

Prabhat Ale

Roll No:22

1. Write a stored procedure named “GetEmployee()” to get the name, birthdate, and address of employees.

```
CREATE PROCEDURE GetEmployee() -- Procedure name matches the call
BEGIN
    -- Select name, birthdate, address of employees.
    SELECT Ename, Bdate, Address
    FROM EMPLOYEE;
END;
```

Name	Value
Updated Rows	0
	CREATE PROCEDURE GetEmployee() -- Procedure name matches the call
	BEGIN
	-- Select name, birthdate, address of employees.
	SELECT Ename, Bdate, Address
	FROM EMPLOYEE;
	END
Start time	Mon May 13 20:59:28 NPT 2024
Finish time	Mon May 13 20:59:28 NPT 2024

2. Execute the procedure in Q.1 and show the result.

```
CALL GetEmployee;
```

	Ename	Bdate	Address	
1	Gaurav	2056-07-24	MacheGaun	
2	Anish	2053-12-05	Budhanilkandha	
3	Prabhat	2054-12-15	Nawalparasi	
4	Suman	2052-07-06	Sunwal	
5	Ashmita	2053-12-05	Brussels	
6	Kabita	2054-12-15	Delhi	
7	Kusum	2052-07-06	Sunwal	
8	Pratigya	2057-11-15	Banglore	
9	Abin	2051-07-09	Sunwal	

3. Write a stored procedure to get the PF category name, Amount, and start date where the amount is greater than the provided input value. Your procedure should contain an IN parameter named amt to take the input value of the amount. Call the procedure with inputs 1000 and 3000 respectively.

```
•CREATE PROCEDURE GetPFDetails(IN amt DECIMAL(10, 2))
BEGIN
    SELECT PFCategoryName, Amount, Start_date
    FROM PF
    WHERE Amount > amt;
END
```

Statistics 1 ×

Name	Value
Updated Rows	0
	CREATE PROCEDURE GetPFDetails(IN amt DECIMAL(10, 2))
	BEGIN
	SELECT PFCategoryName, Amount, Start_date
	FROM PF
	WHERE Amount > amt;
	END
Start time	Mon May 13 21:31:42 NPT 2024
Finish time	Mon May 13 21:31:42 NPT 2024

```
CALL GetPFDetails(1000);
```

PF 1 × Statistics 1

CALL GetPFDetails(1000) Enter a SQL expression to filter results (use Ctrl+Space)

	PFCategoryName	Amount	Start_date
1	Employee_Provident_Fund	4,500	2079-12-05
2	Pension_Fund	5,500	2078-12-15
3	Retirement_Fund	4,700	2076-07-06
4	Children_Savings_Fund	5,800	2080-11-15
5	General_Provident_Fund	2,500	2074-07-09
6	Employee_Provident_Fund	3,000	2078-12-05
7	Pension_Fund	6,000	2077-12-15
8	Retirement_Fund	4,000	2075-07-06
9	Children_Savings_Fund	5,000	2079-11-15
10	General_Provident_Fund	3,500	2073-07-09

CALL GetPFDDetails(3000);

	PFCategoryName	Amount	Start_date
1	Employee_Provident_Fund	4,500	2079-12-05
2	Pension_Fund	5,500	2078-12-15
3	Retirement_Fund	4,700	2076-07-06
4	Children_Savings_Fund	5,800	2080-11-15
5	Pension_Fund	6,000	2077-12-15
6	Retirement_Fund	4,000	2075-07-06
7	Children_Savings_Fund	5,000	2079-11-15
8	General_Provident_Fund	3,500	2073-07-09

4. Write a stored procedure to get a number of PF records where the amount of PF is equal to the provided input value. Your procedure should contain an IN parameter named amt to take the input value of the amount and should contain an OUT parameter named total to return the total number of PF records satisfying the condition.

```

CREATE PROCEDURE GetPFTotal(IN amt DECIMAL(10,2), OUT total INT)
BEGIN
    SELECT COUNT(*) INTO TOTAL FROM PF
    WHERE Amount = amt;
END

