Trigger Requirements & Test Cases

Trigger 1: Before Insert

|  |  |
| --- | --- |
| Trigger name | BeforeWorkCentersInsert |
| Trigger type | Before Insert |
| Description | Updates the total capacity in the WorkCenterStats table before a new work center is inserted into the WorkCenters table |
| Tables | WorkCenters  WorkCentersStats |

Test Case 1: Before Insert

|  |  |  |
| --- | --- | --- |
| Step 1 | Insert a new row in WorkCenters table  INSERT INTO WorkCenters (name, capacity) VALUES (‘Mold Machine’, 100); |  |
| Step 2 | Query data from the WorkCenterStats table  SELECT \*  FROM WorkCenterStats; | totalCapacity  100 |
| Step 3  Step 4 | Insert a new work center  INSERT INTO WorkCenters (name, capacity) VALUES (‘Packing’, 200);  Finally query data from the WorkCenterStats table  SELECT \*  FROM WorkCenterStats; | totalCapacity  300 |

Trigger 2: After Insert

|  |  |
| --- | --- |
| Trigger name | AfterMembersInsert |
| Trigger type | After Insert |
| Description | Insert a reminder into the Reminders table if the birth date of the member is NULL |
| Tables | Members  Reminders |

Test Case 2: After Insert

|  |  |  |
| --- | --- | --- |
| Step 1 | Insert 2 rows in Members table  INSERT INTO Members (name, email, birthDate) VALUES ('John', 'john@example.com', NULL);  INSERT INTO Members (name, email, birthDate) VALUES ('Kim', 'kim@example.com', '2012-05-01'); |  |
| Step 2 | Query data from the Members table  SELECT \*  FROM Members; | Table should contain 2 records, one of the birthDate record value should be NULL. |
| Step 3 | Finally query data from the Reminders table  SELECT \*  FROM Reminders; | 2 rows inserted into Members table. However, only first row has birthDate NULL value. |

Trigger 3: Before Update

|  |  |
| --- | --- |
| Trigger name | BeforeSalesUpdate |
| Trigger type | Before Update |
| Description | If you update the value in the quantity column to a new value that is 3 times greater than the current value, the trigger raises an error and stops the update |
| Tables | Sales |

Test Case 3: Before Update

|  |  |  |
| --- | --- | --- |
| Step 1 | 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 doesn’t violate the rules. |
| Step 2 | 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 find the new quantity caused a violation and raised an error. |

Trigger 4: After Update

|  |  |
| --- | --- |
| Trigger name | AfterSalesUpdate |
| Trigger type | After Update |
| Description | If you update the value in the quantity column to a new value, the trigger insert a new row to log the changes in the SalesChanges table |
| Tables | Sales, SalesChanges |

Test Case 4: After Update

|  |  |  |
| --- | --- | --- |
| Step 1 | 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 be trigger automatically. |
| Step 2 | Update the quantity of all 3 rows by increasing 10%  UPDATE Sales  SET quantity = CAST(quantity \* 1.1 AS UNSIGNED); |  |
| Step 3 | Query data from the SalesChanges table  SELECT \*  FROM SalesChanges; | The trigger should fire 3 times because of the updates of the 3 rows. |

Trigger 5: Before Delete

|  |  |
| --- | --- |
| Trigger name | BeforeSalariesDelete |
| Trigger type | Before Delete |
| Description | Insert a new row into the SalaryArchives table before a row from the Salaries table is deleted |
| Tables | Salaries, SalaryArchives |

Test Case 5: Before Delete

|  |  |  |
| --- | --- | --- |
| Step 1 | Delete the row from Salaries table  DELETE FROM Salaries  WHERE employeeNumber = 1002;  Query data from the SalaryArchives table  SELECT \*  FROM SalaryArchives; | The trigger should invoke and insert a new row into the SalaryArchives table. |
| Step 2 | 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 2 rows from the Salaries table. |

Trigger 6: After Delete

|  |  |
| --- | --- |
| Trigger name | AfterSalariesDelete |
| Trigger type | After Delete |
| Description | Updates the total salary in the SalaryBudgets table after a row is declared from the Salaries table |
| Tables | Salaries, SalaryBudgets |

Test Case 6: After Delete

|  |  |  |
| --- | --- | --- |
| Step 1 | Delete the row from Salaries table  DELETE FROM Salaries  WHERE employeeNumber = 1002;  Query data from the SalaryBudgets table  SELECT \*  FROM SalaryBudgets; | In the output, the total should be reduced by the deleted salary. |
| Step 2 | Delete all the rows from Salaries table  DELETE FROM Salaries;  Finally, query the data from SalaryBudgets table  SELECT \*  FROM SalaryBudgets; | The trigger updated the total to zero. |