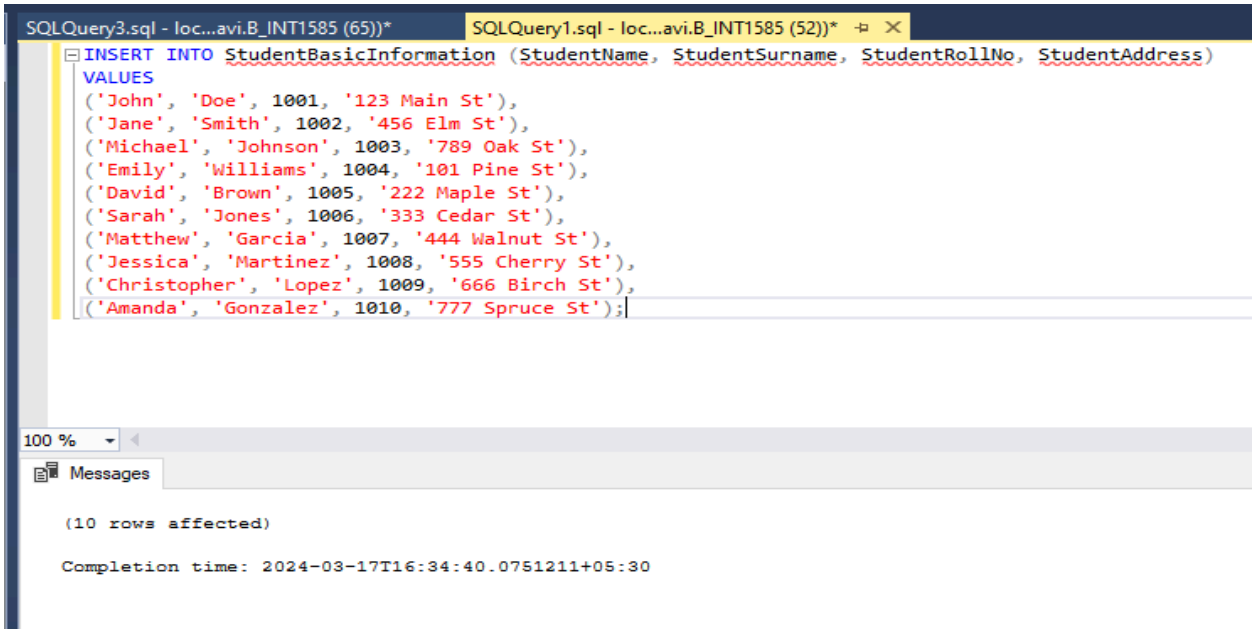
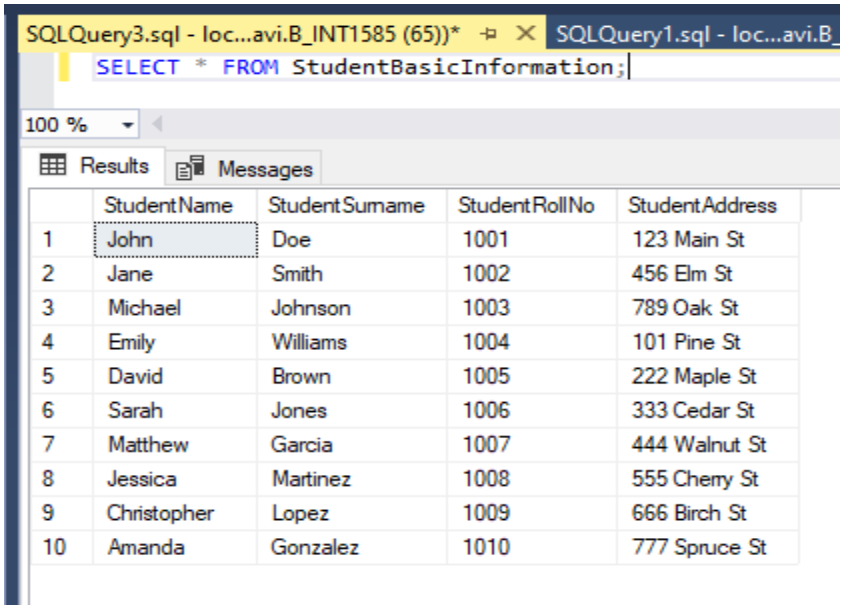
Data has been randomly chosen from the Internet because I forgot the website from where we were supposed to take the data from and hence some questions might have different results.

3. INSERTION INTO THE 4 TABLES

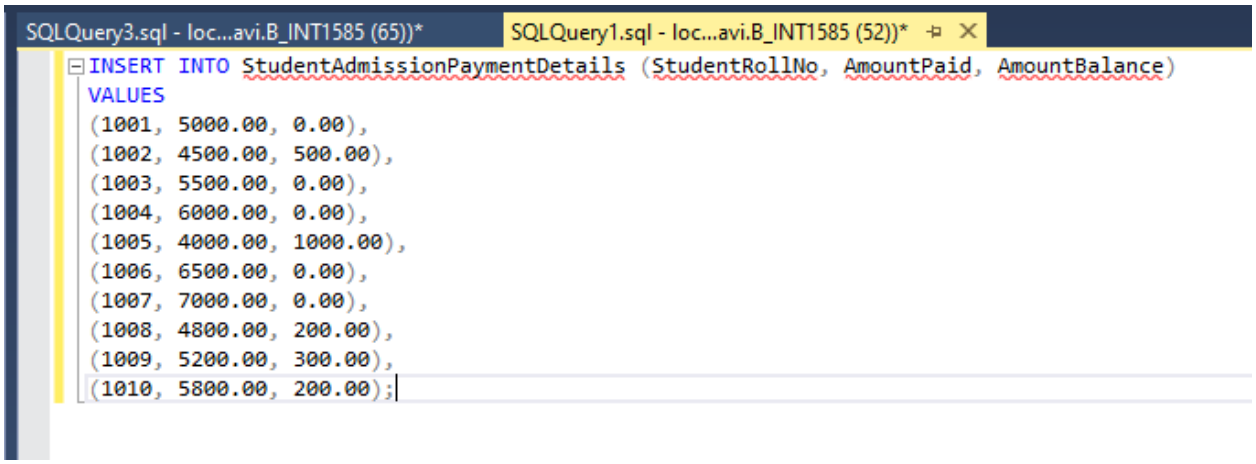
(StudentBasicInformation)