```

Name	Value
Updated Rows	0
	CREATE PROCEDURE GetPFTotal(IN amt DECIMAL(10,2), OUT total INT)
	BEGIN
	SELECT COUNT(*) INTO TOTAL FROM PF
	WHERE Amount = amt;
	END
Start time	Mon May 13 21:41:20 NPT 2024
Finish time	Mon May 13 21:41:20 NPT 2024

5. Call the procedure in Q4. with input of 3000 and print the @total.

```
CALL GetPFTotal(3000, @total)
```

Name	Value
Updated Rows	1
Query	CALL GetPFTotal(3000, @total)
Start time	Mon May 13 21:44:22 NPT 2024
Finish time	Mon May 13 21:44:22 NPT 2024

```
SELECT @total AS total_number_of_pf_records;
```

Results 1 X

SELECT @total AS total_number_of_pf_records

	total_number_of_pf_records
1	1

```
SELECT * FROM PF WHERE Amount = 3000;
```

PF 1 X

SELECT * FROM PF WHERE Amount = 3000

	PFID	SSN	PFCategoryName	Amount	Start_date	Remarks
1	1,111	2	Employee_Provident_Fund	3,000	2078-12-05	[NULL]

6. Write a before insert the trigger before inserting a record into the Employee table. Show some action on the event.

```
CREATE TRIGGER before_insert_trigger BEFORE
INSERT ON EMPLOYEE
FOR EACH ROW
BEGIN
    SET NEW.Salary = NEW.Salary * 1.3;
END
```

Statistics 1 X

Name	Value
Updated Rows	0
Query	CREATE TRIGGER before_insert_trigger BEFORE INSERT ON EMPLOYEE FOR EACH ROW BEGIN SET NEW.Salary = NEW.Salary * 1.3; END
Start time	Mon May 13 21:52:04 NPT 2024
Finish time	Mon May 13 21:52:04 NPT 2024

To execute a trigger, we must first insert data into the EMPLOYEE table. However, the EMPLOYEE table depends on the OFFICE table, as the primary key of the OFFICE table is referenced as a foreign key in the EMPLOYEE table. Therefore, we need to perform an insert operation in the OFFICE table beforehand.

```

•INSERT INTO Office(Onumber, Oname, Country)
VALUES
(36, 'Sunil_Office_36', 'Ireland');

```

Name	Value
Updated Rows	1
Query	INSERT INTO Office(Onumber, Oname, Country) VALUES (36, 'Sunil_Office_36', 'Ireland')
Start time	Mon May 13 22:04:02 NPT 2024
Finish time	Mon May 13 22:04:02 NPT 2024

Then we can insert into the EMPLOYEE table with the appropriate details.

```

•INSERT INTO EMPLOYEE (SSN, Ename, Gender, Bdate, Address, Salary, Ono, Years_of_experience, Marital_Status)
VALUES(11, 'Sunil', 'M', '2055-07-10', 'Gangabu', 100000, 36, 4, 'Single');

```

Name	Value
Updated Rows	1
Query	INSERT INTO EMPLOYEE (SSN, Ename, Gender, Bdate, Address, Salary, Ono, Years_of_experience, Marital_Status) VALUES(11, 'Sunil', 'M', '2055-07-10', 'Gangabu', 100000, 36, 4, 'Single')
Start time	Mon May 13 22:05:17 NPT 2024
Finish time	Mon May 13 22:05:17 NPT 2024

