Experiment No 10
Aim: Implementation of Views and Triggers
Class: SE Comp
Year: 2020-21
Performed by: Danyl Fernandes, 72
Performance Date: 22-04-2021

Views:

- A view is a database object that has no values. Its contents are based on the base table.
- It contains rows and columns similar to the real table. In MySQL, the View is a **virtual table** created by a query by joining one or more tables.
- It is operated similarly to the base table but does not contain any data of its own. The View and table have one main difference that the views are definitions built on top of other tables (or views).
- If any changes occur in the underlying table, the same changes reflected in the View also.

Syntax:

```
CREATE [OR REPLACE] VIEW view_name AS
SELECT columns
FROM tables
[WHERE conditions];
```

Parameters:

The view syntax contains the following parameters:

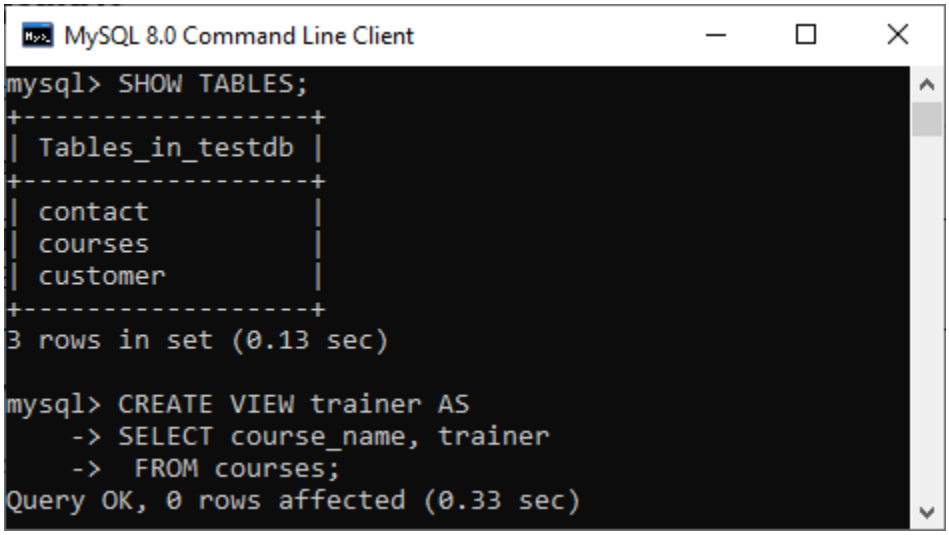
OR REPLACE: It is optional. It is used when a VIEW already exists. If you do not specify this clause and the VIEW already exists, the CREATE VIEW statement will return an error.

view_name: It specifies the name of the VIEW that you want to create in MySQL.

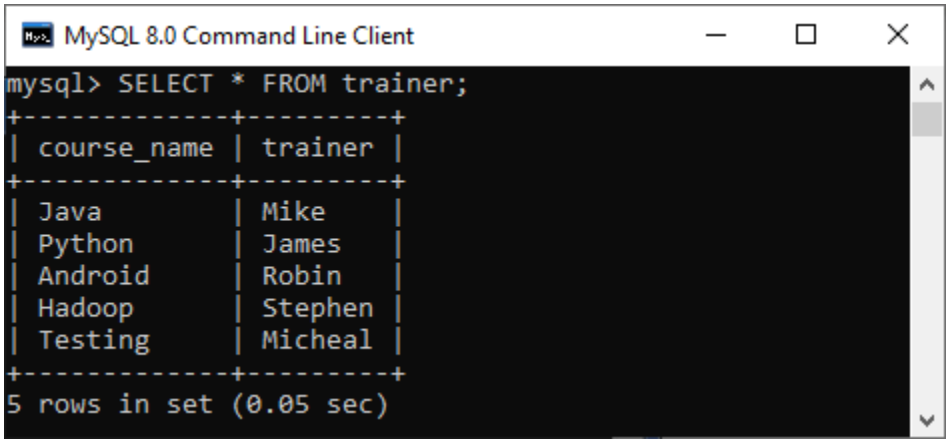
WHERE conditions: It is also optional. It specifies the conditions that must be met for the records to be included in the VIEW.

