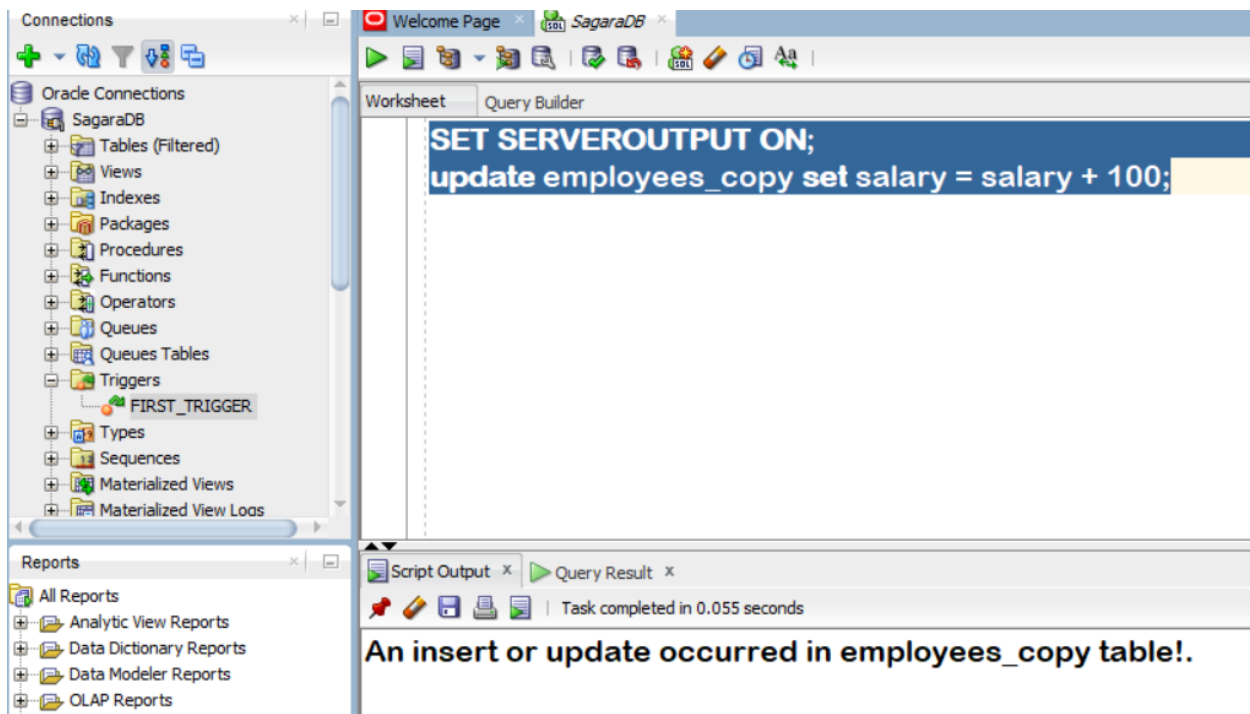
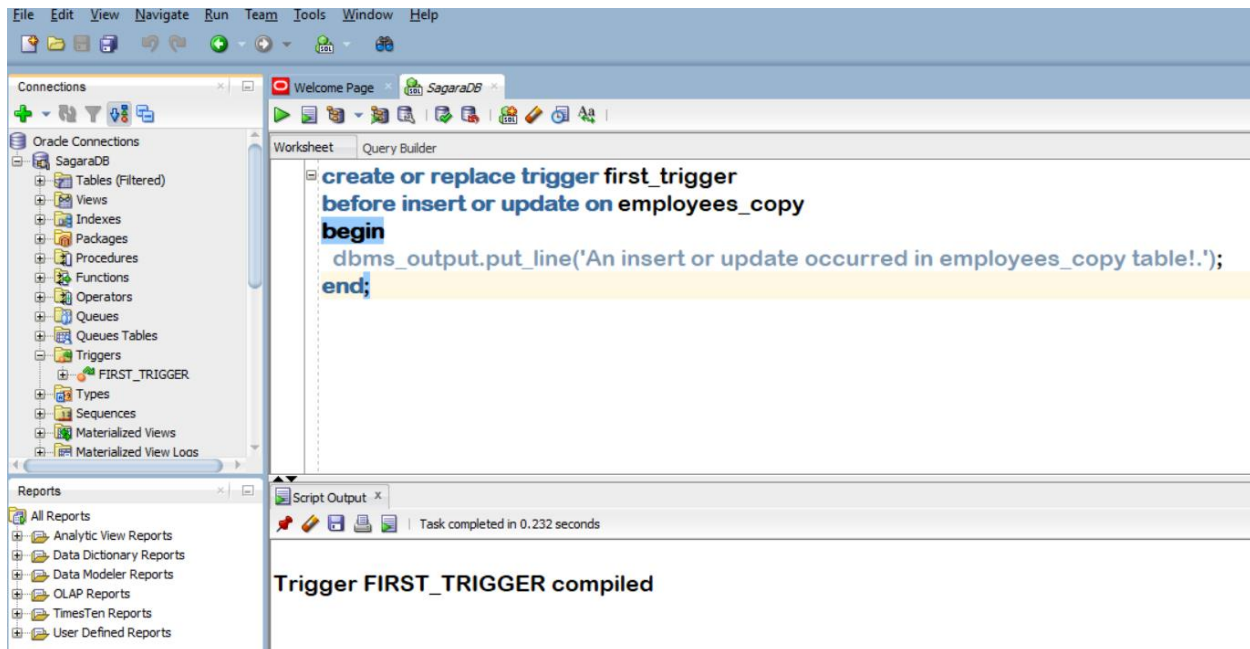
