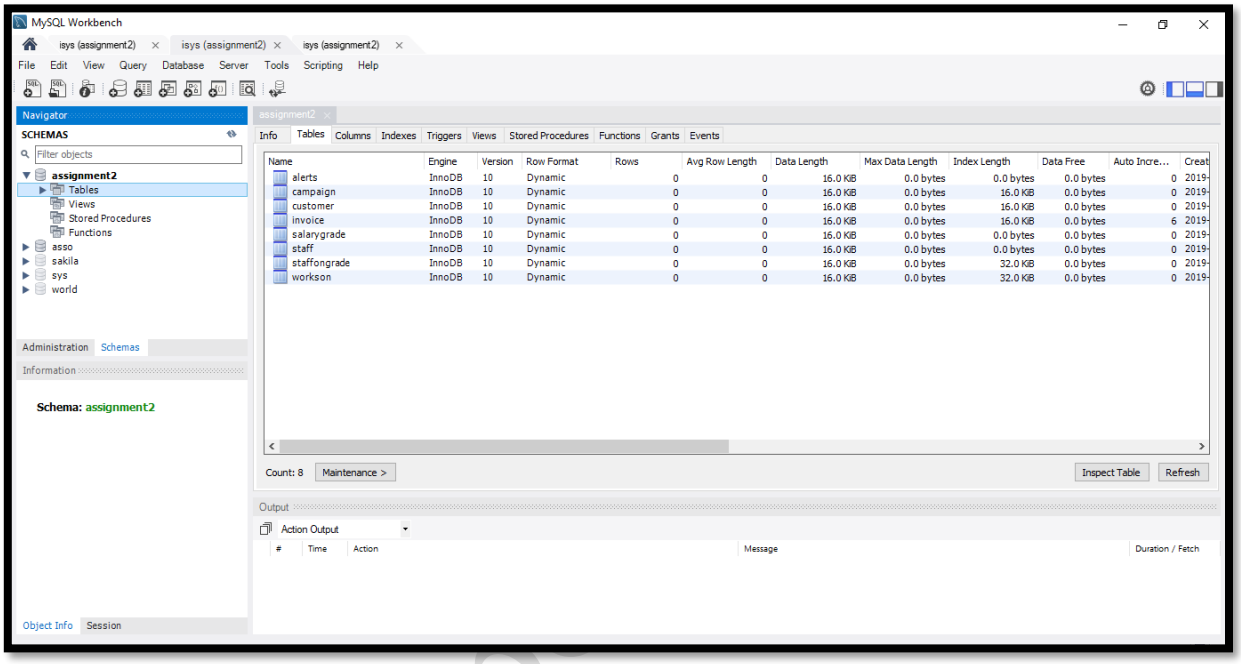
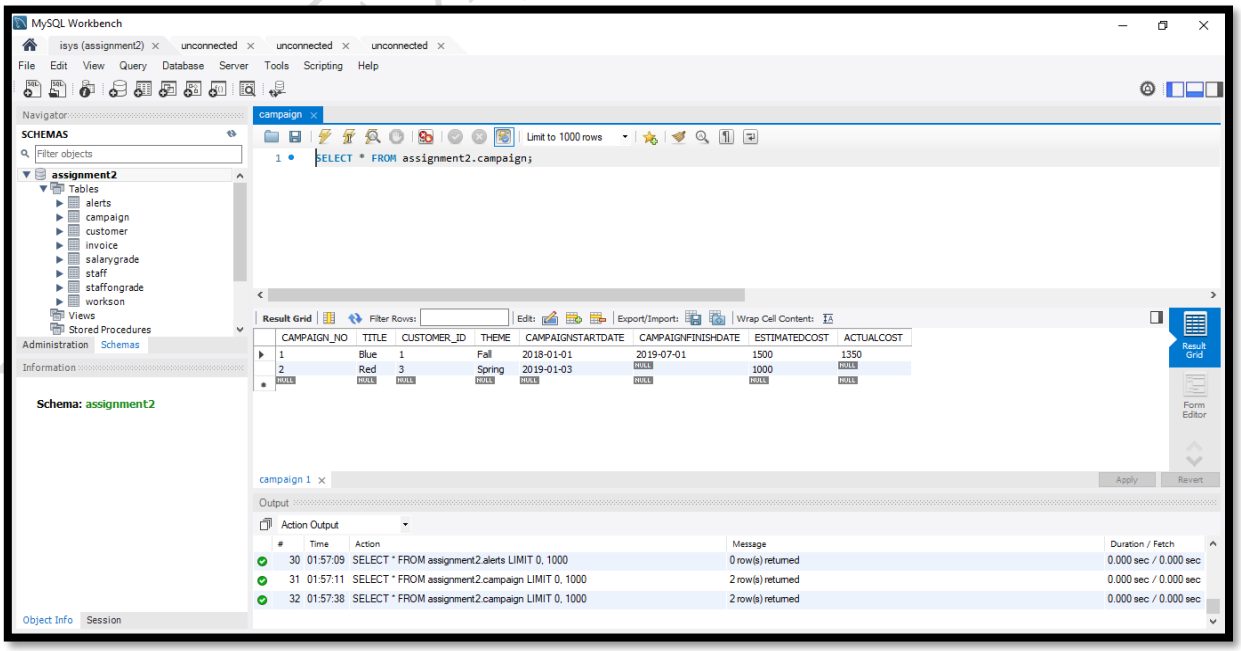
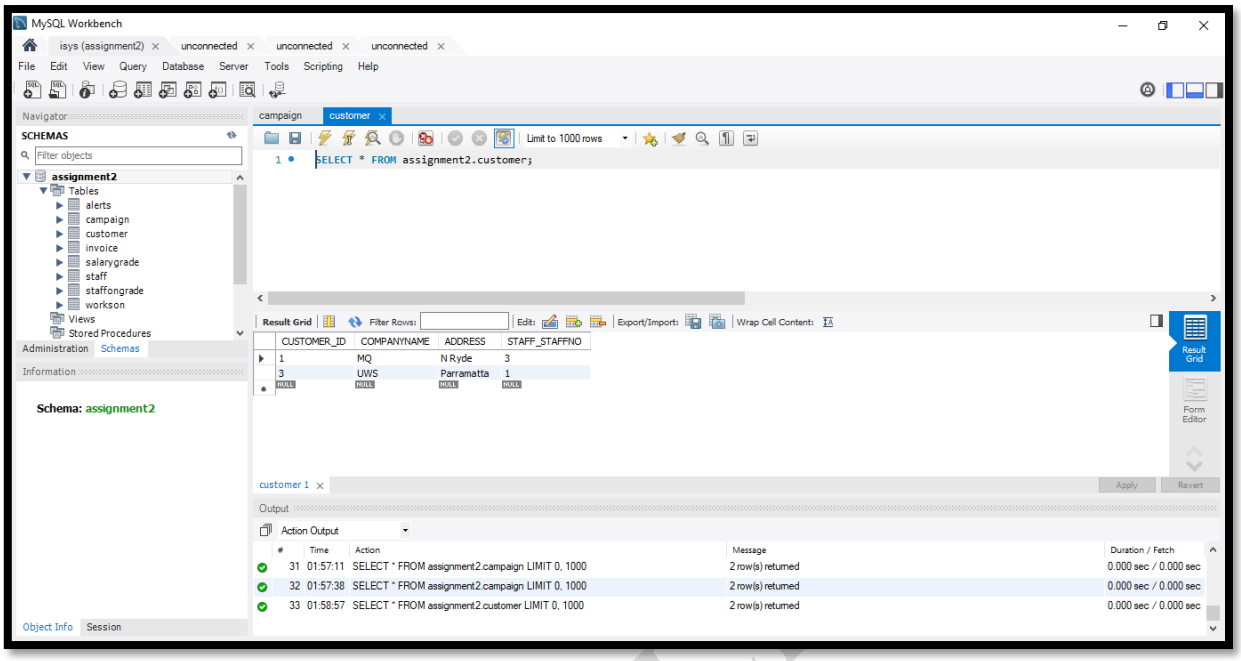
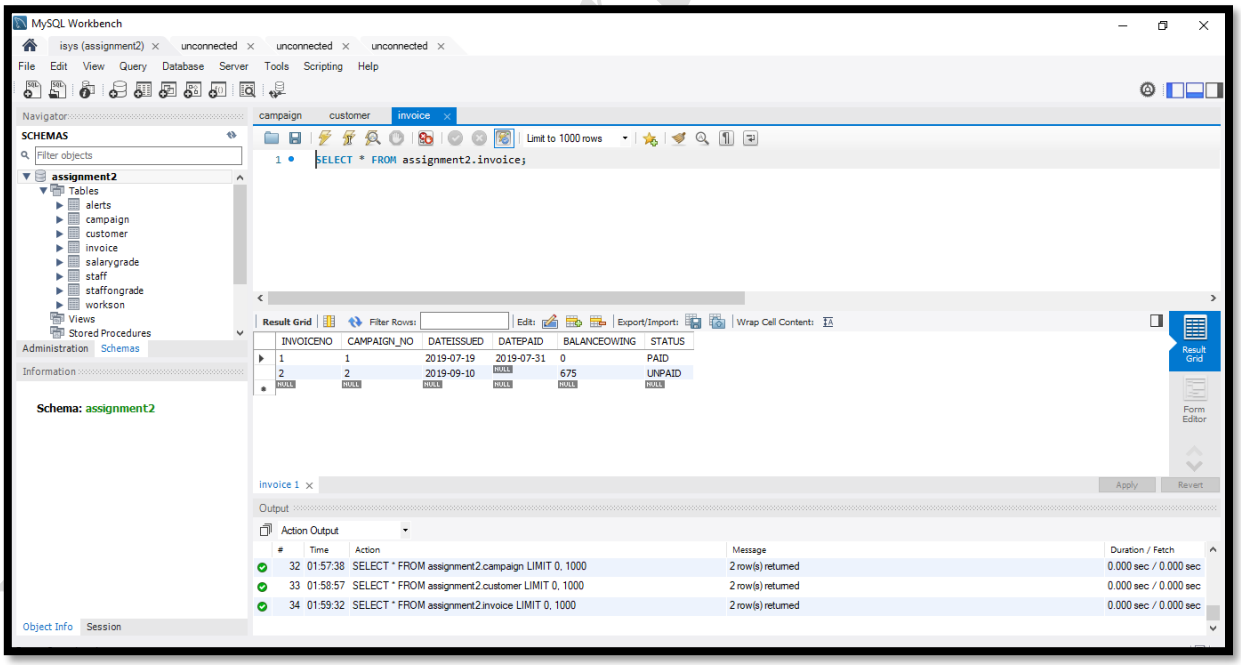
1. Initial State of the database

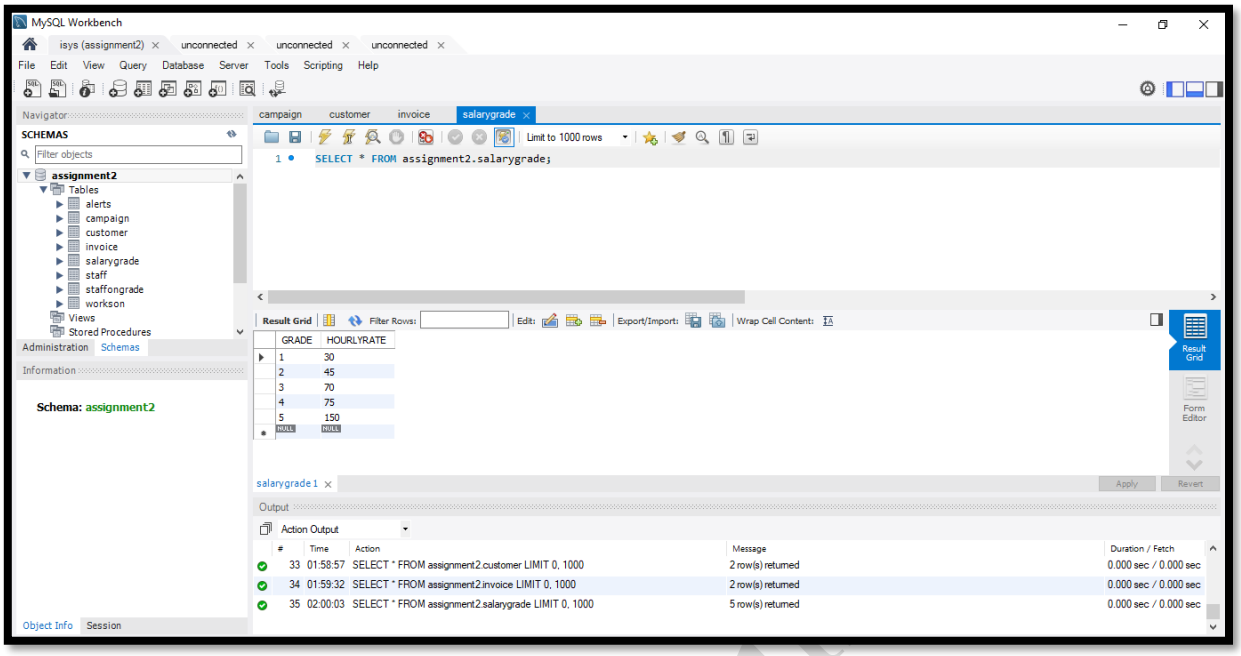
Paste here the screen shots showing the provided sample data in the tables.
Relation data model:

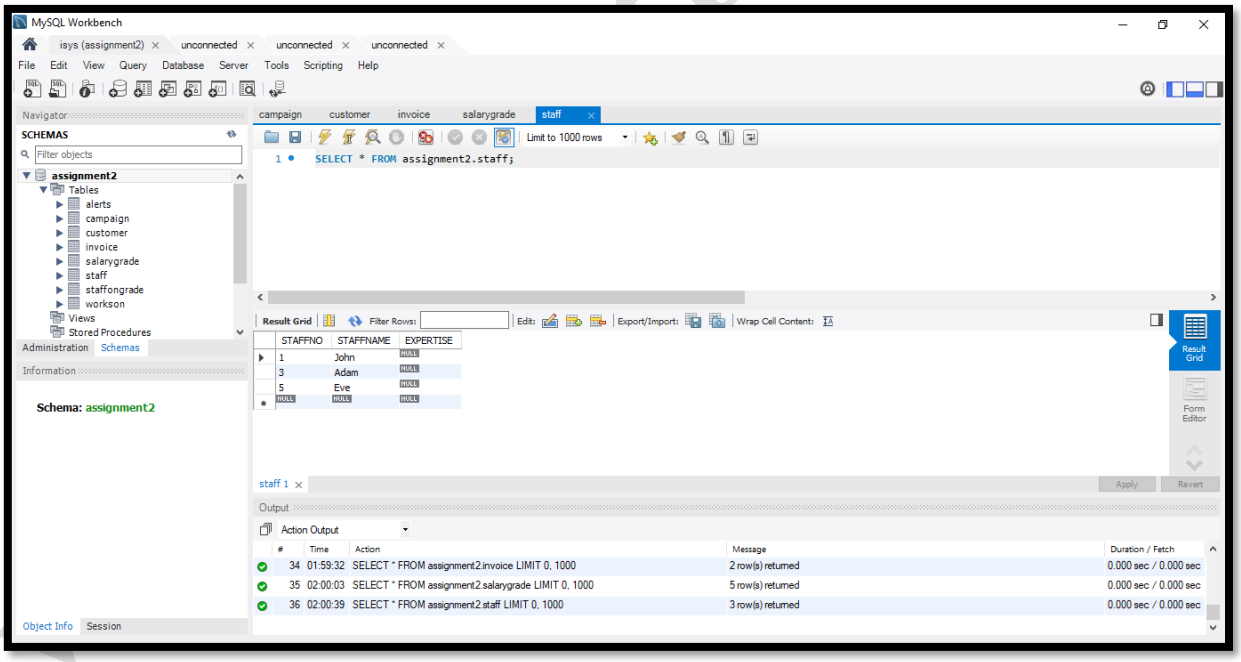


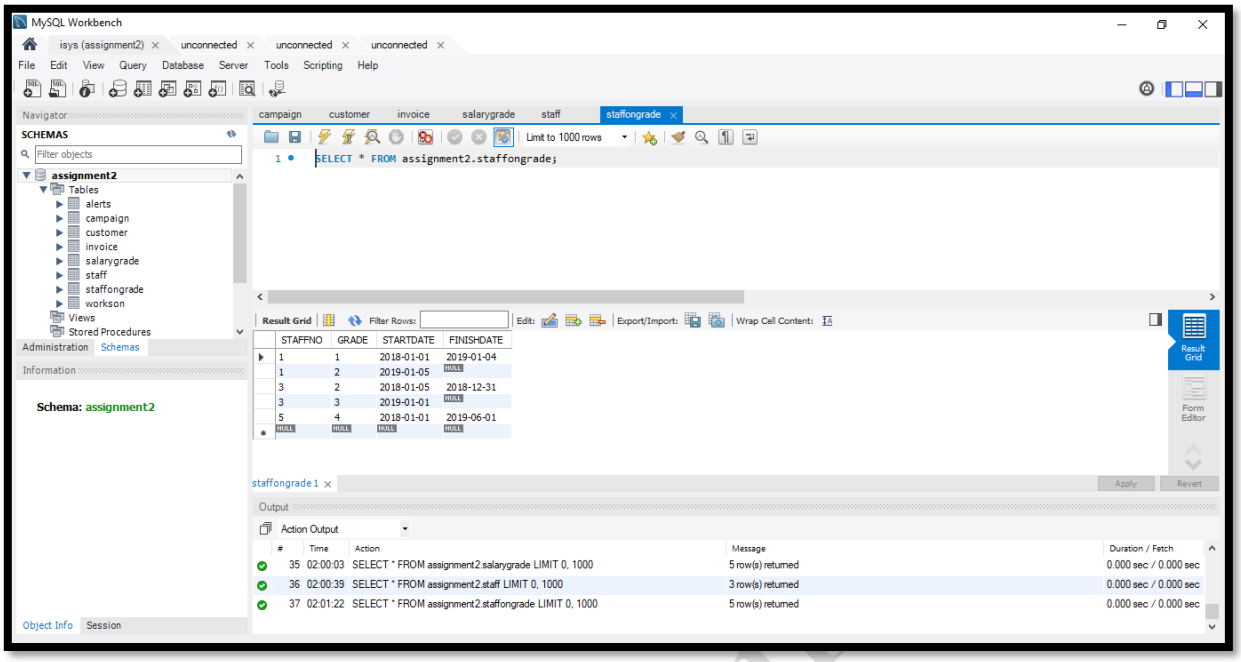


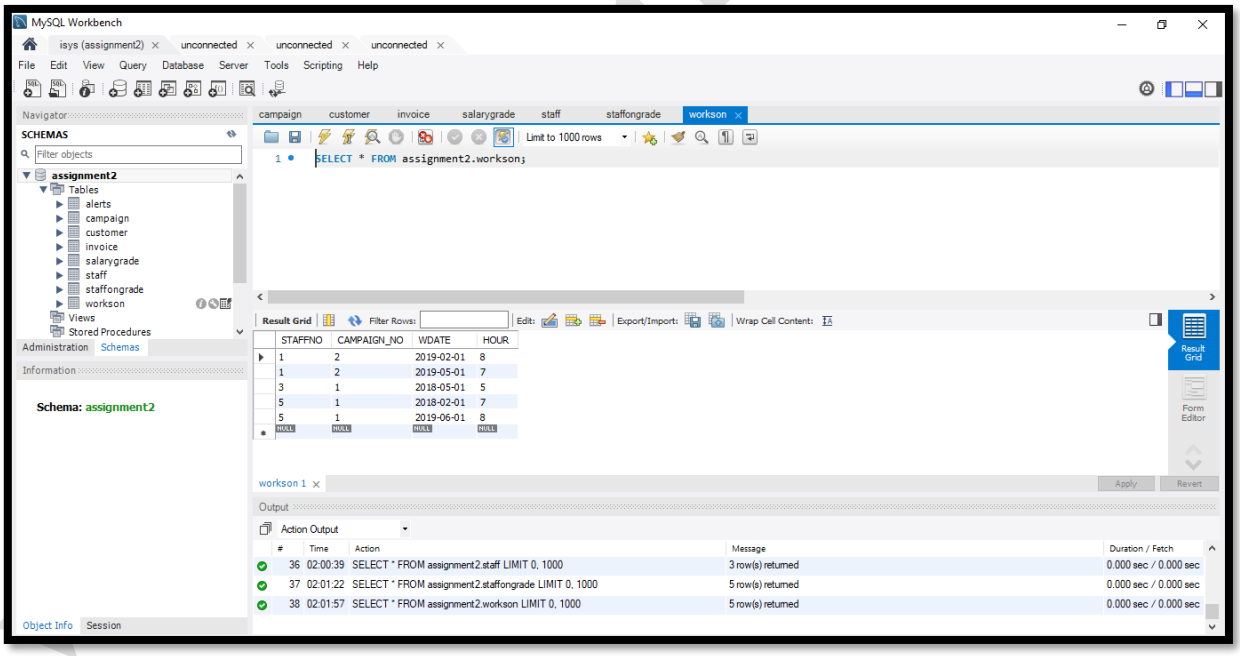

MySQL Workbench
isys (assignment2)
unconnected
unconnected
unconnected
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
assignment2
Tables
alerts
campaign
customer
invoice
salarygrade
staff
staffongrade
workson
Views
Stored Procedures
Administration Schemas
Information
Schema: assignment2
Object Info Session
campaign customer
Limit to 1000 rows
1 SELECT * FROM assignment2.customer;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
CUSTOMER_ID COMPANYNAME ADDRESS STAFF_STAFFNO
1 MQ N Ryde 3
3 UWS Parramatta 1
Result Grid
Form Editor
customer 1
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
31 01:57:11 SELECT * FROM assignment2.campaign LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
32 01:57:38 SELECT * FROM assignment2.campaign LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
33 01:58:57 SELECT * FROM assignment2.customer LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec

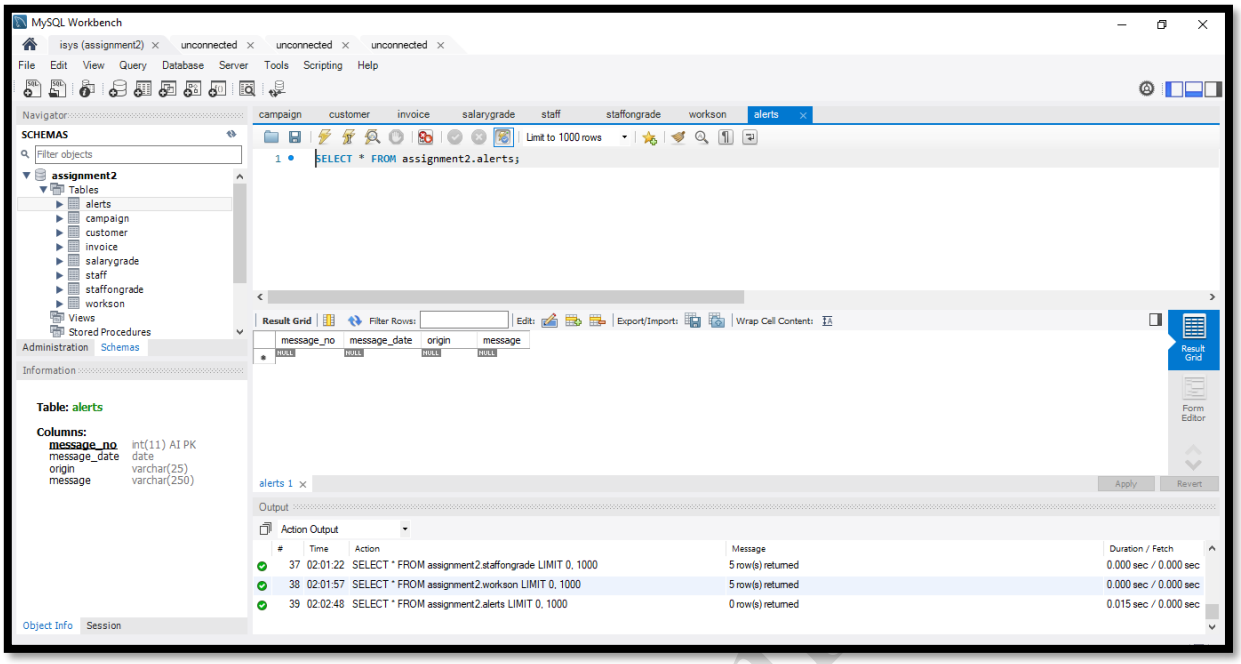

MySQL Workbench
isys (assignment2)
unconnected
unconnected
unconnected
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
assignment2
Tables
alerts
campaign
customer
invoice
salarygrade
staff
staffongrade
workson
Views
Stored Procedures
Administration Schemas
Information
Schema: assignment2
Object Info Session
campaign customer invoice
Limit to 1000 rows
1 SELECT * FROM assignment2.invoice;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
INVOICENO CAMPAIGN_NO DATEISSUED DATEPAID BALANCEOWING STATUS
1 1 2019-07-19 2019-07-31 0 PAID
2 2 2019-09-10 675 UNPAID
Result Grid
Form Editor
invoice 1
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
32 01:57:38 SELECT * FROM assignment2.campaign LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
33 01:58:57 SELECT * FROM assignment2.customer LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
34 01:59:32 SELECT * FROM assignment2.invoice LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec


MySQL Workbench
isys (assignment2)
unconnected
unconnected
unconnected
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
assignment2
Tables
alerts
campaign
customer
invoice
salarygrade
staff
staffongrade
workson
Views
Stored Procedures
Administration Schemas
Information
Schema: assignment2
Object Info Session
campaign customer invoice salarygrade
Limit to 1000 rows
1 SELECT * FROM assignment2.salarygrade;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
GRADE HOURLYRATE
1 30
2 45
3 70
4 75
5 150
NULL NULL
Result Grid
Form Editor
salarygrade 1
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
33 01:58:57 SELECT * FROM assignment2.customer LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
34 01:59:32 SELECT * FROM assignment2.invoice LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
35 02:00:03 SELECT * FROM assignment2.salarygrade LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec


MySQL Workbench
isys (assignment2)
unconnected
unconnected
unconnected
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
assignment2
Tables
alerts
campaign
customer
invoice
salarygrade
staff
staffongrade
workson
Views
Stored Procedures
Administration Schemas
Information
Schema: assignment2
Object Info Session
campaign customer invoice salarygrade staff
Limit to 1000 rows
1 SELECT * FROM assignment2.staff;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
STAFFNO STAFFNAME EXPERTISE
1 John NULL
3 Adam NULL
5 Eve NULL
NULL NULL NULL
Result Grid
Form Editor
staff 1
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
34 01:59:32 SELECT * FROM assignment2.invoice LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
35 02:00:03 SELECT * FROM assignment2.salarygrade LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec
36 02:00:39 SELECT * FROM assignment2.staff LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec


MySQL Workbench
isys (assignment2)
unconnected
unconnected
unconnected
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
assignment2
Tables
alerts
campaign
customer
invoice
salarygrade
staff
staffongrade
workson
Views
Stored Procedures
Administration Schemas
Information
Schema: assignment2
Object Info Session
campaign customer invoice salarygrade staff staffongrade
Limit to 1000 rows
1 SELECT * FROM assignment2.staffongrade;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
STAFFNO GRADE STARTDATE FINISHDATE
1 1 2018-01-01 2019-01-04
1 2 2019-01-05 NULL
3 2 2018-01-05 2018-12-31
3 3 2019-01-01 NULL
5 4 2018-01-01 2019-06-01
NULL NULL NULL NULL
Result Grid
Form Editor
staffongrade 1
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
35 02:00:03 SELECT * FROM assignment2.salarygrade LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec
36 02:00:39 SELECT * FROM assignment2.staff LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
37 02:01:22 SELECT * FROM assignment2.staffongrade LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec


MySQL Workbench
isys (assignment2)
unconnected
unconnected
unconnected
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
assignment2
Tables
alerts
campaign
customer
invoice
salarygrade
staff
staffongrade
workson
Views
Stored Procedures
Administration Schemas
Information
Schema: assignment2
Object Info Session
campaign customer invoice salarygrade staff staffongrade workson
Limit to 1000 rows
1 SELECT * FROM assignment2.workson;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
STAFFNO CAMPAIGN_NO WDATE HOUR
1 2 2019-02-01 8
1 2 2019-05-01 7
3 1 2018-05-01 5
5 1 2018-02-01 7
5 1 2019-06-01 8
NULL NULL NULL NULL
Result Grid
Form Editor
workson 1
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
36 02:00:39 SELECT * FROM assignment2.staff LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
37 02:01:22 SELECT * FROM assignment2.staffongrade LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec
38 02:01:57 SELECT * FROM assignment2.workson LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec

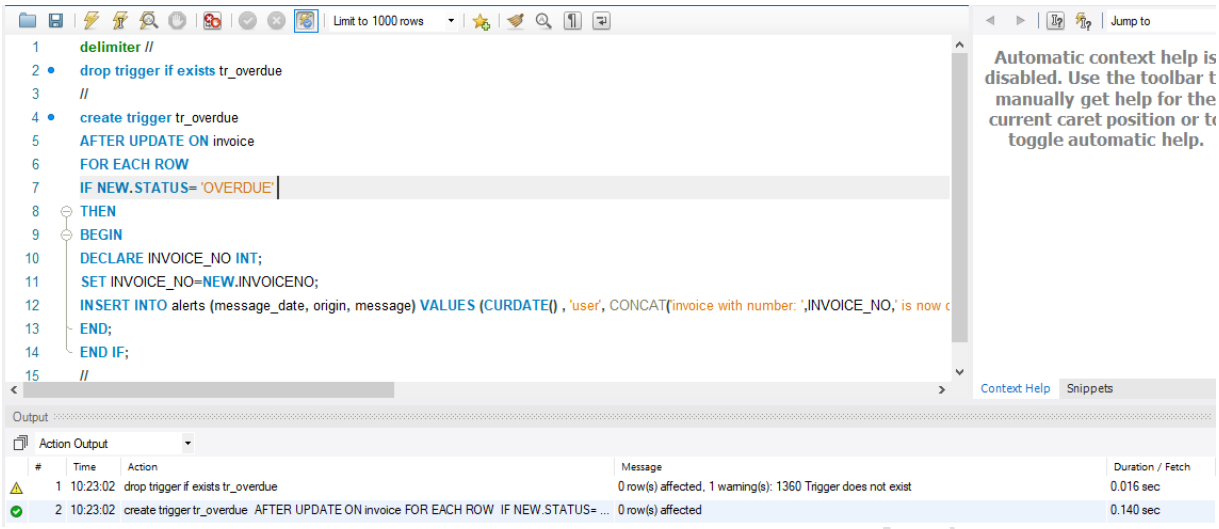


2. Stored Programs.

*/********* Trigger TR_OVERDUE **********/*

delimiter //
drop trigger if exists tr_overdue
//
create trigger tr_overdue
AFTER UPDATE ON invoice
FOR EACH ROW
IF NEW.STATUS= 'OVERDUE'
THEN
BEGIN
DECLARE INVOICE_NO INT;
SET INVOICE_NO=NEW.INVOICENO;
INSERT INTO alerts (message_date, origin, message) VALUES (CURDATE() , 'user', CONCAT('invoice with number: ',INVOICE_NO,' is now overdue!'));
END;
END IF;
//

www.ExpertLi



*/********** Helper Functions/Procedures used, two functions for example **********/*

delimiter //

drop function if exists rate_on_date //

create function rate_on_date(staff_id int, given_date DATE) returns float

-- retuens the hourly salary rate of a given staff (staff_id) on any particular date (given_date)

READS SQL DATA

DETERMINISTIC

begin

DECLARE hour_rate FLOAT;

SET hour_rate = 0;

SELECT HOURLYRATE INTO hour_rate FROM SALARYGRADE WHERE GRADE=(SELECT GRADE FROM STAFFONGRADE WHERE STAFFNO=staff_id AND given_date>=STARTDATE AND given_date<=FINISHDATE);

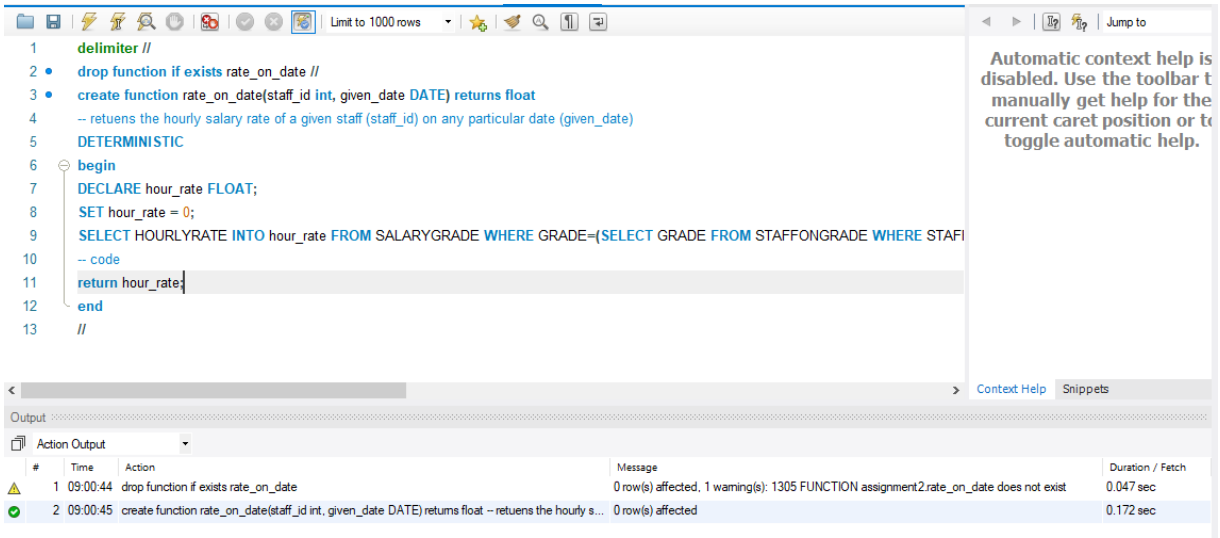
-- code

return hour_rate;

end

//

www.Expertus.in



delimiter //

drop function if exists cost_of_campaign ;

//

create function cost_of_campaign (camp_id int) returns float

-- retuens the total cost incurred due to any given campaigh (camp_id)

READS SQL DATA

DETERMINISTIC

begin

DECLARE total_cost FLOAT;

SET total_cost = 0;

*SELECT SUM(rate_on_date(w.STAFFNO,w.WDATE)*w.HOUR) INTO total_cost FROM workson w WHERE w.CAMPAIGN_NO=camp_id;*

-- code

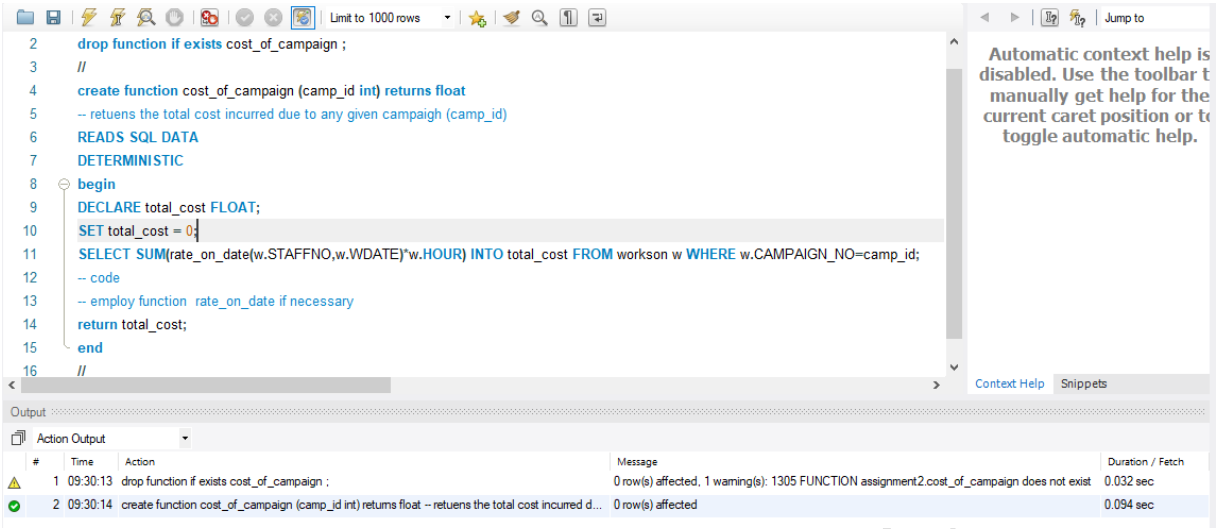
-- employ function rate_on_date if necessary

return total_cost;

end

//

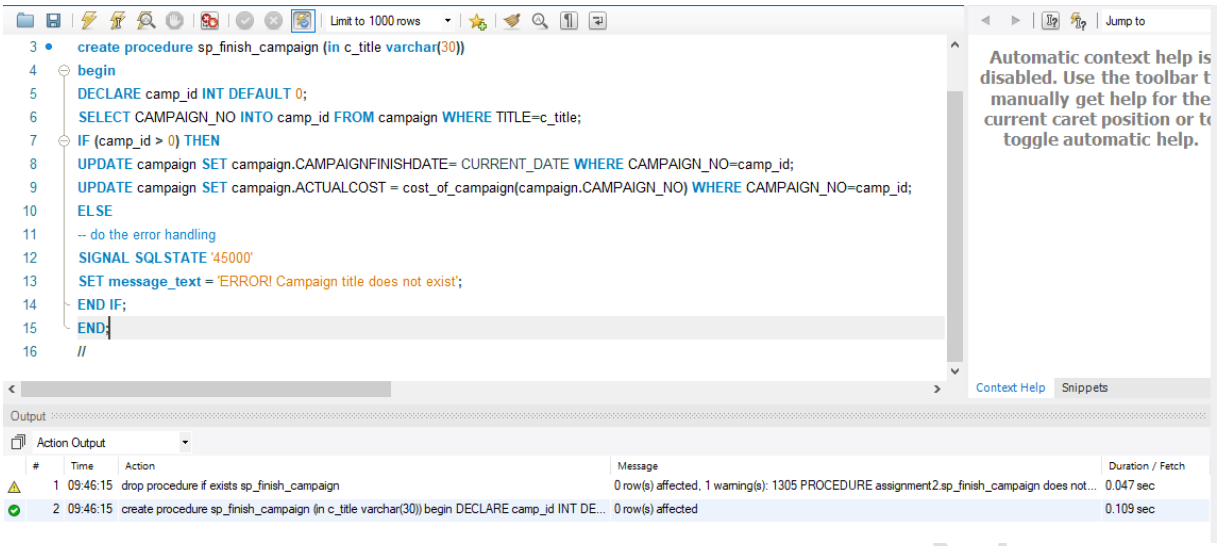
www.ExpertLin.c



*/********** Procedure SP_FINISH_CAMPAIGN**********/*

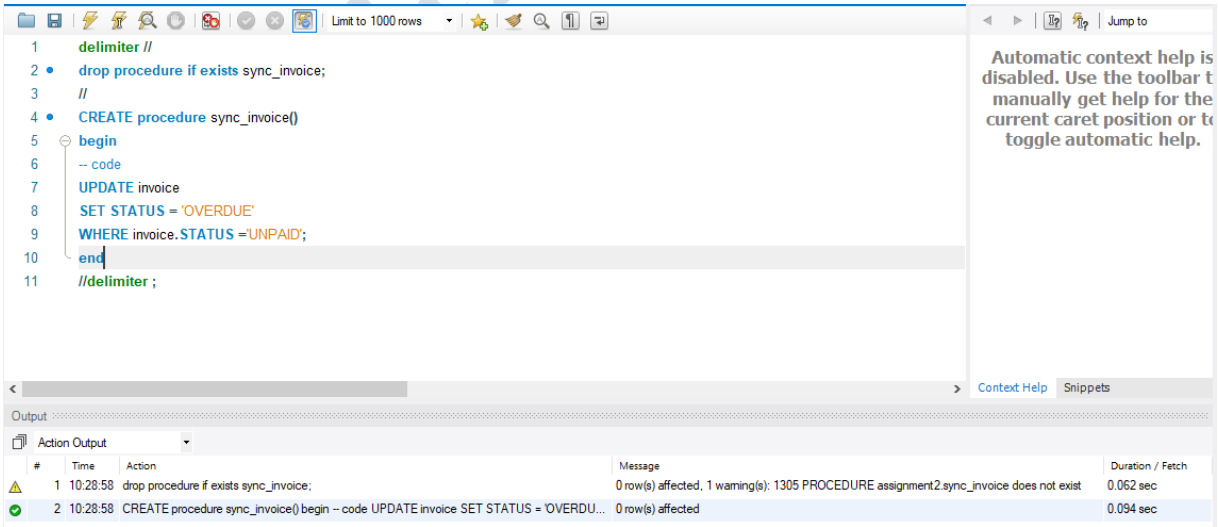
delimiter //
drop procedure if exists sp_finish_campaign //
create procedure sp_finish_campaign (in c_title varchar(30))
begin
DECLARE camp_id INT DEFAULT 0;
SELECT CAMPAIGN_NO INTO camp_id FROM campaign WHERE TITLE=c_title;
IF (camp_id > 0) THEN
UPDATE campaign SET campaign.CAMPAIGNFINISHDATE= CURRENT_DATE WHERE CAMPAIGN_NO=camp_id;
UPDATE campaign SET campaign.ACTUALCOST = cost_of_campaign(campaign.CAMPAIGN_NO) WHERE CAMPAIGN_NO=camp_id;
ELSE
-- do the error handling
SIGNAL SQLSTATE '45000'
SET message_text = 'ERROR! Campaign title does not exist';
END IF;
END;
//

www.ExpertLin.c



*/********** Procedure SYNC_INVOICE**********/*

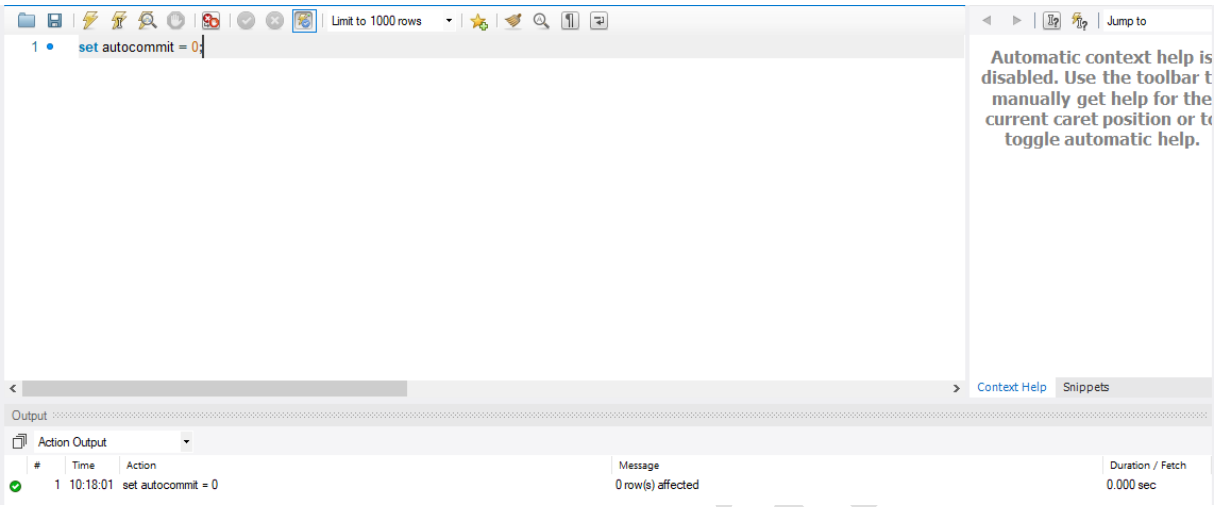
```
delimiter //
drop procedure if exists sync_invoice;
//
CREATE procedure sync_invoice()
begin
-- code
UPDATE invoice
SET STATUS = 'OVERDUE'
WHERE invoice.STATUS ='UNPAID';
end
//delimiter ;
```



