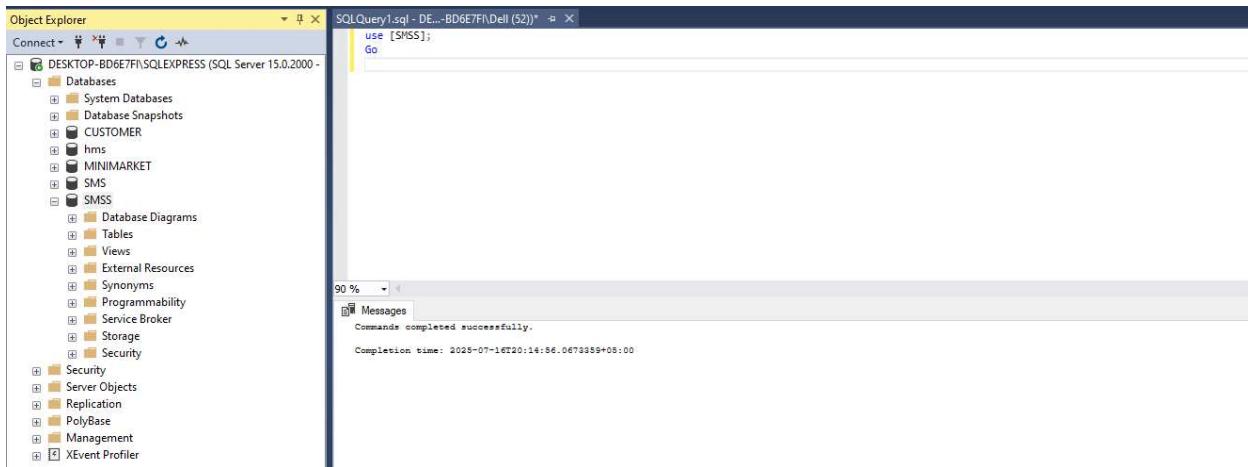


# Database Project

## STUDENT MANAGEMENT SYSTEM

### Creating Database

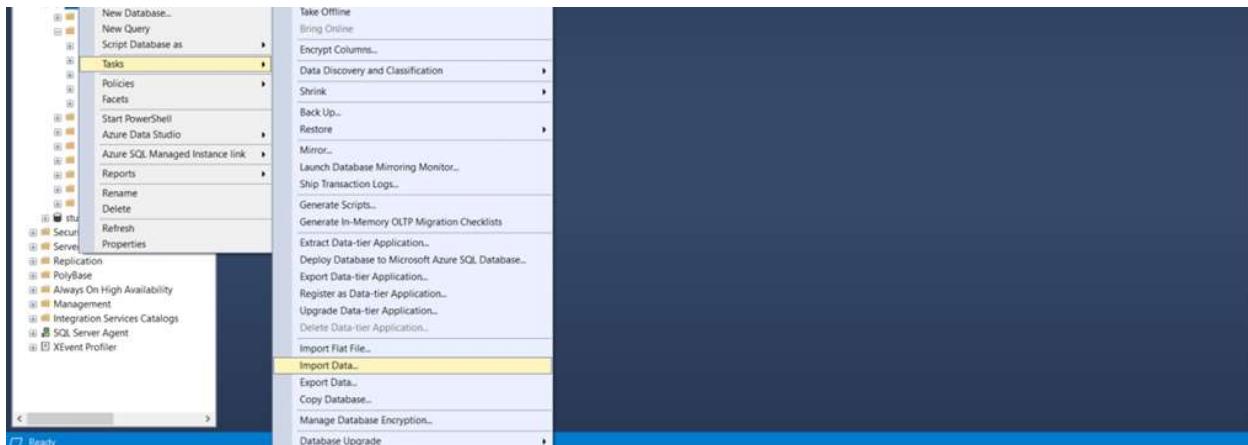


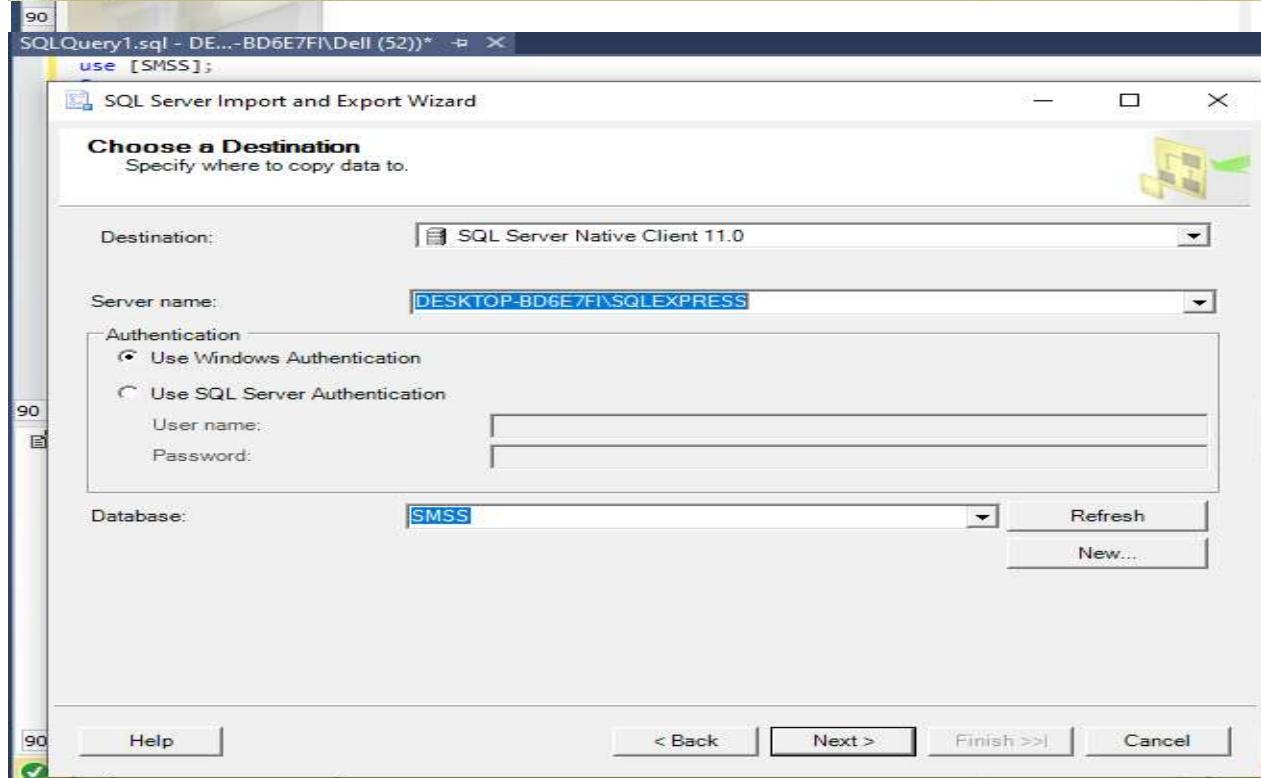
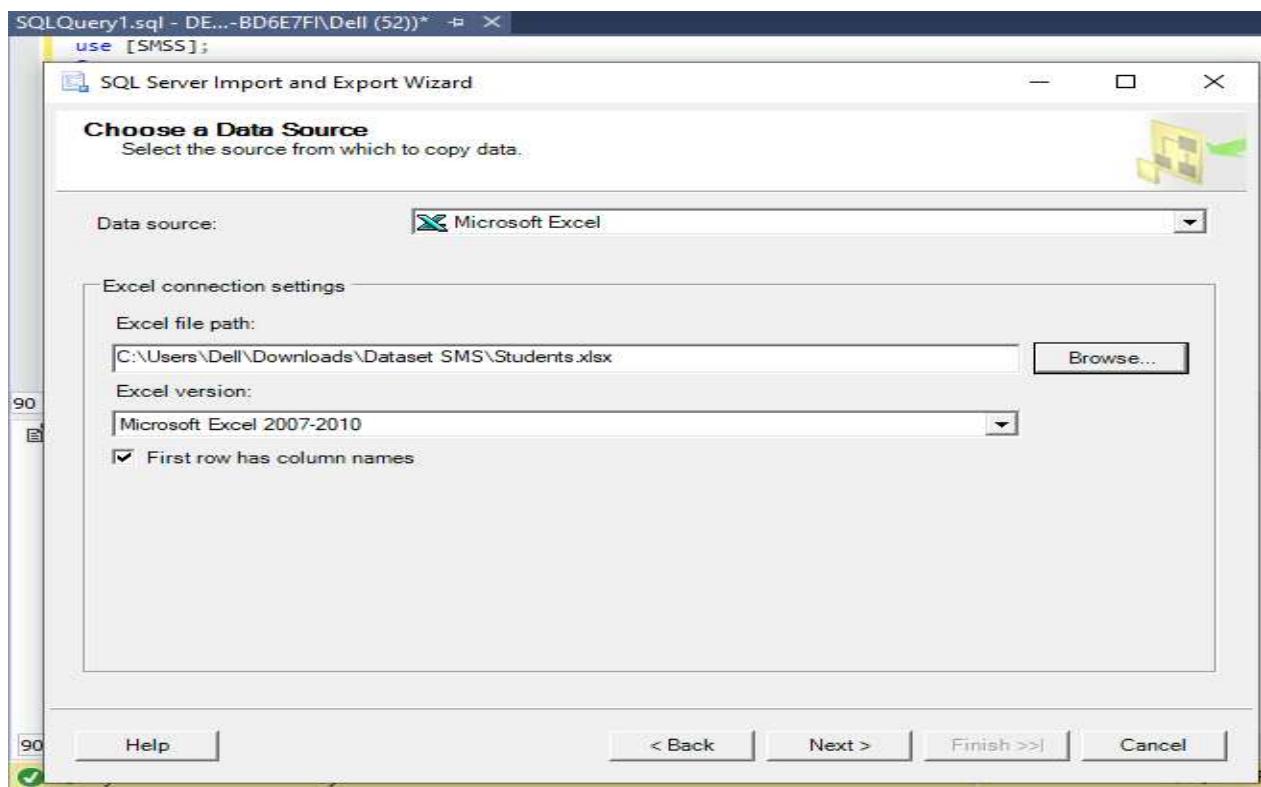
The screenshot shows the SQL Server Object Explorer window. Under the 'DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)' connection, a new database named 'SMSS' has been created. The 'Databases' node now includes 'System Databases', 'Database Snapshots', 'CUSTOMER', 'hms', 'MINIMARKET', 'SMS', and 'SMSS'. The 'SMSS' node is expanded, showing 'Database Diagrams', 'Tables', 'Views', 'External Resources', 'Synonyms', 'Programmability', 'Service Broker', 'Storage', and 'Security'.