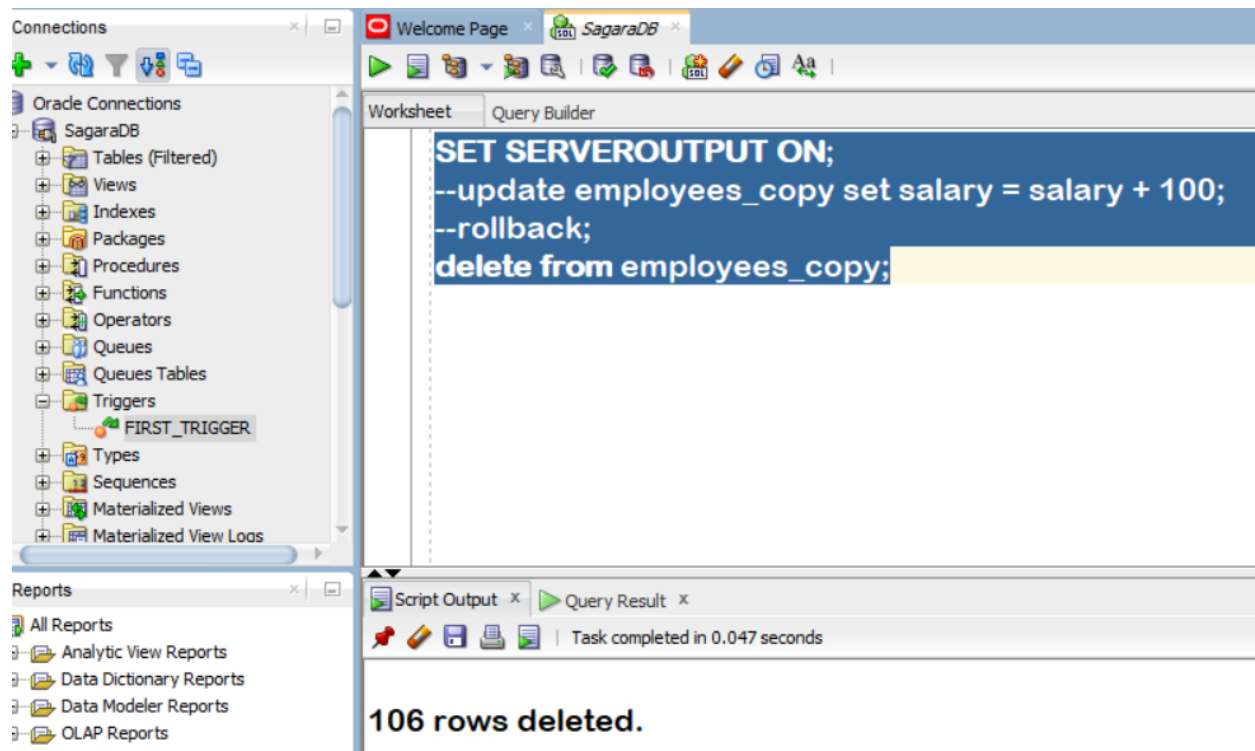
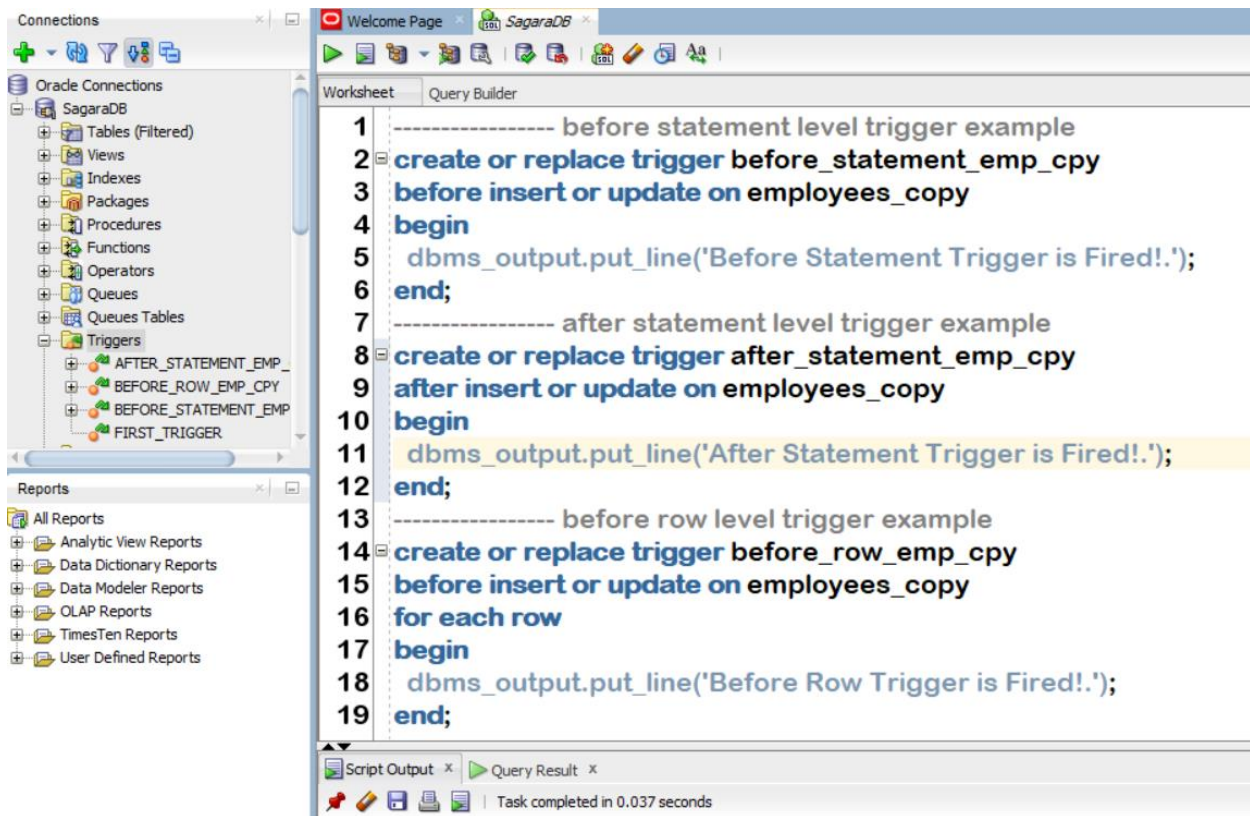