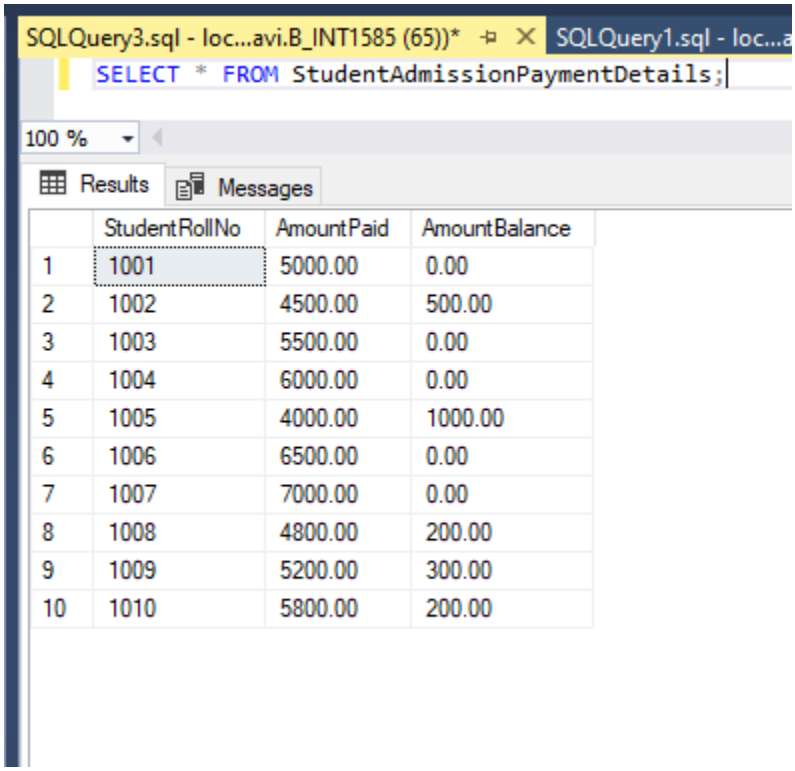




	StudentName	StudentSurname	StudentRollNo	StudentAddress
1	John	Doe	1001	123 Main St
2	Jane	Smith	1002	456 Elm St
3	Michael	Johnson	1003	789 Oak St
4	Emily	Williams	1004	101 Pine St
5	David	Brown	1005	222 Maple St
6	Sarah	Jones	1006	333 Cedar St
7	Matthew	Garcia	1007	444 Walnut St
8	Jessica	Martinez	1008	555 Cherry St
9	Christopher	Lopez	1009	666 Birch St
10	Amanda	Gonzalez	1010	777 Spruce St

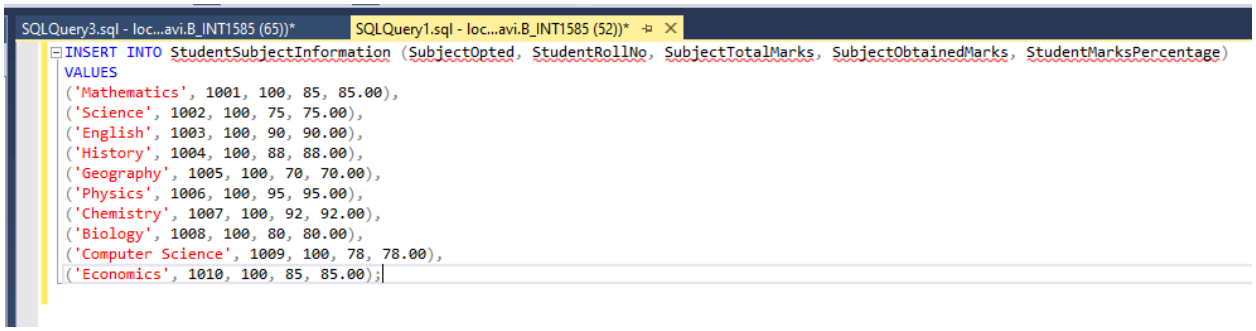
(StudentAdmissionPaymentDetails)

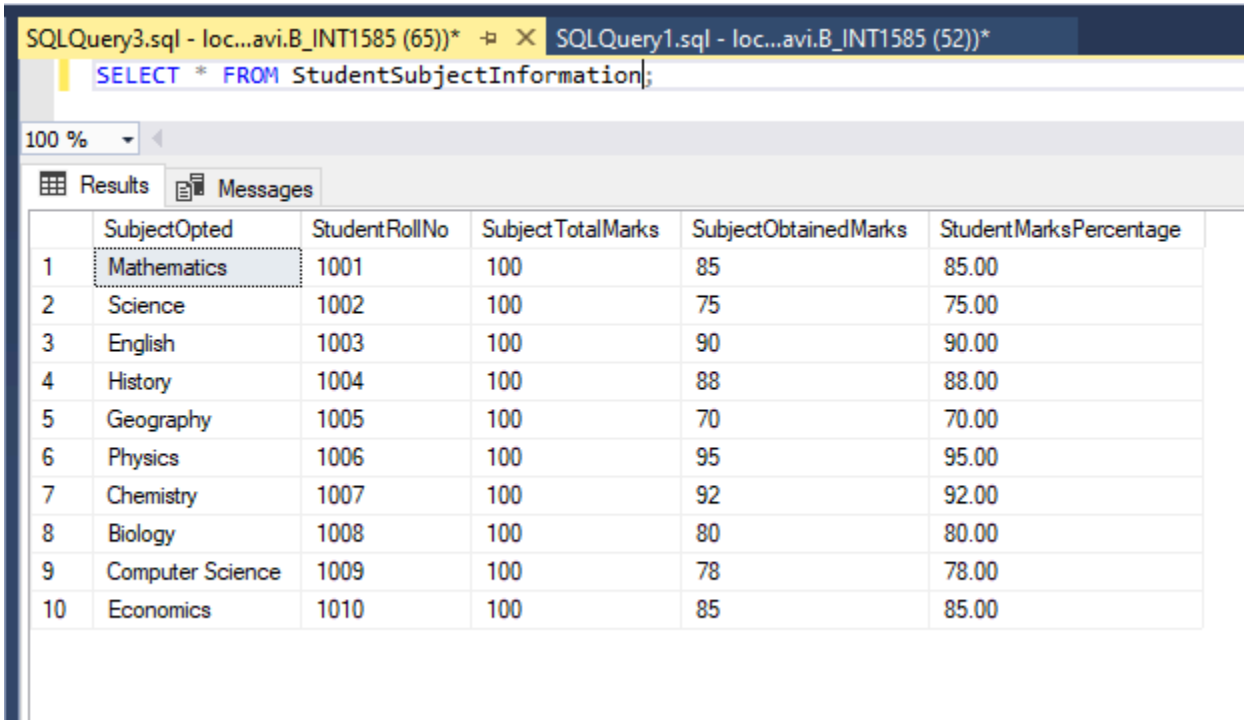




	StudentRollNo	AmountPaid	AmountBalance
1	1001	5000.00	0.00
2	1002	4500.00	500.00
3	1003	5500.00	0.00
4	1004	6000.00	0.00
5	1005	4000.00	1000.00
6	1006	6500.00	0.00
7	1007	7000.00	0.00
8	1008	4800.00	200.00
9	1009	5200.00	300.00
10	1010	5800.00	200.00