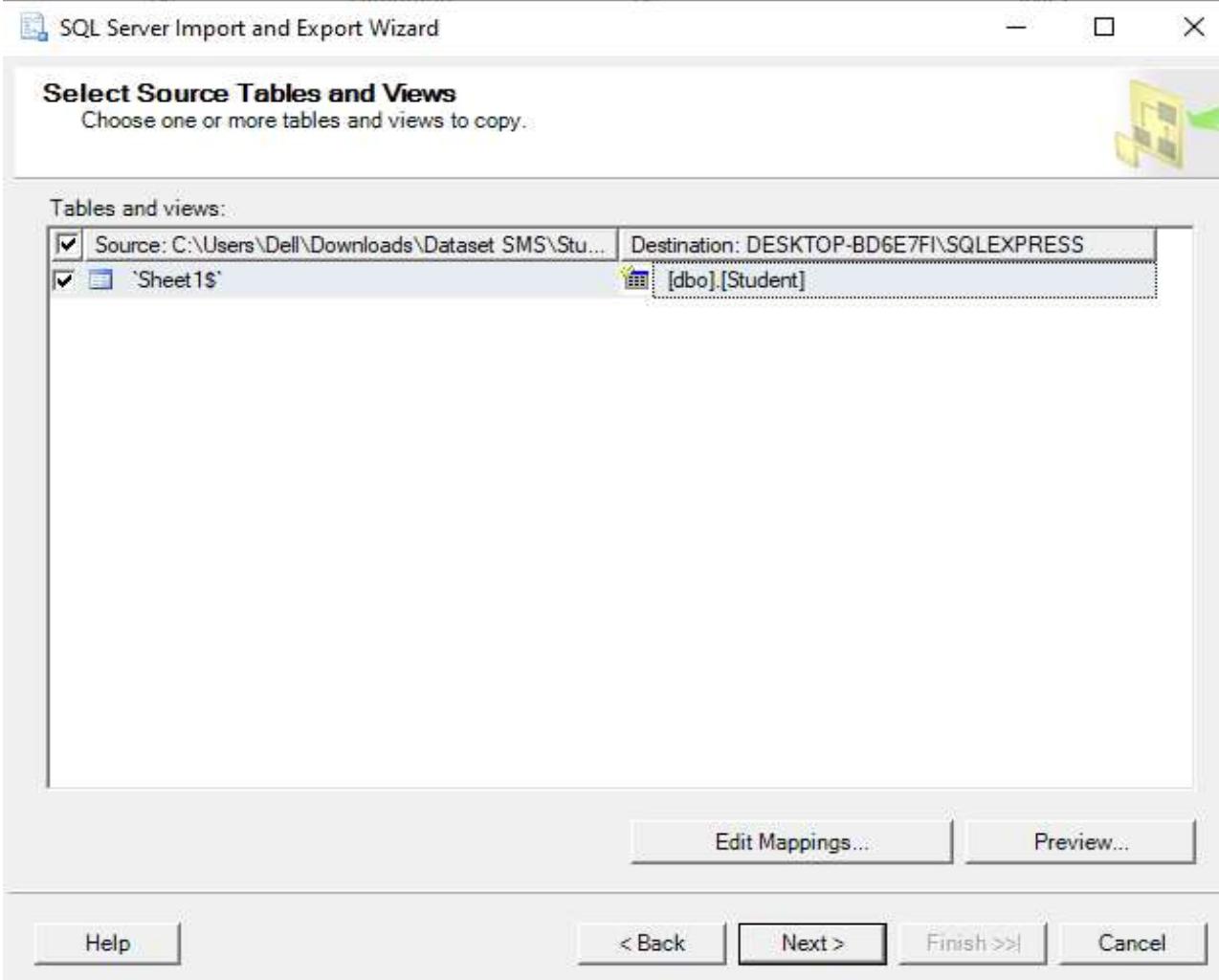
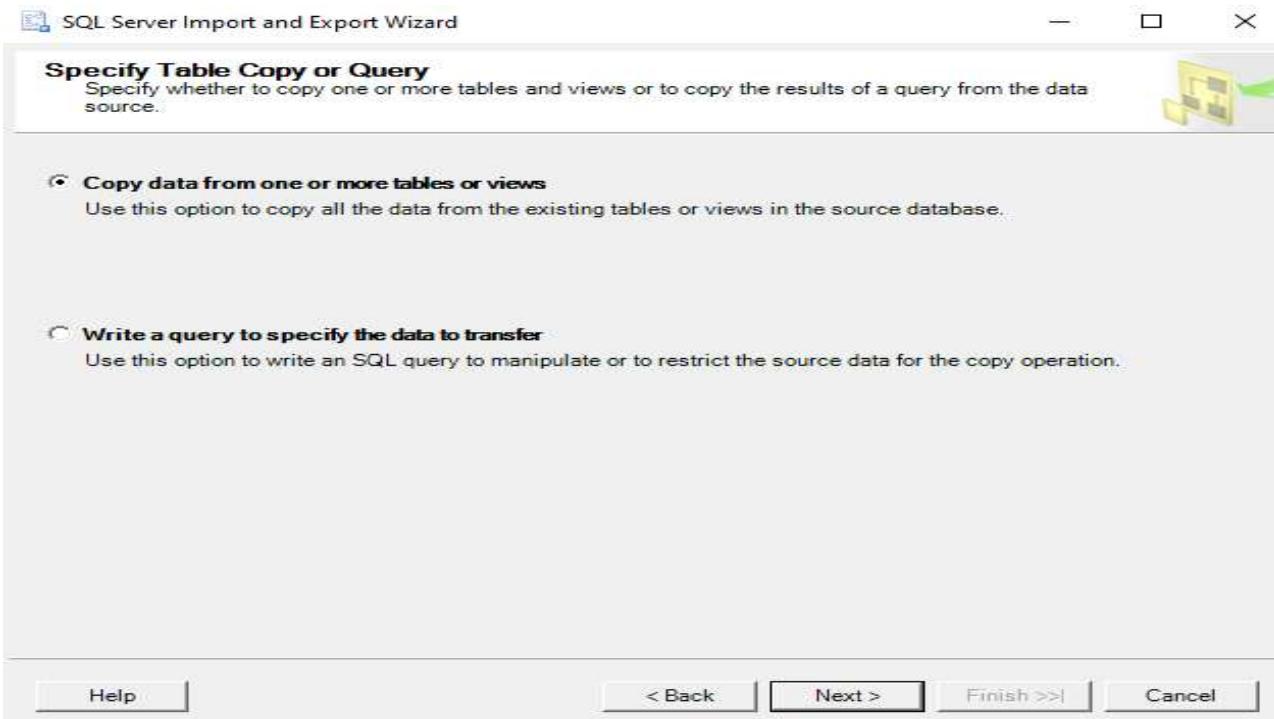
```
use [SMSS];
Go
```

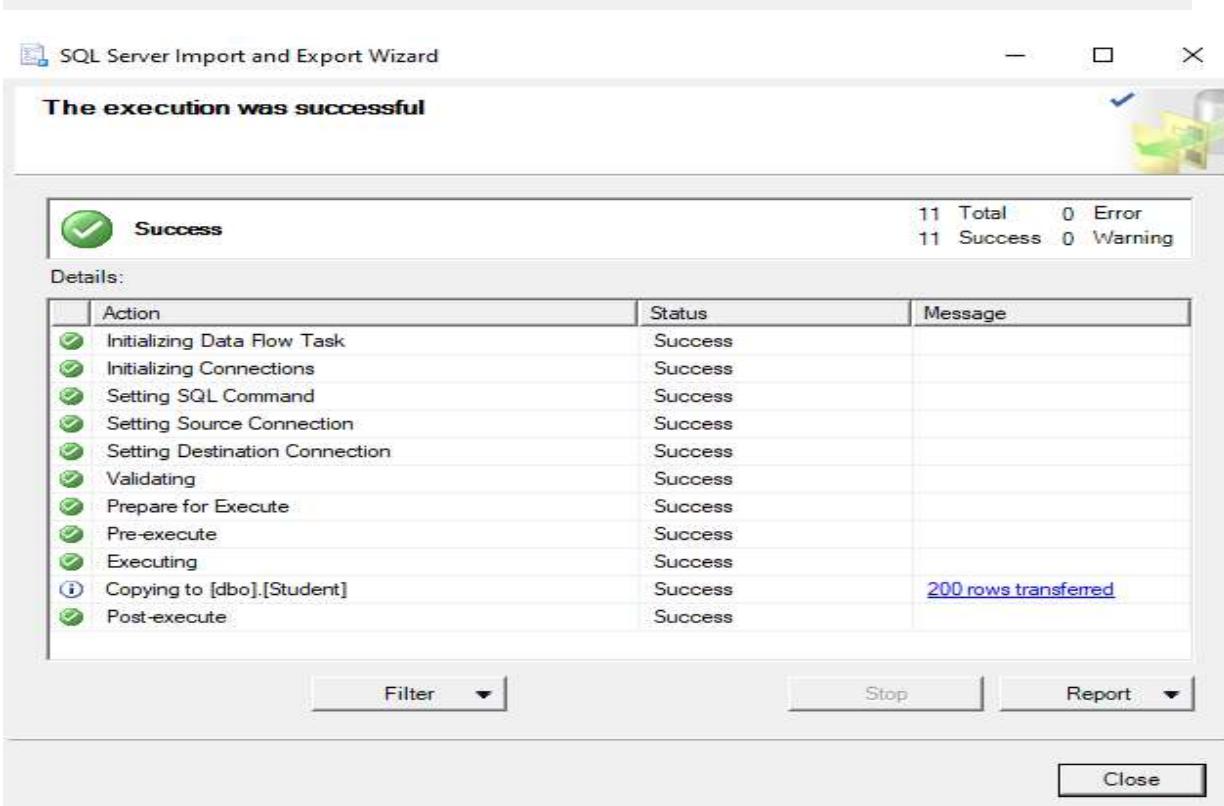
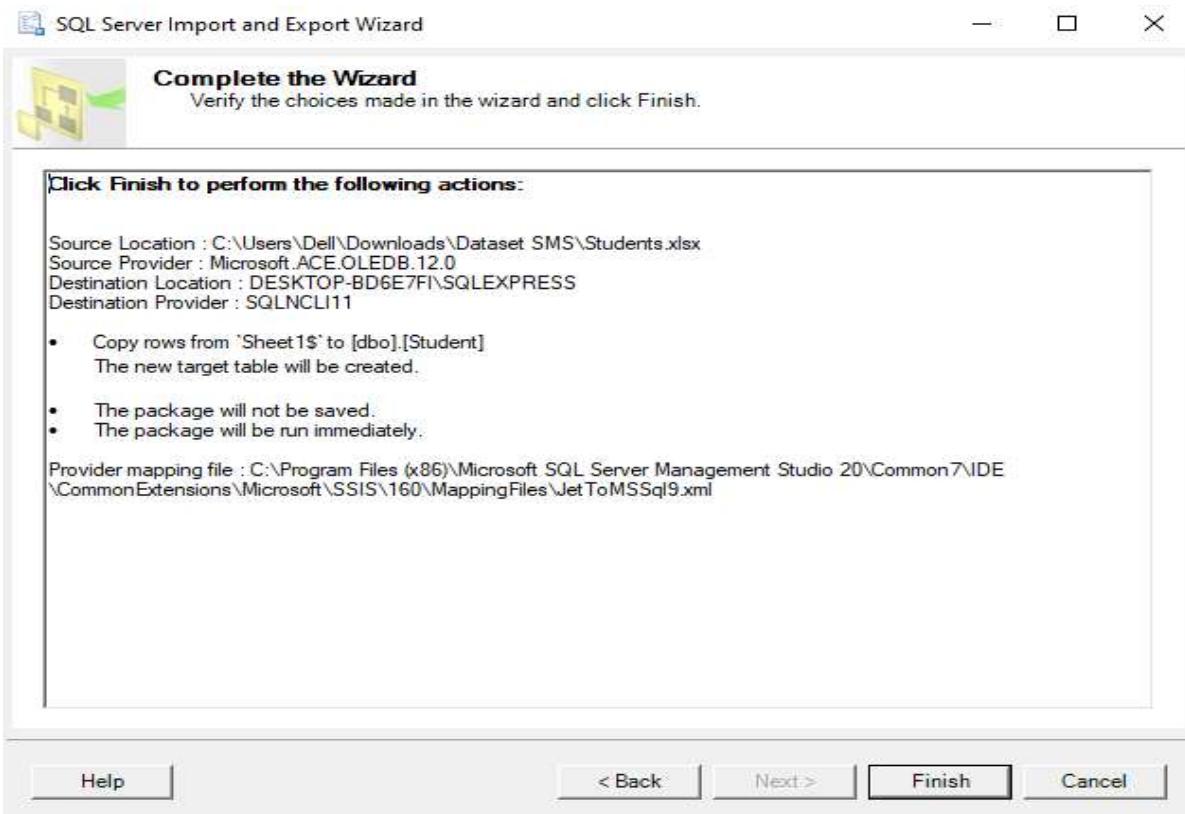
Messages  
Commands completed successfully.  
Completion time: 2025-07-16T20:14:56.0673359+05:00

### Import File Data









The method of importing all the other tables from excel to SQL is same as above.