3. Required Testing against Sample Database.

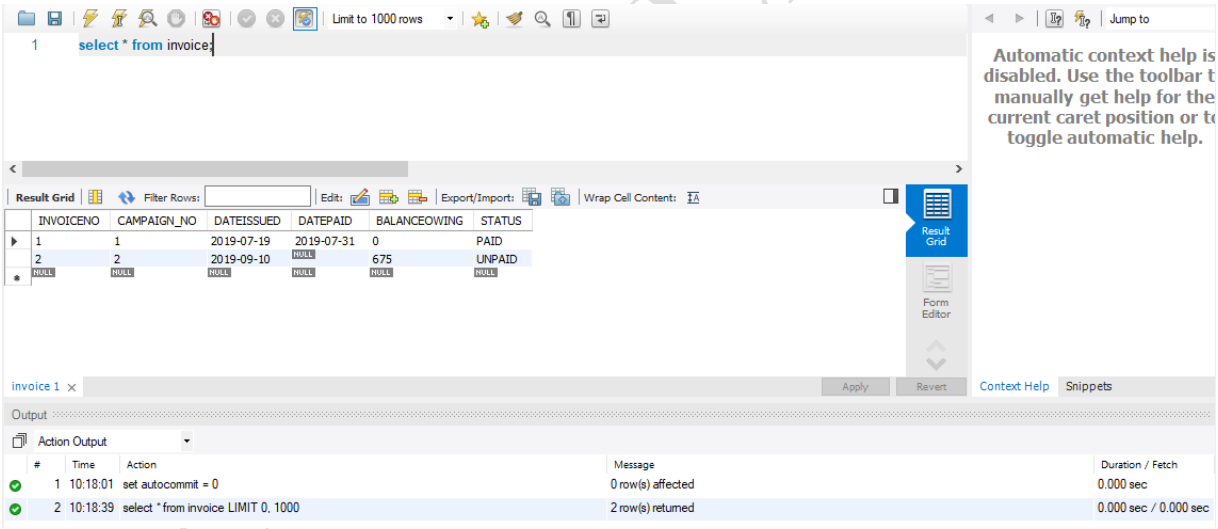
Paste into this section the initial tests you ran (one by one), each followed by the corresponding results as screenshots.

set autocommit = 0;

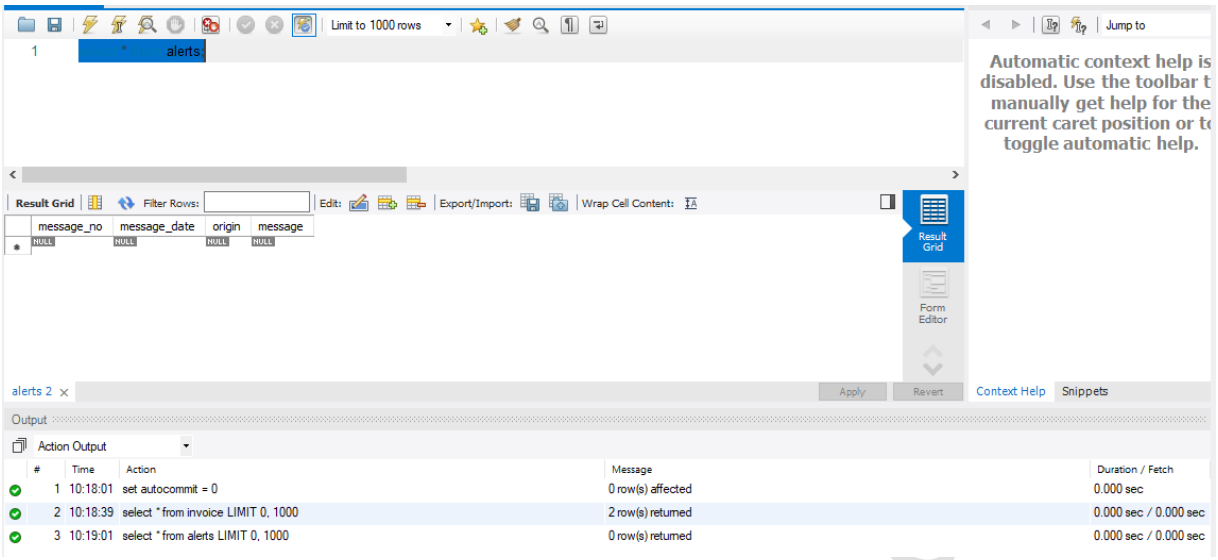


-- Inspect the invoice and the alerts table

*select * from invoice;*

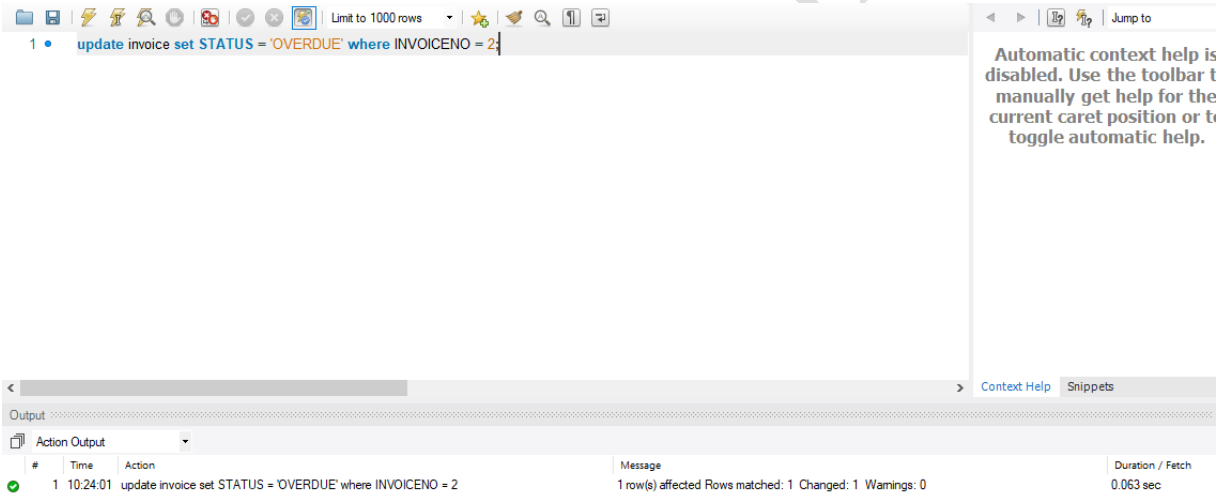


*select * from alerts;*



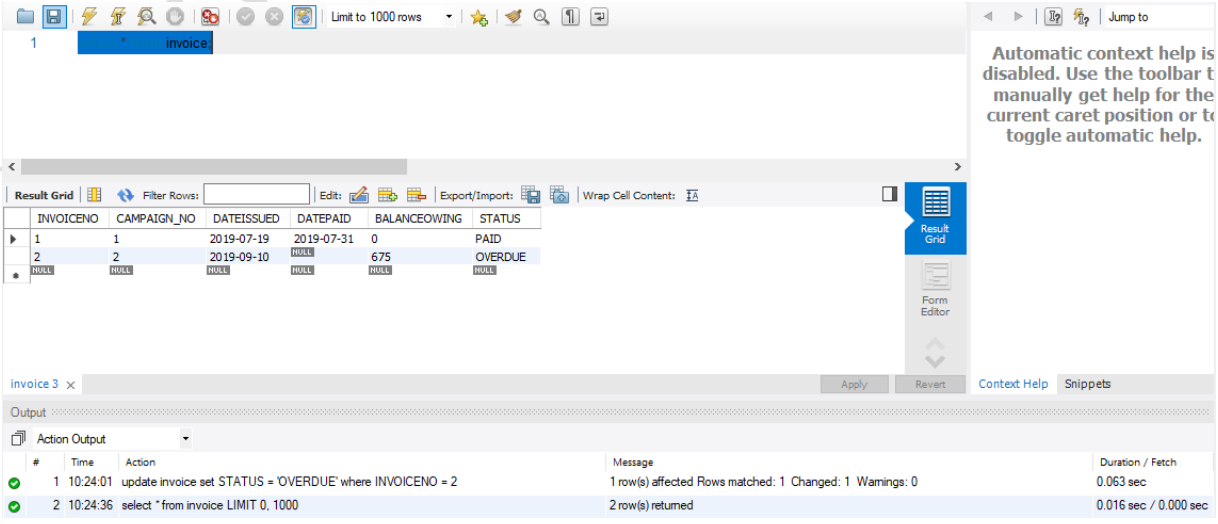
--Update the invoice table

update invoice set STATUS = 'OVERDUE' where INVOICENO = 2;

