Activity No. 11. - Stored Procedures	
Name: Efa, Christian Guevarra, Hans Angelo Mendoza, John Renzo Nicolas, Sean Julian Vinluan, Armando	**Date:** 21/11/2022
Section: CPE21S3	**Instructor:** Dr. Jonathan Vidal Taylar

Objectives:

This activity aims to create and execute stored procedures in databases

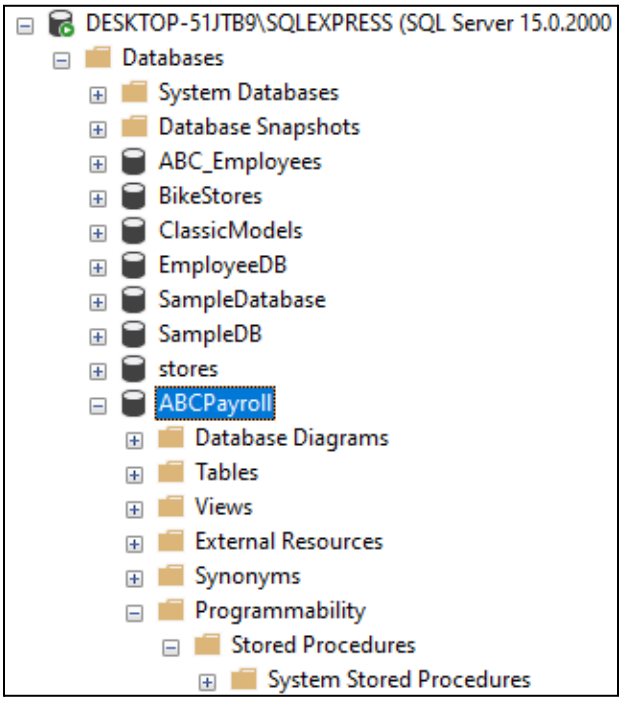
Intended Learning Outcomes (ILOs):

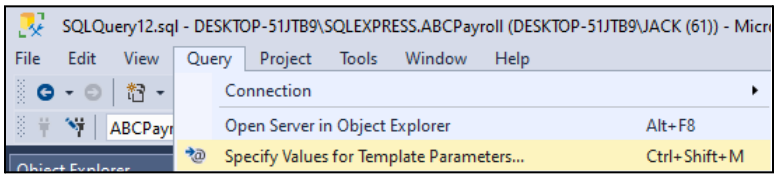
The students should be able to:
2.1 Create stored procedures using Management Studio and T - SQL.
2.2 Apply input parameters and return output data in stored procedures.
2.3 Implement and execute stored procedures

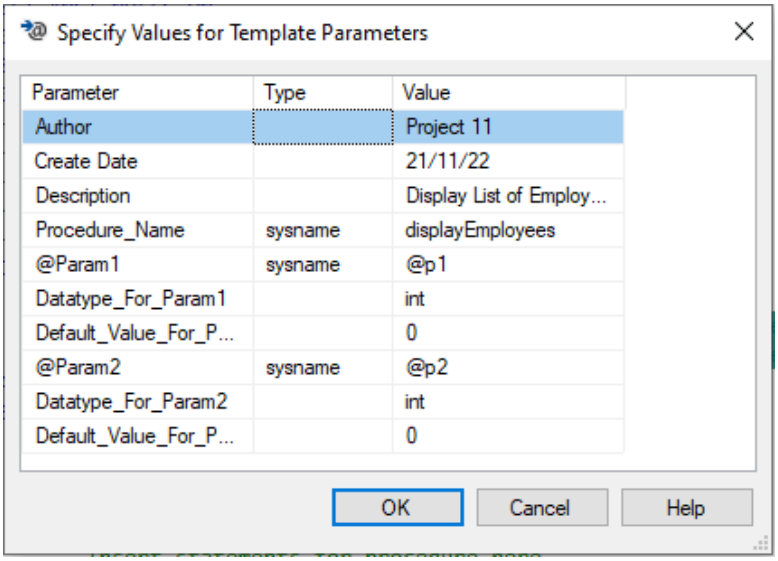
Output

Create a Stored Procedure

A. Using SQL Management Studio



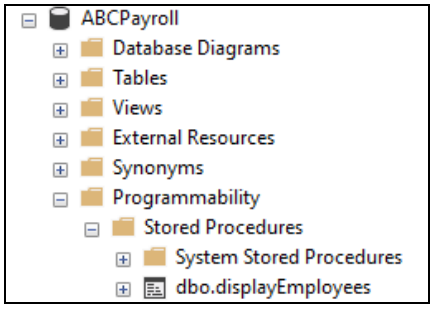




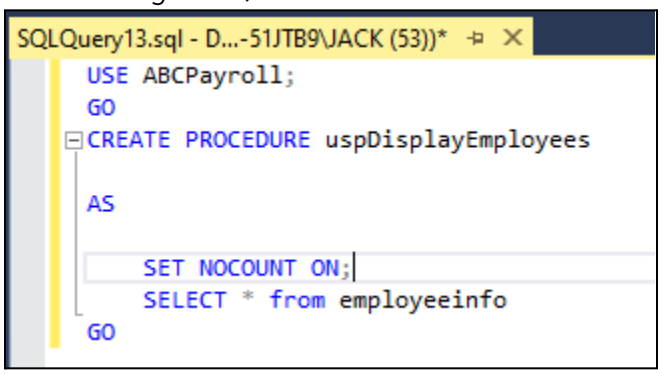

```
-- ================================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author:      Project 11
-- Create date: 21/11/22
-- Description: Display List of Employees
-- ============================================
CREATE PROCEDURE displayEmployees
    -- Add the parameters for the stored procedure here\

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT * from employeeinfo
END
GO
```

