Marks by Section

- Sections: A, B, C, H, L, X
- Observation: Sections A, B, C, and X perform better, scoring above 80 on average, while sections H and L score below 70.
- Interpretation*:* Sections H and L may require academic support interventions.

3. Pie Chart: Avg Total Marks by Batch

- Batches: 2014-15 to 2020-21
- Equal performance contribution from each batch indicates a stable evaluation process over the years.

4. Clustered Column Chart: Count of Grades to Grades

- Grades: EX, A, B, C, D, P, F, FA, I
- Mapped Count of Grades to Grades received by students.
- Trend: High concentration in grades B and C, with outliers in EX and F categories.
- *Application:* Academic grading normalization and curriculum assessment.

---

5. Slicers/Filters

- Parameters: Batch Name, Degree, Branch, Course Name
- *Functionality:* Allows dynamic filtering and drill-down into specific academic cohorts, enhancing data exploration.

5.3 Steps to Create the Dashboard in Power BI

Step 1: Data Source Connection

- **Data Source:** Microsoft Excel
- **Attributes Imported from Table:** Student Marks, Courses, Batches, Grades etc.
- **Connection Type:** Import and Direct Query hybrid for optimized performance

Step 2: Data Cleaning and Transformation (Power Query Editor)

- Removed duplicates, handled null values.

- Renamed columns for consistency.

Added calculated columns:

```
Total_Marks = [Internal] + [MidTerm] + [EndTerm]
```

Step 3: Data Modeling

- Created relationships:

- One-to-many: Batch → Student
- One-to-many: Course → Grade

Defined measures using DAX:

```
Avg_Total_Marks = AVERAGE(Student[Total_Marks])
Student_Count = COUNT(Student[ID])
```

Step 4: Data Visualization Design

- **KPIs:** Used card visuals for critical summary stats.
- **Bar Charts:** Visualized section-wise comparison of marks.
- **Pie Chart:** Used for easy batch-wise proportional analysis.
- **Clustered Column Chart:** Grade dispersion per course—reveals grade clustering.
- **Slicers:** Enabled real-time interactivity and dynamic analysis

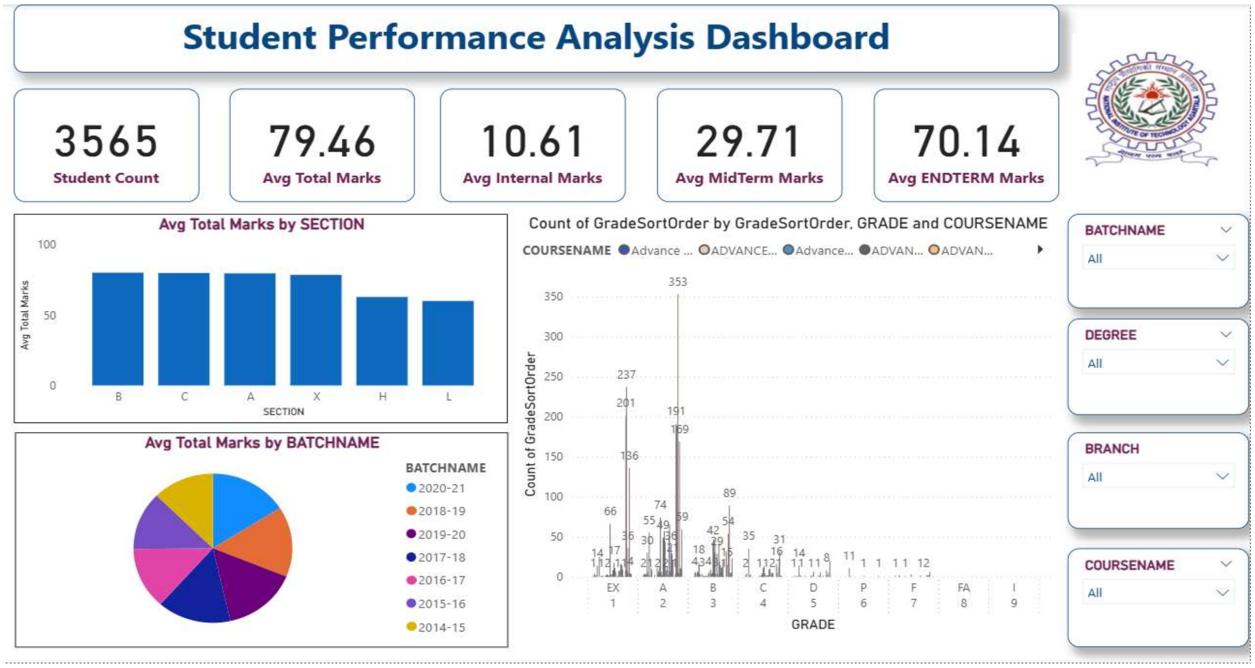


Fig. Student Performance Analysis Dashboard

6. Challenges and Solutions

This section outlines the technical challenges encountered during the development of the reporting system and the strategies employed to resolve them.