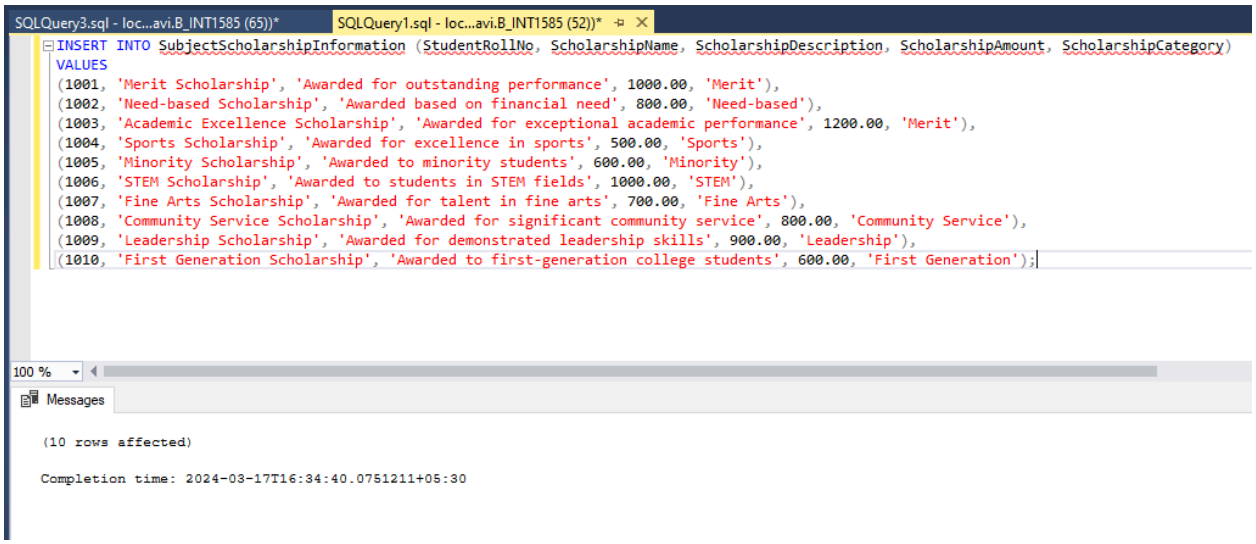
(StudentSubjectInformation)





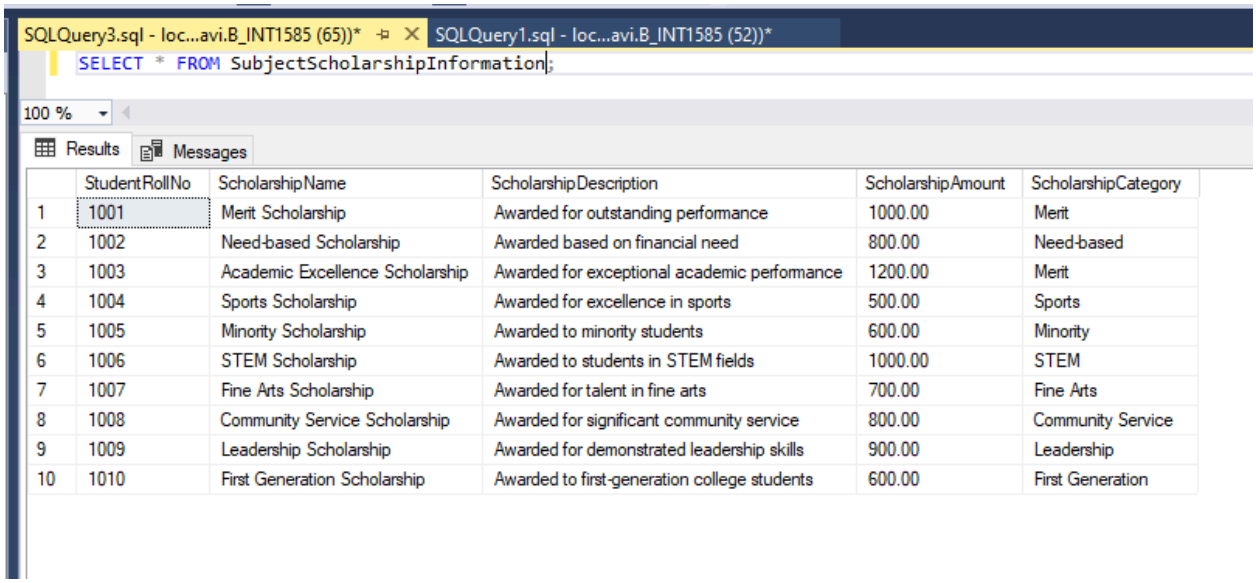
	SubjectOpted	StudentRollNo	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage
1	Mathematics	1001	100	85	85.00
2	Science	1002	100	75	75.00
3	English	1003	100	90	90.00
4	History	1004	100	88	88.00
5	Geography	1005	100	70	70.00
6	Physics	1006	100	95	95.00
7	Chemistry	1007	100	92	92.00
8	Biology	1008	100	80	80.00
9	Computer Science	1009	100	78	78.00
10	Economics	1010	100	85	85.00

(SubjectScholarshipInformation)



```
INSERT INTO SubjectScholarshipInformation (StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount, ScholarshipCategory)
VALUES
(1001, 'Merit Scholarship', 'Awarded for outstanding performance', 1000.00, 'Merit'),
(1002, 'Need-based Scholarship', 'Awarded based on financial need', 800.00, 'Need-based'),
(1003, 'Academic Excellence Scholarship', 'Awarded for exceptional academic performance', 1200.00, 'Merit'),
(1004, 'Sports Scholarship', 'Awarded for excellence in sports', 500.00, 'Sports'),
(1005, 'Minority Scholarship', 'Awarded to minority students', 600.00, 'Minority'),
(1006, 'STEM Scholarship', 'Awarded to students in STEM fields', 1000.00, 'STEM'),
(1007, 'Fine Arts Scholarship', 'Awarded for talent in fine arts', 700.00, 'Fine Arts'),
(1008, 'Community Service Scholarship', 'Awarded for significant community service', 800.00, 'Community Service'),
(1009, 'Leadership Scholarship', 'Awarded for demonstrated leadership skills', 900.00, 'Leadership'),
(1010, 'First Generation Scholarship', 'Awarded to first-generation college students', 600.00, 'First Generation');
```