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T15:01:54.8169978+08:00
```

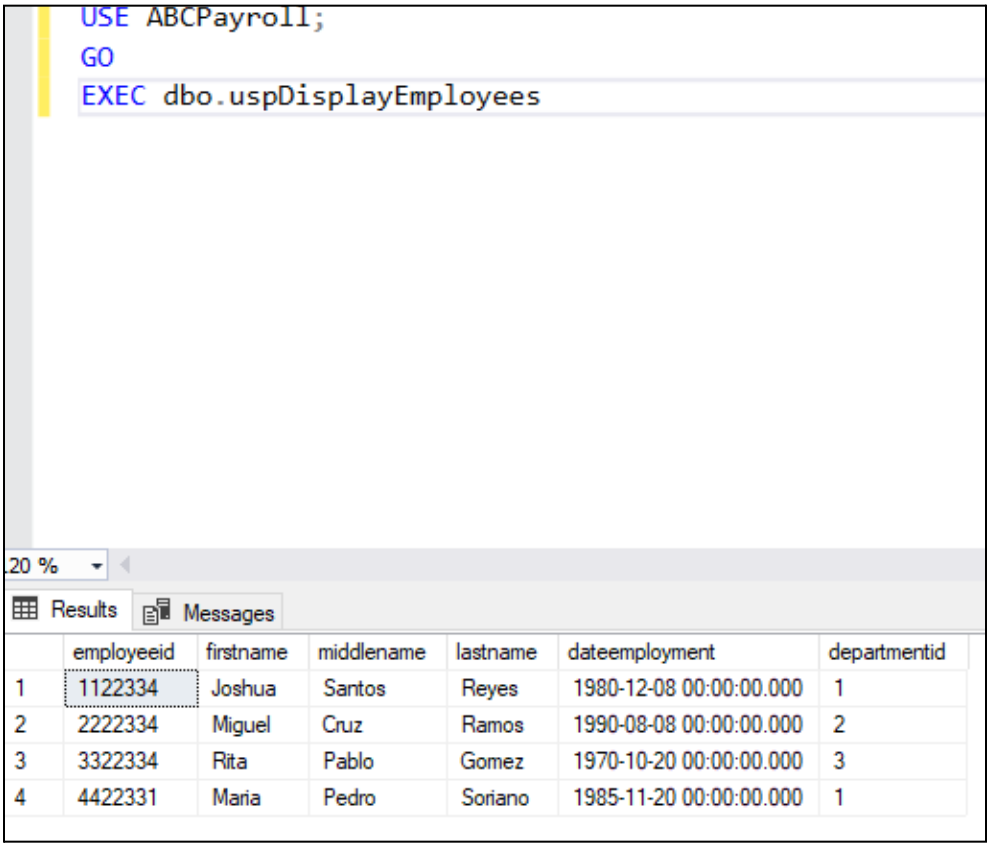


B. Using T - SQL



Observations: In this procedure, the group was able to make stored procedure by specifying values for template parameters and using the CREATE PROCEDURE command. Using the EXEC statement , we are able to execute the stored procedure

Execute a Stored Procedure



	employeeid	firstname	middlename	lastname	dateemployment	departmentid
1	1122334	Joshua	Santos	Reyes	1980-12-08 00:00:00.000	1
2	2222334	Miguel	Cruz	Ramos	1990-08-08 00:00:00.000	2
3	3322334	Rita	Pablo	Gomez	1970-10-20 00:00:00.000	3
4	4422331	Maria	Pedro	Soriano	1985-11-20 00:00:00.000	1

Observations: Using a GO EXEC statement, we are able to execute a stored procedure

Modify a Stored Procedure

```sql
USE [ABCPayroll]
GO
/****** Object:  StoredProcedure [dbo].[uspDisplayEmployees]    Script Date: 21/11/2022 2:59:01 pm ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[uspDisplayEmployees]
        @employeeid char(7)
AS

    SET NOCOUNT ON;
    SELECT * from employeeinfo where employeeid=@employeeid
```

120 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T15:00:25.1285123+08:00
```

```sql
EXEC dbo.uspDisplayEmployees @employeeid=1122334
```

120 %

Results | Messages

	employeeid	firstname	middlename	lastname	dateemployment	departmentid
1	1122334	Joshua	Santos	Reyes	1980-12-08 00:00:00.000	1

Observation: We can modify a stored procedure by simply right clicking on it and selecting modify. We can then edit the query to display specific rows from the table.