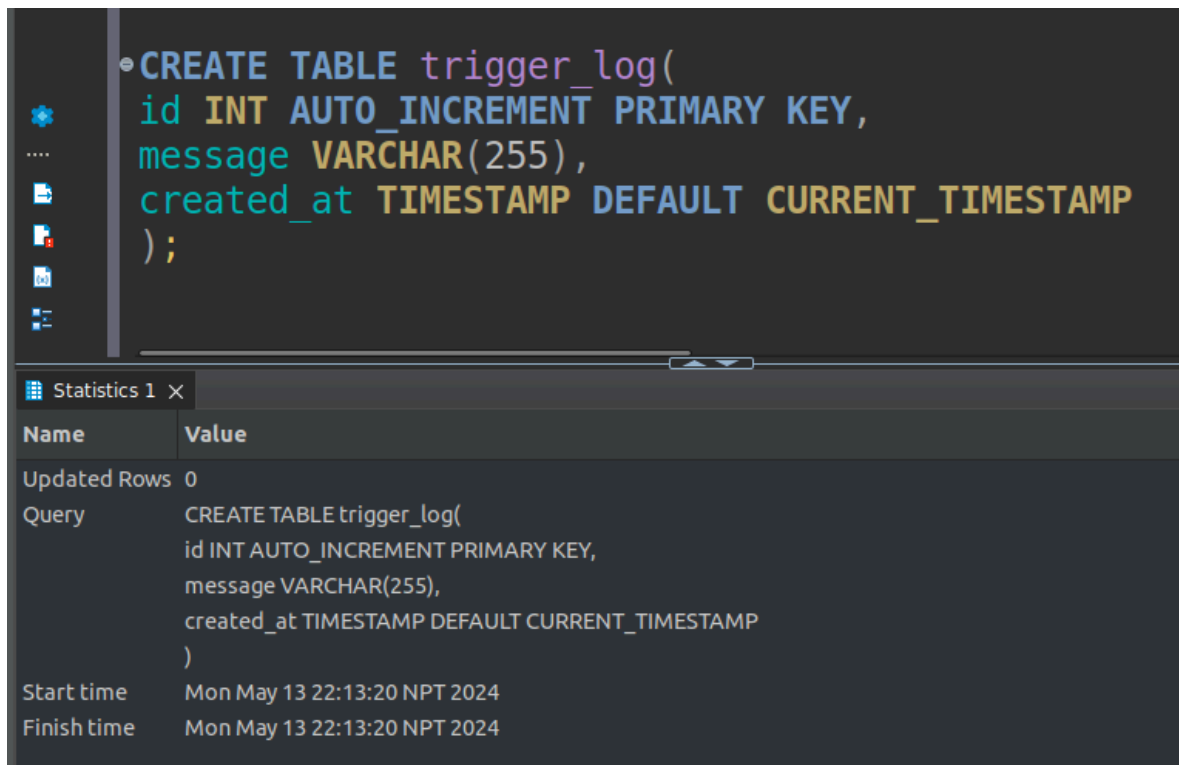
NOTE: We have inserted the Sunil information where his salary was inserted as 100000 but the trigger automatically gets executed before the insert operation and hiked the salary by 30% to 130000.

```
SELECT * FROM EMPLOYEE e ;
```

	SSN	Ename	Gender	Bdate	Address	Salary	Ono	Years_of_experience	Marital_Status
1	2	Gaurav	M	2056-07-24	MacheGaun	30,000	10		1 Married
2	3	Anish	M	2053-12-05	Budhanilkandha	95,000	2		3 Married
3	4	Prabhat	M	2054-12-15	Nawalparasi	80,000	11		2 Divorced
4	5	Suman	M	2052-07-06	Sunwal	75,000	33		3 Divorced
5	6	Ashmita	F	2053-12-05	Brussels	95,000	4		5 Divorced
6	7	Kabita	F	2054-12-15	Delhi	80,000	12		4 Single
7	8	Kusum	F	2052-07-06	Sunwal	75,000	15		2 Single
8	9	Pratigya	F	2057-11-15	Banglore	80,000	27		3 Single
9	10	Abin	M	2051-07-09	Sunwal	75,000	1		2 Single
10	11	Sunil	M	2055-07-10	Gangabu	130,000	36		4 Single

7. Write after the delete trigger on the PF table during the delete operation. Print “It is deleted”.

Create a new table named trigger_log to store the log message so that we can print the trigger message