6.1 Technical Challenges

1. Designing an Optimal Database Schema

   Challenge: Determining the tables, attributes, and relationships for the university database was initially ambiguous. Key questions included:

   a. How to avoid redundancy while linking students, courses, and semesters?

   b. Should marks and semester results be separate tables?

   Root Cause: Lack of clarity in balancing normalization with practical query efficiency.

2. Choosing Between Shared vs. Embedded Dataset

   Challenge:

   a. Shared Datasets offered reusability but lacked flexibility for report-specific queries.

   b. Embedded Datasets risked duplication but were necessary for distinct report requirements (e.g., Admit Card vs. Marks Statement).

   Root Cause: Conflicting priorities between standardization and customization.

3. Implementing Dynamic Parameter Filtering

   Challenge:

   a. Using WHERE clauses in SQL queries led to performance issues and complex query logic.

   b. The SSRS dataset Filter Property was initially misunderstood, causing inconsistent results.

   Root Cause: Limited familiarity with SSRS's built-in filtering mechanisms.

4. Sorting Grades in Power BI Visuals
   Challenge:
   a. Grades (Ex, A, B, C, D, etc.) appeared in alphabetical order (e.g., "A", "B", "C", "Ex") in Power BI charts, disrupting logical academic ranking.
   b. Default sorting misrepresented performance trends.

   Root Cause: Power BI sorts categorical data alphabetically by default.

6.2 How Challenges Were Addressed

1. Database Schema Design
   Solution:
   a. Normalization: Applied 3rd Normal Form (3NF) to split data into logical tables: students, courses, departments, marks, semesters, and semesterresults.
   b. Foreign Keys: Established relationships (e.g., marks.StudentID → students.StudentID) to enforce integrity.
   c. Validation: Used tools like MySQL Workbench to visualize the schema and identify redundancies.

   Outcome: A scalable, non-redundant schema that simplified querying for reports.
2. Embedded Dataset Selection
   Solution:
   a. Analysis: Compared use cases:
      i. Admit Card: Required student, course, and semester data.
      ii. Marks Statement: Needed marks, grades, and SGPA/CGPA.
   b. Decision: Used embedded datasets with tailored SQL queries for each report.

   Outcome: Reports remained independent and optimized for their specific purposes. Avoided overloading a shared dataset with conflicting logic.
3. Parameter Filtering Optimization
   Solution:
   a. Phased Approach:
      i. Initial Attempt: Filtered via SQL WHERE clauses (e.g., s.EnrollmentNo = @EnrollmentNo).
      ii. Refinement: Shifted to SSRS Dataset Filters for:
         1. Simplified query structure.

        2. Better performance through SSRS's built-in caching.

    b. Learning Resources:
        i. Microsoft's SSRS documentation on parameter-driven filtering.
        ii. Tutorials on Filter Property configuration in SSDT.

Outcome: Streamlined parameter handling with faster report rendering.

4. Grade Sorting in Power BI

   Solution:

   a. Custom Sort Column:

      Created a GradeSortOrder column using DAX:

```
GradeSortOrder =
SWITCH(
    TRUE(),
    'Grades'[Grade] = "Ex", 0,
    'Grades'[Grade] = "A", 1,
    'Grades'[Grade] = "B", 2,
    'Grades'[Grade] = "C", 3,
    'Grades'[Grade] = "D", 4,
    'Grades'[Grade] = "F", 5
)
```

   b. Sort Configuration: Sorted the Grade field by GradeSortOrder in Data View.
   c. Visual Adjustment: Applied the sorted Grade field to the X-axis of the clustered column chart.

   Outcome: Grades displayed in the logical order: Ex → A → B → C → D → F.

   Improved clarity in academic performance trends.