Example:

```
CREATE VIEW trainer AS
SELECT course_name, trainer
FROM courses;
```




```
SELECT * FROM trainer;
```



Triggers:

- A trigger in MySQL is a set of SQL statements that reside in a system catalog.
- **It is a special type of stored procedure that is invoked automatically in response to an event**.
- Each trigger is associated with a table, which is activated on any DML statement such as **INSERT, UPDATE**, or **DELETE**.
- A trigger is called a special procedure because it cannot be called directly like a stored procedure.
- The main difference between the trigger and procedure is that a trigger is called automatically when a data modification event is made against a table.
- In contrast, a stored procedure must be called explicitly.

Syntax:

```
CREATE TRIGGER trigger_name
(AFTER | BEFORE) (INSERT | UPDATE | DELETE)
ON table_name FOR EACH ROW
BEGIN
     --variable declarations
     --trigger code
END;
```

Types of Triggers:

We can define the maximum six types of actions or events in the form of triggers:

- **Before Insert:** It is activated before the insertion of data into the table.
- **After Insert:** It is activated after the insertion of data into the table.
- **Before Update:** It is activated before the update of data in the table.
- **After Update:** It is activated after the update of the data in the table.
- **Before Delete:** It is activated before the data is removed from the table.
- **After Delete:** It is activated after the deletion of data from the table.

Limitations of Using Triggers:

- MySQL triggers do not allow the use of all validations; they only provide extended validations. **For example**, we can use the NOT NULL, UNIQUE, CHECK and FOREIGN KEY constraints for simple validations.
- Triggers are invoked and executed invisibly from the client application. Therefore, it isn't easy to troubleshoot what happens in the database layer.
- Triggers may increase the overhead of the database server.

Query:

```
create view full_sailor_view as select * from sailor;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0060 seconds.)

```
create view full_sailor_view as select * from sailor
```

Showing rows 0 - 20 (21 total, Query took 0.0019 seconds.)

```
SELECT * FROM `full_sailor_view`
```

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	sid	sname	address	rating	age
Edit Copy Delete	1	Rahul	Mumbai	5	34
Edit Copy Delete	2	Danyl	Pune	5	21
Edit Copy Delete	3	Shrinath	Hyderabad	5	22
Edit Copy Delete	4	Jack	Pennsylvania	4	32
Edit Copy Delete	5	Sean	Ontario	3	30
Edit Copy Delete	6	Matt	Texas	3	29
Edit Copy Delete	7	Hans	Berlin	4	28
Edit Copy Delete	8	Drishyam	Telangana	2	29

Query:

```
create view rating_3 as select * from sailor where rating = 3;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0065 seconds.)

```
create view rating_3 as select * from sailor where rating = 3
```

Showing rows 0 - 4 (5 total, Query took 0.0012 seconds.)

```
SELECT * FROM `rating_3`
```