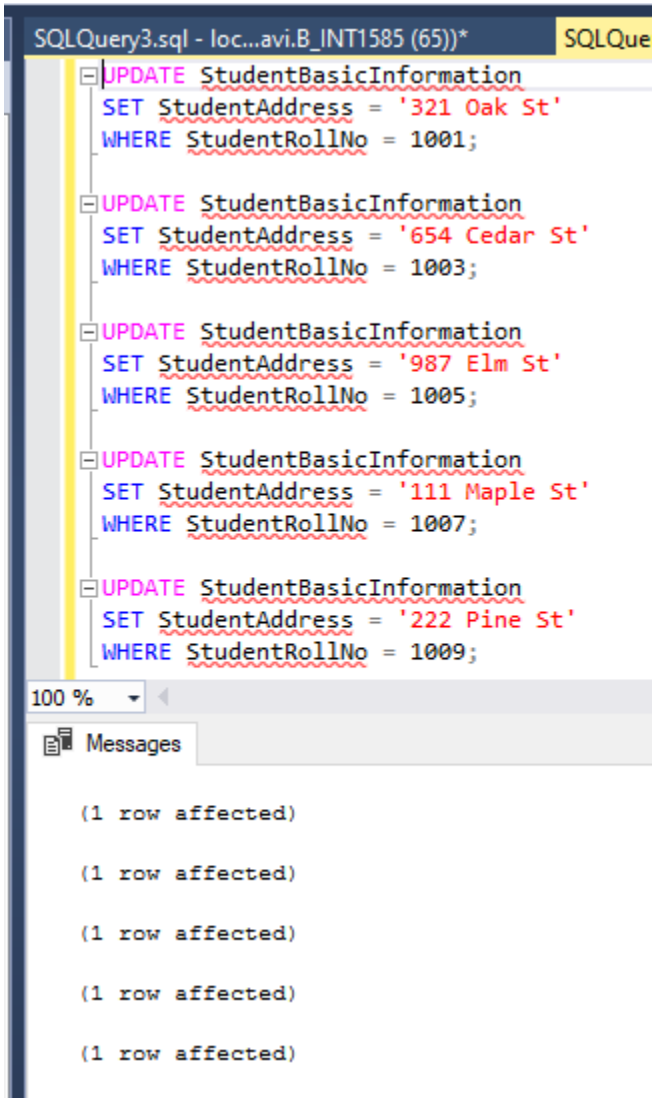
```
(10 rows affected)

Completion time: 2024-03-17T16:34:40.0751211+05:30
```



```
SELECT * FROM SubjectScholarshipInformation;
```

	StudentRollNo	ScholarshipName	ScholarshipDescription	ScholarshipAmount	ScholarshipCategory
1	1001	Merit Scholarship	Awarded for outstanding performance	1000.00	Merit
2	1002	Need-based Scholarship	Awarded based on financial need	800.00	Need-based
3	1003	Academic Excellence Scholarship	Awarded for exceptional academic performance	1200.00	Merit
4	1004	Sports Scholarship	Awarded for excellence in sports	500.00	Sports
5	1005	Minority Scholarship	Awarded to minority students	600.00	Minority
6	1006	STEM Scholarship	Awarded to students in STEM fields	1000.00	STEM
7	1007	Fine Arts Scholarship	Awarded for talent in fine arts	700.00	Fine Arts
8	1008	Community Service Scholarship	Awarded for significant community service	800.00	Community Service
9	1009	Leadership Scholarship	Awarded for demonstrated leadership skills	900.00	Leadership
10	1010	First Generation Scholarship	Awarded to first-generation college students	600.00	First Generation

4. Update any 5 records of your choice in any table like update the StudentAddress with some other address content and likewise so on with any records of any table of your choice and add snapshots of all the tables post update.



SQLQuery3.sql - loc...avi.B_INT1585 (65))* | SQLQuery1.sql - loc...avi.B_INT15

```
SELECT * FROM StudentBasicInformation;
```

100 %

Results | Messages

	StudentName	StudentSurname	StudentRollNo	StudentAddress
1	John	Doe	1001	321 Oak St
2	Jane	Smith	1002	456 Elm St
3	Michael	Johnson	1003	654 Cedar St
4	Emily	Williams	1004	101 Pine St
5	David	Brown	1005	987 Elm St
6	Sarah	Jones	1006	333 Cedar St
7	Matthew	Garcia	1007	111 Maple St
8	Jessica	Martinez	1008	555 Cherry St
9	Christopher	Lopez	1009	222 Pine St
10	Amanda	Gonzalez	1010	777 Spruce St

5. Select the student details records who has received the scholarship more than 800

SQLQuery3.sql - loc...avi.B_INT1585 (65))* | SQLQuery1.sql - loc...avi.B_INT1585 (52))*

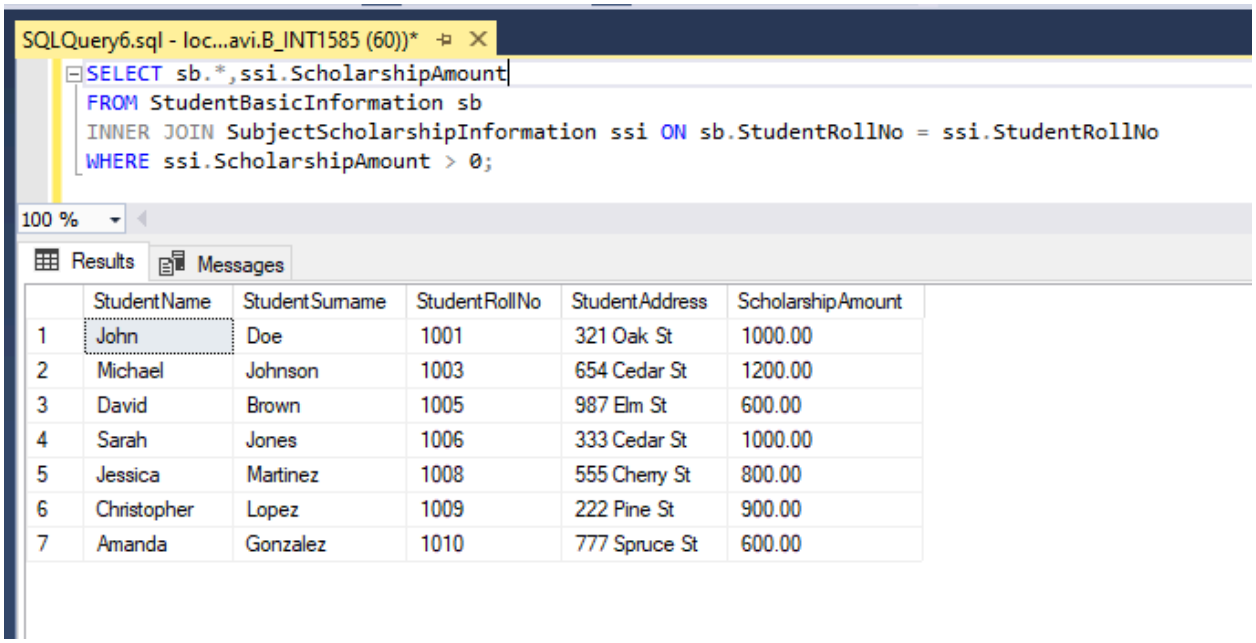
```
SELECT sb.StudentName, sb.StudentSurname, sb.StudentRollNo, sb.StudentAddress, ssi.ScholarshipName, ssi.ScholarshipCategory, ssi.ScholarshipAmount
FROM StudentBasicInformation sb
JOIN SubjectScholarshipInformation ssi ON sb.StudentRollNo = ssi.StudentRollNo
WHERE ssi.ScholarshipAmount > 800;
```