## Table view

The screenshot shows the SSMS interface. On the left is the Object Explorer, which lists the database structure including databases like DESKTOP-BD6E7F1\SQLEXPRESS, tables such as CUSTOMER, hms, MINIMARKET, SMS, and SMSS, and various system objects. The main area contains two tabs: 'SQLQuery2.sql - DE...-BD6E7F1\... (54)' and 'SQLQuery1.sql - DE...-BD6E7F1\... (52)\*'. The 'SQLQuery1' tab displays a query result set:

```
SELECT TOP (1000) [StudentID]
,[Name]
,[Age]
,[Email]
,[Department_name]
,[GPA]
,[GraduationYear]
,[Course_name]
,[gender]
,[final_result]
,[Department_id]
FROM [SMSS].[dbo].[Student]
```

The results grid shows 11 rows of student data:

StudentID	Name	Age	Email	Department_name	GPA	GraduationYear	Course_name	gender	final_result	Department_id	
1	3336	David Palmer	19	sean43@hotmail.com	Mathematics	3.16	2026	Computer Science	Other	Fail	S001
2	8774	Andrew Roach	23	vbecker@harvey.com	Chemistry	3.75	2027	Arts	Male	Pass	S002
3	1396	Jonathan Gonzalez	22	hollydavis@gmail.com	Physics	2.95	2027	Computer Science	Male	Fail	S003
4	6716	Kenneth Morrow	24	ganderson@wheeler-atkins.info	Physics	3.55	2029	Arts	Male	Pass	S004
5	8830	Kaitlyn Martinez	18	hayesdiane@gmail.com	Chemistry	2.29	2025	Computer Science	Other	Pass	S005
6	5305	Tiffany Wolf	23	qanderson@taylor.com	Mathematics	3.3	2029	Engineering	Male	Fail	S006
7	5048	James Reyes	20	drodriguez@nguyen-cooper.info	Chemistry	2.44	2029	Engineering	Female	Pass	S007
8	5986	Samantha Sellers	20	michelle27@hubbard-webster.com	Mathematics	3.44	2025	Computer Science	Other	Pass	S008
9	8721	Michael Kim	25	jacksonhannah@miles.com	Computer Science	3.27	2027	Computer Science	Other	Pass	S009
10	6622	Emily Davis	18	george02@hotmail.com	Physics	3.09	2030	Engineering	Other	Fail	S010
11	3254	Adam Evans	20	bryantmargaret@hernandez.com	Physics	3.42	2026	Arts	Other	Pass	S011

## Functional Dependencies

For Student Table:

The screenshot shows the SSMS interface with the same Object Explorer as the previous screenshot. The 'SQLQuery1' tab displays a query result set:

```
use [SMSS];
Go
SELECT * FROM Student;
Go
```

The results grid shows the same 11 rows of student data as the previous screenshot:

StudentID	Name	Age	Email	Department_name	GPA	GraduationYear	Course_name	gender	final_result	Department_id	
1	3336	David Palmer	19	sean43@hotmail.com	Mathematics	3.16	2026	Computer Science	Other	Fail	S001
2	8774	Andrew Roach	23	vbecker@harvey.com	Chemistry	3.75	2027	Arts	Male	Pass	S002
3	1396	Jonathan Gonzalez	22	hollydavis@gmail.com	Physics	2.95	2027	Computer Science	Male	Fail	S003
4	6716	Kenneth Morrow	24	ganderson@wheeler-atkins.info	Physics	3.55	2029	Arts	Male	Pass	S004
5	8830	Kaitlyn Martinez	18	hayesdiane@gmail.com	Chemistry	2.29	2025	Computer Science	Other	Pass	S005
6	5305	Tiffany Wolf	23	qanderson@taylor.com	Mathematics	3.3	2029	Engineering	Male	Fail	S006
7	5048	James Reyes	20	drodriguez@nguyen-cooper.info	Chemistry	2.44	2029	Engineering	Female	Pass	S007
8	5986	Samantha Sellers	20	michelle27@hubbard-webster.com	Mathematics	3.44	2025	Computer Science	Other	Pass	S008
9	8721	Michael Kim	25	jacksonhannah@miles.com	Computer Science	3.27	2027	Computer Science	Other	Pass	S009
10	6622	Emily Davis	18	george02@hotmail.com	Physics	3.09	2030	Engineering	Other	Fail	S010
11	3254	Adam Evans	20	bryantmargaret@hernandez.com	Physics	3.42	2026	Arts	Other	Pass	S011

Checking one by one the repeating values in a column with the query as follows:

```
SELECT Column_name
FROM Table_name
GROUP BY Column_name
HAVING COUNT(Column_name) > 1;
```

## [Attribute: StudentID]

The screenshot shows the SSMS interface with the Object Explorer on the left and a query window on the right. The query window contains the following T-SQL code:

```
use [SMSS];
Go
SELECT * FROM Student;
GO
SELECT StudentID
FROM Student
GROUP BY StudentID
HAVING COUNT(StudentID) > 1;
```

The results pane shows a single column named "StudentID" with the value "1".

## [Attribute: Name]

The screenshot shows the SSMS interface with the Object Explorer on the left and a query window on the right. The query window contains the following T-SQL code:

```
SELECT * FROM Student;
GO
SELECT Name
FROM Student
GROUP BY Name
HAVING COUNT(Name) > 1;
```

The results pane shows a column named "Name" with four rows: "Carrie Hall", "Christie Smith", "James Reyes", and "Lisa Brown".

## [Attribute: Age]

The screenshot shows the SSMS interface with the Object Explorer on the left and a query window on the right. The query window contains the following T-SQL code:

```
SELECT * FROM Student;
GO
SELECT Name
FROM Student
GROUP BY Name
HAVING COUNT(Name) > 1;
SELECT Age
FROM Student
GROUP BY Age
HAVING COUNT(Age) > 1;
```

The results pane shows a column named "Age" with eight rows: 18, 19, 20, 21, 22, 23, 24, and 25.

## [Attribute: Email]

**[Attribute: Name]**

```

Object Explorer
Connect > DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
      dbo.Student
        Columns
        Keys
        Constraints
        Triggers
        Indexes
        Statistics
      Views
      External Resources

SQLQuery4.sql - DESKTOP-BD6E7F\Dell (59)*
SELECT * FROM Student
GO
SELECT Name
FROM Student
GROUP BY Name
HAVING COUNT(Name) > 1;
SELECT Age
FROM Student
GROUP BY Age
HAVING COUNT(Age) > 1;
SELECT Email
FROM Student
GROUP BY Email
HAVING COUNT(Email) > 1;

Results Messages
Email
1 drodriguez@nguyen-cooper.info
2 rhal@gmail.com

```

### [Attribute: Department\_name]

```

Object Explorer
Connect > DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
      dbo.Student
        Columns
        Keys
        Constraints
        Triggers
        Indexes
        Statistics
      Views
      External Resources

SQLQuery4.sql - DESKTOP-BD6E7F\Dell (59)*
GROUP BY Age
HAVING COUNT(Age) > 1;
SELECT Email
FROM Student
GROUP BY Email
HAVING COUNT(Email) > 1;
SELECT Department_name
FROM Student
GROUP BY Department_name
HAVING COUNT(Department_name) > 1;

Results Messages
Department_name
1 Biology
2 Chemistry
3 Computer Science
4 Mathematics
5 Physics

```

### [Attribute: GPA]

```

Object Explorer
Connect > DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
      dbo.Student
        Columns
        Keys
        Constraints
        Triggers
        Indexes
        Statistics
      Views
      External Resources

SQLQuery4.sql - DESKTOP-BD6E7F\Dell (59)*
SELECT Email
FROM Student
GROUP BY Email
HAVING COUNT(Email) > 1;
SELECT Department_name
FROM Student
GROUP BY Department_name
HAVING COUNT(Department_name) > 1;
SELECT GPA
FROM Student
GROUP BY GPA
HAVING COUNT(GPA) > 1;

Results Messages
GPA
1 2.02
2 2.04
3 2.08
4 2.1
5 2.14
6 2.22
7 2.3
8 2.33
9 2.51
10 2.62
11 2.7

```

### [Attribute: GraduationYear]

```

Object Explorer
Connect - DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.Student
          Columns
          Keys
          Constraints
          Triggers
          Indexes

SQLQuery4.sql - DE...-BD6E7F\Dell (59)* - X
SELECT Department_name
FROM Student
GROUP BY Department_name
HAVING COUNT(Department_name) > 1;
SELECT GPA
FROM Student
GROUP BY GPA
HAVING COUNT(GPA) > 1;
SELECT GraduationYear
FROM Student
GROUP BY GraduationYear
HAVING COUNT(GraduationYear) > 1;

Results Messages
GraduationYear
1 2024
2 2025
3 2026
4 2027
5 2028
6 2029
7 2030

```

### [Attribute: Course\_name]

```

Object Explorer
Connect - DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.Student

SQLQuery4.sql - DE...-BD6E7F\Dell (59)* - X
GROUP BY GPA
HAVING COUNT(GPA) > 1;
SELECT GraduationYear
FROM Student
GROUP BY GraduationYear
HAVING COUNT(GraduationYear) > 1;
SELECT Course_name
FROM Student
GROUP BY Course_name
HAVING COUNT(Course_name) > 1;

Results Messages
Course_name
1 Arts
2 Business
3 Computer Science
4 Engineering

```

### [Attribute: gender]

```

Object Explorer
Connect - DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables

SQLQuery4.sql - DE...-BD6E7F\Dell (59)* - X
GROUP BY GraduationYear
HAVING COUNT(GraduationYear) > 1;
SELECT Course_name
FROM Student
GROUP BY Course_name
HAVING COUNT(Course_name) > 1;
SELECT gender
FROM Student
GROUP BY gender
HAVING COUNT(gender) > 1;

Results Messages
gender
1 Female
2 Male
3 Other

```

### [Attribute: final\_result]

```

Object Explorer
Connect - DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
      Tables
        System Tables
        FileTables

SQLQuery4.sql - DE...-BD6E7F\Dell (59)* - X
GROUP BY Course_name
HAVING COUNT(Course_name) > 1;
SELECT gender
FROM Student
GROUP BY gender
HAVING COUNT(gender) > 1;
SELECT final_result
FROM Student
GROUP BY final_result
HAVING COUNT(final_result) > 1;

Results Messages
final_result
1 Fail
2 Pass

```

we get to know that **StudentID** does not have repetition and it can determine all other attributes. The other columns have repeating values so we cannot consider them as unique identifiers.

So, **StudentID** will be determinant and all other attributes will be dependent. Now we have to check if there are any composite keys that can uniquely identify records by executing the following query:

## Composite Key Identifier

```
SELECT COUNT(*) AS COMPOSITE_IDENTIFIERS  
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE  
WHERE TABLE_NAME='Student' AND TABLE_SCHEMA=DATABASE();
```

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure of 'DESKTOP-BD6E7F1\SQLEXPRESS (SQL Server 15.0.200)'. Under the 'Tables' node, there is a single entry: 'dbo.Student'. On the right, the 'SQLQuery4.sql - DE...BD6E7F1\SQLEXPRESS (SQL Server 15.0.200)\*' window contains the following T-SQL code:

```
SELECT * FROM Student  
GO  
SELECT COUNT(*) AS COMPOSITE_IDENTIFIERS  
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE  
WHERE TABLE_NAME='Student' AND TABLE_SCHEMA=DATABASE();
```

The results pane shows a single row with 'COMPOSITE\_IDENTIFIERS' having a value of 0.

This query tells us if there are any composite keys but as the result of the query is zero we get to know that there are no composite keys which can uniquely identify records. so, Reg-no is the uniquely identifying key(PK).

## Functional Dependency Analysis:

**StudentID → Name**

**StudentID → Age**

**StudentID → Email**

**StudentID → Department\_name**

**StudentID → GPA**

**StudentID → GraduationYear**

**StudentID → Course\_name**

**StudentID → gender**

**StudentID → final\_result**

## From Course table:

The screenshot shows the Object Explorer on the left with databases CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Course_id]
      ,[Course_name]
      ,[Semester]
  FROM [SMSS].[dbo].[Course]
```

The results grid shows the following data:

Course_id	Course_name	Semester
1	Computer Science	6
2	Arts	6
3	Computer Science	6
4	Arts	6
5	Computer Science	6
6	Engineering	6
7	Engineering	6

## [Attribute: Course\_id]

The screenshot shows the Object Explorer on the left with databases CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Course_id]
      ,[Course_name]
      ,[Semester]
  FROM [SMSS].[dbo].[Course]
 WHERE Course_id IN
    (SELECT Course_id
      FROM Course
      GROUP BY Course_id
      HAVING COUNT(Course_id) > 1);
```

The results grid shows the following data:

Course_id
1

## [Attribute: Course\_name]

The screenshot shows the Object Explorer on the left with databases CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Course_id]
      ,[Course_name]
      ,[Semester]
  FROM [SMSS].[dbo].[Course]
 WHERE Course_name IN
    (SELECT Course_name
      FROM Course
      GROUP BY Course_name
      HAVING COUNT(Course_name) > 1);
```

The results grid shows the following data:

Course_name
Arts
Business
Computer Science
Engineering

## [Semester]

The screenshot shows the Object Explorer on the left with databases CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Course_id]
      ,[Course_name]
      ,[Semester]
  FROM [SMSS].[dbo].[Course]
 WHERE Semester IN
    (SELECT Semester
      FROM Course
      GROUP BY Semester
      HAVING COUNT(Semester) > 1);
```

The results grid shows the following data:

Semester
6

Course\_name and Semester are constantly repeating as we can see in the table so they cannot be considered as unique identifier alone. The unique identifier in the course table are [course\_id]. The other columns have repeating values so we cannot consider them as unique identifiers.