Week 10 – Trigger Lab :







Connections

- Oracle Connections
 - SagaraDB
 - Tables (Filtered)
 - Views
 - Indexes
 - Packages
 - Procedures
 - Functions
 - Operators
 - Queues
 - Queues Tables
 - Triggers
 - AFTER_ROW_EMP_CPY
 - AFTER_STATEMENT_EMP_
 - BEFORE_ROW_EMP_CPY
 - BEFORE_STATEMENT_EMP
 - FIRST_TRIGGER
 - Types
 - Sequences
 - Materialized Views

Reports

- All Reports
 - Analytic View Reports
 - Data Dictionary Reports
 - Data Modeler Reports
 - OLAP Reports
 - TimesTen Reports
 - User Defined Reports

Worksheet | **Query Builder**

```

1 ----- after row level trigger example
2 create or replace trigger after_row_emp_cpy
3 after insert or update on employees_copy
4 for each row
5 begin
6   dbms_output.put_line('After Row Trigger is Fired!');
7 end;
8
9 SET SERVEROUTPUT ON;
10 update employees_copy set salary = salary + 100 where employee_id = 100;
11 --update employees_copy set salary = salary + 100 where employee_id = 99;
12 --update employees_copy set salary = salary + 100
13 --where department_id = 30;
  
```

Script Output | **Query Result**

Task completed in 0.045 seconds

Before Statement Trigger is Fired!.
An insert or update occurred in employees_copy table!.
After Statement Trigger is Fired!.