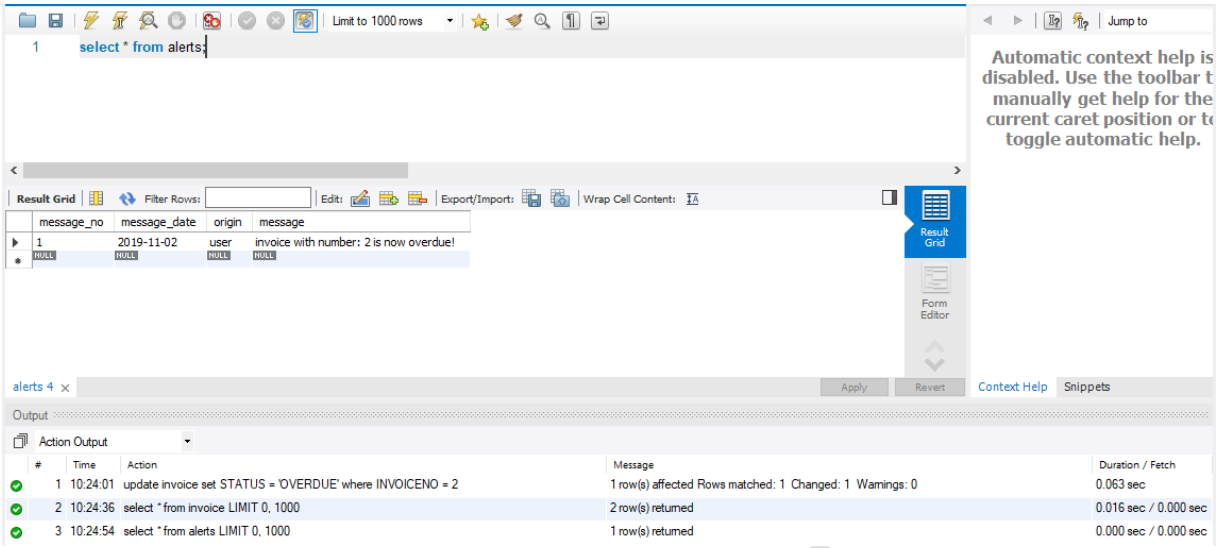


-- Verify that the trigger you implemented works

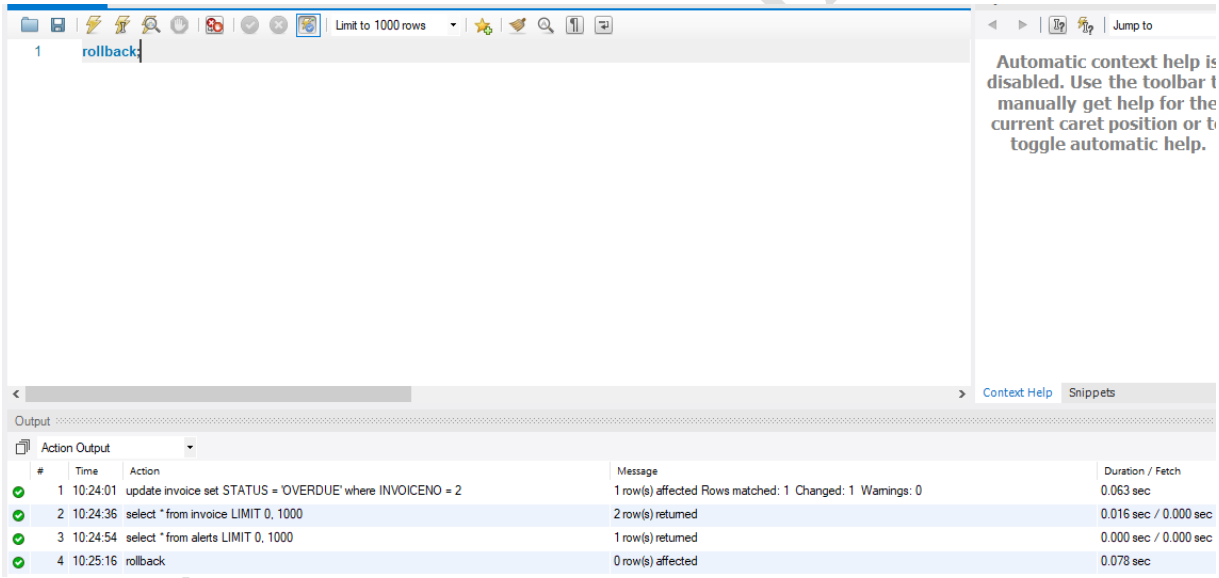
*select * from invoice;*



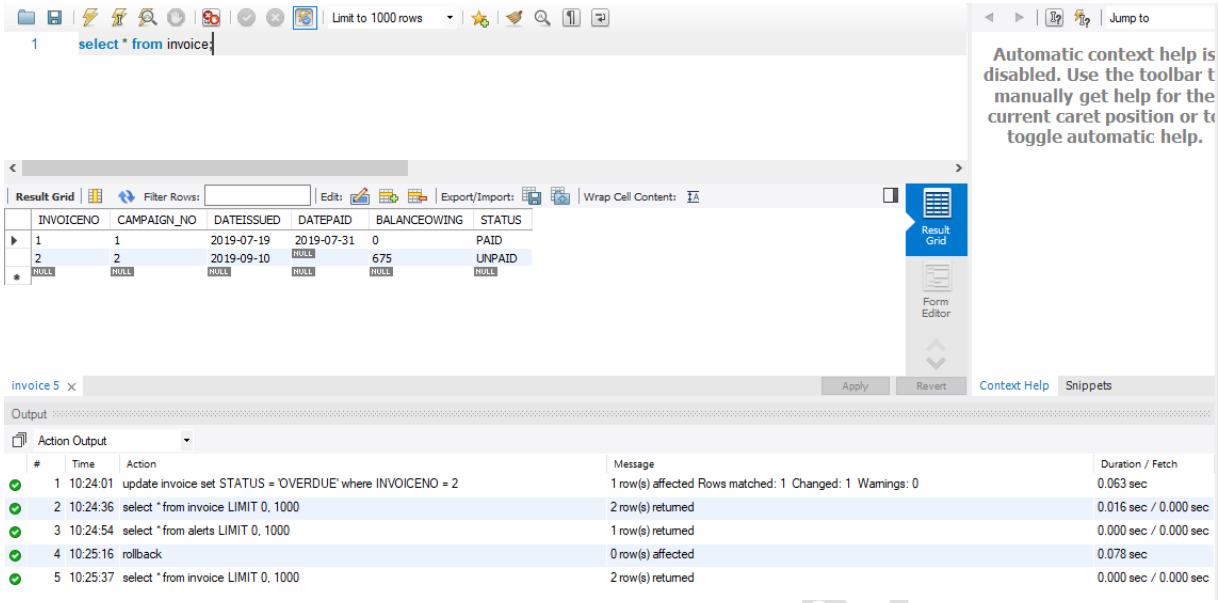
*select * from alerts;*



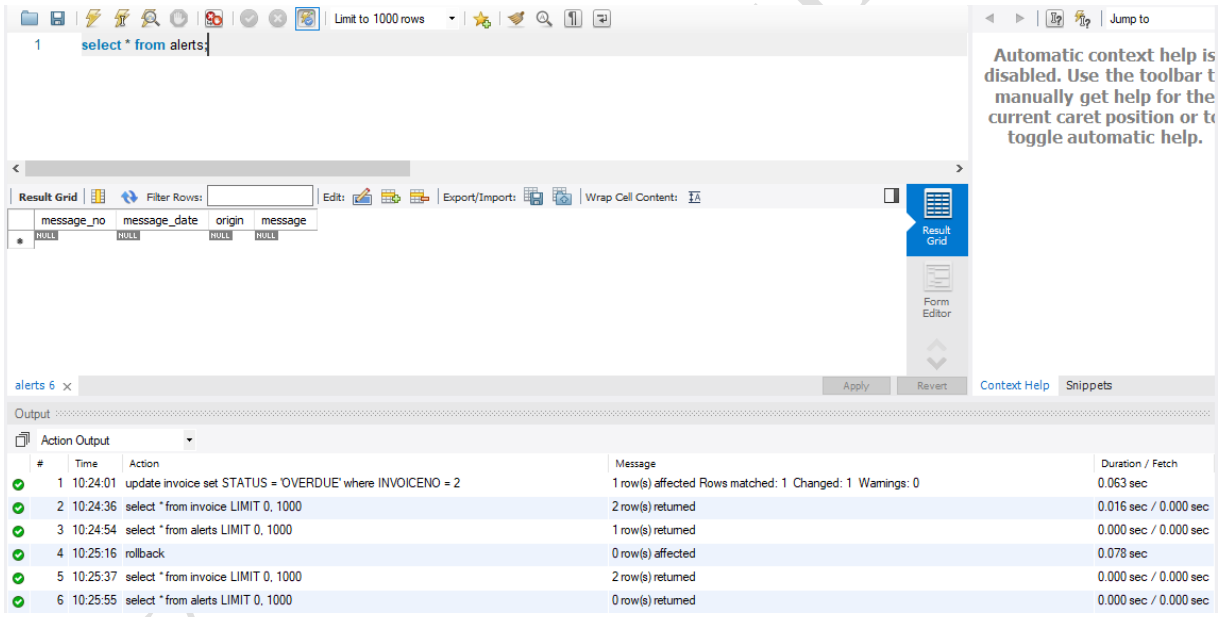
-- Bring DB back to original state; re-check rollback;



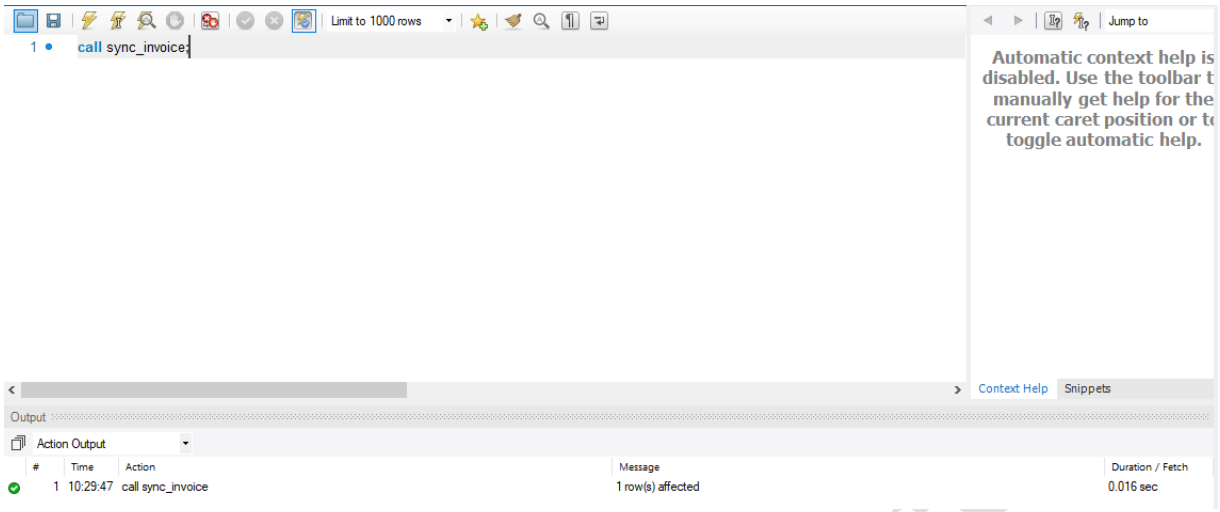
*select * from invoice;*



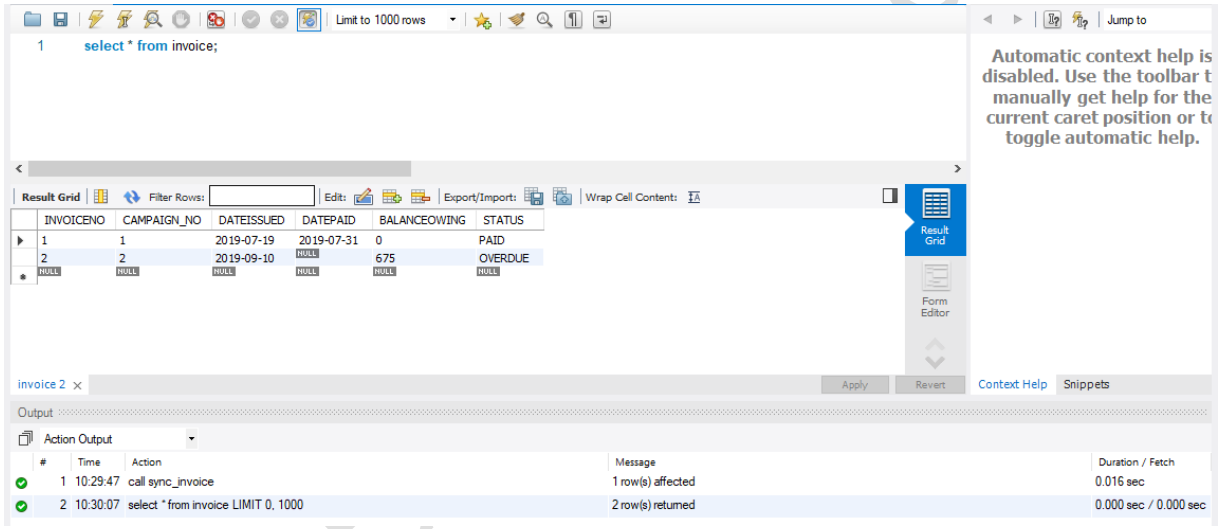
*select * from alerts;*



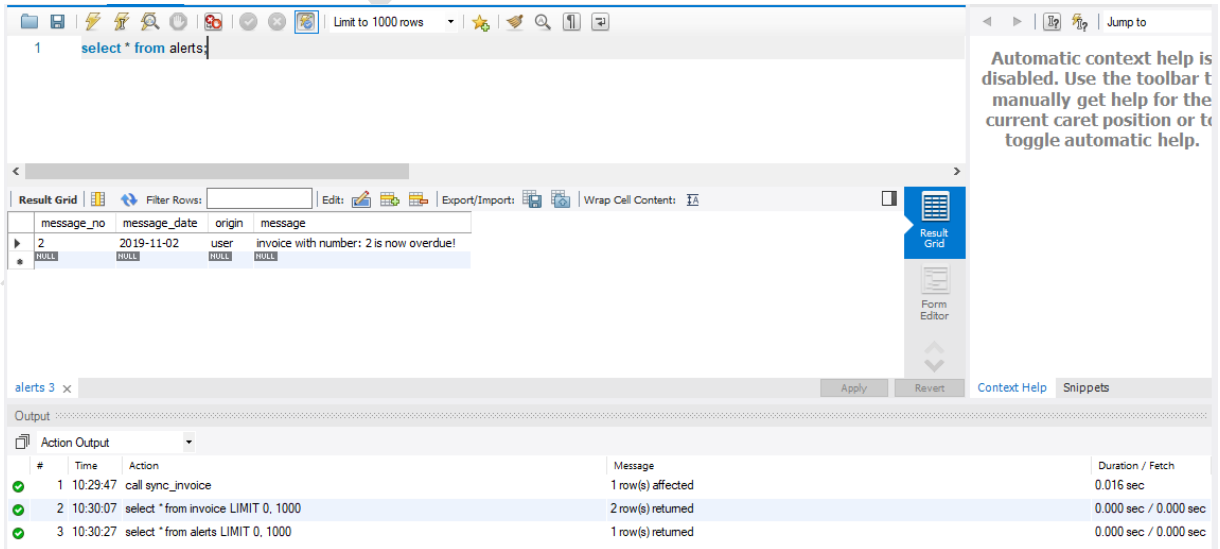
-- Synchronise the invoice table and verify the procedure behaves as desired
call sync_invoice;