## Composite Key Identifier

```
SELECT COUNT(*) AS COMPOSITE_IDENTIFIERS FROM  
INFORMATION_SCHEMA.KEY_COLUMN_USAGE WHERE TABLE_NAME='Customer' AND  
TABLE_SCHEMA=DB_NAME();
```

The screenshot shows the SSMS interface with a query window open. The query retrieves data from the 'Course' table, filtering by Semester count greater than 1, and then counts composite identifiers used in the table. The results show a single row with a value of 0.

```
SELECT TOP (1000) [Course_id]
      ,[Course_name]
      ,[Semester]
     FROM [SMSS].[dbo].[Course]

    SELECT Semester
      FROM Course
     GROUP BY Semester
    HAVING COUNT(Semester) > 1;

    SELECT COUNT(*) AS COMPOSITE_IDENTIFIERS
      FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
     WHERE TABLE_NAME='Course' AND TABLE_SCHEMA=OBJECT_NAME();
```

COMPOSITE_IDENTIFIERS
1   0

So functional dependency that exist in Course table is:

## PARTIAL FUNCTIONAL DEPENDENCY:

[course\_id] → [course\_name]

[course\_id] → [Semester]

This means that in Course table [course\_id] determines {[course\_name],[Semester]}

Determinants: [course id]

Dependents: [course name], [Semester]

## For Department table:

The screenshot shows the SSMS interface with the following details:

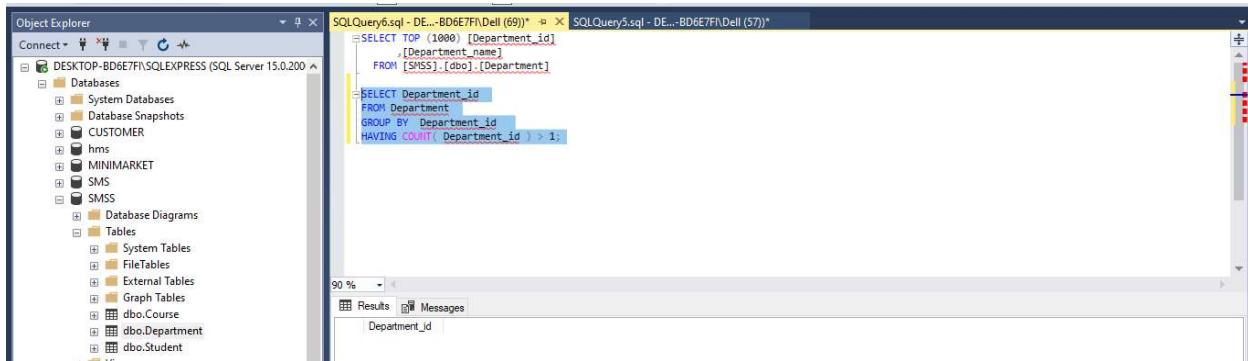
- Object Explorer:** Shows the database structure for "DESKTOP-BD6E7F1\SOLEXPRESS".
- SQLQuery6.sql - DE... - BD6E7F1\DELL (69):** Contains the query:

```
SELECT TOP (1000) [Department_id]
      ,[Department_name]
  FROM [SMSS].[dbo].[Department]
```
- SQLQuery5.sql - DE... - BD6E7F1\DELL (57)\*:** Contains the query:

```
SELECT TOP (1000) [Department_id]
      ,[Department_name]
  FROM [SMSS].[dbo].[Department]
```
- Results Grid:** Displays the results of the query from the first tab, showing 7 rows of department data.

	Department_id	Department_name
1	S001	Mathematics
2	S002	Chemistry
3	S003	Physics
4	S004	Physics
5	S005	Chemistry
6	S006	Mathematics
7	S007	Chemistry

## [Attribute: Department\_id]



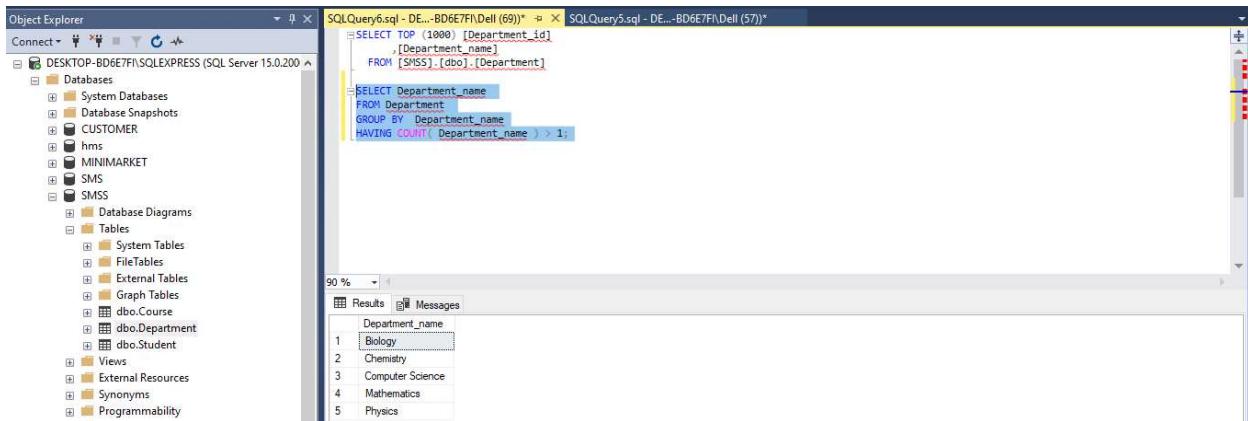
The screenshot shows the Object Explorer on the left with databases CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Department_id]
FROM [SMSS].[dbo].[Department]

SELECT Department_id
FROM Department
GROUP BY Department_id
HAVING COUNT(Department_id) > 1;
```

The Results pane shows a single column 'Department\_id' with the value 1.

## [Attribute: Department\_name]



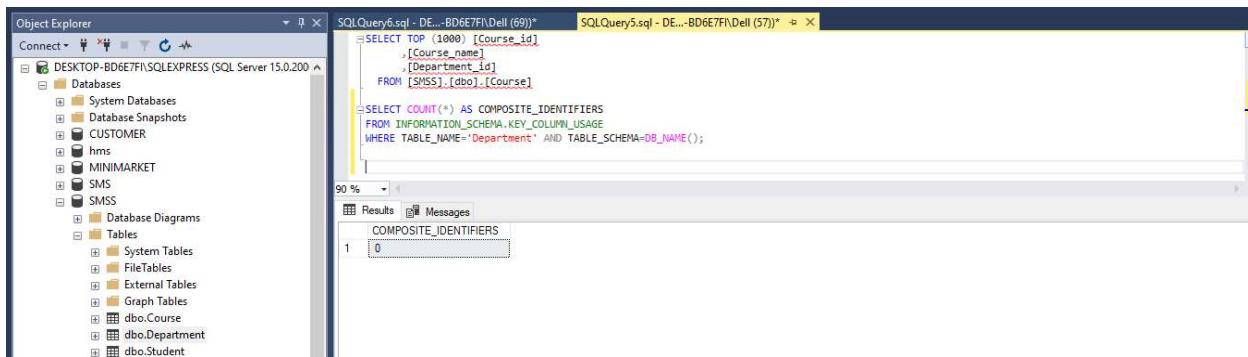
The screenshot shows the Object Explorer on the left with databases CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Department_id]
FROM [SMSS].[dbo].[Department]

SELECT Department_name
FROM Department
GROUP BY Department_name
HAVING COUNT(Department_name) > 1;
```

The Results pane shows a single column 'Department\_name' with values Biology, Chemistry, Computer Science, Mathematics, and Physics.

## Composite key identifier



The screenshot shows the Object Explorer on the left with databases CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Course_id]
      ,[Course_name]
      ,[Department_id]
  FROM [SMSS].[dbo].[Course]

SELECT COUNT(*) AS COMPOSITE_IDENTIFIERS
  FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
 WHERE TABLE_NAME='Department' AND TABLE_SCHEMA=DATABASE_NAME();
```

The Results pane shows a single row with the value 0 under 'COMPOSITE\_IDENTIFIERS'.

Given that dep-id and dep-name both uniquely identify a department, we can establish the following functional dependencies:

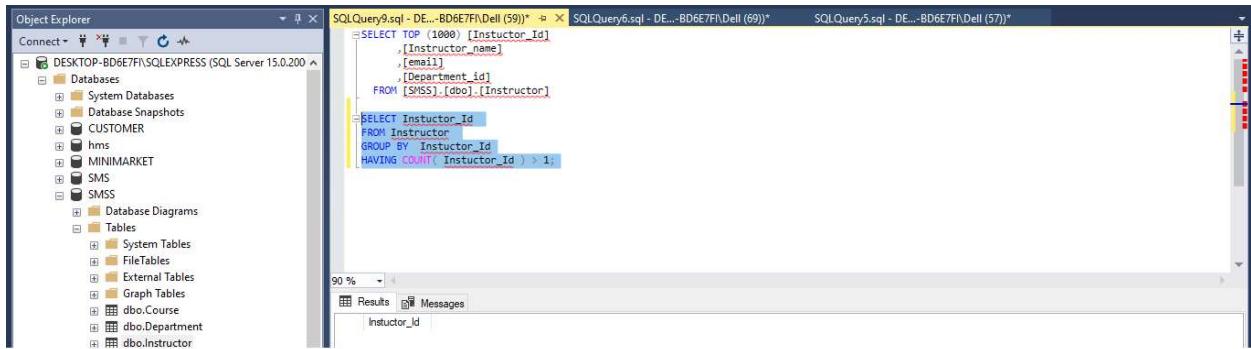
Department\_id uniquely identifies the department's name : dep-id → dep-name

Determinant: [dep-id], [dep-name]

Dependents: [dep-id]

## For Instructor Table:

### [Attributes: Instructor\_id]



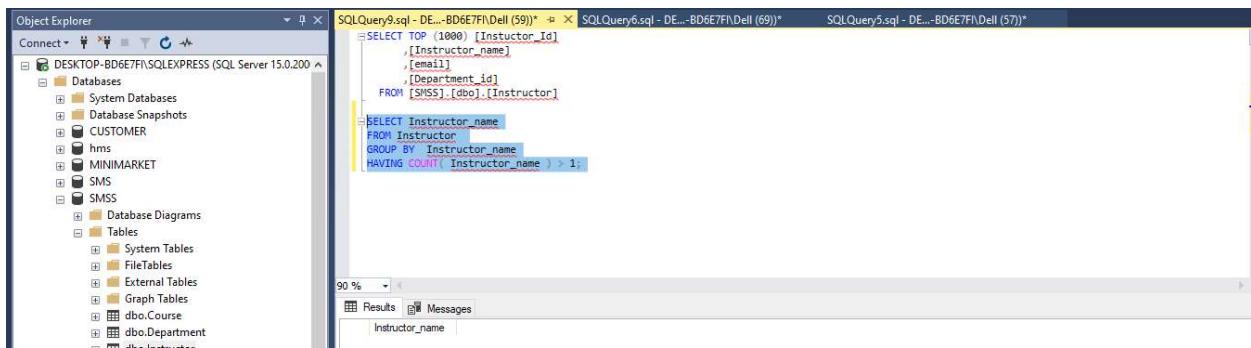
The screenshot shows the Object Explorer on the left with the database 'DESKTOP-BD6E7F\SQLEXPRESS' selected. The 'Tables' node under 'SMS' is expanded, showing 'dbo.Instructor'. The 'Results' tab in the center displays the output of the following query:

```
SELECT TOP (1000) [Instructor_Id]
      ,[Instructor_name]
      ,[email]
      ,[Department_id]
  FROM [SMS$].[dbo].[Instructor]