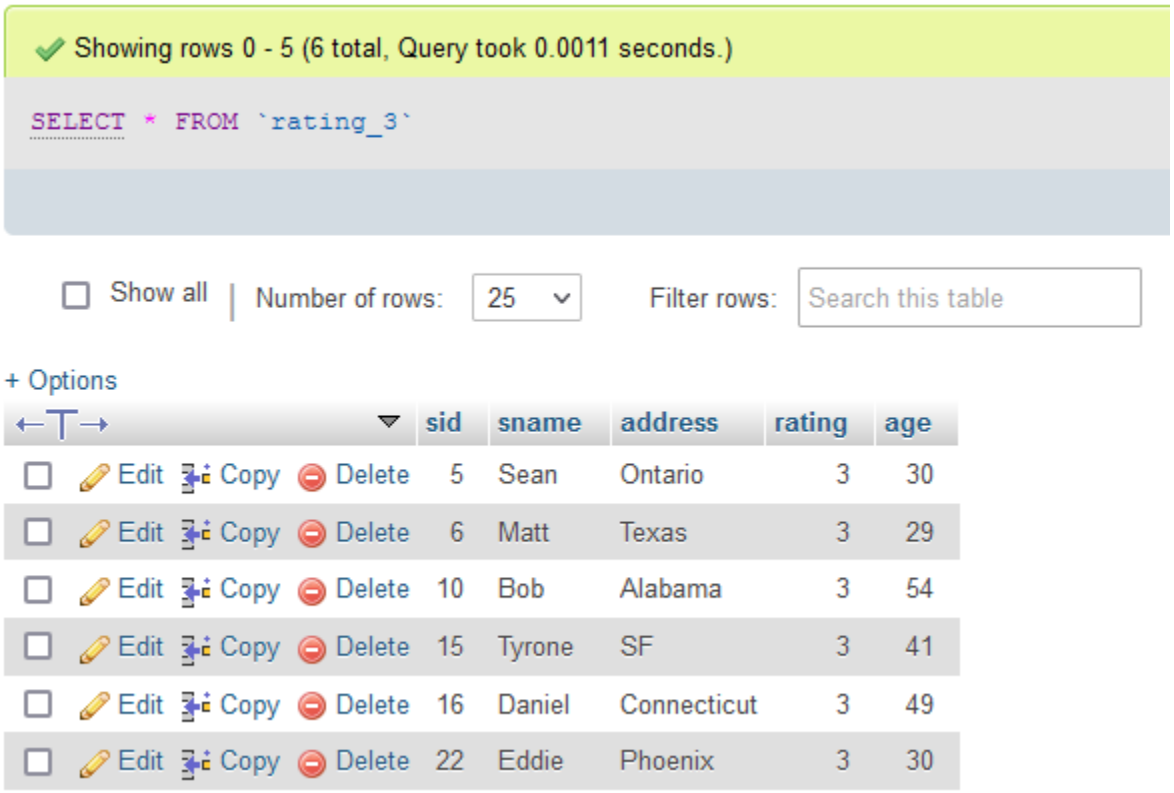
Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	sid	sname	address	rating	age
Edit Copy Delete	5	Sean	Ontario	3	30
Edit Copy Delete	6	Matt	Texas	3	29
Edit Copy Delete	10	Bob	Alabama	3	54
Edit Copy Delete	15	Tyrone	SF	3	41
Edit Copy Delete	16	Daniel	Connecticut	3	49

Query:

```
insert into sailor values (22, "Eddie", "Phoenix", 3, 30);
```

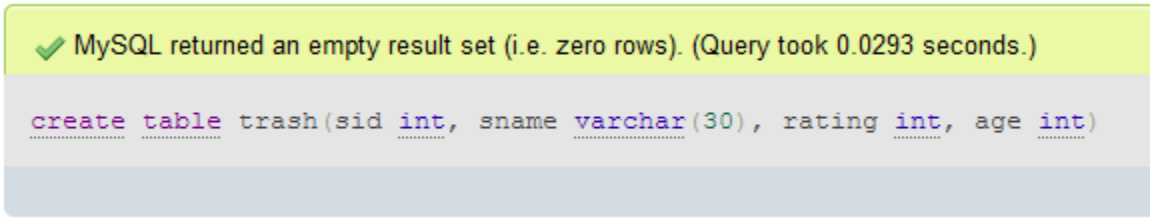


	sid	sname	address	rating	age
Edit Copy Delete	5	Sean	Ontario	3	30
Edit Copy Delete	6	Matt	Texas	3	29
Edit Copy Delete	10	Bob	Alabama	3	54
Edit Copy Delete	15	Tyrone	SF	3	41
Edit Copy Delete	16	Daniel	Connecticut	3	49
Edit Copy Delete	22	Eddie	Phoenix	3	30

Delete Trigger:

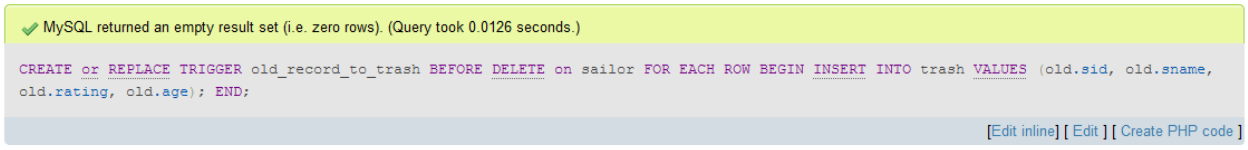
Query:

```
create table trash (
     sid int,
     sname varchar(30),
     rating int,
     age int
);
```