100 %

Results | Messages

	StudentName	StudentSurname	StudentRollNo	StudentAddress	ScholarshipName	ScholarshipCategory	ScholarshipAmount
1	John	Doe	1001	321 Oak St	Merit Scholarship	Merit	1000.00
2	Michael	Johnson	1003	654 Cedar St	Academic Excellence Scholarship	Merit	1200.00
3	Sarah	Jones	1006	333 Cedar St	STEM Scholarship	STEM	1000.00
4	Christopher	Lopez	1009	222 Pine St	Leadership Scholarship	Leadership	900.00

6. Select the students who opted for scholarship but did not get the scholarship.



	StudentName	StudentSurname	StudentRollNo	StudentAddress	ScholarshipAmount
1	John	Doe	1001	321 Oak St	1000.00
2	Michael	Johnson	1003	654 Cedar St	1200.00
3	David	Brown	1005	987 Elm St	600.00
4	Sarah	Jones	1006	333 Cedar St	1000.00
5	Jessica	Martinez	1008	555 Cherry St	800.00
6	Christopher	Lopez	1009	222 Pine St	900.00
7	Amanda	Gonzalez	1010	777 Spruce St	600.00

7. Fill in data for the percentage column i.e. StudentMarksPercentage in the table StudentSubjectInformation by creating and using the stored procedure created.

SQLQuery7.sql - loc...avi.B_INT1585 (52))* SQLQuery6.sql - loc...avi.B_INT1585 (60))*

```
CREATE PROCEDURE CalculateStudentMarksPercentage
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE StudentSubjectInformation
    SET StudentMarksPercentage = (SubjectObtainedMarks * 100.0) / NULLIF(SubjectTotalMarks, 0);
END;

EXEC CalculateStudentMarksPercentage;

SELECT * FROM StudentSubjectInformation;
```

100 %

Results Messages

	SubjectOpted	StudentRollNo	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage
1	Mathematics	1001	100	85	85.00
2	Science	1002	100	75	75.00
3	English	1003	100	90	90.00
4	History	1004	100	88	88.00
5	Geography	1005	100	70	70.00
6	Physics	1006	100	95	95.00
7	Chemistry	1007	100	92	92.00
8	Biology	1008	100	80	80.00
9	Computer Science	1009	100	78	78.00
10	Economics	1010	100	85	85.00

8. Decide the category of the scholarship depending upon the marks/percentage obtained by the student and likewise update the ScholarshipCategory column, create a stored procedure in order to handle this operation.

SQLQuery7.sql - loc...avi.B_INT1585 (52))* | SQLQuery6.sql - loc...avi.B_INT1585 (60))*

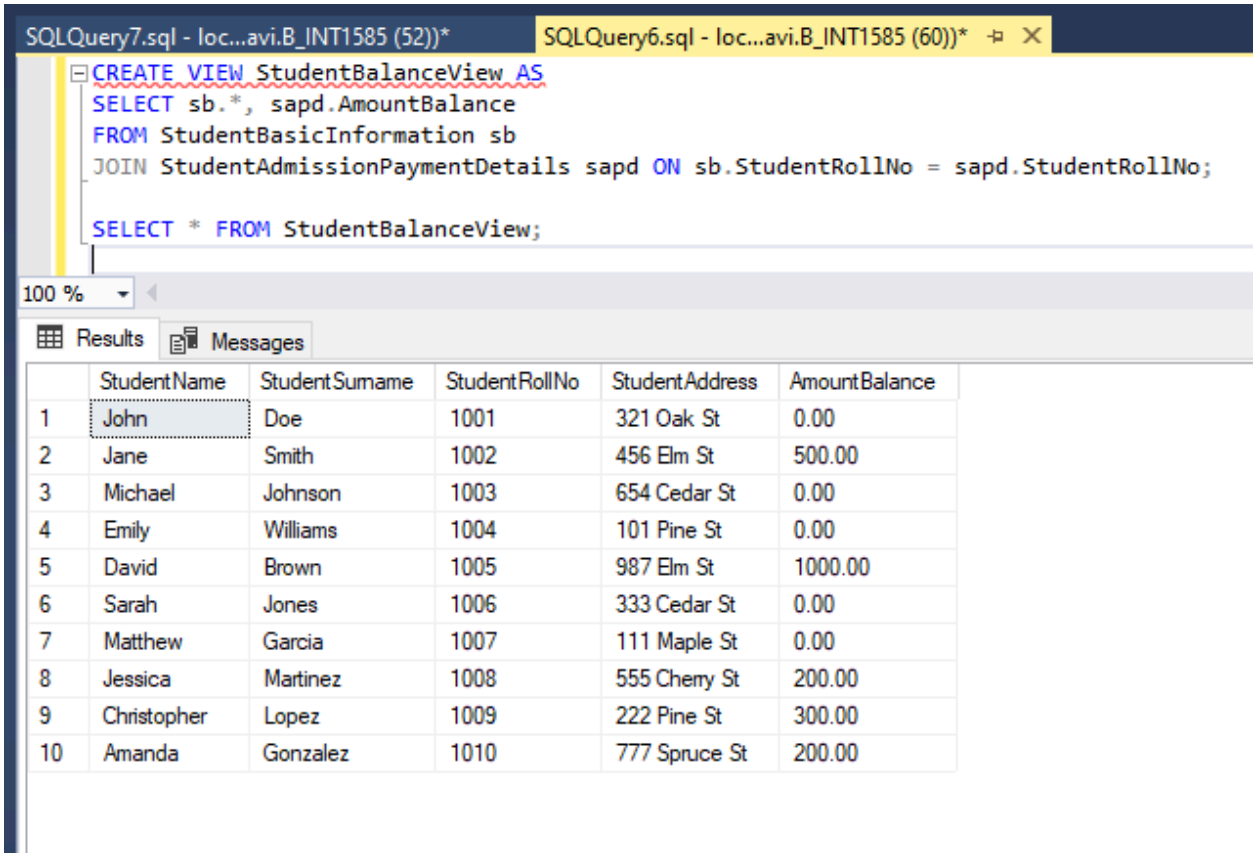
```
CREATE PROCEDURE DetermineScholarshipCategory
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE SubjectScholarshipInformation
    SET ScholarshipCategory =
        CASE
            WHEN ssi.StudentMarksPercentage >= 90 THEN 'High Achiever'
            WHEN ssi.StudentMarksPercentage >= 80 THEN 'Excellent'
            WHEN ssi.StudentMarksPercentage >= 70 THEN 'Good'
            WHEN ssi.StudentMarksPercentage >= 60 THEN 'Average'
            ELSE 'No Scholarship'
        END
    FROM SubjectScholarshipInformation ssi1
    JOIN StudentSubjectInformation ssi ON ssi.StudentRollNo = ssi1.StudentRollNo;
END;
```