Connections

- Oracle Connections
 - SagaraDB
 - Tables (Filtered)
 - Views
 - Indexes
 - Packages
 - Procedures
 - Functions
 - Operators
 - Queues
 - Queues Tables
 - Triggers
 - AFTER_ROW_EMP_CPY
 - AFTER_STATEMENT_EMP_
 - BEFORE_ROW_EMP_CPY
 - BEFORE_STATEMENT_EMP
 - FIRST_TRIGGER
 - Types
 - Sequences
 - Materialized Views

Reports

- All Reports
 - Analytic View Reports
 - Data Dictionary Reports
 - Data Modeler Reports
 - OLAP Reports
 - TimesTen Reports
 - User Defined Reports

Worksheet | **Query Builder**

```

1 ----- after row level trigger example
2 create or replace trigger after_row_emp_cpy
3 after insert or update on employees_copy
4 for each row
5 begin
6   dbms_output.put_line('After Row Trigger is Fired!');
7 end;
8
9 SET SERVEROUTPUT ON;
10 --update employees_copy set salary = salary + 100 where employee_id = 100;
11 update employees_copy set salary = salary + 100 where employee_id = 99;
12 --update employees_copy set salary = salary + 100
13 --where department_id = 30;
  
```

Script Output | **Query Result**

Task completed in 0.067 seconds

Before Statement Trigger is Fired!.
An insert or update occurred in employees_copy table!.
After Statement Trigger is Fired!.

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows a connection to 'SagaraDB'. The 'Schema Browser' on the left lists various database objects, including 'Triggers' which contains several triggers like 'AFTER_ROW_EMP_CPY', 'AFTER_STATEMENT_EMP_', 'BEFORE_ROW_EMP_CPY', 'BEFORE_STATEMENT_EMP_', and 'FIRST_TRIGGER'. The main 'Worksheet' area shows a SQL script for creating and testing an after-row trigger. The script includes comments and SQL commands to create the trigger, set server output, and perform updates on the 'employees_copy' table. The 'Script Output' pane at the bottom shows the results of the script execution, indicating that the trigger was successfully created and fired.

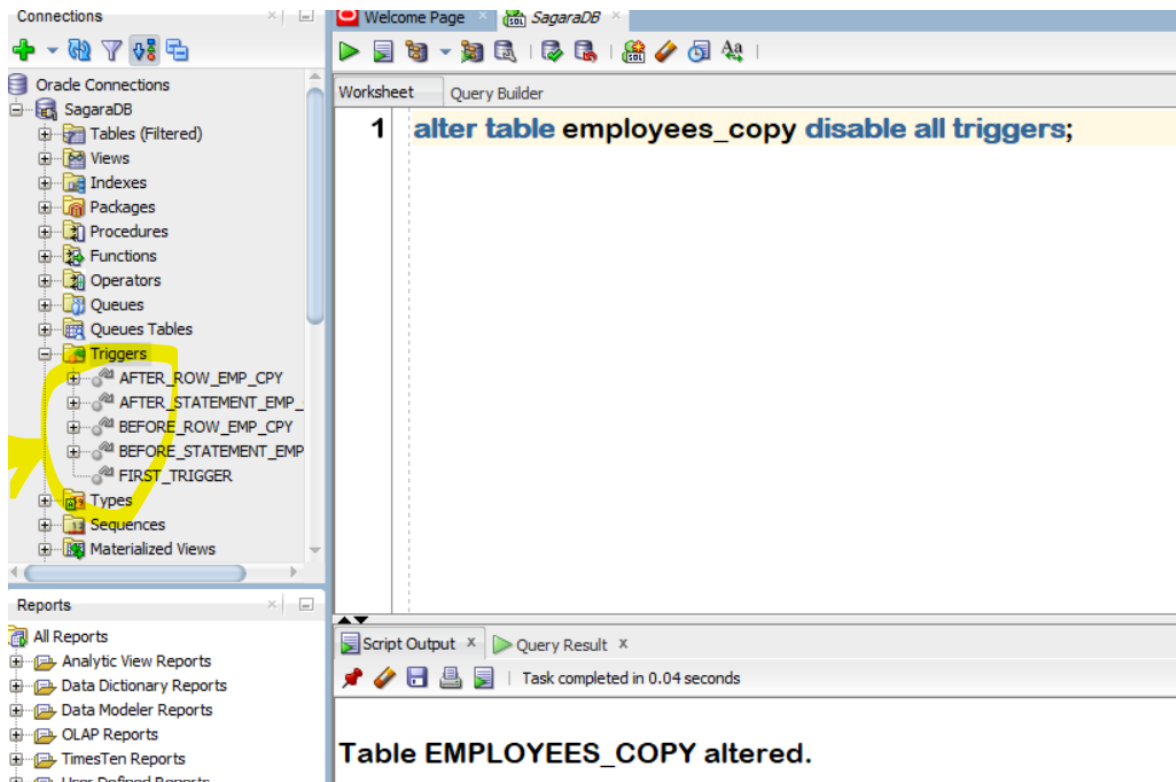
```

1 ----- after row level trigger example
2 create or replace trigger after_row_emp_cpy
3 after insert or update on employees_copy
4 for each row
5 begin
6   dbms_output.put_line('After Row Trigger is Fired!.');
7 end;
8
9 SET SERVEROUTPUT ON;
10 --update employees_copy set salary = salary + 100 where employee_id = 100;
11 --update employees_copy set salary = salary + 100 where employee_id = 99;
12 update employees_copy set salary = salary + 100
13 where department_id = 30;