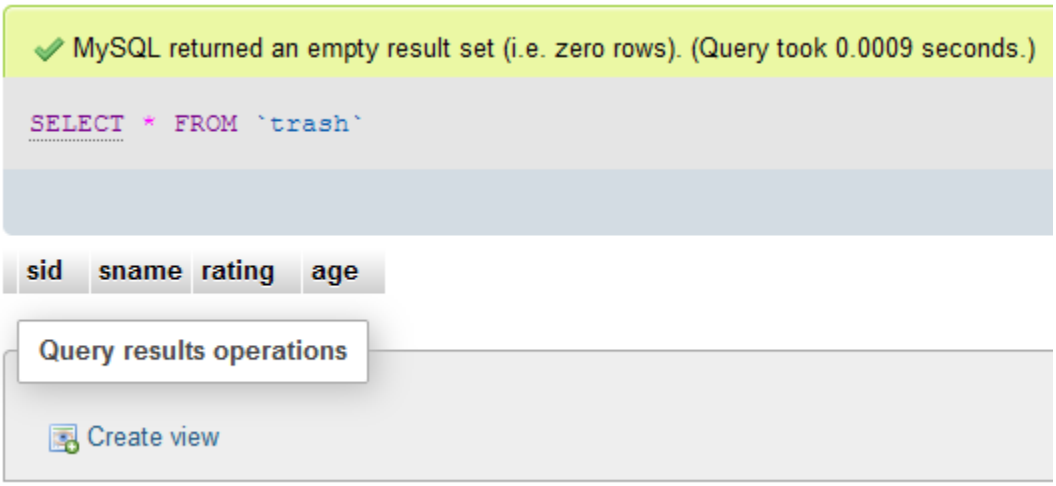
Query:

```
DELIMITER $
CREATE or REPLACE TRIGGER old_record_to_trash BEFORE DELETE on
sailor
FOR EACH ROW
BEGIN
    INSERT INTO trash VALUES (old.sid, old.sname, old.rating,
old.age);
END;
$
```



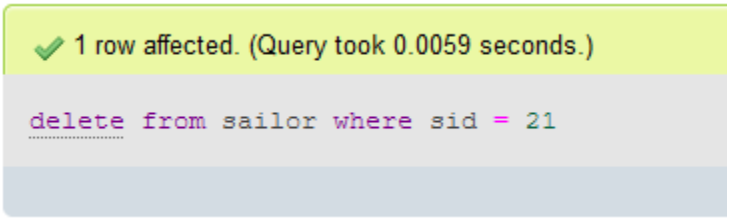
Query:

```
select * from trash
```



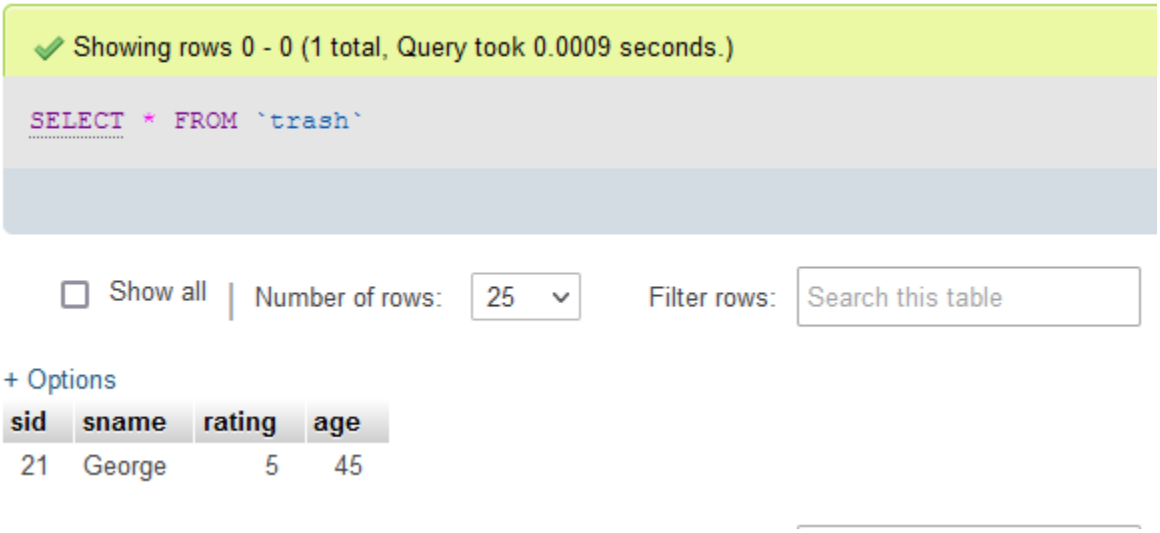
Query:

```
delete from sailor where sid = 21
```



Query:

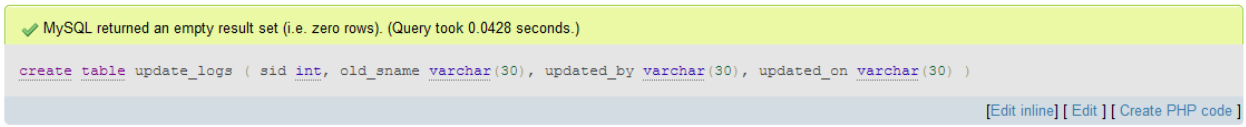
```
select * from trash
```



Before Update Trigger:

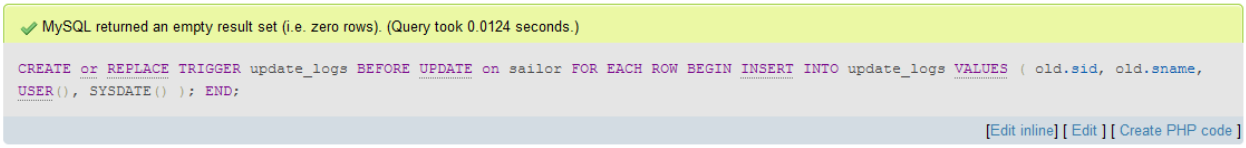
Query:

```
create table update_logs (
     sid int,
     old_sname varchar(30),
     updated_by varchar(30),
     Updated_on varchar(30)
);
```



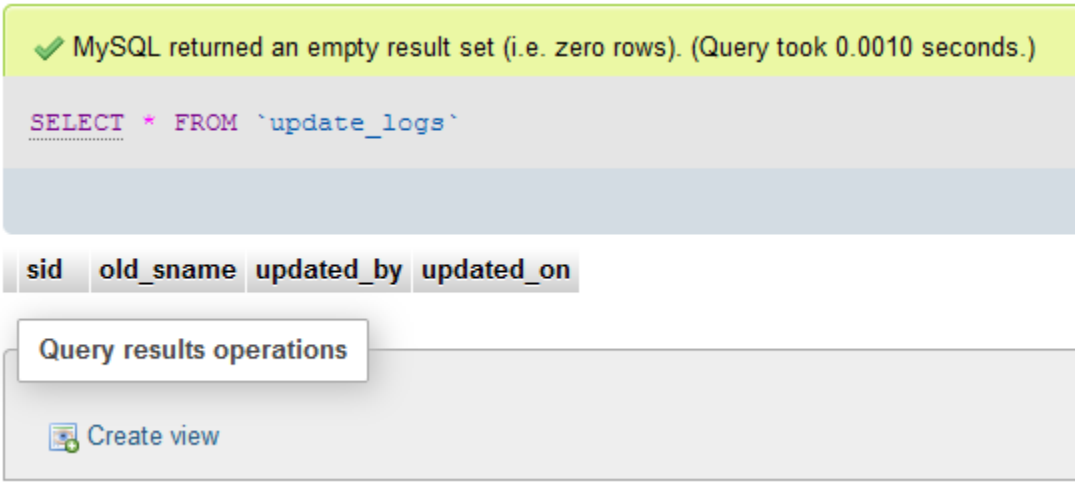
Query:

```
DELIMITER $
CREATE or REPLACE TRIGGER update_logs BEFORE UPDATE on sailor
FOR EACH ROW
BEGIN
    INSERT INTO update_logs VALUES (
          old.sid,
          old.sname,
          USER(),
          SYSDATE()
     );
END;
$
```



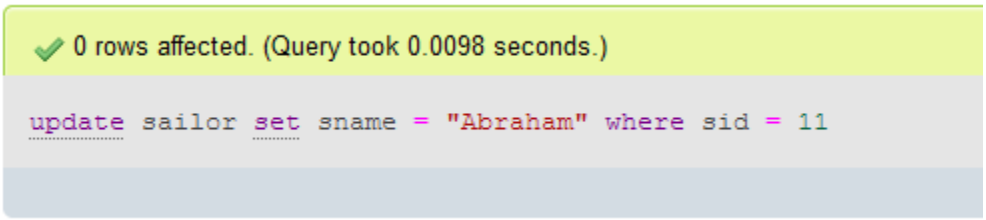
Query:

```
select * from update_logs
```