100 %

Results | Messages

	StudentRollNo	ScholarshipName	ScholarshipDescription	ScholarshipAmount	ScholarshipCategory
1	1001	Merit Scholarship	Awarded for outstanding performance	1000.00	Excellent
2	1002	Need-based Scholarship	Awarded based on financial need	0.00	Good
3	1003	Academic Excellence Scholarship	Awarded for exceptional academic performance	1200.00	High Achiever
4	1004	Sports Scholarship	Awarded for excellence in sports	0.00	Excellent
5	1005	Minority Scholarship	Awarded to minority students	600.00	Good
6	1006	STEM Scholarship	Awarded to students in STEM fields	1000.00	High Achiever
7	1007	Fine Arts Scholarship	Awarded for talent in fine arts	0.00	High Achiever
8	1008	Community Service Scholarship	Awarded for significant community service	800.00	Excellent
9	1009	Leadership Scholarship	Awarded for demonstrated leadership skills	900.00	Good
10	1010	First Generation Scholarship	Awarded to first-generation college students	600.00	Excellent

9.Create a View which shows the balance amount to be paid by the student along with the student's detailed information. (use join)



SQLQuery7.sql - loc...avi.B_INT1585 (52))* | SQLQuery6.sql - loc...avi.B_INT1585 (60))*

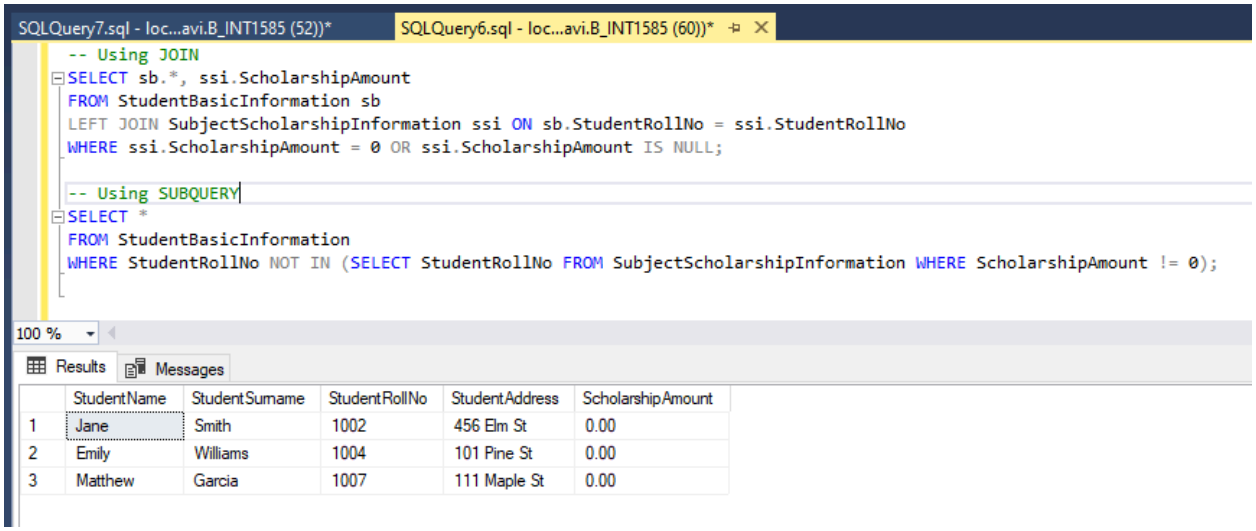
```
CREATE VIEW StudentBalanceView AS
SELECT sb.*, sapd.AmountBalance
FROM StudentBasicInformation sb
JOIN StudentAdmissionPaymentDetails sapd ON sb.StudentRollNo = sapd.StudentRollNo;

SELECT * FROM StudentBalanceView;
```

100 %

Results | Messages

	StudentName	StudentSurname	StudentRollNo	StudentAddress	AmountBalance
1	John	Doe	1001	321 Oak St	0.00
2	Jane	Smith	1002	456 Elm St	500.00
3	Michael	Johnson	1003	654 Cedar St	0.00
4	Emily	Williams	1004	101 Pine St	0.00
5	David	Brown	1005	987 Elm St	1000.00
6	Sarah	Jones	1006	333 Cedar St	0.00
7	Matthew	Garcia	1007	111 Maple St	0.00
8	Jessica	Martinez	1008	555 Cherry St	200.00
9	Christopher	Lopez	1009	222 Pine St	300.00
10	Amanda	Gonzalez	1010	777 Spruce St	200.00

10. Get the details of the students who haven't got any scholarship. (use joins/subqueries)



SQLQuery7.sql - loc...avi.B_INT1585 (52))* | SQLQuery6.sql - loc...avi.B_INT1585 (60))*

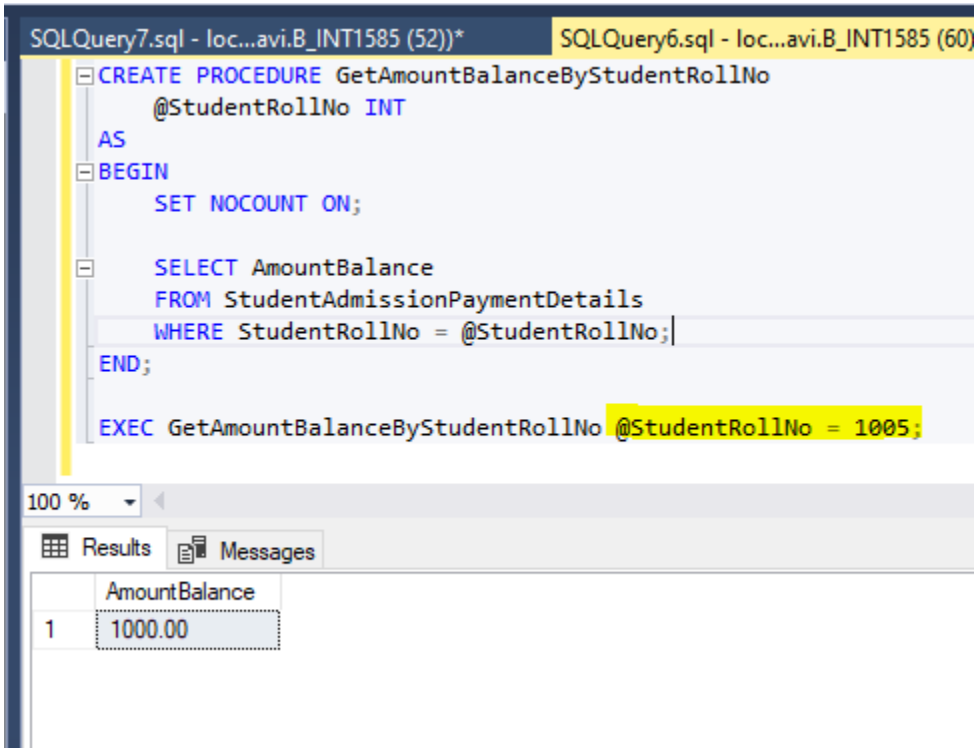
```
-- Using JOIN
SELECT sb.*, ssi.ScholarshipAmount
FROM StudentBasicInformation sb
LEFT JOIN SubjectScholarshipInformation ssi ON sb.StudentRollNo = ssi.StudentRollNo
WHERE ssi.ScholarshipAmount = 0 OR ssi.ScholarshipAmount IS NULL;

-- Using SUBQUERY
SELECT *
FROM StudentBasicInformation
WHERE StudentRollNo NOT IN (SELECT StudentRollNo FROM SubjectScholarshipInformation WHERE ScholarshipAmount != 0);
```

100 %

Results | Messages

	StudentName	StudentSurname	StudentRollNo	StudentAddress	ScholarshipAmount
1	Jane	Smith	1002	456 Elm St	0.00
2	Emily	Williams	1004	101 Pine St	0.00
3	Matthew	Garcia	1007	111 Maple St	0.00

11. Create Stored Procedure which will return the amount balance to be paid by the student as per the student roll number passed through the stored procedure as the input parameter.