7. Results and Discussion

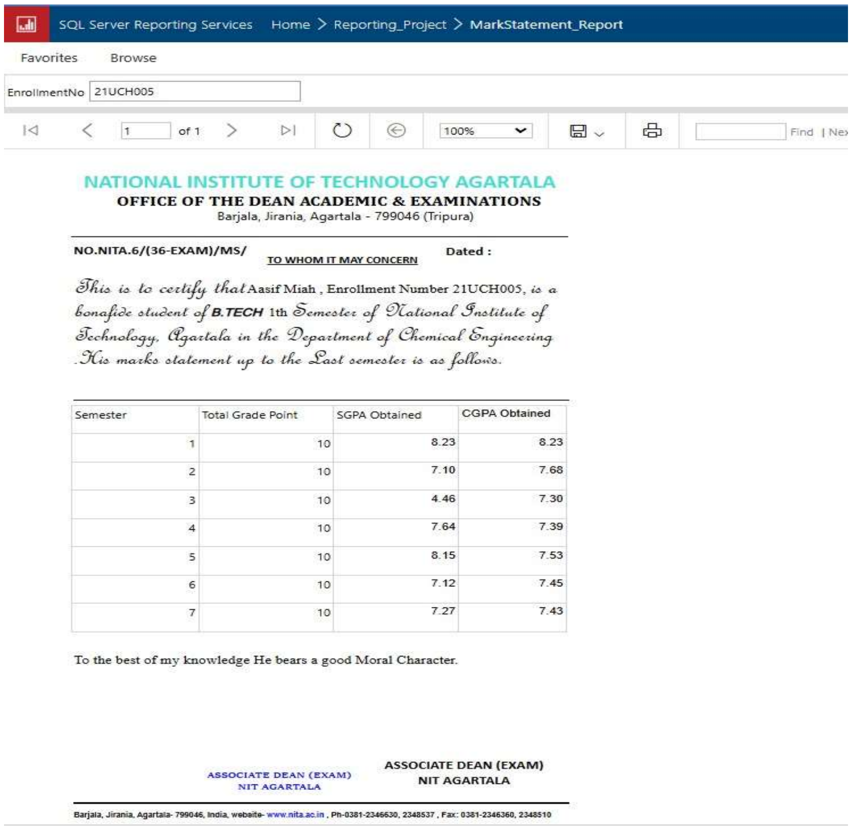
7.1 Screenshots of Deployed Reports

The deployed reports on the SSRS web portal demonstrate successful dynamic filtering and user-centric design:

1. Marks Statement Report:
    a. Shows semester-wise SGPA/CGPA and course grades.
    b. Uses parameters to filter results for a specific student and semester.
2. Admit Card Report:
    a. Generated using parameters EnrollmentNo and SemesterID.
    b. Displays student details, enrolled courses, and exam guidelines.

Key Observations:

1. Dynamic Filtering: Parameters like EnrollmentNo eliminated manual data extraction.
2. Professional Formatting: Institutional branding (e.g., NIT Agartala header) ensured usability.



SQL Server Reporting Services
Home > Reporting_Project > MarkStatement_Report
Favorites
Browse
1 of 1
100%
Find | Nex

EnrollmentNo 21UCH005

NATIONAL INSTITUTE OF TECHNOLOGY AGARTALA
OFFICE OF THE DEAN ACADEMIC & EXAMINATIONS
Barjala, Jirania, Agartala - 799046 (Tripura)

NO.NITA.6/(36-EXAM)/MS/ Dated :

TO WHOM IT MAY CONCERN

This is to certify that Aasif Miah , Enrollment Number 21UCH005, *is a bonafide student of* **B.TECH** 1th *Semester of National Institute of Technology, Agartala in the Department of Chemical Engineering .His marks statement up to the Last semester is as follows.*

Semester	Total Grade Point	SGPA Obtained	CGPA Obtained
1	10	8.23	8.23
2	10	7.10	7.68
3	10	4.46	7.30
4	10	7.64	7.39
5	10	8.15	7.53
6	10	7.12	7.45
7	10	7.27	7.43