  SELECT Instructor_Id
  FROM Instructor
  GROUP BY Instructor_Id
  HAVING COUNT( Instructor_Id ) > 1;
```

The results show one row with Instructor\_Id: 1.

### [Attributes: Instructor\_name]



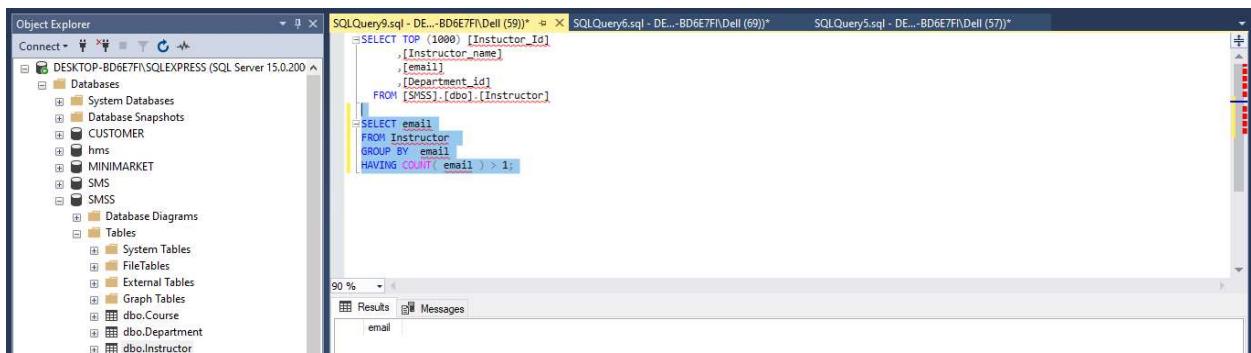
The screenshot shows the Object Explorer on the left with the database 'DESKTOP-BD6E7F\SQLEXPRESS' selected. The 'Tables' node under 'SMS' is expanded, showing 'dbo.Instructor'. The 'Results' tab in the center displays the output of the following query:

```
SELECT TOP (1000) [Instructor_Id]
      ,[Instructor_name]
      ,[email]
      ,[Department_id]
  FROM [SMS$].[dbo].[Instructor]

  SELECT Instructor_name
  FROM Instructor
  GROUP BY Instructor_name
  HAVING COUNT( Instructor_name ) > 1;
```

The results show one row with Instructor\_name: 'John Doe'.

### [Attribute: Email]



The screenshot shows the Object Explorer on the left with the database 'DESKTOP-BD6E7F\SQLEXPRESS' selected. The 'Tables' node under 'SMS' is expanded, showing 'dbo.Instructor'. The 'Results' tab in the center displays the output of the following query:

```
SELECT TOP (1000) [Instructor_Id]
      ,[Instructor_name]
      ,[email]
      ,[Department_id]
  FROM [SMS$].[dbo].[Instructor]

  SELECT email
  FROM Instructor
  GROUP BY email
  HAVING COUNT( email ) > 1;
```

The results show one row with email: 'johndoe@example.com'.

### [Attribute: department\_id]

```

Object Explorer
Connect + DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.2000 -)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
    Database Diagrams
      Tables
      Views
      External Resources
      Synonyms
      Programmability
      Service Broker
      Storage
      Security
  Security

SQLQuery9.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*
SELECT TOP (1000) [Instructor_id]
      ,[Instructor_name]
      ,[email]
      ,[Department_id]
  FROM [SMSS].[dbo].[Instructor]

SELECT Department_id
  FROM Instructor
 GROUP BY Department_id
 HAVING COUNT(Department_id) > 1;

SQLQuery6.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*
SQLQuery5.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*

Results Messages
Department_id
1 S001

```

Since we can see that dep-id is not unique it can become foreign key in this table.

department-id are constantly repeating as we can see in the table so they cannot be considered as unique identifier alone. The unique identifier in the Instructor table are [Instructor\_id] and [Instructor\_Email],[Instructor\_Name].

## Composite keys Identifier

```

Object Explorer
Connect + DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.2000 -)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
    Database Diagrams
      Tables
      Views
      External Resources
      Synonyms
      Programmability
      Service Broker
      Storage
      Security
  Security

SQLQuery9.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*
SELECT TOP (1000) [Instructor_Id]
      ,[Instructor_name]
      ,[email]
      ,[Department_id]
  FROM [SMSS].[dbo].[Instructor]

SELECT Department_id
  FROM Instructor
 GROUP BY Department_id
 HAVING COUNT(Department_id) > 1;

SELECT COUNT(*) AS COMPOSITE_IDENTIFIERS
  FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
 WHERE TABLE_NAME='Instructor' AND TABLE_SCHEMA=DATABASE_NAME();

SQLQuery6.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*
SQLQuery5.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*

Results Messages
COMPOSITE_IDENTIFIERS
1 0

```

So functional dependency that exist in Instructor table is:

$$[\text{Instructor\_Id}], [\text{Instructor\_Name}], [\text{Instructor\_Email}] \rightarrow [\text{dep-id}]$$

This suggests that the combination of Instructor\_Id, Instructor\_Name, and Instructor\_Email determines dep-id.

Determinants: [Instructor\_Id], [Instructor\_Name] , [Instructor\_Email]

Dependents: [dep-id]

## For Enrollment Table:

```

Object Explorer
Connect + DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.2000 -)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
    Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.Course
        dbo.Department
        dbo.Instructor
        dbo.Student
        Views
  Security

SQLQuery5.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*
SELECT TOP (1000) [Enrollment_id]
      ,[Course_id]
      ,[StudentID]
      ,[year]
      ,[grade]
  FROM [SMSS].[dbo].[Enrollment]

SQLQuery4.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*
SQLQuery2.sql - DE...BD6E7F\SQLEXPRESS (SQL Server 15.0.2000)*

Results Messages
Enrollment_id Course_id StudentID year grade
1 E1 1 3336 2002 B
2 E2 2 9774 2003 C
3 E3 3 1396 2004 D
4 E4 4 6716 2005 F

```

[Attribute: Enrollment\_id]

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left lists databases: DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200), CUSTOMER, hms, MINIMARKET, SMS, and SMSS. The SMSS database is expanded, showing System Tables, Tables (with Course, Department, Enrollment), and System Views. Two query panes are open: SQLQuery3.sql and SQLQuery2.sql. SQLQuery3.sql contains a query to select top 1000 rows from the Enrollment table, including columns Course\_id, StudentID, year, and grade. SQLQuery2.sql contains a query to find Enrollment\_ids where the count of rows is greater than 1.

```
Object Explorer
Connect ▾
DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200) ▾
  Databases
    System Databases
    Database Snapshots
  CUSTOMER
  hms
  MINIMARKET
  SMS
  SMSS
    Tables
      Course
      Department
      Enrollment
    System Tables
    FileTables
    External Tables
    Graph Tables
  System Views

SQLQuery3.sql - DE...-BD6E7FDell (61)* x SQLQuery2.sql - DE...-BD6E7FDell (58)*
=SELECT TOP (1000) [Enrollment_id]
,[Course_id]
,[StudentID]
,[year]
,[grade]
FROM [SMSS].[dbo].[Enrollement]

=SELECT Enrollment_id
FROM Enrollment
GROUP BY Enrollment_id
HAVING COUNT(Enrollment_id) > 1;
```

[Course\_id]

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left lists databases including 'DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)', 'CUSTOMER', 'hms', 'MINIMARKET', 'SMS', and 'SMSS'. The 'Tables' node under 'CUSTOMER' is expanded, showing 'Course', 'Department', and 'Enrollment'. The central pane displays two open queries. The top query is:

```
SELECT TOP (1000) [Enrollment_id],  
       [Course_id],  
       [StudentID],  
       [year],  
       [grade]  
  FROM [SMSS].[dbo].[Enrollement]
```

The bottom query is:

```
=> SELECT Course_id  
      FROM Enrollment  
      GROUP BY Course_id  
      HAVING COUNT(Course_id) > 1;
```

The status bar at the bottom indicates a progress of 90%.

[student\_id]



The screenshot shows the Object Explorer pane on the left with a tree view of database objects. The central pane displays two SQL queries in the SQL Editor. The top query retrieves enrollment details for the top 1000 students, while the bottom query identifies students with more than one enrollment.

```
SELECT TOP (1000) [Enrollment_id]
      ,[Course_ID]
      ,[StudentID]
      ,[Year]
      ,[Grade]
  FROM [SWS5].[dbo].[Enrollement]

  SELECT StudentID
    FROM Enrollment
   GROUP BY StudentID
 HAVING COUNT(StudentID) > 1;
```

[year]

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer on the left, there is a tree view of databases, including 'DESKTOP-BD6E7F\SQLExpress' (SQL Server 15.0.200) which contains 'CUSTOMER', 'hms', 'MINIMARKET', 'SMS', and 'SMS'. The 'SMS' database is expanded to show 'Tables' and 'Views'. In the center, a query window titled 'SQLQuery4.sql - DE... - BD6E7F\Del (69)' is open with the following T-SQL code:

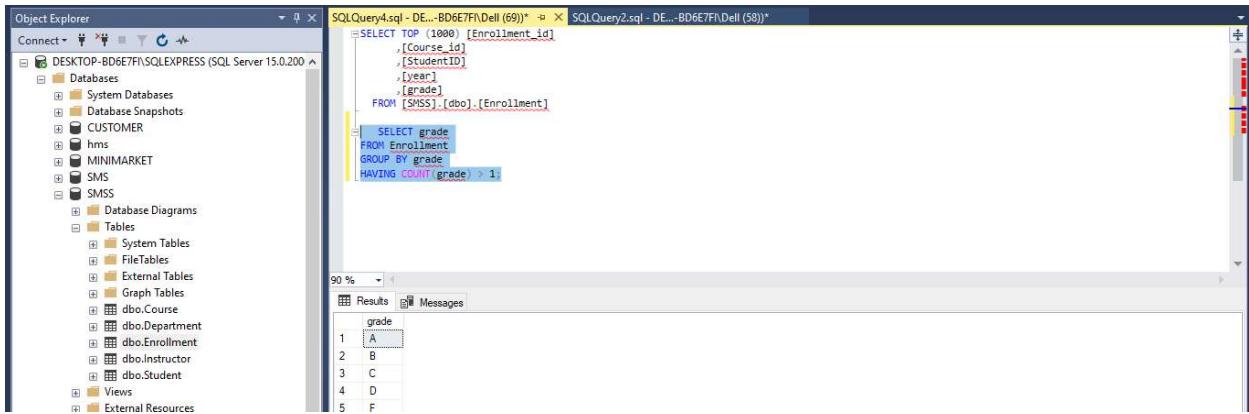
```
SELECT TOP (1000) [Enrollment_id]
,[Course_id]
,[StudentID]
,[Year]
,[Term]
,[Semester]
FROM [LSSS5].[dbo].[Enrollment]

SELECT *
FROM Enrollment
GROUP BY Year
HAVING COUNT(Year) > 1;
```

The 'Results' tab at the bottom shows the output of the query:

year
2006

[grade]



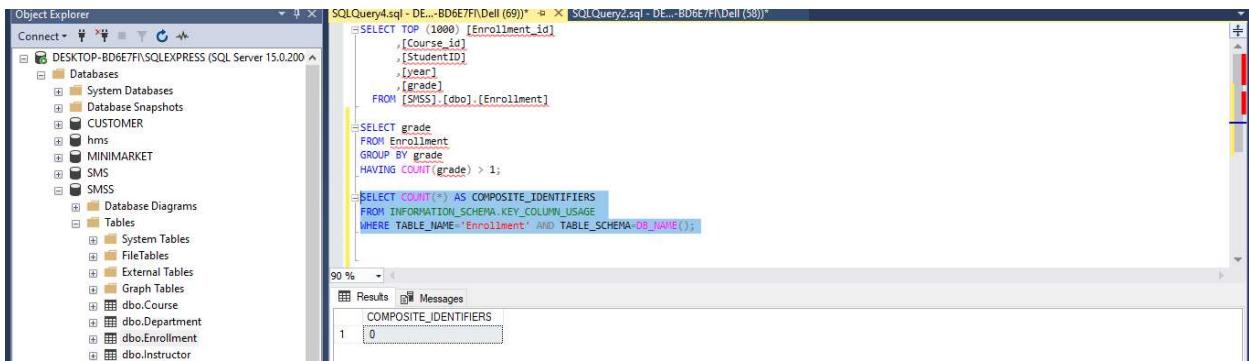
The screenshot shows the Object Explorer on the left with databases like CUSTOMER, hms, MINIMARKET, SMS, and SMSS selected. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Enrollment_id]
      ,[Course_id]
      ,[StudentID]
      ,[year]
      ,[grade]
  FROM [SMSS].[dbo].[Enrollment]

  SELECT grade
  FROM Enrollment
  GROUP BY grade
  HAVING COUNT(grade) > 1;
```

The Results pane shows a table with one column 'grade' containing values 1, 2, 3, 4, and 5.