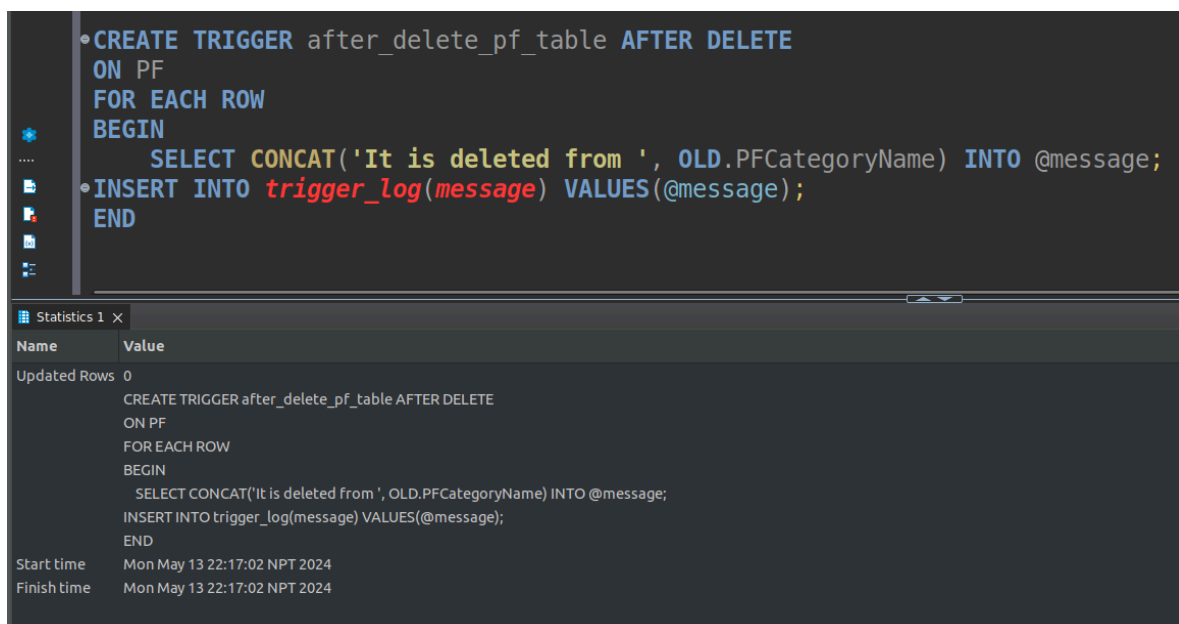


The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL script for creating the trigger_log table. The bottom pane shows the execution statistics for the query.

```
CREATE TABLE trigger_log(  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    message VARCHAR(255),  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Name	Value
Updated Rows	0
Query	CREATE TABLE trigger_log(id INT AUTO_INCREMENT PRIMARY KEY, message VARCHAR(255), created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP)
Start time	Mon May 13 22:13:20 NPT 2024
Finish time	Mon May 13 22:13:20 NPT 2024

Create an after-delete trigger on the PF table

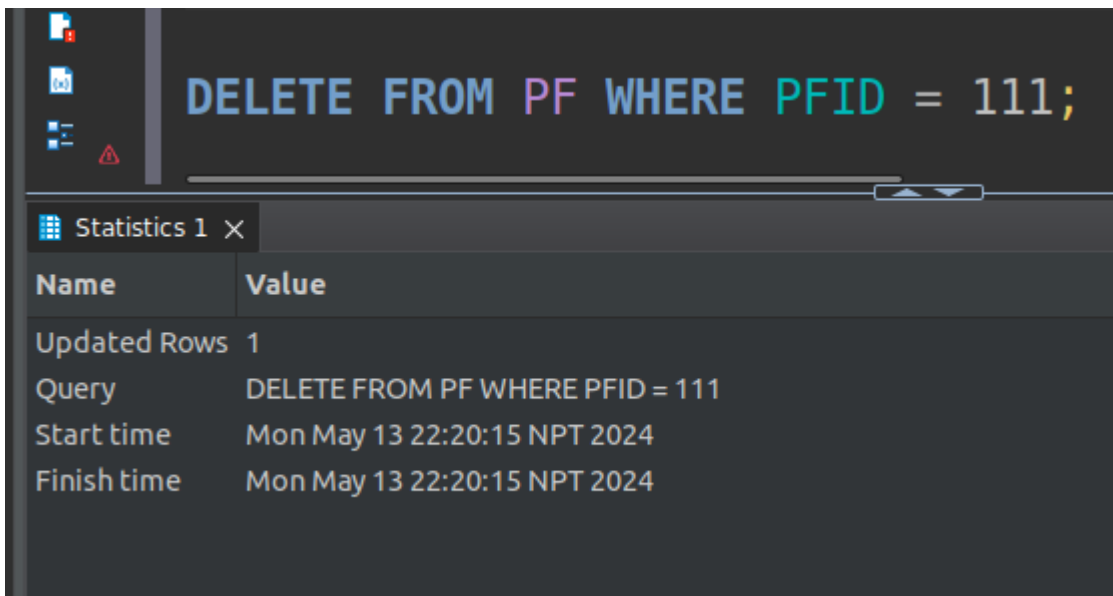


The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL script for creating an after-delete trigger on the PF table. The bottom pane shows the execution statistics for the query.

