Testing trigger		*Trigger 1-Type: Before Insert*
Steps	**Description & Query**	**Expected Result**

Step1	Insert a new row into WorkCenters table:
	
	INSERT INTO WorkCenters(name, capacity)
	VALUES('Mold Machine', 100);
		NA

Step2	Query data from the WorkCenterStats table: SELECT * FROM WorkCenterstats;	totalCapacity 100

Step3	Insert a new work center:
	
	INSERT INTO WorkCenters(name, capacity)
	VALUES('Packing', 200);
		NA

Step4	Finally query data from the WorkCenterStats table SELECT * FROM WorkCenterStats;	totalCapacity 300

Testing trigger		*Trigger 2-Type: After Insert*
Steps	**Description & Query**	**Expected Result**

Step1	Insert 2 rows into members table:
	
	INSERT INTO members(name, email, birthDate)
	VALUES('Bezzi', 'bezzi@example.com', NULL);
	INSERT INTO members(name, email, birthDate)
	VALUES('Begli', 'begli@example.com', '1999-03-06');
		NA

Step2	Query data from the members table: SELECT * FROM members;	Table should contain 2 records, one of the record birthdate value should be NULL.
Step3	Finally query data from the reminders table: SELECT * FROM reminders;	Message Hi Bezzi, please update your date of birth.

Testing trigger		*Trigger 3-Type: Before Update*
Steps	**Description & Query**	**Expected Result**

Step1	Update the quantity of the row with id 1 to 150
	UPDATE sales SET quantity = 150 WHERE id = 1;
	
	Query data from the sales table to verify update
	SELECT * FROM sales;
		It should update sales table because the new quantity does not violate the rule.

Step2	Update the quantity of the row with id 1 to 500 UPDATE sales SET quantity = 500 WHERE id = 1;	Error Code : 1644. The new quantity 500 cannot be 3 times greater than the current quantity 150. In this case, the trigger should found the new quantity caused a violation and raised an error.

Testing trigger		*Trigger 4-Type: After Update*
Steps	**Description & Query**	**Expected Result**

Step1	Update the quantity of the row with id 1 to 350
	UPDATE sales SET quantity = 350 WHERE id = 1;
	
	Query data from the SalesChanges table to verify update
	SELECT * FROM SalesChanges;
		The trigger should triggered automatically.

Step2	Update the quantity of all 3 rows by increasing 10% UPDATE Sales SET quantity = CAST(quantity * 1.1 AS UNSIGNED);	NA
Step3	Query data from the SalesChanges table SELECT * FROM SalesChanges;	The trigger should fire three times because of the updates of the three rows

Testing trigger		***Trigger 5-Type: Before Delete***
Steps	**Description & Query**	**Expected Result**

Step1	Delete the row from salaries table
	DELETE FROM salaries WHERE employeeNumber = 1002;
	
	Query the data from SalaryArchives table
	SELECT * FROM SalaryArchives;
		The trigger should invoke and insert a new row into the SalaryArchives table

Step2	Delete all the rows from salaries table
	DELETE FROM salaries;
	
	Finally, query the data from SalaryArchives table
	SELECT * FROM SalaryArchives;
		The trigger should trigger 2 times because the DELETE statement deleted two rows from the Salaries table

Testing trigger		***Trigger 6-Type: After Delete***
Steps	**Description & Query**	**Expected Result**

Step1	Delete the row from salaries table
	DELETE FROM salaries WHERE employeeNumber = 1002;
	
	Query salary from SalaryBudgets table
	SELECT * FROM SalaryBudgets;
		In the output, the total should be reduced by the deleted salary

Step2	Delete all the rows from salaries table
	DELETE FROM salaries;
	
	Finally, query the total from SalaryBudgets table
	SELECT * FROM SalaryBudgets;
		The trigger updated the total to zero