## Composite key identifier



The screenshot shows the Object Explorer on the left with the same database selection. The SQL Query window on the right contains the following query:

```
SELECT TOP (1000) [Enrollment_id]
      ,[Course_id]
      ,[StudentID]
      ,[year]
      ,[grade]
  FROM [SMSS].[dbo].[Enrollment]

  SELECT grade
  FROM Enrollment
  GROUP BY grade
  HAVING COUNT(grade) > 1;

  SELECT COUNT(*) AS COMPOSITE_IDENTIFIERS
  FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
  WHERE TABLE_NAME='Enrollment' AND TABLE_SCHEMA=DATABASE();
```

The Results pane shows a table with one column 'COMPOSITE\_IDENTIFIERS' containing the value 1.

If we have defined only Enrollment\_id as PK, no composite PK will appear.  
If we define (StudentID, Course\_id) as composite PK, both will appear under the PK constraint.

## Using Query

### Step-wise implementation for Exam table

#### 1. Create Exam Table

```
CREATE TABLE Exam (
    Exam_id INT PRIMARY KEY,
    StudentID INT,
    Course_id INT,
    exam_date DATE,
    marks_obtained INT,
    total_marks INT,
    grade CHAR(1)
);
```

```

CREATE TABLE Exam (
    Exam_id INT PRIMARY KEY,
    StudentID INT,
    Course_id INT,
    exam_date DATE,
    marks_obtained INT,
    total_marks INT,
    grade CHAR(1)
);

```

## 2. Insert Data into Exam Table

### Insert single record:

```

INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained,
total_marks, grade)
VALUES (1, 3336, 1, '2025-07-16', 85, 100, 'A');

```

```

CREATE TABLE Exam (
    Exam_id INT PRIMARY KEY,
    StudentID INT,
    Course_id INT,
    exam_date DATE,
    marks_obtained INT,
    total_marks INT,
    grade CHAR(1)
);

INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained, total_marks, grade)
VALUES (1, 3336, 1, '2025-07-16', 85, 100, 'A');

```

```

SELECT TOP (1000) [Exam_id],
       [StudentID],
       [Course_id],
       [exam_date],
       [marks_obtained],
       [total_marks],
       [grade]
  FROM [SMSS].[dbo].[Exam]

```

Exam_id	StudentID	Course_id	exam_date	marks_obtained	total_marks	grade
1	3336	1	2025-07-16	85	100	A

### Insert multiple records:

```

INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained,
total_marks, grade)
VALUES
(2, 8774, 2, '2025-07-17', 70, 100, 'B'),

```

(3, 1396, 3, '2025-07-18', 60, 100, 'C');

The screenshot shows the Object Explorer on the left with the database 'DESKTOP-BD6E7F\SQLEXPRESS' selected. The 'Tables' node under 'dbo' contains 'Exam'. Two queries are open in the center:

```
CREATE TABLE Exam (
    Exam_id INT PRIMARY KEY,
    StudentID INT,
    Course_id INT,
    exam_date DATE,
    marks_obtained INT,
    total_marks INT,
    grade CHAR(1)
);
INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained, total_marks, grade)
VALUES (1, 3336, 1, '2025-07-16', 85, 100, 'A');
INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained, total_marks, grade)
VALUES
(2, 8774, 2, '2025-07-17', 70, 100, 'B'),
(3, 1396, 3, '2025-07-18', 60, 100, 'C');
```

The 'Messages' pane at the bottom shows '(2 rows affected)' and the completion time: 2025-07-17 19:31:26.4916202+05:00.

### 3. Update Data in Exam Table

Update marks and grade for a specific exam:

The screenshot shows the same database structure and query window setup as the previous screenshot. A new update statement is added to the second query:

```
UPDATE Exam
SET marks_obtained = 75, grade = 'B'
WHERE Exam_id = 3;
```

The 'Messages' pane shows '(1 row affected)' and the completion time: 2025-07-17 19:32:36.7744492+05:00.

The screenshot shows the same database structure and query window setup. A new select statement is added to the third query:

```
SELECT TOP (1000) [Exam_id]
,[StudentID]
,[Course_id]
,[exam_date]
,[marks_obtained]
,[total_marks]
,[grade]
FROM [SMSS].[dbo].[Exam]
```

The 'Results' pane displays the following data:

Exam_id	StudentID	Course_id	exam_date	marks_obtained	total_marks	grade
1	3336	1	2025-07-16	85	100	A
2	8774	2	2025-07-17	70	100	B
3	1396	3	2025-07-18	60	100	B

### 4. Delete Data from Exam Table

Delete a specific record:

DELETE FROM Exam

WHERE Exam\_id = 2;

```

Object Explorer
Connect ▾ DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
        Tables
          System Tables
          FileTables
          External Tables
          Graph Tables
          dbo.Course
          dbo.Department
          dbo.Enrollment
          dbo.Exam

SQLQuery8.sql - DE...-BD6E7F\DEll (58)* SQLQuery7.sql - DE...-BD6E7F\DEll (57) SQLQuery6.sql - DE...-BD6E7F\DEll (52)*
INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained, total_marks, grade)
VALUES (1, 3336, 1, '2025-07-16', 85, 100, 'A');

INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained, total_marks, grade)
VALUES
(2, 8774, 2, '2025-07-17', 70, 100, 'B'),
(3, 1396, 3, '2025-07-18', 60, 100, 'C');

UPDATE Exam
SET marks_obtained = 75, grade = 'B'
WHERE Exam_id = 3;

DELETE FROM Exam
WHERE Exam_id = 2;

90 % ▾ Messages
(1 row affected)
Completion time: 2025-07-17T19:35:32.5090047+08:00

```

```

Object Explorer
Connect ▾ DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
        Tables
          System Tables
          FileTables
          External Tables
          Graph Tables
          dbo.Course
          dbo.Department
          dbo.Enrollment
          dbo.Exam

SQLQuery9.sql - DE...-BD6E7F\DEll (59) SQLQuery6.sql - DE...-BD6E7F\DEll (52)*
SELECT TOP (1000) [Exam_id]
,[StudentID]
,[Course_id]
,[exam_date]
,[marks_obtained]
,[total_marks]
,[grade]
FROM [SMSS].[dbo].[Exam]

90 % ▾ Results ▾ Messages
Exam_id StudentID Course_id exam_date marks_obtained total_marks grade
1 3336 1 2025-07-16 85 100 A
2 1396 3 2025-07-18 75 100 B

```

## 5. Basic Retrieval Queries

Retrieve all records:

`SELECT * FROM Exam;`

```

Object Explorer
Connect ▾ DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)
  Databases
    System Databases
    Database Snapshots
    CUSTOMER
    hms
    MINIMARKET
    SMS
    SMSS
      Database Diagrams
        Tables
          System Tables
          FileTables
          External Tables
          Graph Tables
          dbo.Course
          dbo.Department
          dbo.Enrollment
          dbo.Exam

SQLQuery9.sql - DE...-BD6E7F\DEll (56) SQLQuery6.sql - DE...-BD6E7F\DEll (52)*
INSERT INTO Exam (Exam_id, StudentID, Course_id, exam_date, marks_obtained, total_marks, grade)
VALUES
(2, 8774, 2, '2025-07-17', 70, 100, 'B'),
(3, 1396, 3, '2025-07-18', 60, 100, 'C');

UPDATE Exam
SET marks_obtained = 75, grade = 'B'
WHERE Exam_id = 3;

DELETE FROM Exam
WHERE Exam_id = 2;

SELECT * FROM Exam

90 % ▾ Results ▾ Messages
Exam_id StudentID Course_id exam_date marks_obtained total_marks grade
1 3336 1 2025-07-16 85 100 A
2 1396 3 2025-07-18 75 100 B

```

Retrieve specific columns:

`SELECT StudentID, marks_obtained, grade`

`FROM Exam;`

Object Explorer

SQLQuery9.sql - DE...-BD6E7F\DEll (56)    SQLQuery6.sql - DE...-BD6E7F\DEll (52)\*

```

VALUES
(2, 8774, 2, '2025-07-17', 70, 100, 'B'),
(3, 1396, 3, '2025-07-18', 60, 100, 'C');

UPDATE Exam
SET marks_obtained = 75, grade = 'B'
WHERE Exam_id = 3;

DELETE FROM Exam
WHERE Exam_id = 2;

SELECT * FROM Exam;

SELECT StudentID, marks_obtained, grade
FROM Exam;

```

Results

	StudentID	marks_obtained	grade
1	3336	85	A
2	1396	75	B

### Retrieve data with condition (WHERE clause):

`SELECT *`

`FROM Exam`

`WHERE grade = 'A';`

Object Explorer

SQLQuery9.sql - DE...-BD6E7F\DEll (56)    SQLQuery6.sql - DE...-BD6E7F\DEll (52)\*

```

UPDATE Exam
SET marks_obtained = 75, grade = 'B'
WHERE Exam_id = 3;

DELETE FROM Exam
WHERE Exam_id = 2;

SELECT * FROM Exam;

SELECT StudentID, marks_obtained, grade
FROM Exam;

SELECT *
FROM Exam
WHERE grade = 'A';

```

Results

	Exam_id	StudentID	Course_id	exam_date	marks_obtained	total_marks	grade
1	1	3336	1	2025-07-16	85	100	A

## 6. Delete Specific Column

Delete a column (e.g. total\_marks):

Object Explorer

SQLQuery10.sql - DE...-BD6E7F\DEll (57)    SQLQuery6.sql - DE...-BD6E7F\DEll (52)\*

```

DELETE FROM Exam
WHERE Exam_id = 2;

SELECT * FROM Exam;

SELECT StudentID, marks_obtained, grade
FROM Exam;

SELECT *
FROM Exam
WHERE grade = 'A';

ALTER TABLE Exam
DROP COLUMN total_marks;

```

Messages

Command completed successfully.

Completion time: 2025-07-17T19:40:19.9828661+05:00

The screenshot shows the SSMS interface. The Object Explorer on the left lists the database structure for 'DESKTOP-BD6E7F\SQLExpress'. The 'Tables' node under 'CUSTOMER' is expanded, showing tables like 'hms', 'MINIMARKET', 'SMS', and 'SMSS'. The 'dbo.Exam' table is selected. The main window displays a T-SQL query and its results.

```
SELECT TOP (1000) [Exam_id]
      ,[StudentID]
      ,[Course_id]
      ,[exam_date]
      ,[marks_obtained]
      ,[grade]
  FROM [SMSS].[dbo].[Exam]
```

	Exam_id	StudentID	Course_id	exam_date	marks_obtained	grade
1	1	3336	1	2025-07-16	85	A
2	3	1396	3	2025-07-18	75	B

## 7. Alter Table: Add Columns

Add a single column (e.g. examiner\_name):

## ALTER TABLE Exam

ADD examiner\_name VARCHAR(50);

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure for "DESKTOP-BD6E7F\SQLEXPRESS (SQL Server 15.0.200)". It includes nodes for Databases, Tables, and System Tables.
- SQL Query10.sql - DE...BD6E7F\DELL (57) :** Contains the following T-SQL code:

```
SELECT * FROM Exam;  
SELECT StudentID, marks_obtained, grade  
FROM Exam;  
  
SELECT *  
FROM Exam  
WHERE grade = 'A';  
  
ALTER TABLE Exam  
DROP COLUMN total_marks;  
  
ALTER TABLE Exam  
ADD examiner_name VARCHAR(50);
```
- SQLQuery6.sql - DE...BD6E7F\DELL (52)\* :** Shows the message "Commands completed successfully." and the completion time: 2025-07-17T19:43:06.026053+08:00.

Add multiple columns (e.g. exam room and duration):

## ALTER TABLE Exam

ADD exam\_room VARCHAR(20),

duration INT;

The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'DESKTOP-BD6E7F\SQLExpress' (SQL Server 15.0.200). The 'Tables' node under 'Exam' contains the following columns: Exam\_ID (int), Exam\_name (varchar(50)), grade (char(1)), total\_marks (int), examiner\_name (varchar(50)), exam\_room (varchar(20)), and duration (int). Two queries are running in the center pane:

```
SELECT *  
FROM Exam  
WHERE grade = 'A';  
  
ALTER TABLE Exam  
DROP COLUMN total_marks;  
  
ALTER TABLE Exam  
ADD examiner_name VARCHAR(50);  
  
ALTER TABLE Exam  
ADD exam_room VARCHAR(20),  
duration INT;
```

The status bar at the bottom indicates a completion time of 2025-07-17T19:42:46.2331250+05:00.