```
CREATE TRIGGER after_delete_pf_table AFTER DELETE  
ON PF  
FOR EACH ROW  
BEGIN  
    SELECT CONCAT('It is deleted from ', OLD.PFCategoryName) INTO @message;  
    INSERT INTO trigger_log(message) VALUES(@message);  
END
```

Name	Value
Updated Rows	0
Query	CREATE TRIGGER after_delete_pf_table AFTER DELETE ON PF FOR EACH ROW BEGIN SELECT CONCAT('It is deleted from ', OLD.PFCategoryName) INTO @message; INSERT INTO trigger_log(message) VALUES(@message); END
Start time	Mon May 13 22:17:02 NPT 2024
Finish time	Mon May 13 22:17:02 NPT 2024

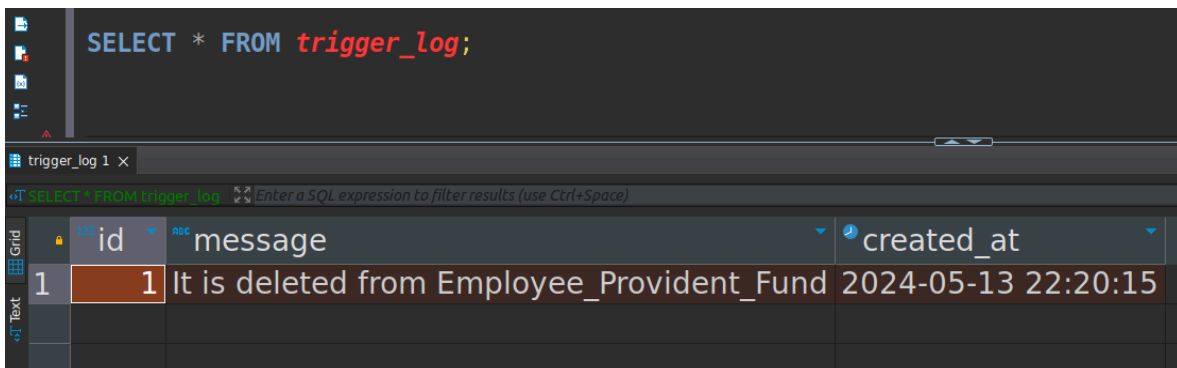
Delete records From the PF table to see whether the trigger message gets populated in the trigger_log table



The screenshot shows a database IDE with a SQL query editor at the top containing the command: `DELETE FROM PF WHERE PFID = 111;`. Below the editor, a 'Statistics 1' window is open, displaying the following data:

Name	Value
Updated Rows	1
Query	DELETE FROM PF WHERE PFID = 111
Start time	Mon May 13 22:20:15 NPT 2024
Finish time	Mon May 13 22:20:15 NPT 2024

Check logs in the trigger_log table



The screenshot shows a database IDE with a SQL query editor at the top containing the command: `SELECT * FROM trigger_log;`. Below the editor, a 'trigger_log 1' window is open, displaying the results of the query in a table grid. The table has three columns: 'id', 'message', and 'created_at'.

	id	message	created_at
1	1	It is deleted from Employee_Provident_Fund	2024-05-13 22:20:15

Note: Displaying a message After the Delete Trigger message is not possible in MYSQL, so creating a separate trigger_log table and creating after the delete trigger to store the message in this table.