12. Retrieve the top five student details as per the StudentMarksPercentage values. (use subqueries)

SQLQuery7.sql - loc...avi.B_INT1585 (52))* | SQLQuery6.sql - loc...avi.B_INT1585 (60))*

```
SELECT sb.*, ssi.StudentMarksPercentage
FROM StudentBasicInformation sb
JOIN (
    SELECT TOP 5 StudentRollNo, StudentMarksPercentage
    FROM StudentSubjectInformation
    ORDER BY StudentMarksPercentage DESC
) AS top_students ON sb.StudentRollNo = top_students.StudentRollNo
JOIN StudentSubjectInformation ssi ON sb.StudentRollNo = ssi.StudentRollNo;
```

100 %

Results | Messages

	StudentName	StudentSurname	StudentRollNo	StudentAddress	StudentMarksPercentage
1	John	Doe	1001	321 Oak St	85.00
2	Michael	Johnson	1003	654 Cedar St	90.00
3	Emily	Williams	1004	101 Pine St	88.00
4	Sarah	Jones	1006	333 Cedar St	95.00
5	Matthew	Garcia	1007	111 Maple St	92.00

13. Write a query using Group By to retrieve roll numbers on the basis of ScholarshipName. (I had unique ScholarshipNames so I'm grouping on the basis of ScholarshipCategory)

SQLQuery7.sql - loc...avi.B_INT1585 (52))* | SQLQuery6.sql -

```
SELECT StudentRollNo, ScholarshipCategory
FROM SubjectScholarshipInformation
GROUP BY ScholarshipCategory,StudentRollNo;
```

100 %

Results | Messages

	StudentRollNo	ScholarshipCategory
1	1001	Excellent
2	1004	Excellent
3	1008	Excellent
4	1010	Excellent
5	1002	Good
6	1005	Good
7	1009	Good
8	1003	High Achiever
9	1006	High Achiever
10	1007	High Achiever

14.Try to use the three types of join - INNER, LEFT, RIGHT learned today in a relevant way, and explain the same why you thought of using that particular join for your selected scenarios. (try to cover relevant and real time scenarios for all the three studied joins)

Consider a scenario where we have two tables:

Employees: Contains information about employees, including their ID, name, department ID, and salary.
Departments: Contains information about departments, including their ID, name, and location.

LEFT JOIN:

Scenario: We want to retrieve a list of all departments, along with the total number of employees in each department.
Reasoning: We use a LEFT JOIN because we want to include all departments in the result, even if there are no employees assigned to them. If a department has no employees, we still want to display the department details with a count of 0.
Explanation: This query performs a LEFT JOIN between the Departments table and the Employees table based on the ID and DepartmentID columns respectively. It ensures that all departments from the Departments table are included in the result, regardless of whether there are matching records in the Employees table. The COUNT function is used to calculate the number of employees in each department.

RIGHT JOIN:

Scenario: We want to retrieve a list of all employees, along with their department names. We also want to include departments that have no employees assigned to them.
Reasoning: We use a RIGHT JOIN because we want to include all departments in the result, even if there are no employees assigned to them. If a department has no employees, we still want to display the department details with a NULL value for employee-related columns.
Explanation: This query performs a RIGHT JOIN between the Employees table and the Departments table based on the DepartmentID and ID columns respectively. It ensures that all departments from the Departments table are included in the result, regardless of whether there are matching records in the Employees table. If a department has no matching employees, NULL values will be returned for employee-related columns.

INNER JOIN:

Scenario: We want to retrieve a list of employees along with their department names.
Reasoning: We use an INNER JOIN because we only want to retrieve employees who are assigned to a department. If an employee does not have an associated department, we do not want to include them in the result.

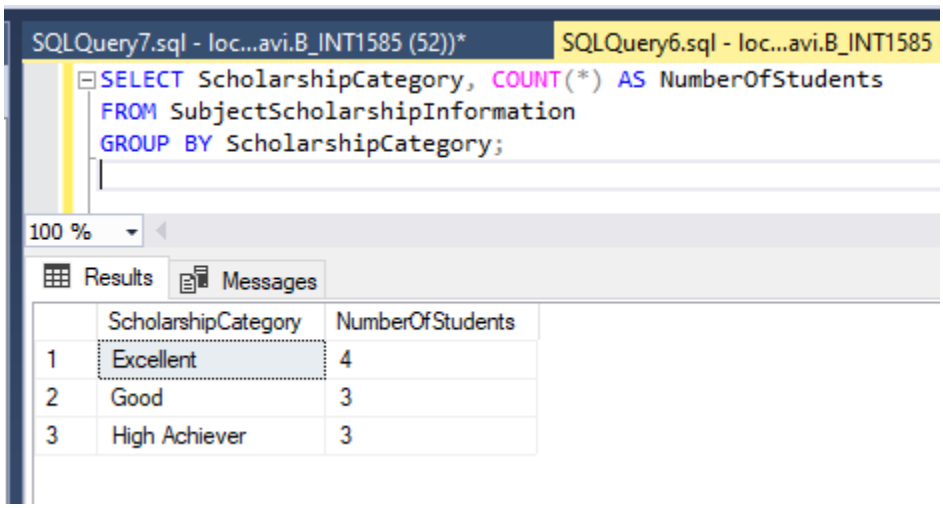
Explanation: This query joins the Employees table with the Departments table based on the DepartmentID column. It only includes rows where there is a match between the DepartmentID in the Employees table and the ID in the Departments table, ensuring that we only retrieve employees who are associated with a department.

15.Mention the differences between delete, drop and truncate commands with examples.