## 8. Modify Column

Change data type of duration column:

ALTER TABLE Exam

ALTER COLUMN duration VARCHAR(10);

The screenshot shows the Object Explorer on the left with two databases selected: DESKTOP-BD6E7F\SQLEXPRESS and DESKTOP-BD6E7F\SQLEXPRESS. The right side has three query panes. The top pane contains the following SQL code:

```
SELECT *  
FROM Exam  
WHERE grade = 'A';  
  
ALTER TABLE Exam  
DROP COLUMN total_marks;  
  
ALTER TABLE Exam  
ADD examiner_name VARCHAR(50);  
  
ALTER TABLE Exam  
ADD exam_room VARCHAR(20),  
duration INT;  
  
ALTER TABLE Exam  
ALTER COLUMN duration VARCHAR(10);
```

The middle pane shows the execution results with the message "Commands completed successfully." and a completion time of 2025-07-17T19:44:29.8282695+05:00.

The bottom pane shows the following SQL code:

```
SELECT TOP (1000) [Exam_id]  
,[StudentID]  
,[Course_id]  
,[exam_date]  
,[marks_obtained]  
,[grade]  
,[examiner_name]  
,[exam_room]  
,[duration]  
FROM [SNSS].[dbo].[Exam]
```

The Results pane displays the data from the Exam table:

Exam_id	StudentID	Course_id	exam_date	marks_obtained	grade	examiner_name	exam_room	duration
1	1	3336	1	2025-07-16	85	A	NULL	NULL
2	3	1396	3	2025-07-18	75	B	NULL	NULL

## 9. Delete Entire Exam Table

Drop table:

DROP TABLE Exam;

The screenshot shows the Object Explorer on the left with the same two databases selected. The right side has three query panes. The top pane contains the following SQL code:

```
SELECT *  
FROM Exam  
WHERE grade = 'A';  
  
ALTER TABLE Exam  
DROP COLUMN total_marks;  
  
ALTER TABLE Exam  
ADD examiner_name VARCHAR(50);  
  
ALTER TABLE Exam  
ADD exam_room VARCHAR(20),  
duration INT;  
  
ALTER TABLE Exam  
ALTER COLUMN duration VARCHAR(10);  
  
DROP TABLE Exam;
```

The middle pane shows the execution results with the message "Commands completed successfully." and a completion time of 2025-07-17T19:46:56.8412542+05:00.

The screenshot shows the SSMS interface. The Object Explorer on the left lists databases like CUSTOMER, MINIMARKET, hms, SMS, and SMSS, along with their tables such as Exam, Department, Enrollment, and Course. Three query panes are open at the top:

- SQLQuery12.sql - D...-BD6E7F1\DELL (58)
- SQLQuery11.sql - D...-BD6E7F1\DELL (56)
- SQLQuery10.sql - D...-BD6E7F1\DELL (57)

The middle pane contains the following SQL query and its execution results:

```
SELECT TOP (1000) * FROM [SMSS].[dbo].[Exam]
```

Messages pane output:

```
Msg 208, Level 16, State 1, Line 1
Invalid object name 'SMSS.dbo.Exam'.
Completion time: 2025-07-17T19:47:12.1616559+05:00
```

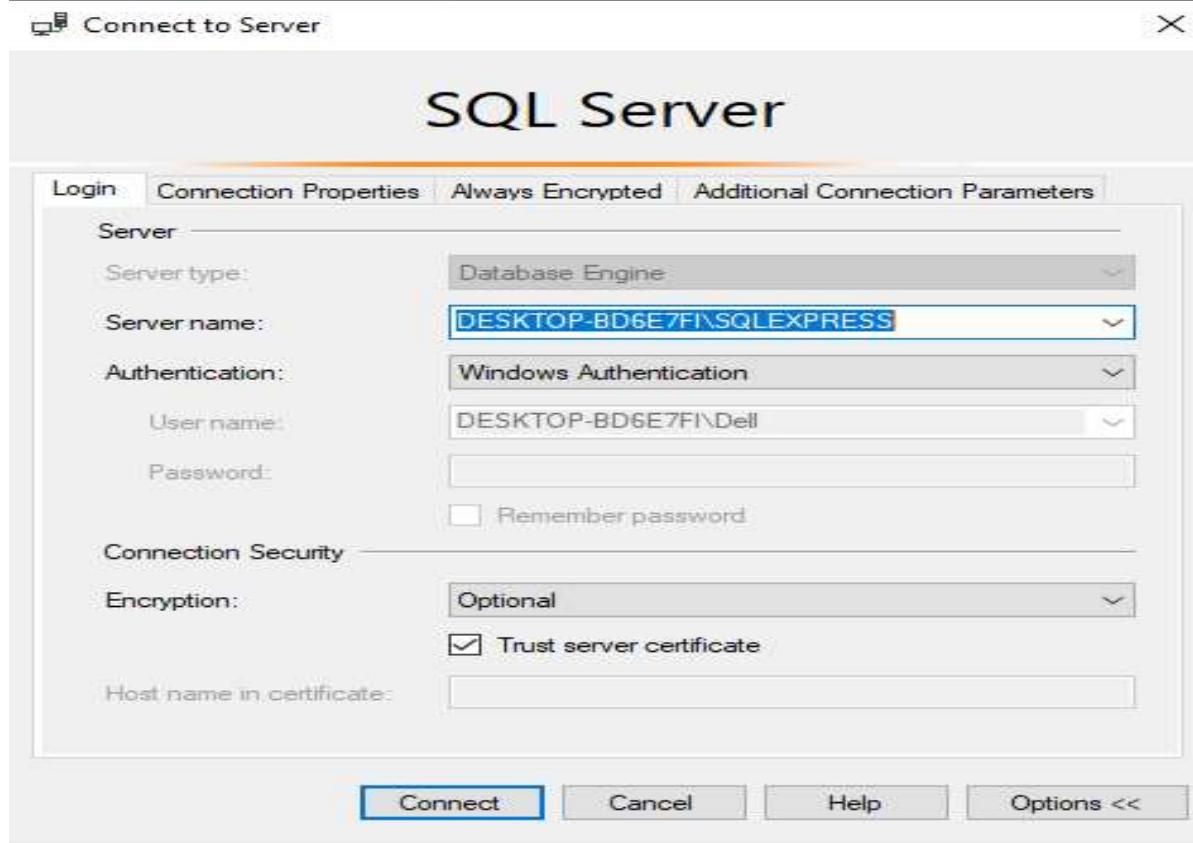
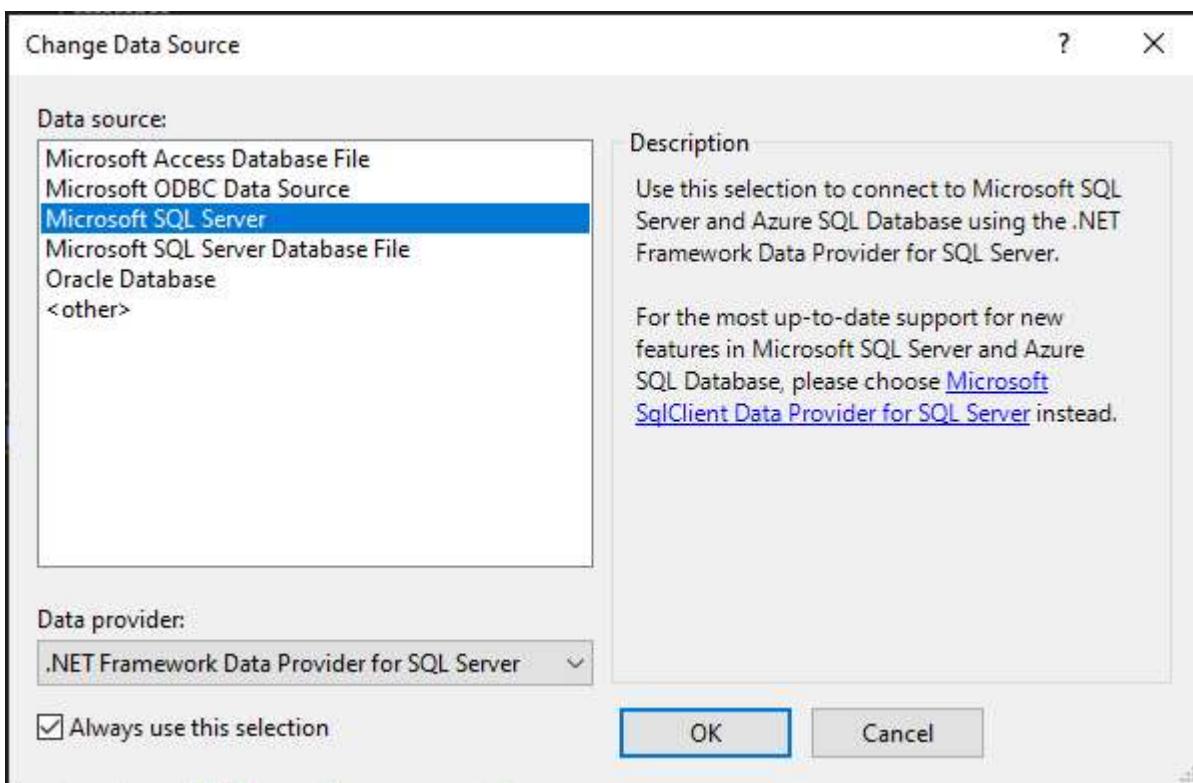
## Frontend Form using Visual Studio Code

The screenshot shows a Windows application window titled "Student". The title bar also displays "Student Management System". The application has a green background with school-related icons like a calculator, ruler, and books. It features a grid of input fields for student information:

StudentID	Department_name
Name	GraduationYear
Age	Course_name
Email	gender
GPA	final_result

At the bottom of the form are three buttons: "INSERT", "DELETE", and "UPDATE".

## Connection With DB



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: DESKTOP-BD6E7FI\SQLEXPRESS

Log on to the server

Authentication: Windows Authentication

User name:

Password:

Encrypt: Mandatory (True)

Trust Server Certificate  
 Save my password

Connect to a database

Select or enter a database name: SMSS

Attach a database file:   
Logical name:

Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: DESKTOP-BD6E7FI\SQLEXPRESS

Log on to the server

Authentication: Windows Authentication

User name:

Password:

Encrypt:  Options  
 Trust  
 Save

Connect to a database

Select or enter a database name: SMSS

Attach a database file:   
Logical name:

Microsoft Visual Studio  
Test connection succeeded.

## INSERT Button

```
private void button1_Click(object sender, EventArgs e)
{
    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
BD6E7FI\\SQLEXPRESS;Initial Catalog=SMSS;Integrated Security=True;Encrypt=False"))
    {
        string Query = @"INSERT INTO Student
(StudentID, Name, Age, Email, Department_name, GPA, GraduationYear, Course_name, gender,
final_result)

VALUES
(@StudentID, @Name, @Age, @Email, @Department_name, @GPA, @GraduationYear,
@Course_name, @gender, @final_result)";

        using (SqlCommand cmd = new SqlCommand(Query, con))
        {
            cmd.Parameters.AddWithValue("@StudentID", txtID.Text);
            cmd.Parameters.AddWithValue("@Name", txtname.Text);
            cmd.Parameters.AddWithValue("@Age", txtage.Text);
            cmd.Parameters.AddWithValue("@Email", txtemail.Text);
            cmd.Parameters.AddWithValue("@GPA", txtgpa.Text);
            cmd.Parameters.AddWithValue("@Department_name", txtdepartment_name.Text);
            cmd.Parameters.AddWithValue("@GraduationYear", txtgraduationyear.Text);
            cmd.Parameters.AddWithValue("@Course_name", txtcoursename.Text);
            cmd.Parameters.AddWithValue("@gender", txt_gnder.Text);
            cmd.Parameters.AddWithValue("@final_result", txtfinalresult.Text);

            con.Open();
            cmd.ExecuteNonQuery();

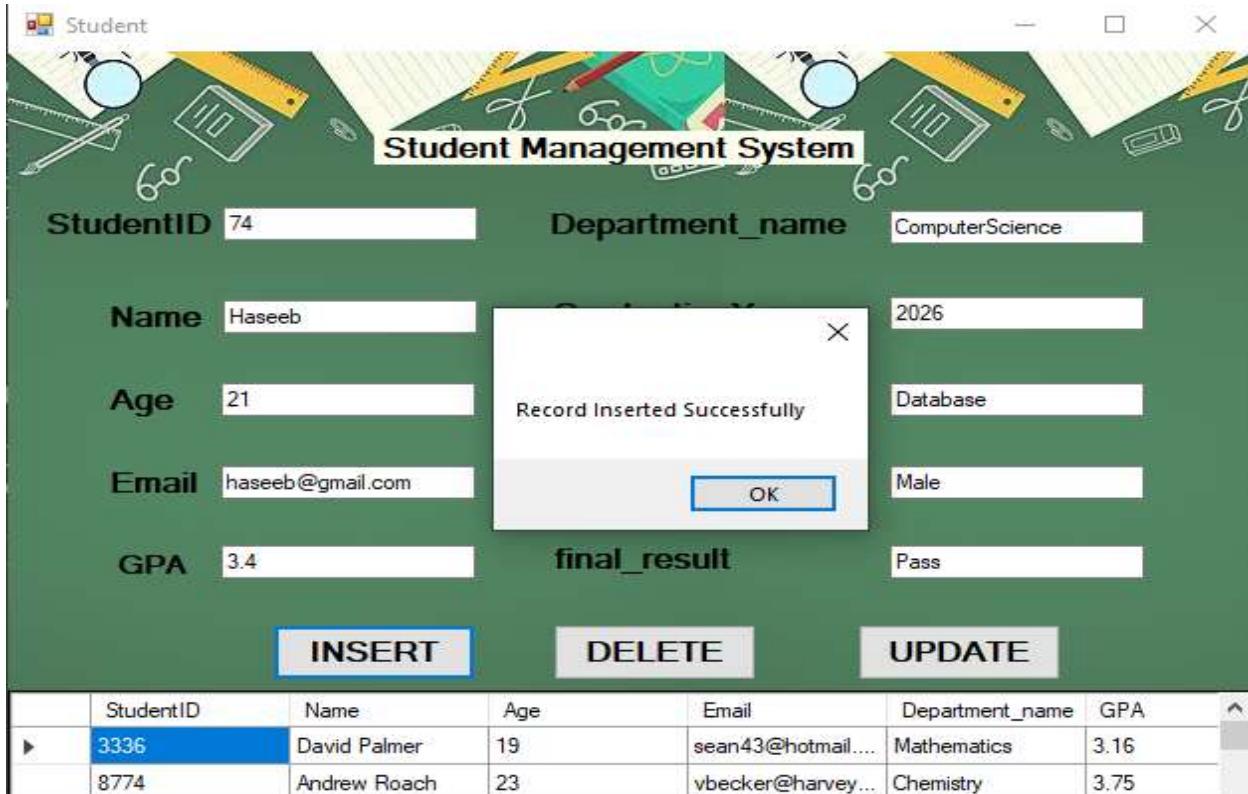
            con.Close();
        }
    }
}
```

```

        MessageBox.Show("Record Inserted Successfully");

        ClearFields(); // Clear fields after insertion

        GetStudentRecord(); // refresh DataGridView to show new data }}}
    
```



