STORED PROCEDURES

DATABASES

PEACE SALOMY PHIRI
ST10163788

WHAT IS A STORED PROCEDURE?

IN MYSQL

INTRODUCTION STORED PROCEDURE

- Stored procedures are a set of SQL statements that perform a specific task
- They are used to encapsulate a series of SQL statements into a single, reusable unit
- Stored procedures can accept input parameters and return output parameters.

BENEFITS OF USING STORED PROCEDURES

- Reusability: Stored procedures can be called multiple times, reducing the need to write the same code repeatedly
- Performance: Stored procedures are often pre-compiled, which can improve the performance of database operations
- Security: Stored procedures can help improve security by controlling access to database operations
- Modularity: Stored procedures allow you to break down complex database operations into smaller, more manageable units

CREATING A PROCEDURE

```
DELIMITER //
CREATE PROCEDURE procedure_name(
    IN param1 datatype,
    IN param2 datatype
)
BEGIN
    -- SQL statements
END //
DELIMITER ;
```

Explanation:

- The DELIMITER statement is used to change the default delimiter (usually semicolon) to double slash (//) to avoid conflicts with the SQL statements inside the procedure
- The CREATE PROCEDURE statement defines the procedure name and its parameters
- The BEGIN and END keywords wrap the SQL statements that make up the procedure's logic