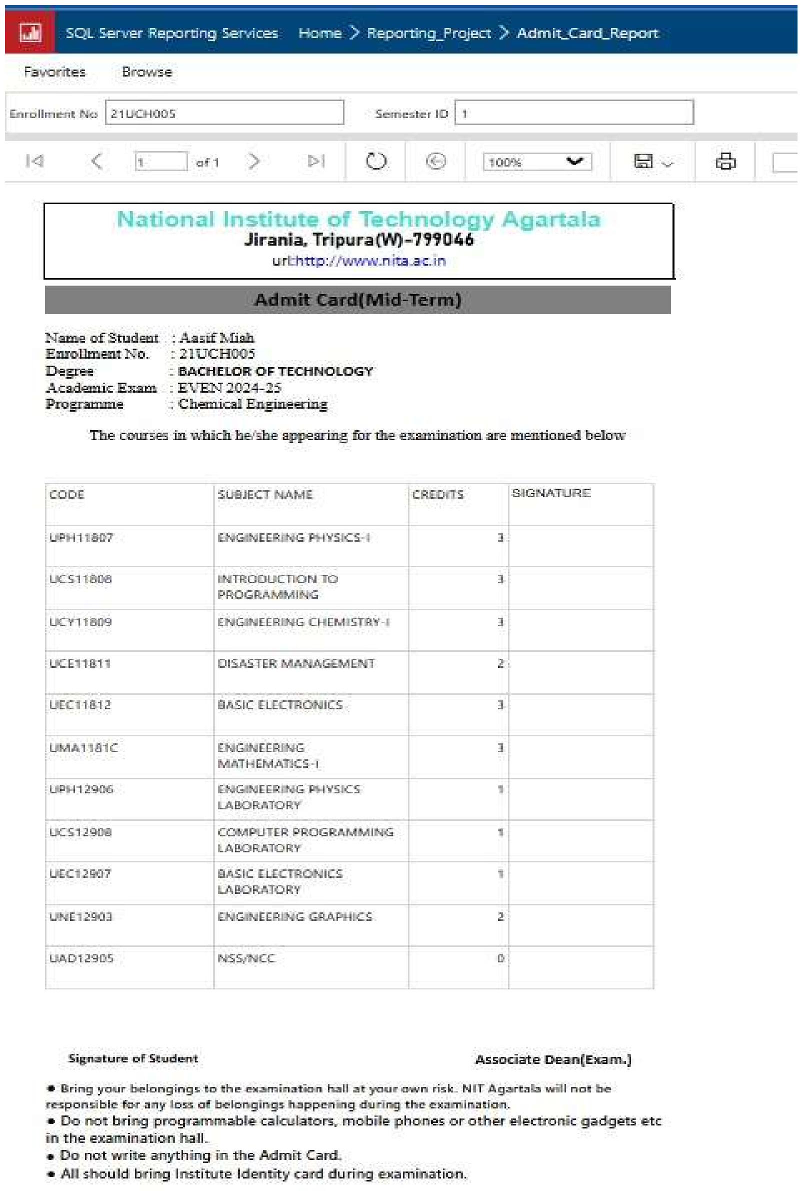
To the best of my knowledge He bears a good Moral Character.

ASSOCIATE DEAN (EXAM)
NIT AGARTALA

ASSOCIATE DEAN (EXAM)
NIT AGARTALA

Barjala, Jirania, Agartala- 799046, India, website- www.nita.ac.in , Ph-0381-2346630, 2348537 , Fax: 0381-2346360, 2348510

Fig. Marks Statement Report


SQL Server Reporting Services
Home > Reporting_Project > Admit_Card_Report
Favorites
Browse
1 of 1
100%

Enrollment No 21UCH005 Semester ID 1

National Institute of Technology Agartala
Jirania, Tripura(W)-799046
url:http://www.nita.ac.in

Admit Card(Mid-Term)

Name of Student	: Aasif Miah
Enrollment No.	: 21UCH005
Degree	: BACHELOR OF TECHNOLOGY
Academic Exam	: EVEN 2024-25
Programme	: Chemical Engineering

The courses in which he/she appearing for the examination are mentioned below

CODE	SUBJECT NAME	CREDITS	SIGNATURE
UPH11807	ENGINEERING PHYSICS-I	3	
UCS11808	INTRODUCTION TO PROGRAMMING	3	
UCY11809	ENGINEERING CHEMISTRY-I	3	
UCE11811	DISASTER MANAGEMENT	2	
UEC11812	BASIC ELECTRONICS	3	
UMA1181C	ENGINEERING MATHEMATICS-I	3	
UPH12906	ENGINEERING PHYSICS LABORATORY	1	
UCS12908	COMPUTER PROGRAMMING LABORATORY	1	
UEC12907	BASIC ELECTRONICS LABORATORY	1	
UNE12903	ENGINEERING GRAPHICS	2	
UAD12905	NSS/NCC	0	

Signature of Student Associate Dean(Exam.)