Specify a Parameter

SQLQuery14.sql - D...-51JTB9\JACK (55))*

```sql
USE [ABCPayroll]
GO

CREATE PROCEDURE uspDisplayEmployeeperDepartment
        @departmentid int
AS

    SET NOCOUNT ON;
    SELECT e.employeeid as 'employee id', CONCAT(e.firstname,' ',
    e.middlename,' ', e.lastname) as 'employee name', d.department
    from employeeinfo e
    inner join department d
    on e.departmentid = d.departmentid
    where e.departmentid = @departmentid
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T15:14:18.8069196+08:00
```

SQLQuery15.sql - D...-51JTB9\JACK (58))* | SQLQuery14.sql - D...-51JTB9\JACK (

```sql
USE ABCPayroll
EXEC dbo.uspDisplayEmployeeperDepartment @departmentid = 1
```

100 %

Results | Messages

	employee id	employee name	department
1	1122334	Joshua Santos Reyes	Marketing
2	4422331	Maria Pedro Soriano	Marketing

SQLQuery15.sql - D...-51JTB9\JACK (58))* | SQLQuery14.sql - D...-51JTB9\JACK

```sql
USE ABCPayroll
EXEC dbo.uspDisplayEmployeeperDepartment @departmentid = 2
```

100 %

Results | Messages

	employee id	employee name	department
1	2222334	Miguel Cruz Ramos	Administrator

SQLQuery15.sql - D...-51JTB9\JACK (58))* | SQLQuery14.sql - D...-51JTB9\JACK

```sql
USE ABCPayroll
EXEC dbo.uspDisplayEmployeeperDepartment @departmentid = 3
```

100 %

Results | Messages

	employee id	employee name	department
1	3322334	Rita Pablo Gomez	Sales

SQLQuery15.sql - D...-51JTB9\JACK (58))* | SQLQuery14.sql - D...-51JTB9\JACK

```sql
USE ABCPayroll
EXEC dbo.uspDisplayEmployeeperDepartment @departmentid = 4
```

100 %

Results | Messages

	employee id	employee name	department

Observation: By using parameters, we can specify the data that will be displayed

Return Data

SQLQuery16.sql - D...-51JTB9\JACK (53))*

```sql
USE [ABCPayroll]
GO

CREATE PROCEDURE dbo.uspCountTotalEmployeesperDepartment
        @departmentid int
AS

    SET NOCOUNT ON;
    DECLARE @total int
    SELECT @total = count(*) from employeeinfo
    WHERE departmentid = @departmentid

    IF @total = 0
        BEGIN
            PRINT 'There is no existing employee'
            RETURN
        END
    ELSE
        BEGIN
            SELECT * from employeeinfo where departmentid = @departmentid
            PRINT 'Total No. of Employee:' +
            convert (varchar(10),@total)
        END
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T15:27:17.0045239+08:00
```

SQLQuery17.sql - D...-51JTB9\JACK (54))* | SQLQuery16.sql - D...-51JTB9\JACK (53))*

```sql
USE ABCPayroll
EXEC dbo.uspCountTotalEmployeesperDepartment @departmentid = 1
```

100 %

Results | Messages

	employeeid	firstname	middlename	lastname	dateemployment	departmentid
1	1122334	Joshua	Santos	Reyes	1980-12-08 00:00:00.000	1
2	4422331	Maria	Pedro	Soriano	1985-11-20 00:00:00.000	1

SQLQuery17.sql - D...-51JTB9\JACK (54))* | SQLQuery16.sql - D...-51JTB9\JACK (53))*

```sql
USE ABCPayroll
EXEC dbo.uspCountTotalEmployeesperDepartment @departmentid = 4
```