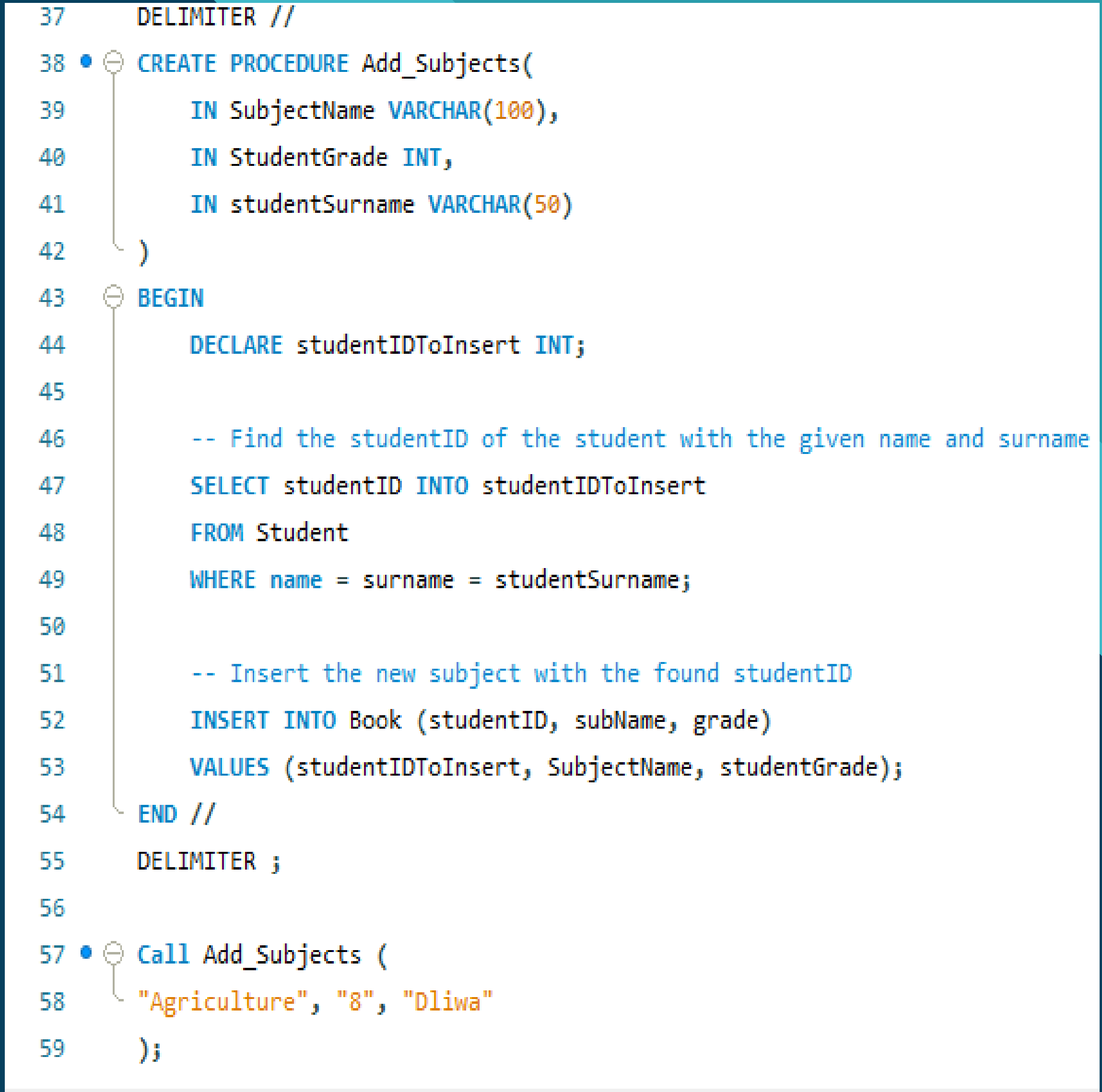
Example 1 of a Stored Procedure

The stored procedure is named "Add_Subjects".

- It takes three input parameters:
- SubjectName: a VARCHAR(100) parameter that represents the name of the new subject.
- StudentGrade: an INT parameter that represents the grade of the new subject.
- studentSurname: a VARCHAR(50) parameter that represents the surname of the student the new subject is associated with.

Stored Procedure Implementation

- The stored procedure starts by declaring a variable called studentIDToInsert of type INT.
- It then uses a SELECT statement to find the studentID of the student with the given studentSurname.



```
DELIMITER //
CREATE PROCEDURE Add_Subjects(
    IN SubjectName VARCHAR(100),
    IN StudentGrade INT,
    IN studentSurname VARCHAR(50)
)
BEGIN
    DECLARE studentIDToInsert INT;

    -- Find the studentID of the student with the given name and surname
    SELECT studentID INTO studentIDToInsert
    FROM Student
    WHERE name = surname = studentSurname;

    -- Insert the new subject with the found studentID
    INSERT INTO Book (studentID, subName, grade)
    VALUES (studentIDToInsert, SubjectName, studentGrade);
END //
DELIMITER ;

Call Add_Subjects (
"Agriculture", "8", "Dliwa"
);
```

Procedure Creation Continued…

```
SELECT studentID INTO studentIDToInsert
FROM Student
WHERE surname = studentSurname;
```

- This statement retrieves the studentID value and stores it in the studentIDToInsert variable.
- After finding the studentID, the stored procedure inserts a new record into the Subject table.

```
INSERT INTO Subject (studentID, subName, grade)
VALUES (studentIDToInsert, SubjectName, StudentGrade);
```

- The new subject record is inserted with the found studentIDToInsert value, the provided SubjectName, and the given StudentGrade.

Calling the Stored Procedure

- To use the stored procedure, you can call it using the CALL statement.

```
 CALL Add_Subjects(  "New Subject", 8,  "Dliwa"
);
```