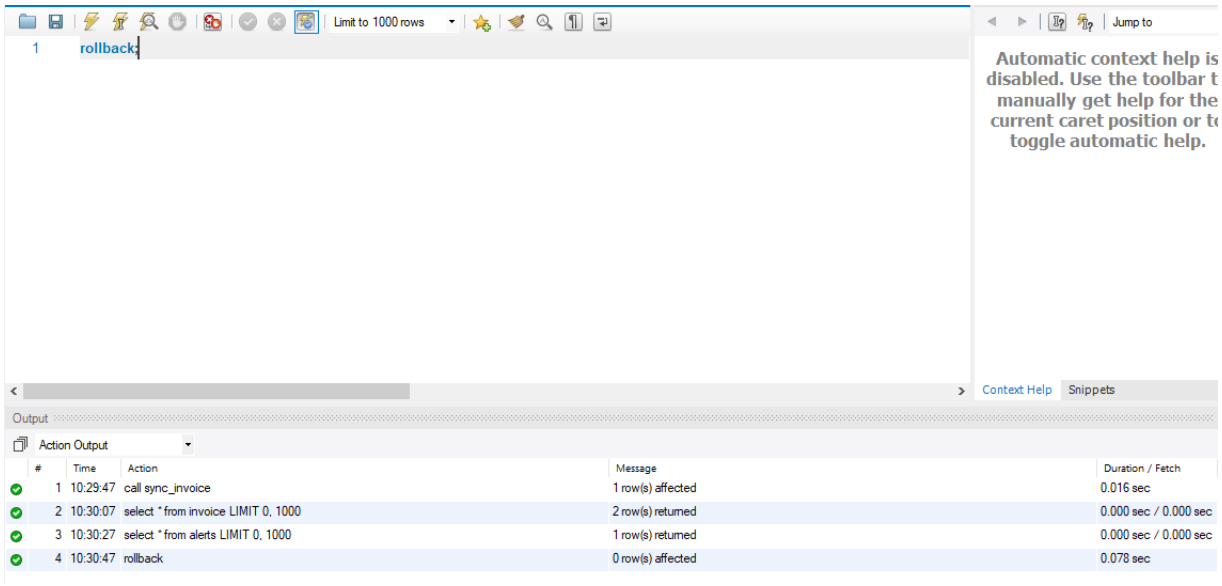
*select * from invoice;*



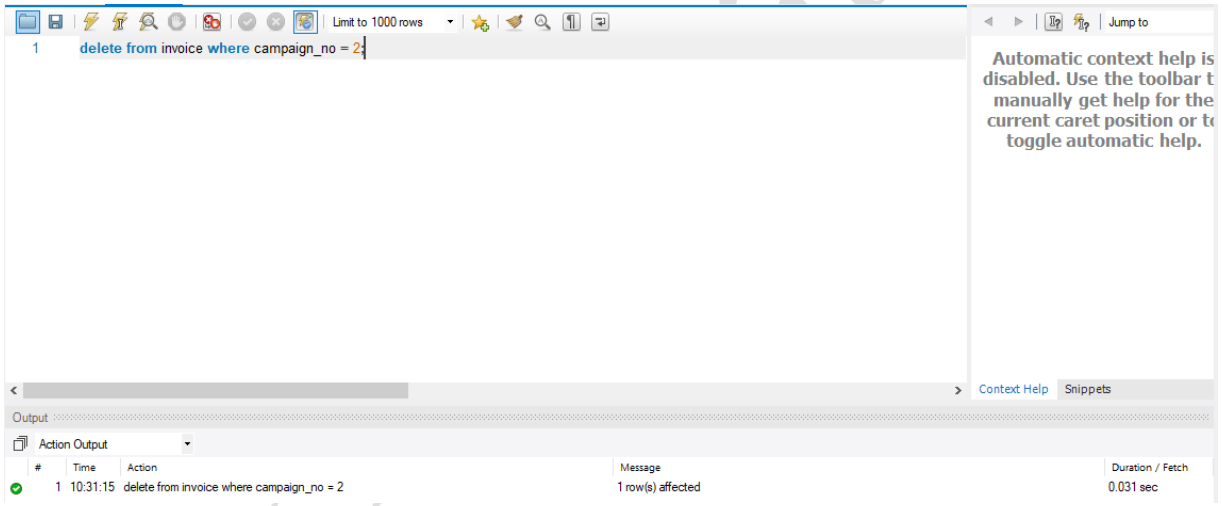
*select * from alerts;*



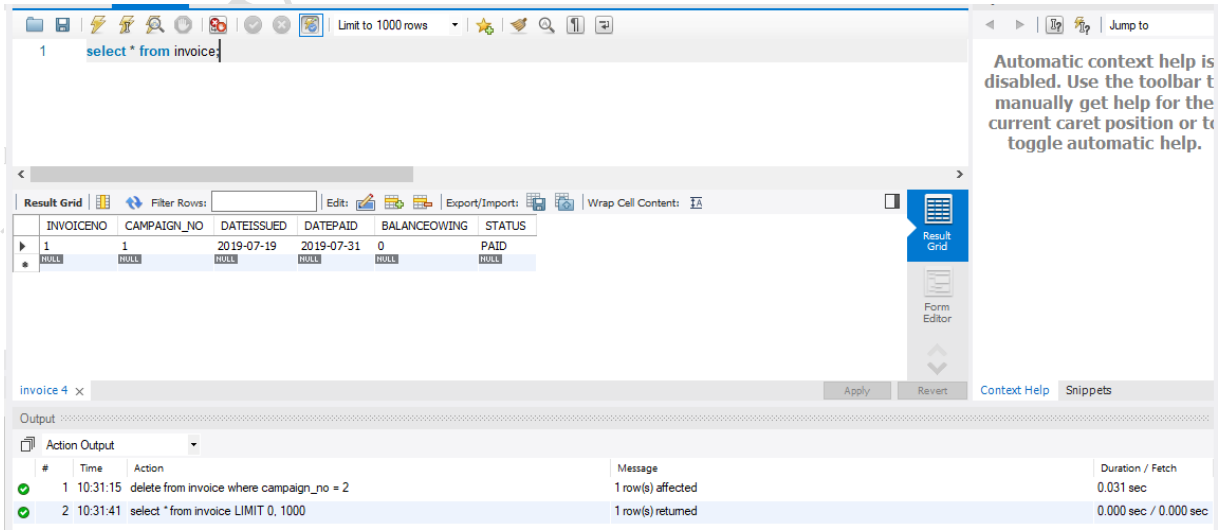
-- Bring DB back to original state; delete campaign# 2; check all relevant tables rollback;



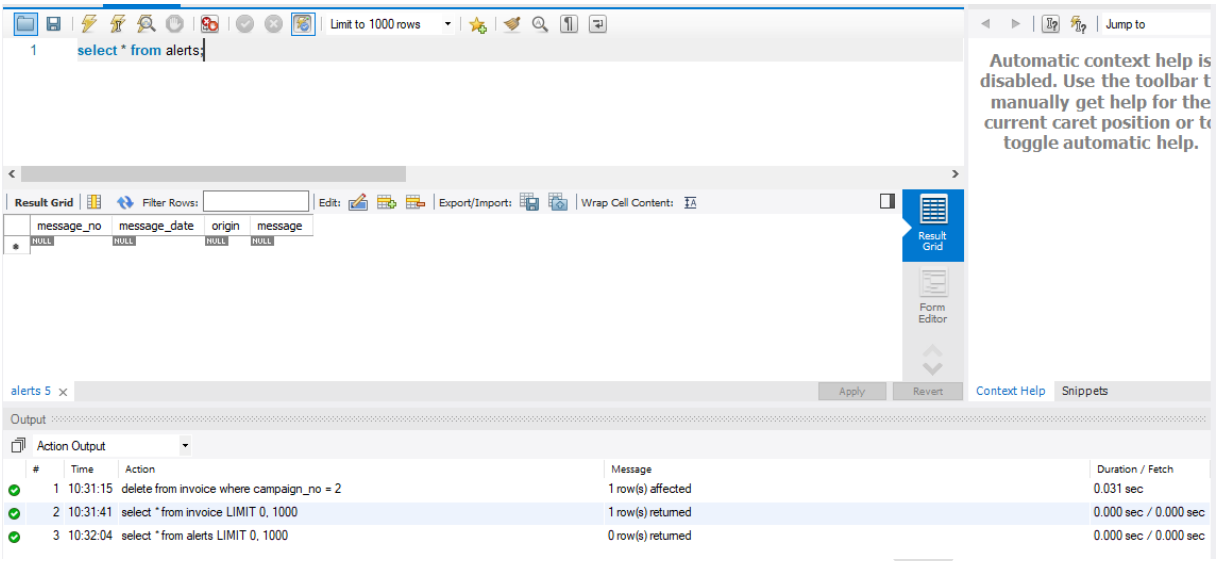
delete from invoice where campaign_no = 2;



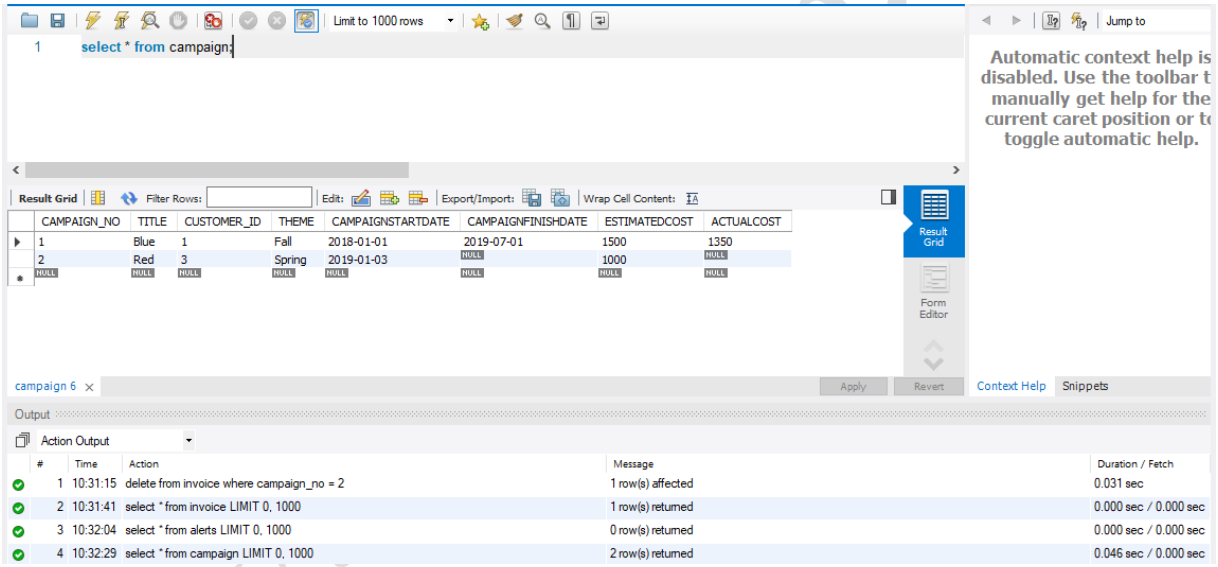
*select * from invoice;*



*select * from alerts;*



*select * from campaign;*



*select * from salarygrade;*

```
1  select * from salarygrade;
```

GRADE	HOURLYRATE
1	30
2	45
3	70
4	75
5	150
NULL	NULL

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:31:15	delete from invoice where campaign_no = 2	1 row(s) affected	0.031 sec
2	10:31:41	select * from invoice LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	10:32:04	select * from alerts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
4	10:32:29	select * from campaign LIMIT 0, 1000	2 row(s) returned	0.046 sec / 0.000 sec
5	10:32:53	select * from salarygrade LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec

*select * from staff;*

```
1  select * from staff;
```

STAFFNO	STAFFNAME	EXPERTISE
1	John	NULL
3	Adam	NULL
5	Eve	NULL
NULL	NULL	NULL

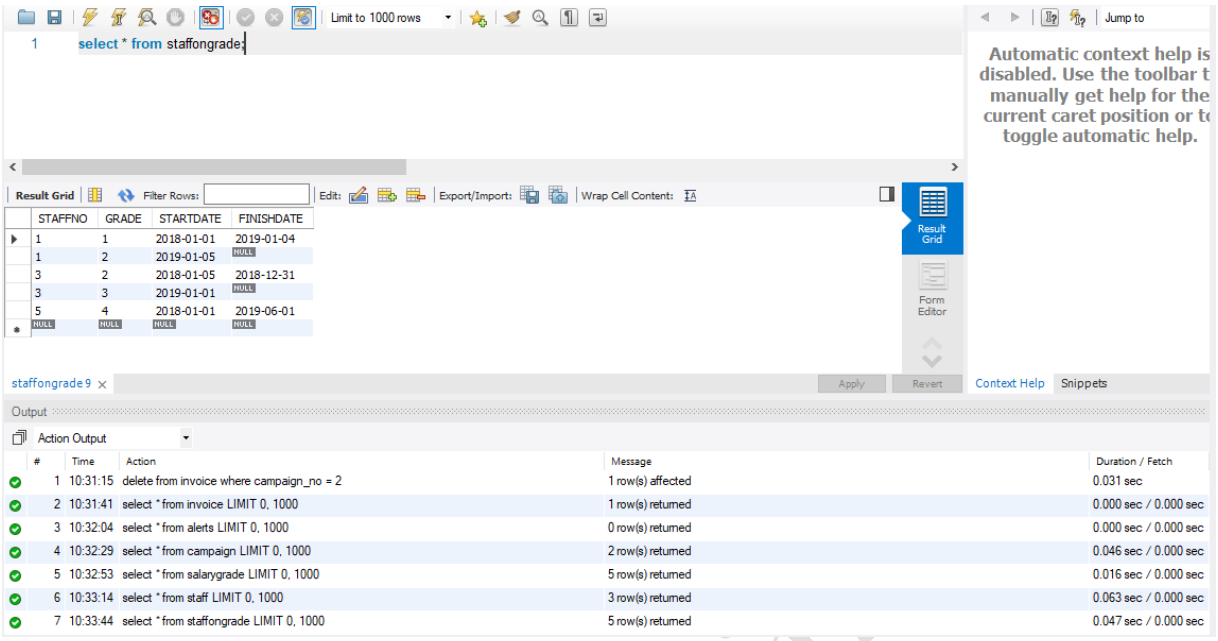
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

Output

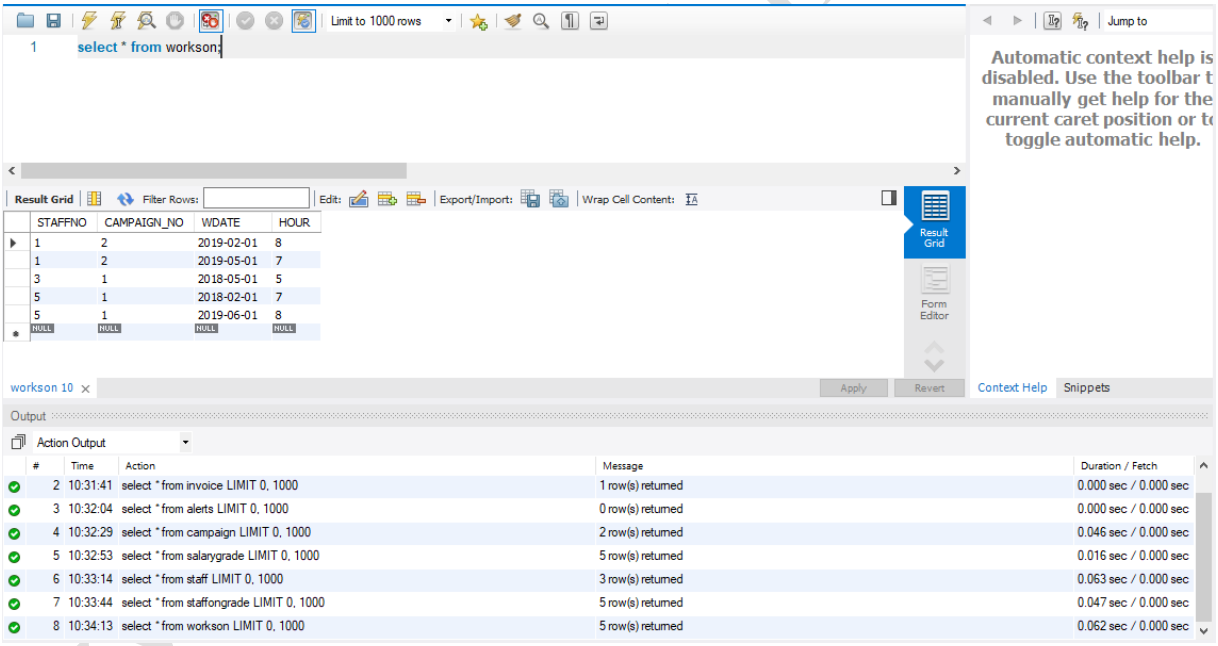
Action Output

#	Time	Action	Message	Duration / Fetch
1	10:31:15	delete from invoice where campaign_no = 2	1 row(s) affected	0.031 sec
2	10:31:41	select * from invoice LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	10:32:04	select * from alerts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
4	10:32:29	select * from campaign LIMIT 0, 1000	2 row(s) returned	0.046 sec / 0.000 sec
5	10:32:53	select * from salarygrade LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
6	10:33:14	select * from staff LIMIT 0, 1000	3 row(s) returned	0.063 sec / 0.000 sec