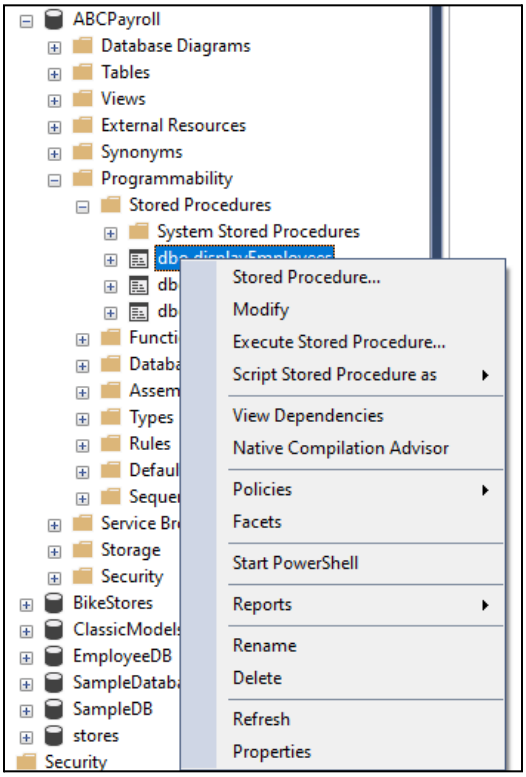
100 %

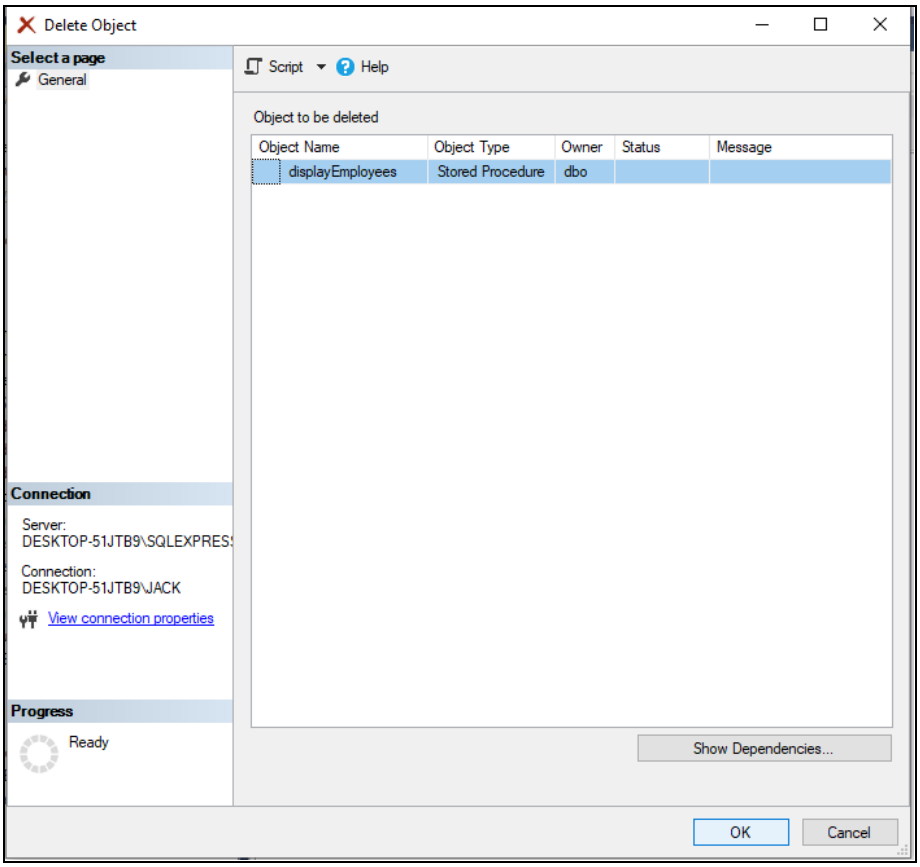
Messages

```
There is no existing employee

Completion time: 2022-11-21T15:29:32.8691499+08:00
```

Observation: Instead of displaying the table itself, we can output integers to count the data within the table that matches the given parameter in this procedure.

Delete a Stored Procedure





Observation: We can simply delete a stored procedure by right clicking on it and selecting the delete option

Supplementary Activity

Table name: TRUCK
Primary key: TRUCK_NUM
Foreign key: BASE_CODE, TYPE_CODE

TRUCK_NUM	BASE_CODE	TYPE_CODE	TRUCK_MILES	TRUCK_BUY_DATE	TRUCK_SERIAL_NUM
1001	501	1	32123.5	23-Sep-07	AA-322-12212-W11
1002	502	1	76984.3	05-Feb-06	AC-342-22134-Q23
1003	501	2	12346.6	11-Nov-06	AC-445-78656-Z99
1004		1	2894.3	06-Jan-07	WQ-112-23144-T34
1005	503	2	45673.1	01-Mar-06	FR-998-32245-W12
1006	501	2	193245.7	15-Jul-03	AD-456-00845-R45
1007	502	3	32012.3	17-Oct-04	AA-341-96573-Z84
1008	502	3	44213.6	07-Aug-05	DR-559-22189-D33
1009	503	2	10932.9	12-Feb-08	DE-887-98453-E94

Table name: BASE
Primary key: BASE_CODE
Foreign key: none

BASE_CODE	BASE_CITY	BASE_STATE	BASE_AREA_CODE	BASE_PHONE	BASE_MANAGER
501	Murfreesboro	TN	615	123-4567	Andrea D. Gallager
502	Lexington	KY	568	234-5678	George H. Delarosa
503	Cape Girardeau	MO	456	345-6789	Maria J. Talindo
504	Dalton	GA	901	456-7890	Peter F. McAvee

Table name: TYPE
Primary key: TYPE_CODE
Foreign key: none

TYPE_CODE	TYPE_DESCRIPTION
1	Single box, double-axle
2	Single box, single-axle
3	Tandem trailer, single-axle

1. Create a script to create the Trucking database and the following tables. Use the appropriate data types and assign the primary keys and foreign keys.

Step 1.sql - DESKT...Bryan Mendoza (52)) | Step 2.sql - DESKT...Bryan Mendoza (54))

```sql
-- Supplementary Activity | Script for Step 1 --
CREATE DATABASE TRUCKING
GO

USE TRUCKING
GO

CREATE TABLE BASE(
    BASE_CODE INT PRIMARY KEY NOT NULL,
    BASE_CITY VARCHAR(50) NOT NULL,
    BASE_STATE VARCHAR(5) NOT NULL,
    BASE_AREA_CODE INT NOT NULL,
    BASE_PHONE INT NOT NULL,
    BASE_MANAGER VARCHAR(50)
    )
GO

CREATE TABLE TYPE(
    TYPE_CODE INT PRIMARY KEY NOT NULL,
    TYPE_DESCRIPTION TEXT NOT NULL
    )
GO

CREATE TABLE TRUCK(
    TRUCK_NUM INT PRIMARY KEY NOT NULL,
    BASE_CODE INT,
    TYPE_CODE INT NOT NULL,
    TRUCK_MILES DECIMAL(10,1) NOT NULL,
    TRUCK_BUY_DATE DATE NOT NULL,
    TRUCK_SERIAL_NUM VARCHAR(50) NOT NULL
    FOREIGN KEY (BASE_CODE) REFERENCES BASE(BASE_CODE),
    FOREIGN KEY (TYPE_CODE) REFERENCES TYPE(TYPE_CODE)
    )
GO
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T16:13:00.0668377+08:00
```

Observation:

Using a new query, we created the new database “Trucking” and the table under it based on the information given from the table.