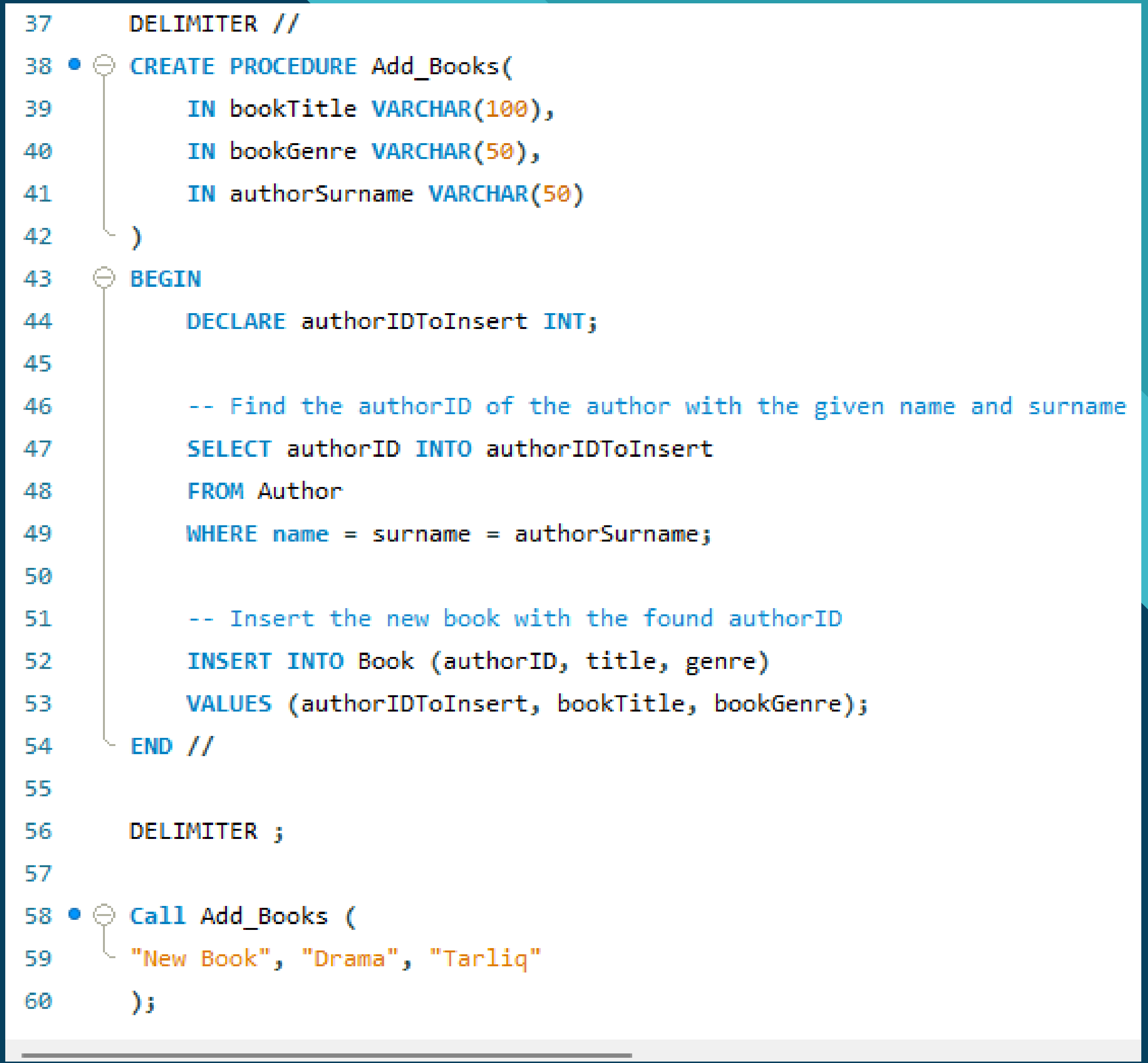
- This call to the Add_Subjects procedure will insert a new subject record with the name "New Subject", a grade of 8, and associate it with the student who has the surname "Dliwa".

The stored procedure is named "Add_Books".

- It takes three input parameters:
- bookTitle: a VARCHAR(100) parameter that represents the name of the new book.
- bookGenre: an INT parameter that represents the genre of the new book.
- authorSurname: a VARCHAR(50) parameter that represents the surname of the author the new book is associated with.

Stored Procedure Implementation

- The stored procedure starts by declaring a variable called authorIDToInsert of type INT.
- It then uses a SELECT statement to find the authorID of the author with the given authorSurname.



```
DELIMITER //
CREATE PROCEDURE Add_Books(
    IN bookTitle VARCHAR(100),
    IN bookGenre VARCHAR(50),
    IN authorSurname VARCHAR(50)
)
BEGIN
    DECLARE authorIDToInsert INT;

    -- Find the authorID of the author with the given name and surname
    SELECT authorID INTO authorIDToInsert
    FROM Author
    WHERE name = surname = authorSurname;

    -- Insert the new book with the found authorID
    INSERT INTO Book (authorID, title, genre)
    VALUES (authorIDToInsert, bookTitle, bookGenre);
END //

DELIMITER ;

Call Add_Books (
"New Book", "Drama", "Tarliq"
);
```

```
SELECT authorID INTO authorIDToInsert
FROM Author
WHERE surname = authorSurname;
```

- This statement retrieves the authorID value and stores it in the authorIDToInsert variable.
- After finding the authorID, the stored procedure inserts a new record into the Book table.

- INSERT INTO Book (authorID, authorName, genre)
- VALUES (authorIDToInsert, authorName, genre;

- The new book record is inserted with the found authorIDToInsert value, the provided authorName, and the given genre.

Calling the Stored Procedure

- To use the stored procedure, you can call it using the CALL statement.

```
 CALL Add_Books(  "New Book", Where Love Lies,  “Nazo"
);
```

- This call to the Add_Subjects procedure will insert a new subject record with the name "New Subject", a grade of 8, and associate it with the student who has the surname "Dliwa".

The End!

Thank You

Peace.