*select * from staffongrade;*

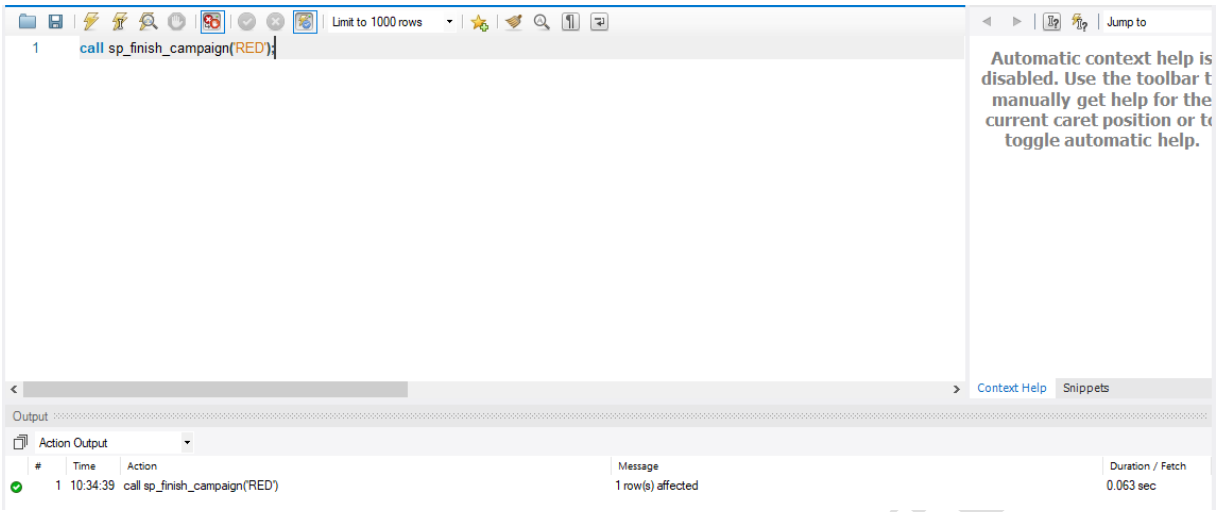


*select * from workson;*

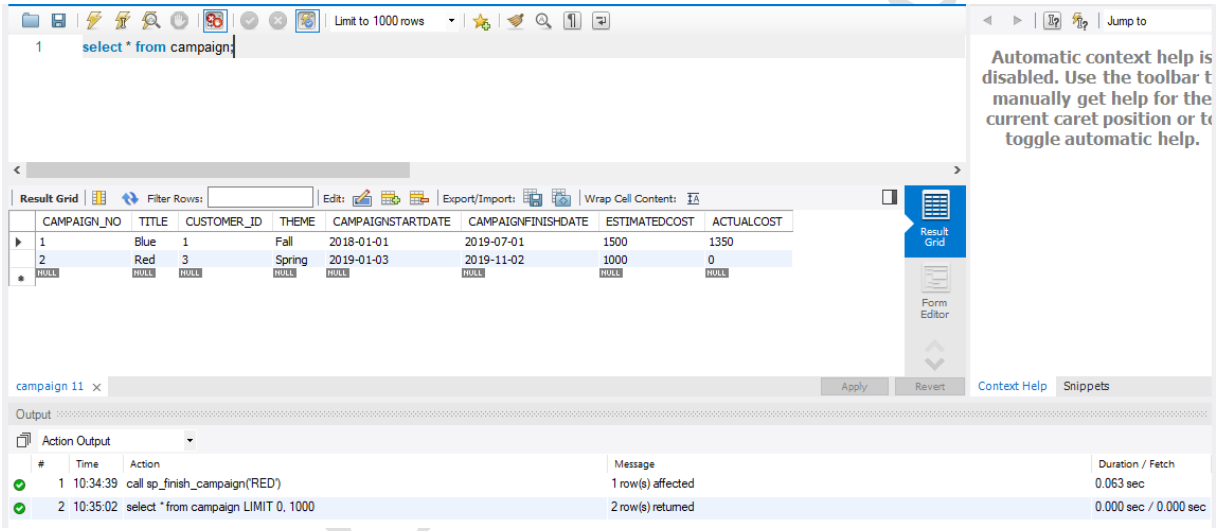


-- Finish the campaign titled RED. Verify that it behaves as desired.

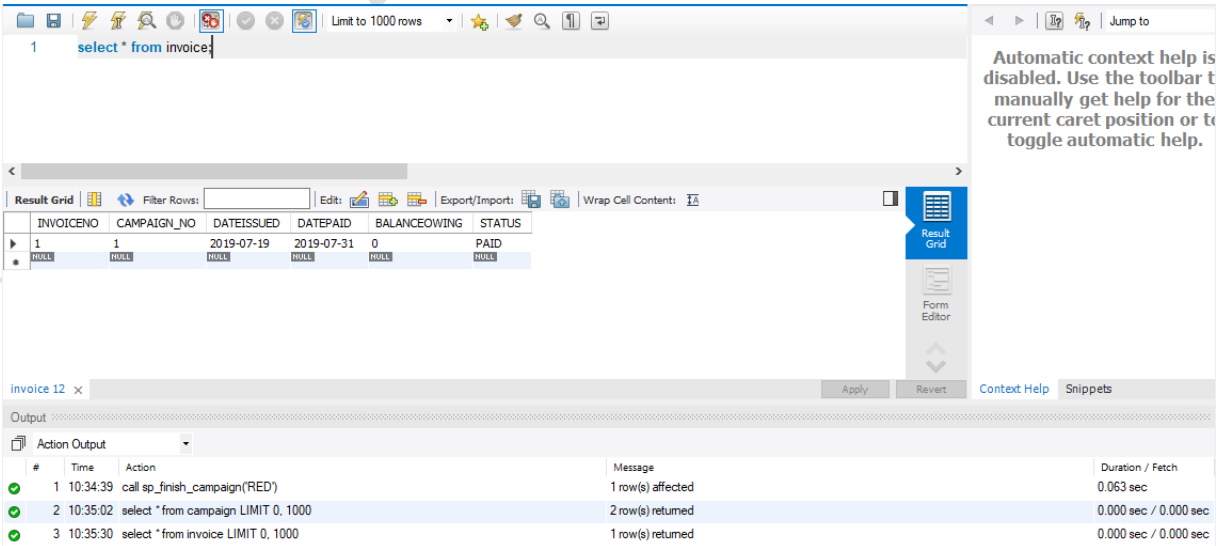
call sp_finish_campaign('RED');



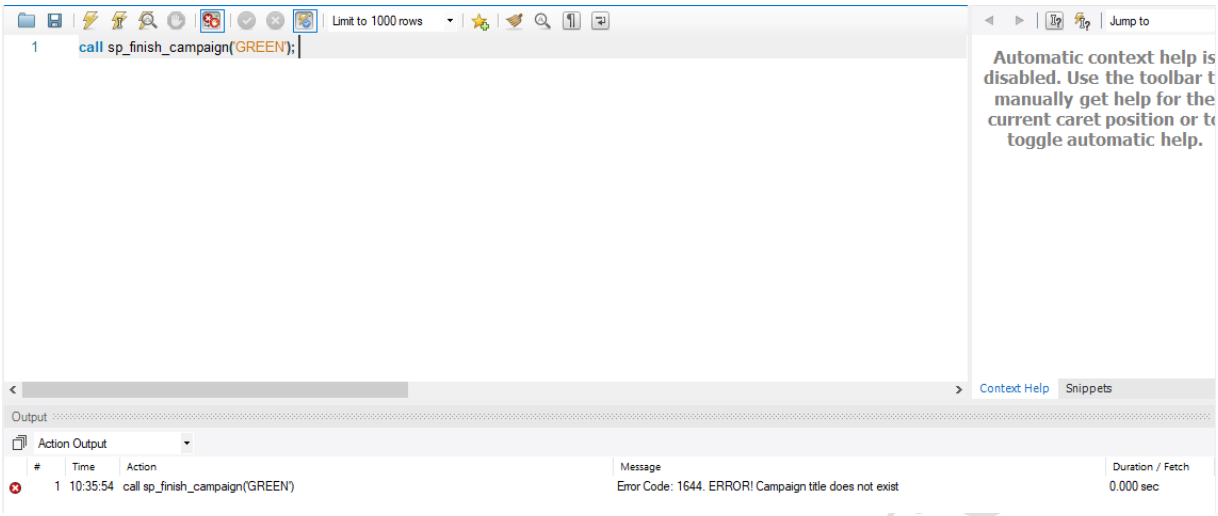
*select * from campaign;*



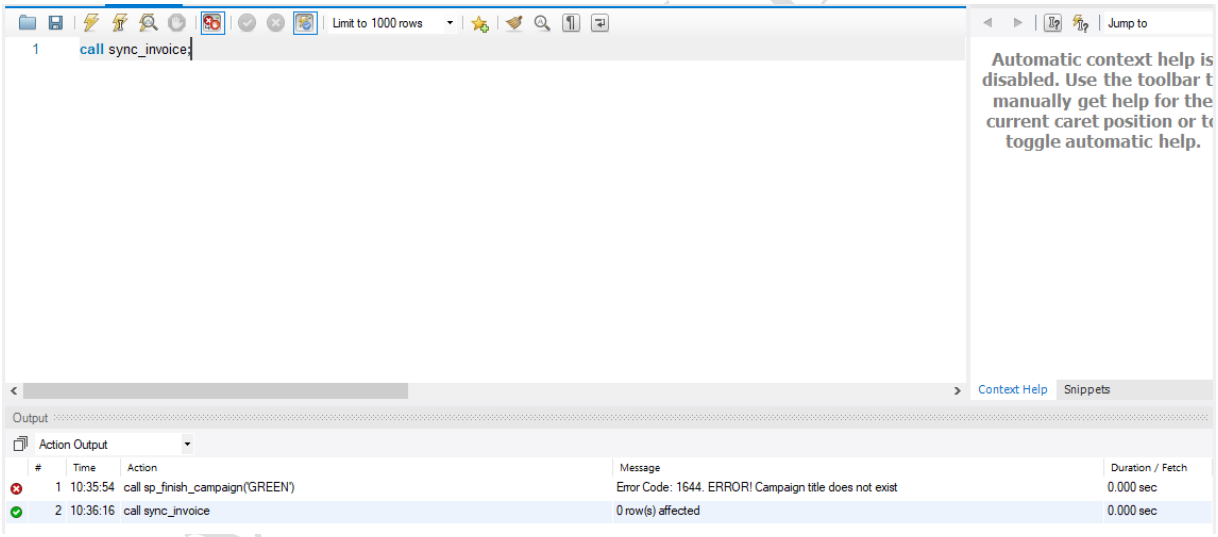
*select * from invoice;*



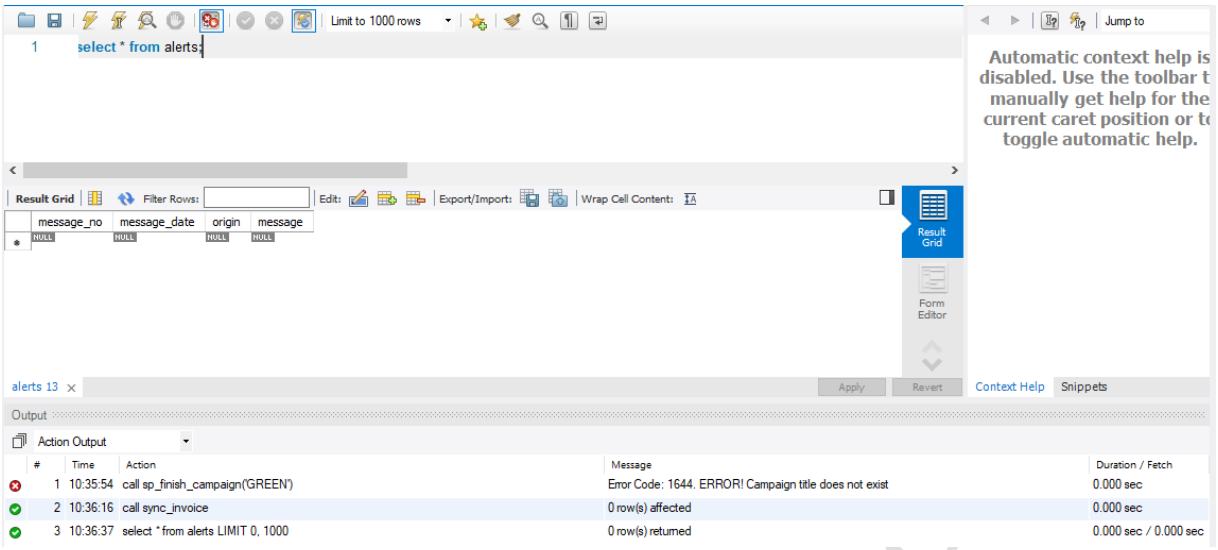
call sp_finish_campaign('GREEN'); -- should SIGNAL error condition



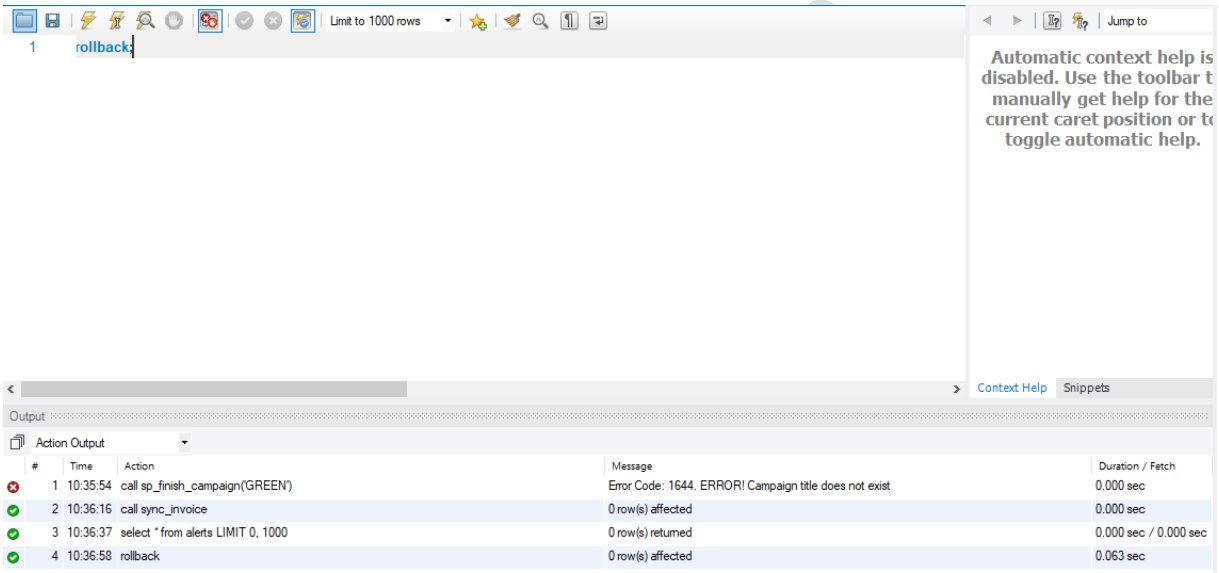
-- Synchronise the invoice table and verify the procedure behaves as desired
call sync_invoice;



*select * from alerts;*



rollback;