2. Create a script to insert the given values using the Trucking database.

Step 1.sql - DESKT...Bryan Mendoza (52)) | Step 2.sql - DESKT...Bryan Mendoza (54))*

```sql
-- Supplementary Activity | Script for Step 2 --
USE TRUCKING
GO

INSERT INTO BASE
    VALUES
    (501, 'Murfreesboro', 'TN', 615, 123-4567, 'Andrea D. Gallager'),
    (502, 'Lexington', 'KY', 568, 234-5678, 'George H. Delarosa'),
    (503, 'Cape Girardeau', 'MO', 456, 345-67897, 'Maria J. Talndo'),
    (504, 'Dalton', 'GA', 901, 456-7890, 'Peter F. McAvee')
GO

INSERT INTO TYPE
    VALUES
    (1, 'Single box, double-axle'),
    (2, 'Single box, single-axle'),
    (3, 'Tandem trailer, single-axle')
GO

INSERT INTO TRUCK
    VALUES
    (1001, 501, 1, 32123.5, '2007-09-23', 'AA-322-12212-W11'),
    (1002, 502, 1, 76984.3, '2006-02-05', 'AC-342-22134-Q23'),
    (1003, 501, 2, 12346.6, '2006-11-11', 'AC-445-78656-Z99'),
    (1004, NULL, 1, 2894.3, '2007-01-06', 'WQ-112-23144-T34'),
    (1005, 503, 2, 45673.1, '2006-03-01', 'FR-998-32245-W12'),
    (1006, 501, 2, 193245.7, '2003-07-15', 'AD-456-00845-R45'),
    (1007, 502, 3, 32012.3, '2004-10-17', 'AA-341-96573-Z84'),
    (1008, 503, 3, 44213.6, '2005-08-07', 'DR-559-22189-D33'),
    (1009, 503, 2, 10932.9, '2008-02-12', 'DE-887-98456-E94')
GO
```

110 %

Messages

```
(4 rows affected)

(3 rows affected)

(9 rows affected)

Completion time: 2022-11-21T16:13:29.3393934+08:00
```

Observation:

Given the information from the table, we inserted values for each table following the data type implemented on step 1.

3. Create and execute a stored procedure to display the total number of trucks per base. Display the truck number, base code, and base manager. Arrange the list from highest to lowest.

Stored Procedure Creation

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 4.sql - DESKT...Bryan Mendoza (59))* | Step 3.sql - DESKT...Bryan Mer

```sql
-- Supplementary Activity | Script for Step 3 --
USE TRUCKING
GO

CREATE PROCEDURE dbo.DisplayNumberOfTrucks

AS
    SET NOCOUNT ON;
    DECLARE @Base1 INT
    SELECT @Base1 = count(*)
    FROM TRUCK
    WHERE BASE_CODE = 501;

    DECLARE @Base2 INT
    SELECT @Base2 = count(*)
    FROM TRUCK
    WHERE BASE_CODE = 502;

    DECLARE @Base3 INT
    SELECT @Base3 = count(*)
    FROM TRUCK
    WHERE BASE_CODE = 503;

    BEGIN
        SELECT a.TRUCK_NUM, a.BASE_CODE, b.BASE_MANAGER
        FROM TRUCK a
        INNER JOIN BASE b ON a.BASE_CODE = b.BASE_CODE
        ORDER BY a.BASE_CODE DESC

        PRINT 'Total Trucks for base code 501 is ' + CONVERT(VARCHAR(10), @Base1)
        PRINT 'Total Trucks for base code 502 is ' + CONVERT(VARCHAR(10), @Base2)
        PRINT 'Total Trucks for base code 503 is ' + CONVERT(VARCHAR(10), @Base3)
    END
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T16:53:41.3298467+08:00
```

Calling the Stored Procedure

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 4.sql - DESKT...Bryan

```sql
USE TRUCKING
GO

EXEC dbo.DisplayNumberOfTrucks
```

110 %

Results | Messages

	TRUCK_NUM	BASE_CODE	BASE_MANAGER
1	1005	503	Maria J. Talndo
2	1008	503	Maria J. Talndo
3	1009	503	Maria J. Talndo
4	1007	502	George H. Delarosa
5	1002	502	George H. Delarosa
6	1003	501	Andrea D. Gallager
7	1006	501	Andrea D. Gallager
8	1001	501	Andrea D. Gallager

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 4.sql - DESKT...Bryan Mend

```sql
USE TRUCKING
GO

EXEC dbo.DisplayNumberOfTrucks
```

110 %

Results | Messages

```
Total Trucks for base code 501 is 3
Total Trucks for base code 502 is 2
Total Trucks for base code 503 is 3

Completion time: 2022-11-21T16:54:02.0941376+08:00
```

Observation:

We declared 3 variables in order to store the BASE_CODE present on the given table. Using these three variables, we will be able to print the searched occurrences of trucks having the same base code (501, 502, 0r 503). The result of these three variables are observed on the messages tab. While the command to display the trucks with their base code and manage are seen on the results tab.

4. Create and execute a stored procedure to display the total number of trucks per type code. Display the truck number, type code and type description. Use TYPE_CODE as input parameter.