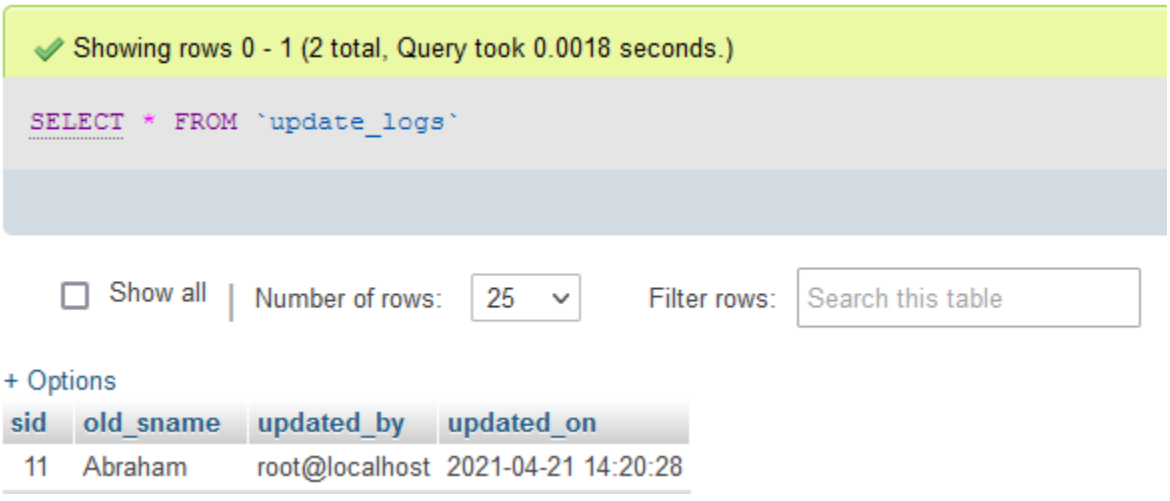
Query:

```
update sailor set name = “Abraham” where sid = 11
```



Query:

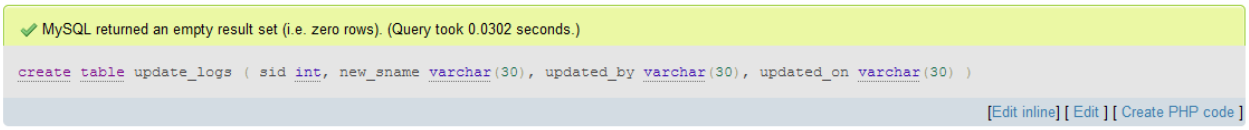
```
select * from update_logs
```



After Update Trigger:

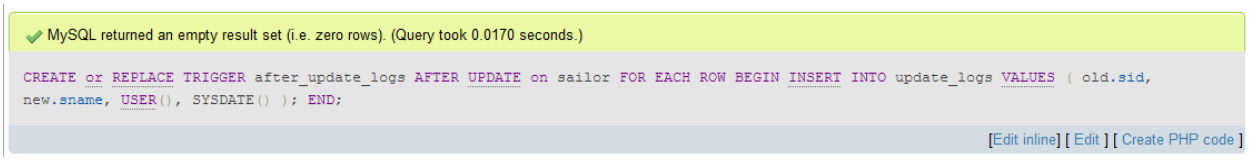
Query:

```
create table update_logs (
     sid int,
     new_sname varchar(30),
     updated_by varchar(30),
     updated_on varchar(30)
);
```