```

Script Output: Task completed in 0.055 seconds

Before Statement Trigger is Fired!.
 An insert or update occurred in employees_copy table!.
 After Statement Trigger is Fired!.



The screenshot displays the Oracle SQL Developer environment. On the left, the 'Connections' pane shows a tree view of the 'SagaraDB' schema, including tables, views, indexes, packages, procedures, functions, operators, queues, queues tables, triggers, types, sequences, and materialized views. The 'Triggers' folder is expanded, showing a list of triggers: AFTER_ROW_EMP_CPY, AFTER_STATEMENT_EMP_CPY, BEFORE_ROW_EMP_CPY, BEFORE_STATEMENT_EMP_CPY, and FIRST_TRIGGER. Below this is the 'Reports' pane with a list of report types: All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports.

The main workspace is titled 'Query Builder' and contains a SQL script for creating a trigger. The script is as follows:

```
1 alter table employees_copy disable all triggers;
2
3 create or replace trigger before_row_emp_cpy
4 before insert or update or delete on employees_copy
5 --referencing old as O new as N
6 for each row
7 begin
8   dbms_output.put_line('Before Row Trigger is Fired!.');
9   dbms_output.put_line('The Salary of Employee '||:old.employee_id
10   ||' -> Before:|| :old.salary||' After:||:new.salary);
11 end;
```

Below the script, the 'Script Output' pane shows the message: 'Trigger BEFORE_ROW_EMP_CPY compiled'. The 'Query Result' pane is empty. The status bar at the bottom indicates 'Task completed in 0.043 seconds'.

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows a tree view of the database schema, including tables, views, indexes, and triggers. The 'Triggers' folder is expanded, showing a trigger named 'BEFORE_ROW_EMP_CPY'. The main window shows a SQL script in the 'Query Builder' tab. The script is as follows:

```

1 create or replace trigger before_row_emp_cpy
2 before insert or update or delete on employees_copy
3 for each row
4 begin
5   dbms_output.put_line('Before Row Trigger is Fired!.');
6   dbms_output.put_line('The Salary of Employee '||:old.employee_id
7   ||' -> Before:|| :old.salary||' After:||:new.salary);
8 end;
9 /
10 SET SERVEROUTPUT ON;
11 update employees_copy set salary = salary + 100
12 where department_id=30;

```

Below the script, the 'Script Output' pane shows the results of the execution. The output is as follows:

```

Before Row Trigger is Fired!.
The Salary of Employee 114 -> Before:11000 After:11100
Before Row Trigger is Fired!.
The Salary of Employee 115 -> Before:3100 After:3200
Before Row Trigger is Fired!.
The Salary of Employee 116 -> Before:2900 After:3000
Before Row Trigger is Fired!.

```