Creating the Stored Procedure

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 4.sql - DESKT...Bryan Mendoza (59))*

```sql
-- Supplementary Activity | Script for Step 4 --
USE TRUCKING
GO

CREATE PROCEDURE dbo.DisplayNumberOfTrucksPerCode
        @TYPE_CODE INT
AS
    SET NOCOUNT ON;
    DECLARE @TotalperCode INT
    SELECT @TotalperCode = count(*)
    FROM TRUCK
    WHERE TYPE_CODE = @TYPE_CODE;

    BEGIN
        SELECT a.TRUCK_NUM, b.TYPE_CODE, b.TYPE_DESCRIPTION
        FROM TRUCK a
        INNER JOIN TYPE b ON a.TYPE_CODE = b.TYPE_CODE
        WHERE a.TYPE_CODE = @TYPE_CODE

        PRINT 'Total Trucks with type code ' +
        CONVERT(VARCHAR(10), @TYPE_CODE) + ' is ' +
        CONVERT(VARCHAR(10), @TotalperCode)
    END
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T16:56:36.3176183+08:00
```

Calling the Stored Procedure

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 4.sql - DESKT...Bryan Mendoza (59))*

```sql
USE TRUCKING
GO

EXEC dbo.DisplayNumberOfTrucksPerCode @TYPE_CODE = 1
```

110 %

Results | Messages

	TRUCK_NUM	TYPE_CODE	TYPE_DESCRIPTION
1	1001	1	Single box, double-axle
2	1002	1	Single box, double-axle
3	1004	1	Single box, double-axle

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 4.sql - DESKT...Bryan Mendoza (59

```sql
USE TRUCKING
GO

EXEC dbo.DisplayNumberOfTrucksPerCode @TYPE_CODE = 1
```

110 %

Results | Messages

```
Total Trucks with type code 1 is 3

Completion time: 2022-11-21T16:57:07.8969306+08:00
```

Observation:

By declaring a variable to store the user input parameter which is the TYPE_CODE, we are able to filter the results. The variable will be the one used to store the user input in order to select specific results from the trucks.

5. Create and execute a stored procedure to display the truck number, base city, base manager and type description.

Creating the Stored Procedure

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 5.sql - DESKT...Bryan Mendoza (57)) | Step 4.sql - DESKT

```sql
-- Supplementary Activity | Script for Step 5 --
USE TRUCKING
GO

CREATE PROCEDURE dbo.DisplayTruckBaseCityManagerAndType
AS
    SET NOCOUNT ON;
    SELECT a.TRUCK_NUM, b.BASE_CITY, b.BASE_MANAGER, c.TYPE_DESCRIPTION
    FROM TRUCK a
    INNER JOIN BASE b ON a.BASE_CODE = b.BASE_CODE
    INNER JOIN TYPE c ON a.TYPE_CODE = c.TYPE_CODE
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T16:58:28.5634095+08:00
```

Calling the Stored Procedure

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 5.sql - DESKT...Bryan Mendoza (57))

```sql
USE TRUCKING
GO

EXEC dbo.DisplayTruckBaseCityManagerAndType
```

110 %

Results | Messages

	TRUCK_NUM	BASE_CITY	BASE_MANAGER	TYPE_DESCRIPTION
1	1001	Murfreesboro	Andrea D. Gallager	Single box, double-axle
2	1002	Lexington	George H. Delarosa	Single box, double-axle
3	1003	Murfreesboro	Andrea D. Gallager	Single box, single-axle
4	1005	Cape Girardeau	Maria J. Talndo	Single box, single-axle
5	1006	Murfreesboro	Andrea D. Gallager	Single box, single-axle
6	1007	Lexington	George H. Delarosa	Tandem trailer, single-axle
7	1008	Cape Girardeau	Maria J. Talndo	Tandem trailer, single-axle
8	1009	Cape Girardeau	Maria J. Talndo	Single box, single-axle

Observation:

On this procedure, we created a stored procedure that will simply display the TRUCK_NUM, BASE_CITY, BASE_MANAGE, and TYPE_DESCRIPTION. In order to display the results, we joined three tables using the keys which are referenced by the TRUCK table to the other two tables.

6. Create and execute a stored procedure to return the total number of trucks purchased per month. Arrange the list from highest to lowest number of trucks. Print the total number of trucks and the corresponding month. Use month as input parameter and total as output.

Creating the stored procedure

Step 8.sql - DESKT...Bryan Mendoza (56)) | Step 6.sql - DESKT...Bryan Mendoza (54))*

```sql
-- Supplementary Activity | Script for Step 6 --
USE TRUCKING
GO

CREATE PROCEDURE dbo.DisplayTruckPerMonth
        @TRUCK_MONTH INT
AS
    SET NOCOUNT ON;
    DECLARE @TRUCK_MONTH_TOTAL INT

    SELECT @TRUCK_MONTH_TOTAL = COUNT(*) FROM TRUCK
    WHERE MONTH(TRUCK_BUY_DATE) = @TRUCK_MONTH

    SELECT TRUCK_NUM, MONTH(TRUCK_BUY_DATE) AS 'Month'
    FROM TRUCK
    WHERE MONTH(TRUCK_BUY_DATE) = @TRUCK_MONTH

    IF @TRUCK_MONTH_TOTAL = 0
        BEGIN
            PRINT 'No Trucks Purchased on Month ' +
            CONVERT(VARCHAR(10), @TRUCK_MONTH)
            RETURN
        END
    ELSE
        BEGIN
            PRINT 'Trucks Purchase on Month ' +
            CONVERT(VARCHAR(10), @TRUCK_MONTH) + ' is ' +
            CONVERT(VARCHAR(10), @TRUCK_MONTH_TOTAL)
        END
```