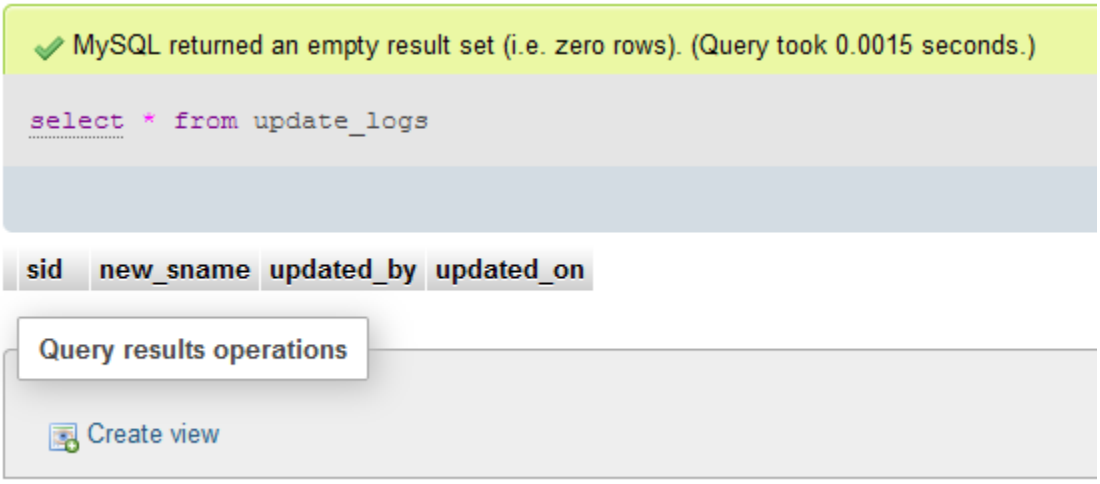
Query:

```
DELIMITER $
CREATE or REPLACE TRIGGER after_update_logs AFTER UPDATE on
sailor
FOR EACH ROW
BEGIN
    INSERT INTO update_logs VALUES (
          old.sid,
          new.sname,
          USER(),
          SYSDATE()
     );
END;
$
```



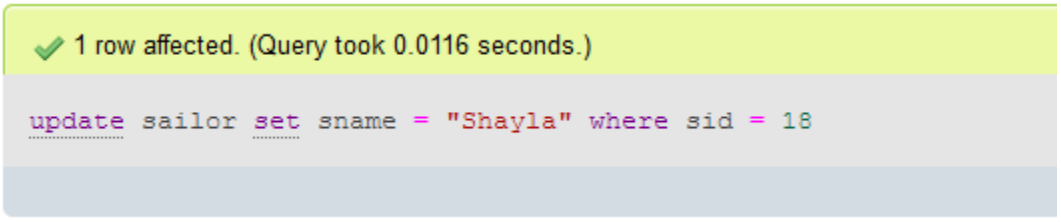
Query:

```
select * from update_logs
```



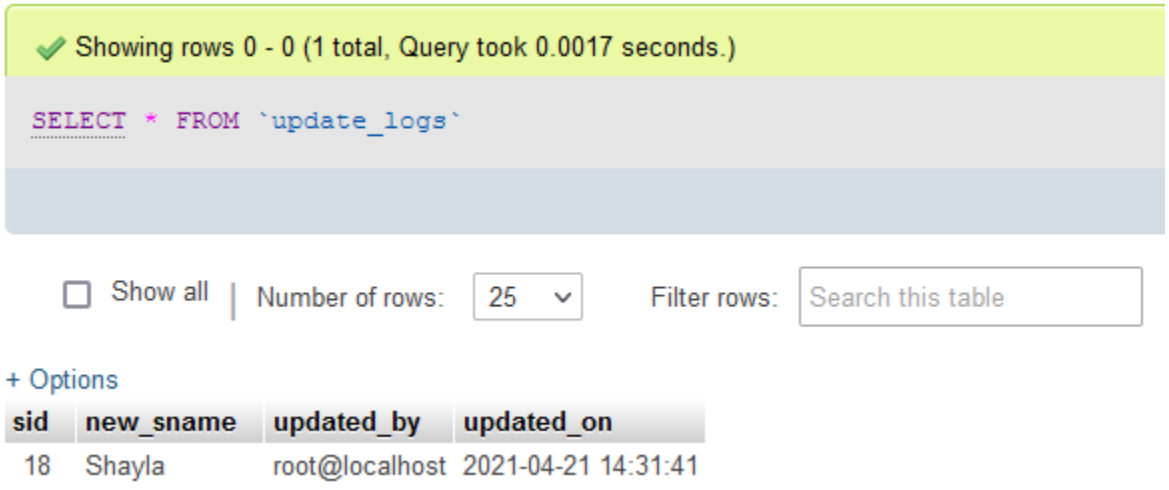
Query:

```
update sailor set sname = “Shayla” where sid = 18
```



Query:

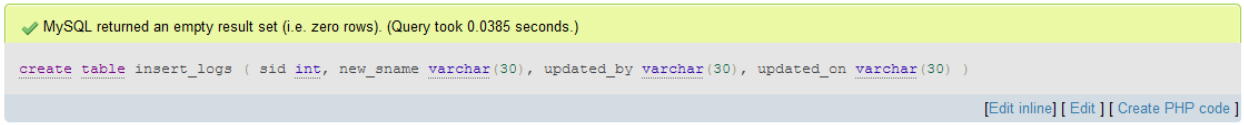
```
select * from update_logs
```



After Insert Trigger:

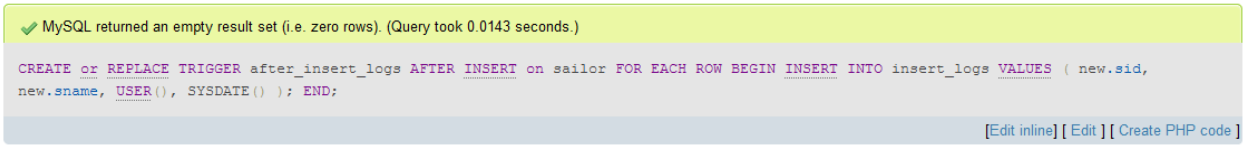
Query:

```
create table insert_logs (
     sid int,
     new_sname varchar(30),
     updated_by varchar(30),
     updated_on varchar(30)
);
```



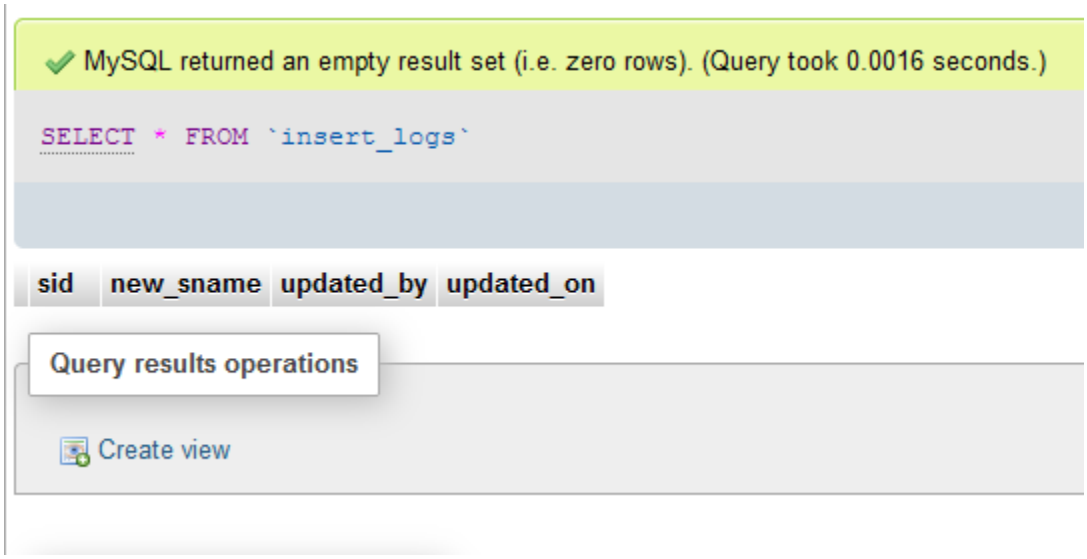
Query:

```
DELIMITER $
CREATE or REPLACE TRIGGER after_update_logs AFTER INSERT on
sailor
FOR EACH ROW
BEGIN
    INSERT INTO update_logs VALUES (
          old.sid,
          new.sname,
          USER(),
          SYSDATE()
     );
END;
$
```



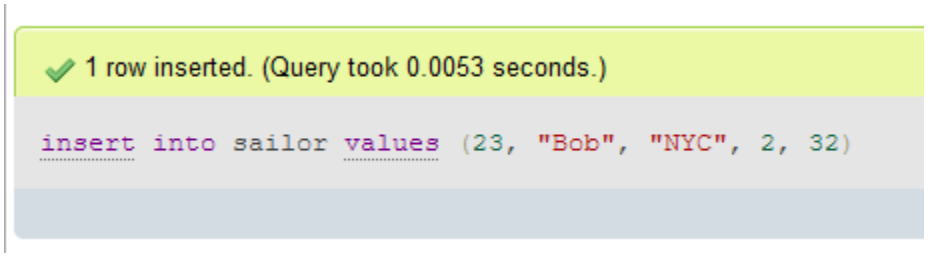
Query:

```
select * from insert_logs
```



Query:

```
insert into sailor values (23, “Bob”, “NYC”, 2, 32)
```



Query:

```
select * from insert_logs
```