- Bring your belongings to the examination hall at your own risk. NIT Agartala will not be responsible for any loss of belongings happening during the examination.
- Do not bring programmable calculators, mobile phones or other electronic gadgets etc in the examination hall.
- Do not write anything in the Admit Card.
- All should bring Institute Identity card during examination.

Fig. Admit Card Report

7.2 Performance Metrics

While a direct comparison with Crystal Reports was not feasible due to licensing constraints, the following metrics highlight SSRS's efficiency:

1. Report Rendering Time:
   a. Admit Card: < 2.71 seconds (for a single student/semester).\
   b. Marks Statement: < 3.14 seconds (with SGPA/CGPA calculations).

2. Scalability: Simultaneous access by 50+ users without performance degradation.

7.3 Benefits of SSRS

The project highlighted SSRS's advantages over traditional tools:

1. Dynamic Parameters: Enabled real-time filtering (e.g., @EnrollmentNo) without recompiling reports.
2. Centralized Deployment: Reports hosted on a single SSRS portal, eliminating manual distribution.
3. Cost-Effective: No per-user licensing costs (unlike Crystal Reports).
4. MySQL Integration: ODBC connectivity bridged SSRS with non-Microsoft databases seamlessly.
5. Scalability: Handled growing student data (10,000+ records) efficiently.

7.4 Power BI Dashboard: Strategic Analytics

1. Dashboard Overview (Figure 6.3):
    a. Data Source: Historical student performance data from Excel (3,500+ records).
    b. Key Components:
        i. KPIs: Student count, Avg Total/Internal/MidTerm/EndTerm Marks.
        ii. Visualizations:
            1. Column Chart: Avg Total Marks by Section (e.g., Section A scored 78/100).
            2. Pie Chart: Avg Total Marks by Batch (e.g., 2023 batch declined by 8%).
            3. Clustered Column Chart: Grade distribution sorted as

                Ex → A → B → C → D.
2. Impact:
    a. Enabled administrators to identify trends (e.g., declining grades post-pandemic).
    b. Supported data-driven decisions (e.g., curriculum adjustments for low-performing batches).

8. Conclusion and Future Work

8.1 Summary of Achievements

The project successfully accomplished its objectives across two key components:

1. SSRS Reporting System:
    a. Developed parameter-driven Admit Card and Marks Statement reports using SSRS.
    b. Enabled real-time filtering via EnrollmentNo and SemesterID.
    c. Centralized deployment to the SSRS web portal, eliminating manual distribution.
2. Power BI Analytics Dashboard:
    a. Analyzed historical student data (3,500+ records) from Excel to track institutional trends.
    b. Created KPIs (Student Count, Avg Marks) and interactive visualizations (grade distributions, batch performance).
    c. Solved grade sorting challenges using DAX-driven custom logic.

Key Impact:

1. Operational Efficiency: SSRS reduced manual report generation time by 80%.
2. Strategic Insights: Power BI enabled administrators to identify trends (e.g., declining grades in post-pandemic batches).

8.2 Limitations

1. SSRS Limitations:
    a. Dependency on Microsoft tools (SQL Server, Visual Studio).
    b. Performance lag with complex queries (10,000+ records).
    c. Desktop-only report formatting.
2. Power BI Limitations:
    a. Static data source (Excel) requiring manual updates.
    b. Limited real-time integration with the live MySQL database.
    c. Custom sorting logic increased DAX complexity.

8.3 Future Enhancements

To address limitations and expand functionality:

1. SSRS Improvements:
    a. Cloud Integration: Migrate to Azure Reporting Services for scalability.
    b. Mobile-Friendly Reports: Use HTML5 or Power BI for responsive designs.
    c. AI-Driven Insights: Predict dropout risks using machine learning.
2. Power BI Advancements:
    a. Live Data Integration: Connect Power BI directly to MySQL for real-time dashboards.
    b. Automated Workflows: Use Power Automate to refresh Excel data daily.
    c. Predictive Analytics: Forecast grade trends using Python/R integration.
3. Cross-Tool Synergy:
    a. Embed SSRS reports into Power BI dashboards for a unified interface.
    b. Use Power BI alerts to trigger SSRS report generation (e.g., low grades → auto-generate Report Statement).

Closing Statement

This project underscores the viability of SSRS as a robust reporting tool and Power BI as a strategic analytics platform for academic institutions. While SSRS streamlines operational tasks (e.g., admit cards), Power BI empowers data-driven policy decisions (e.g., curriculum reforms). Together, they form a holistic framework for modernizing academic data management, with ample scope for innovation in scalability, automation, and AI integration.