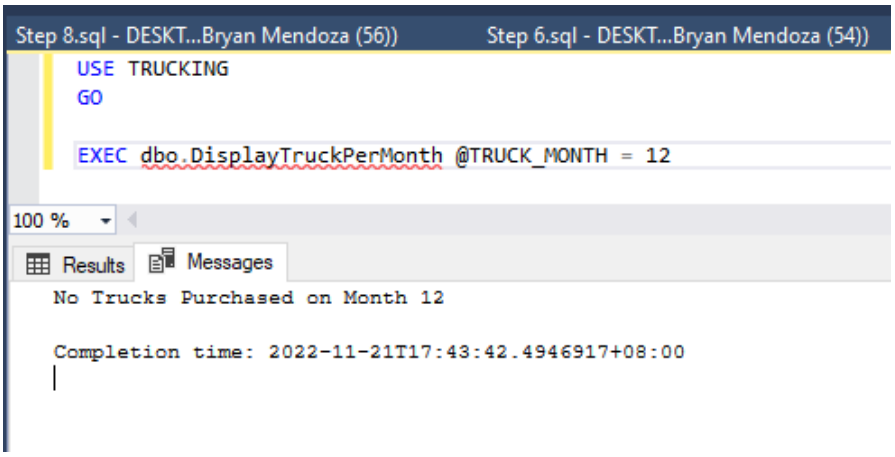
100 %

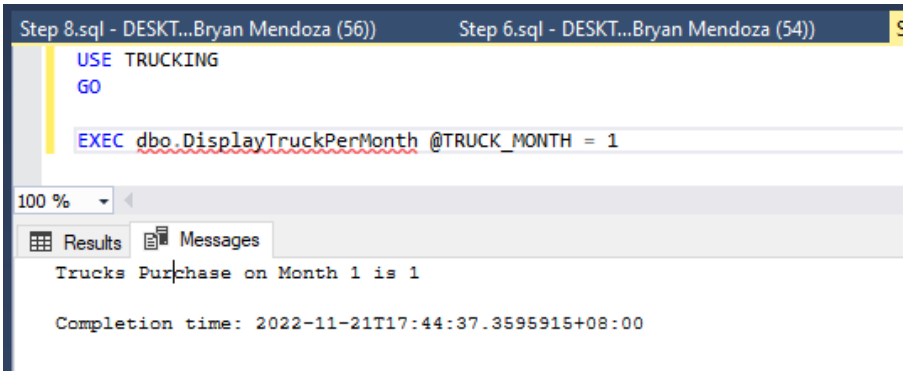
Messages

```
Commands completed successfully.

Completion time: 2022-11-21T17:43:38.7287139+08:00
```

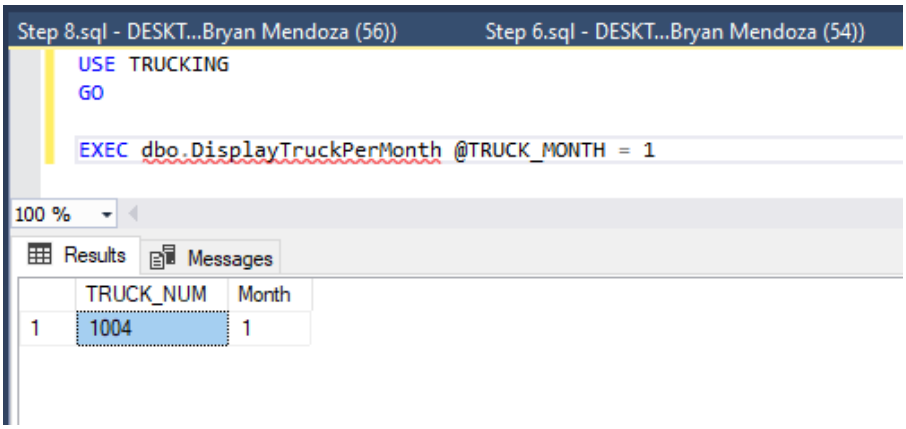
Calling the stored procedure

When There are no truck Purchase on Input Month





When There are purchases on Input Month



Observation:

In this procedure, we used a variable TRUCK_MONTH_TOTAL in order to keep track of the number of trucks purchased on the input month. The if-else statement determines which string to output which corresponds to the results.

7. Create and execute a stored procedure to display the truck with truck miles ranging from 5000 to 40000. Arrange the list from highest to lowest.

Creating the Stored Procedure

SQLQuery31.sql - D...ryan Mendoza (57))* | SQLQuery30.sql - D...ryan Mendo

```sql
-- Supplementary Activity | Script for Step 7 --
USE TRUCKING
GO

CREATE PROCEDURE dbo.DisplayTruckMiles
AS
    SET NOCOUNT ON;
    SELECT TRUCK_NUM, TRUCK_MILES
    FROM TRUCK
    WHERE TRUCK_MILES BETWEEN 5000 AND 40000
    ORDER BY TRUCK_MILES ASC
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T17:01:36.1779737+08:00
```

Calling the Stored Procedure

SQLQuery31.sql - D...ryan Mendoza (57))* | SQLQuery30.sql -

```sql
USE TRUCKING
GO

EXEC dbo.DisplayTruckMiles
```

110 %

Results | Messages

	TRUCK_NUM	TRUCK_MILES
1	1009	10932.9
2	1003	12346.6
3	1007	32012.3
4	1001	32123.5

Observation: We used the BETWEEN command in order to filter the output of the stored procedure. As we can see on the results, we can observe that the TRUCK_MILES of the trucks displayed ranged between 5000 to 40 000.