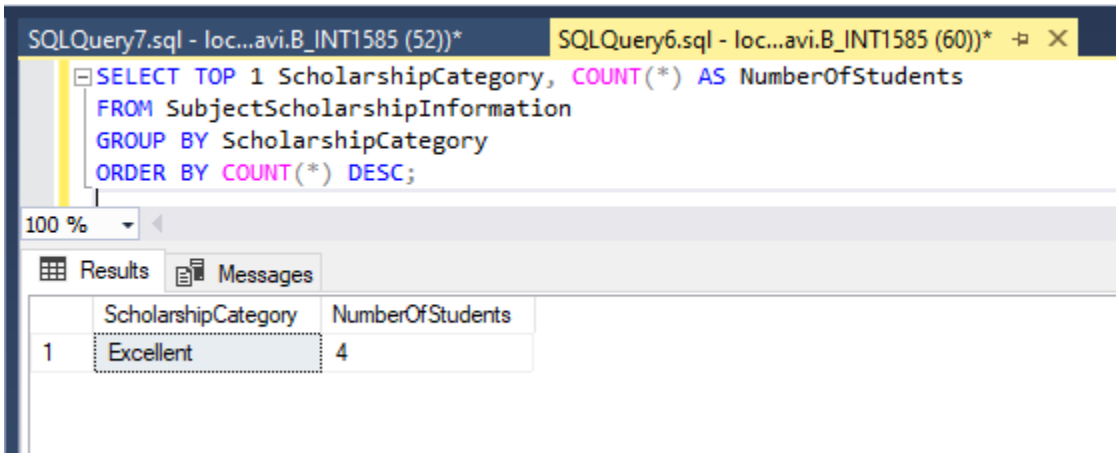
Key Differences:

- DELETE is used to remove specific rows from a table based on conditions, while TRUNCATE removes all rows from a table.
- DROP removes entire database objects (e.g., tables, views) from the database, while DELETE and TRUNCATE operate on rows within a table.
- TRUNCATE is faster than DELETE because it does not generate individual delete operations for each row, but it cannot be rolled back like DELETE.
- DROP permanently removes objects from the database, while DELETE and TRUNCATE only remove data from tables.

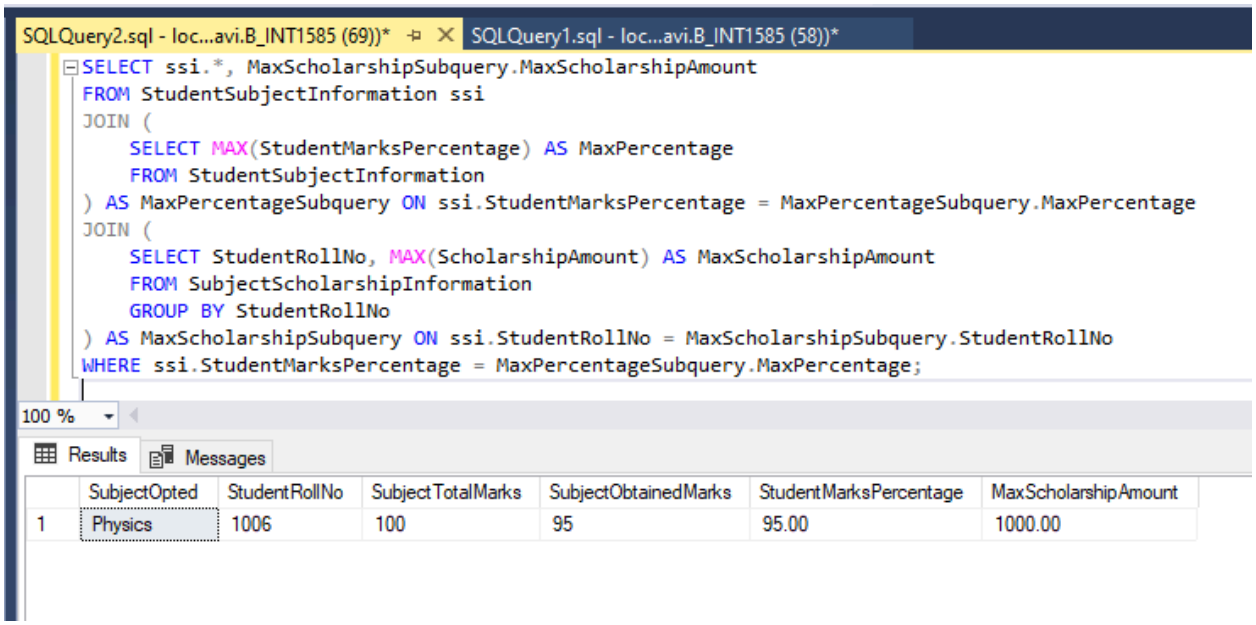
16.Get the count of the Scholarship category which is highly availed by the students, i.e. get the count of the total number of students corresponding to each scholarships category



17. Along with question no.16 try to retrieve the maximum used scholarship category.



18. Retrieve the percentage of the students along with detailed information who has scored the highest percentage along with availing the maximum scholarship amount.



19. Difference between the Triggers, Stored Procedures, Views and Functions with examples.

Here are the key differences between triggers, stored procedures, views, and functions:

Purpose:
Triggers: They are used to automatically execute a set of SQL statements in response to specific events or actions performed on a table, such as INSERT, UPDATE, or DELETE operations.
Stored Procedures: They are precompiled SQL statements stored in the database, designed to encapsulate business logic or commonly performed tasks, which can be executed repeatedly by calling their name.
Views: They are virtual tables that represent the result set of a SELECT query and are used to simplify complex queries, present data in a structured format, or restrict access to sensitive data.
Functions: They are reusable blocks of SQL code that accept parameters, perform calculations, and return a single value or a table.

Execution Context:
Triggers: They execute automatically in response to specific events or actions performed on a table, such as INSERT, UPDATE, or DELETE operations.
Stored Procedures: They are executed explicitly by calling their name from an application or another SQL statement.
Views: They are not executed directly; instead, they are queried like tables, and the underlying SELECT statement is executed when the view is referenced.
Functions: They are executed when called from SQL statements, such as SELECT queries or WHERE clauses.

Input/Output:
Triggers: They do not accept parameters directly but can access data from the triggering event.
Stored Procedures: They can accept input parameters and return output parameters or result sets.
Views: They do not accept parameters but provide a filtered or simplified representation of data from underlying tables.
Functions: They accept input parameters and can return a single value or a table.

Data Modification:
Triggers: They can modify data directly or perform additional actions based on the triggering event.
Stored Procedures: They can modify data, but the changes are explicit within the stored procedure's code.

Views: They cannot directly modify data as they are virtual representations of data from underlying tables.
Functions: They can modify data indirectly by performing calculations or transformations on input parameters.

Usage:
Triggers: They are used for enforcing business rules, maintaining data integrity, and performing auditing tasks.
Stored Procedures: They are used for encapsulating business logic, improving performance, and enhancing security.
Views: They are used for simplifying complex queries, presenting data in a structured format, and restricting access to sensitive data.
Functions: They are used for performing calculations, data transformations, and reusable computations.

20. Difference between clustered and non-clustered indexes.

(I've read the theory part and this is the best way to differentiate between them)

Aspect	Clustered Index	Non-Clustered Index
Organization of Data	Data is physically sorted on disk based on index	Index structure is separate from actual data
Number of Indexes	Only one per table	Multiple per table
Performance	Efficient for range-based queries	Efficient for specific value searches
Impact on Data Modification	May require reordering of data on disk	Data modifications typically do not reorder data
Index Size	Consumes space within the table	Consumes additional space
Use Cases	Range queries, sorting operations	Specific value searches, different access paths