## Output

	StudentID	Name	Age	Email	Department_name	GPA
	9179	Theresa Hughes	21	mstanley@yahoo...	Mathematics	3.23
	4463	Raza	22	raza@gmail.com	SoftwareEnginee...	2.9
	74	Haseeb	21	haseeb@gmail.c...	ComputerScience	3.4

## DELETE button

```
using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
BD6E7FI\\SQLEXPRESS;Initial Catalog=SMSS;Integrated
Security=True;TrustServerCertificate=True"))

{
    string query = "DELETE FROM Student WHERE StudentID = @StudentID";

    using (SqlCommand cmd = new SqlCommand(query, con))

    {
        cmd.Parameters.AddWithValue("@StudentID", txtID.Text);

        con.Open();

        cmd.ExecuteNonQuery();

        con.Close();

        MessageBox.Show("Record Deleted Successfully");

        ClearFields(); // clear input fields after deletion

        GetStudentRecord(); // refresh DataGridView to show updated records
    }
}
```

Student

### Student Management System

StudentID	4463	Department_name	
Name		GraduationYear	
Age			
Email			
GPA		Final_Result	

StudentID
Name
Age
Email
Department\_name
GPA

X

Record Deleted Successfully

## Output

	StudentID	Name	Age	Email	Department_name	GPA
	3274	Catherine Velasq...	22	fsutton@gmail.com	Physics	2.71
	3609	Joyce Thomas	19	zanderson@barr...	Chemistry	2.22
	9179	Theresa Hughes	21	mstanley@yahoo...	Mathematics	3.23
	74	Haseeb	21	haseeb@gmail.c...	ComputerScience	3.4

## UPDATE Button

```
private void button4_Click(object sender, EventArgs e)

{

    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
BD6E7FI\\SQLEXPRESS;Initial Catalog=SMSS;Integrated
Security=True;TrustServerCertificate=True"))

    {

        string query = @"UPDATE Student

SET Name = @Name,
Age = @Age,
Email = @Email,
Department_name = @Department_name,
GPA = @GPA,
GraduationYear = @GraduationYear,
Course_name = @Course_name,
gender = @gender,
final_result = @final_result

WHERE StudentID = @StudentID";

        using (SqlCommand cmd = new SqlCommand(query, con))

        {

            cmd.Parameters.AddWithValue("@StudentID", txtID.Text);
            cmd.Parameters.AddWithValue("@Name", txtname.Text);
            cmd.Parameters.AddWithValue("@Age", txtage.Text);
            cmd.Parameters.AddWithValue("@Email", txtemail.Text);
        }
    }
}
```

```

cmd.Parameters.AddWithValue("@Department_name", txtdepartment_name.Text);

cmd.Parameters.AddWithValue("@GPA", txtgpa.Text);

cmd.Parameters.AddWithValue("@GraduationYear", txtgraduationyear.Text);

cmd.Parameters.AddWithValue("@Course_name", txtcoursename.Text);

cmd.Parameters.AddWithValue("@gender", txt_gnder.Text);

cmd.Parameters.AddWithValue("@final_result", txtfinalresult.Text);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Record Updated Successfully");

ClearFields(); // clear input fields after updating

GetStudentRecord(); // refresh DataGridView to show updated data}}}

```

The screenshot shows a Windows application titled "Student Management System". The interface includes fields for StudentID (9179), Department\_name (Software Engineering), Name (Taqi), Age (30), Email (Taqi@gmail.com), GPA (3.0), GraduationYear (2028), and final\_result (pass). A modal dialog box is displayed in the center, stating "Record Updated Successfully" with an "OK" button. The application has three main buttons at the bottom: "INSERT", "DELETE", and "UPDATE" (which is highlighted in blue).

	StudentID	Name	Age	Email	Department_name	GPA
	3274	Catherine Velasq...	22	fsutton@gmail.com	Physics	2.71
	3609	Joyce Thomas	19	zanderson@barr...	Chemistry	2.22
	9179	Taqi	30	Taqi@gmail.com	SoftwareEnginee...	3

## Code Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Xml.Linq;
using static System.Windows.Forms.VisualStyles.VisualStudioElement.ListView;
using System.Text.RegularExpressions;
namespace Studentform
{
    public partial class Form1 : Form
    {
        private void ClearFields()
        {
            txtID.Clear();
            txtname.Clear();
            txtage.Clear();
        }
    }
}
```

```
txtemail.Clear();

txtdepartment_name.Clear();

txtgpa.Clear();

txtgraduationyear.Clear();

txtcoursename.Clear();

txt_gnder.Clear();

txtfinalresult.Clear();

}

public Form1()

{

InitializeComponent();

GetStudentRecord();

}

private void GetStudentRecord()

{

// Define the connection string

SqlConnection con = new SqlConnection("Data Source=DESKTOP-BD6E7FI\\SQLEXPRESS;Initial Catalog=SMSS;Integrated Security=True;Encrypt=False");

// Define the SQL query

SqlCommand cmd = new SqlCommand("SELECT * FROM Student", con);

// Initialize a DataTable to hold the query results

DataTable dt = new DataTable();

try
```

```
{  
    // Open the connection  
  
    con.Open();  
  
    // Execute the query and load the results into the DataTable  
  
    SqlDataReader sdr = cmd.ExecuteReader();  
  
    dt.Load(sdr);  
  
}  
  
catch (Exception ex)  
  
{  
  
    // Handle any errors that may have occurred  
  
    MessageBox.Show("An error occurred: " + ex.Message);  
  
}  
  
finally  
  
{  
  
    // Close the connection  
  
    con.Close();  
  
}  
  
// Bind the DataTable to the DataGridView  
  
StudentdataGridView1.DataSource = dt;  
  
}  
  
private void textBox1_TextChanged(object sender, EventArgs e)  
  
{  
  
}  
  
private void textBox6_TextChanged(object sender, EventArgs e)
```

```
{  
}  
  
private void label9_Click(object sender, EventArgs e)  
  
{  
}  
  
private void textBox5_TextChanged(object sender, EventArgs e)  
  
{  
}  
  
private void textBox7_TextChanged(object sender, EventArgs e)  
  
{  
}  
  
private void button3_Click(object sender, EventArgs e)  
  
{  
}  
  
private void button1_Click(object sender, EventArgs e)  
  
{  
  
    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-  
BD6E7FI\\SQLEXPRESS;Initial Catalog=SMSS;Integrated Security=True;Encrypt=False"))  
  
    {  
  
        string Query = @"INSERT INTO Student  
(StudentID, Name, Age, Email, Department_name, GPA, GraduationYear, Course_name, gender,  
final_result)  
  
VALUES  
(@StudentID, @Name, @Age, @Email, @Department_name, @GPA, @GraduationYear,  
@Course_name, @gender, @final_result)";  
    }  
}
```

```
using (SqlCommand cmd = new SqlCommand(Query, con))

{
    cmd.Parameters.AddWithValue("@StudentID", txtID.Text);
    cmd.Parameters.AddWithValue("@Name", txtname.Text);
    cmd.Parameters.AddWithValue("@Age", txtage.Text);
    cmd.Parameters.AddWithValue("@Email", txtemail.Text);
    cmd.Parameters.AddWithValue("@GPA", txtgpa.Text);
    cmd.Parameters.AddWithValue("@Department_name", txtdepartment_name.Text);
    cmd.Parameters.AddWithValue("@GraduationYear", txtgraduationyear.Text);
    cmd.Parameters.AddWithValue("@Course_name", txtcoursename.Text);
    cmd.Parameters.AddWithValue("@gender", txt_gnder.Text);
    cmd.Parameters.AddWithValue("@final_result", txtfinalresult.Text);

    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();

    MessageBox.Show("Record Inserted Successfully");

    ClearFields(); // Clear fields after insertion

    GetStudentRecord(); // refresh DataGridView to show new data
}

}

private void Form1_Load(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{
```

```
        using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
BD6E7FI\\SQLEXPRESS;Initial Catalog=SMSS;Integrated Security=True;TrustServerCertificate=True"))

    {

        string query = "DELETE FROM Student WHERE StudentID = @StudentID";

        using (SqlCommand cmd = new SqlCommand(query, con))

    {

        cmd.Parameters.AddWithValue("@StudentID", txtID.Text);

        con.Open();

        cmd.ExecuteNonQuery();

        con.Close();

        MessageBox.Show("Record Deleted Successfully");

        ClearFields(); // clear input fields after deletion

        GetStudentRecord(); // refresh DataGridView to show updated records

    }

}

    
```

```
private void button4_Click(object sender, EventArgs e)

{

    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
BD6E7FI\\SQLEXPRESS;Initial Catalog=SMSS;Integrated Security=True;TrustServerCertificate=True"))

    {

        string query = @"UPDATE Student

SET Name = @Name,

Age = @Age,

Email = @Email,

Department_name = @Department_name,

GPA = @GPA,
```

```
GraduationYear = @GraduationYear,  
Course_name = @Course_name,  
gender = @gender,  
final_result = @final_result  
  
WHERE StudentID = @StudentID";  
  
using (SqlCommand cmd = new SqlCommand(query, con))  
{  
    cmd.Parameters.AddWithValue("@StudentID", txtID.Text);  
    cmd.Parameters.AddWithValue("@Name", txtname.Text);  
    cmd.Parameters.AddWithValue("@Age", txtage.Text);  
    cmd.Parameters.AddWithValue("@Email", txtemail.Text);  
    cmd.Parameters.AddWithValue("@Department_name", txtdepartment_name.Text);  
    cmd.Parameters.AddWithValue("@GPA", txtgpa.Text);  
    cmd.Parameters.AddWithValue("@GraduationYear", txtgraduationyear.Text);  
    cmd.Parameters.AddWithValue("@Course_name", txtcoursename.Text);  
    cmd.Parameters.AddWithValue("@gender", txt_gnder.Text);  
    cmd.Parameters.AddWithValue("@final_result", txtfinalresult.Text);  
    con.Open();  
    cmd.ExecuteNonQuery();  
    con.Close();  
    MessageBox.Show("Record Updated Successfully");  
    ClearFields(); // clear input fields after updating  
    GetStudentRecord(); // refresh DataGridView to show updated data  
}
```