8. Create and execute a stored procedure to display the truck number, truck_buy_date and type description that was purchased for a specific year. Use year as input parameter.

Creating the Stored Procedure

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 8.sql - DESKT...Bryan Mendoza (54))

```sql
-- Supplementary Activity | Script for Step 8 --
USE TRUCKING
GO

CREATE PROCEDURE dbo.DisplayTrucksPurchasedOnGivenYear
        @TRUCK_BUY_DATE INT
AS
    SET NOCOUNT ON;
    SELECT a.TRUCK_NUM, a.TRUCK_BUY_DATE, b.TYPE_DESCRIPTION
    FROM TRUCK a
    INNER JOIN TYPE b ON a.TYPE_CODE = b.TYPE_CODE
    WHERE YEAR(a.TRUCK_BUY_DATE) = @TRUCK_BUY_DATE
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-11-21T17:08:50.8080649+08:00
```

Calling the Stored Procedure

SQLQuery30.sql - D...ryan Mendoza (55))* | Step 8.sql - DESKT...Bryan Mendoza (54)) | Step 4.s

```sql
GO

EXEC dbo.DisplayTrucksPurchasedOnGivenYear @TRUCK_BUY_DATE = 2007
```

110 %

Results | Messages

	TRUCK_NUM	TRUCK_BUY_DATE	TYPE_DESCRIPTION
1	1001	2007-09-23	Single box, double-axle
2	1004	2007-01-06	Single box, double-axle

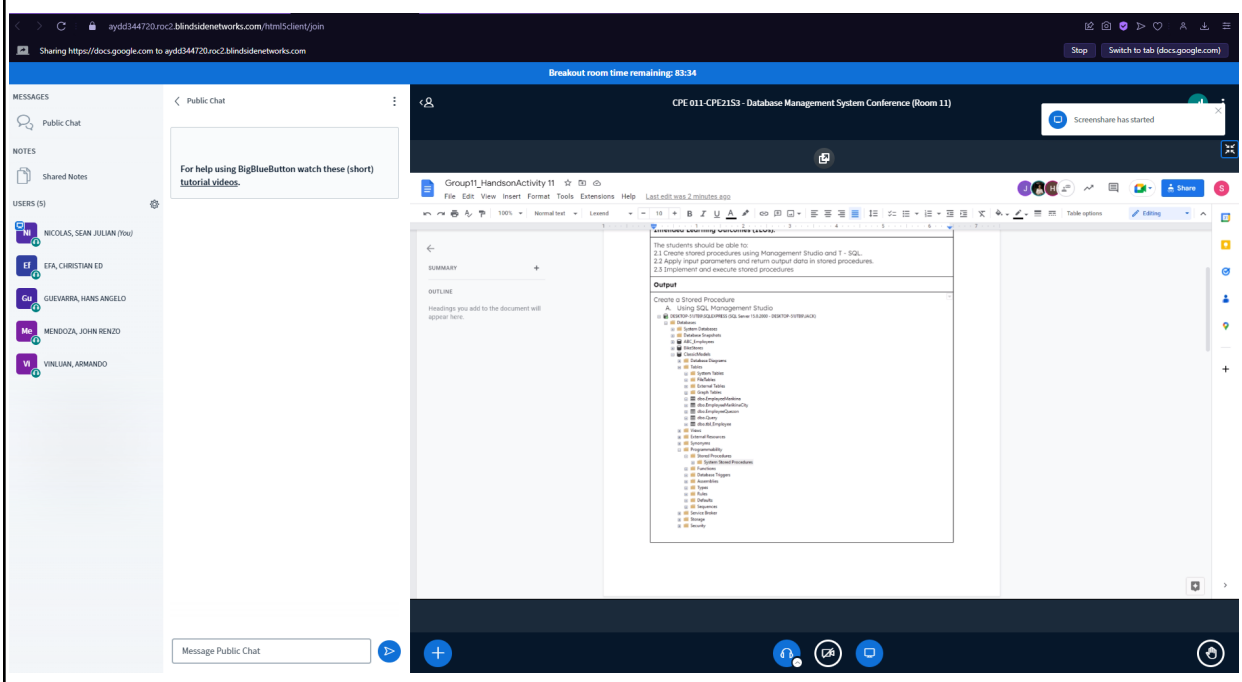
Observation: Using the Year Function, we are able to identify the user input to search for the trucks which were purchased on the inputted year. We also joined the TYPE and TRUCK table in order to arrive with the output.

Conclusion

An important layer of protection is added by a stored procedure between the user interface and the database. Because end users can add or modify data but not develop procedures, it enables security through data access controls. A stored procedure groups SQL statements together so that they can all be called at once and performed. As a result, fewer slow networks are used, less network traffic is generated, and round-trip response times are enhanced. The elimination of network bottlenecks via result set processing makes OLTP applications in particular advantageous.

In addition, the stored procedures are also considered as an easier way to implement commands to our database since the commands are saved and therefore can be reused over and over again in a way that they obtain the desired results. Moreover, stored procedures also gives the user a simpler way in order to run commands on the database server as it only requires few lines of code and understanding on how the stored procedure works.

Proof of Collaboration



Honor Pledge

"We accept responsibility for our role in ensuring the integrity of the work submitted by the group in which we participated."