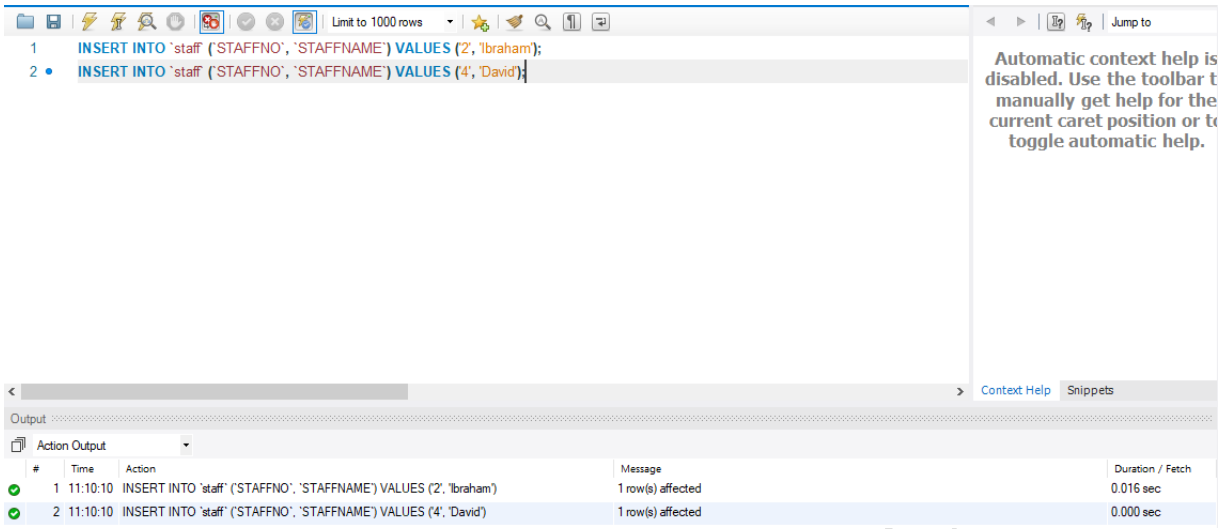


4. More Extensive Testing.

Explain what changes you made to which tables, what tests you ran, and why. Copy and paste from your file `<yourLastname_yourFirstname>_test_script.sql` *the DML statements you used for this purpose, followed by the screenshots of the records in those tables. Then copy and paste the procedure calls you made, and the screenshots of the records in the relevant tables (or the error messages).*

STEP 1:

I have inserted two new staff records in staff table using below insert statements:
INSERT INTO `staff` (`STAFFNO`, `STAFFNAME`) VALUES ('2', 'Ibraham');
INSERT INTO `staff` (`STAFFNO`, `STAFFNAME`) VALUES ('4', 'David');



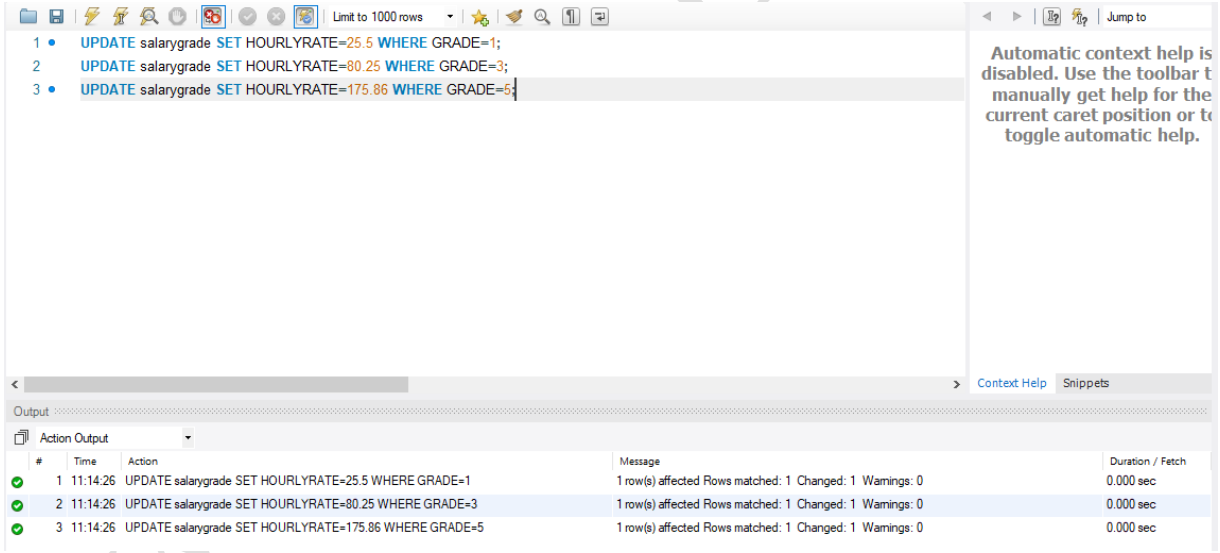
STEP 2:

Update the Salary Grades using below commands:

UPDATE salarygrade SET HOURLYRATE=25.5 WHERE GRADE=1;

UPDATE salarygrade SET HOURLYRATE=80.25 WHERE GRADE=3;

UPDATE salarygrade SET HOURLYRATE=175.86 WHERE GRADE=5;



STEP 3:

Insert grade for employee for newly staff members:

INSERT INTO staffongrade(STAFFNO, GRADE, STARTDATE, FINISHDATE) VALUES ('2', '1', '2015-01-01','2017-12-31');

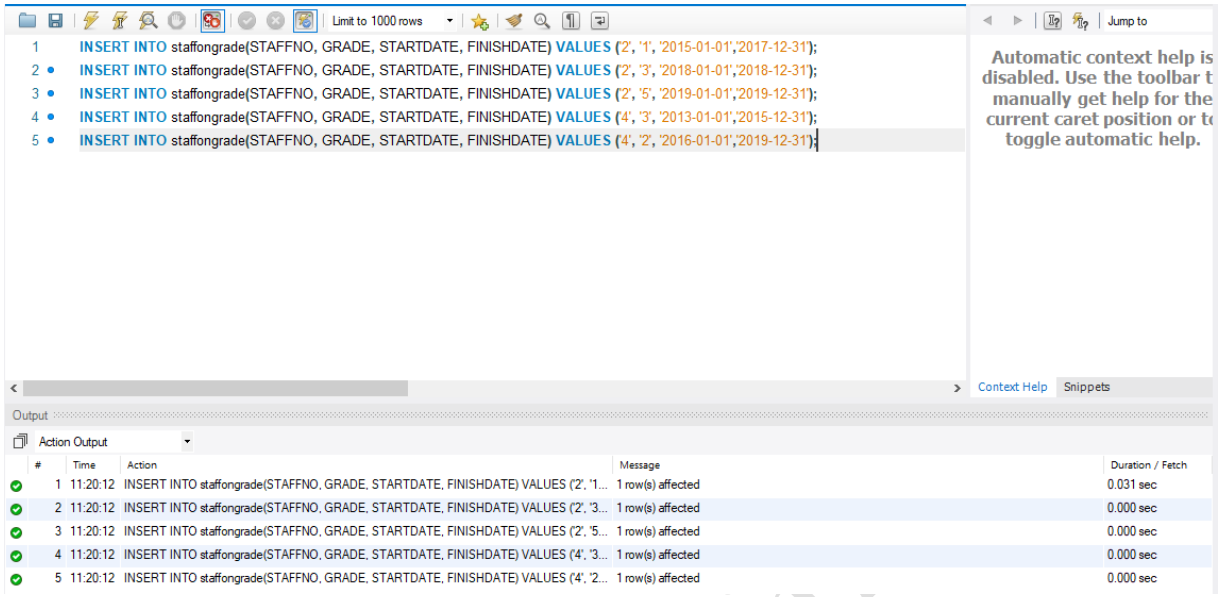
INSERT INTO staffongrade(STAFFNO, GRADE, STARTDATE, FINISHDATE) VALUES ('2', '3', '2018-01-01','2018-12-31');

INSERT INTO staffongrade(STAFFNO, GRADE, STARTDATE, FINISHDATE) VALUES ('2', '5', '2019-01-01','2019-12-31');

INSERT INTO staffongrade(STAFFNO, GRADE, STARTDATE, FINISHDATE) VALUES ('4', '3', '2013-01-01','2015-12-31');

INSERT INTO staffongrade(STAFFNO, GRADE, STARTDATE, FINISHDATE) VALUES

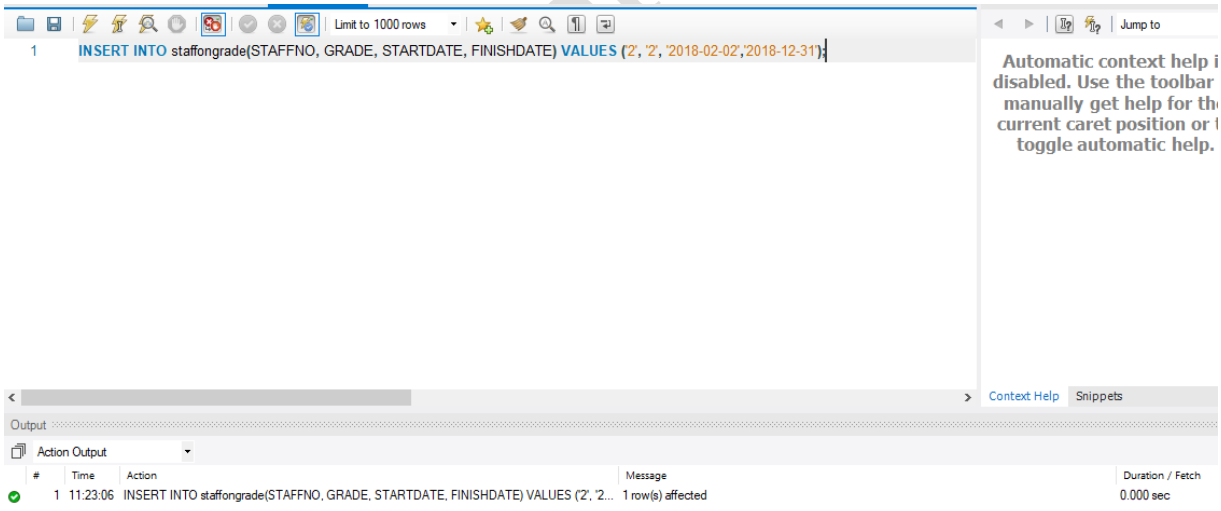
('4', '2', '2016-01-01','2019-12-31');



STEP 4:

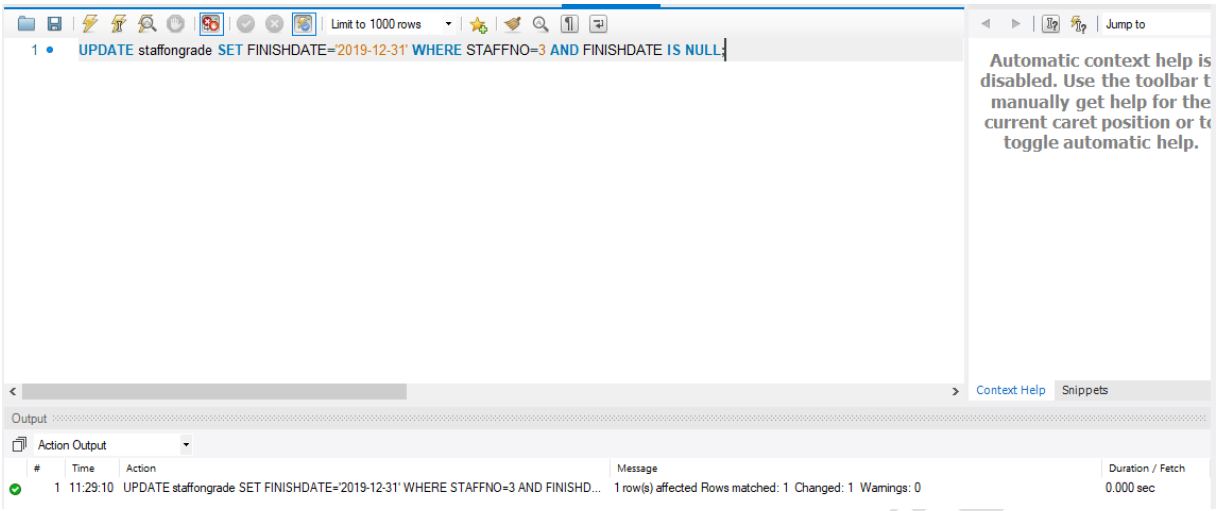
INSERT MULTIPLE GRADE FOR A STAFF for Different grade FOR SAME PERIOD to CHECK either our procedures/functions work or not.

INSERT INTO staffongrade(STAFFNO, GRADE, STARTDATE, FINISHDATE) VALUES ('2', '2', '2018-02-02','2018-12-31');

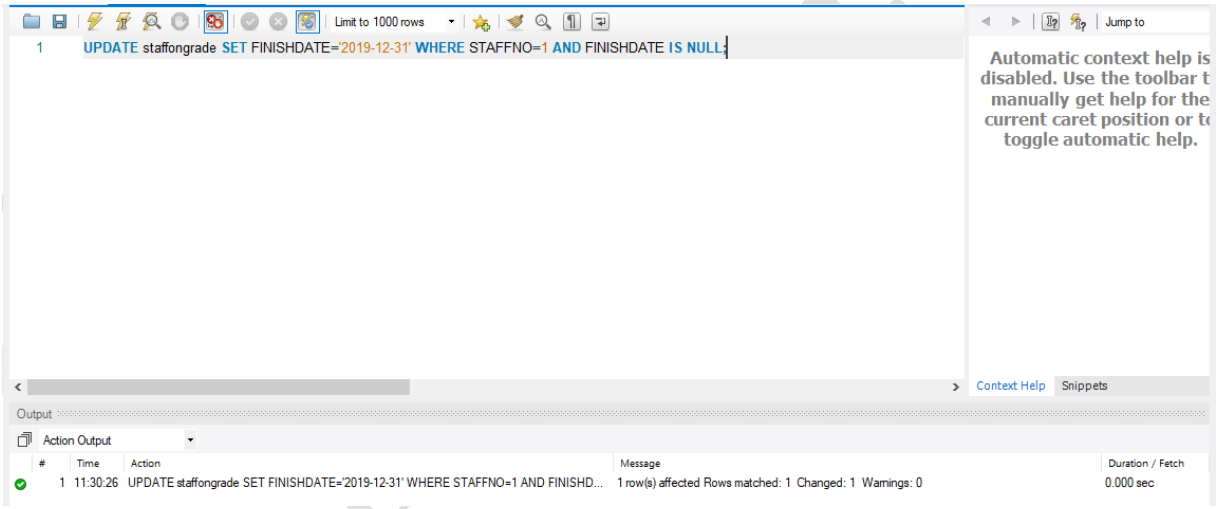


UPDATE those *staffongrade* records WHERE FINISHDATE is NULL.

UPDATE staffongrade SET FINISHDATE='2019-12-31' WHERE STAFFNO=3 AND FINISHDATE IS NULL;

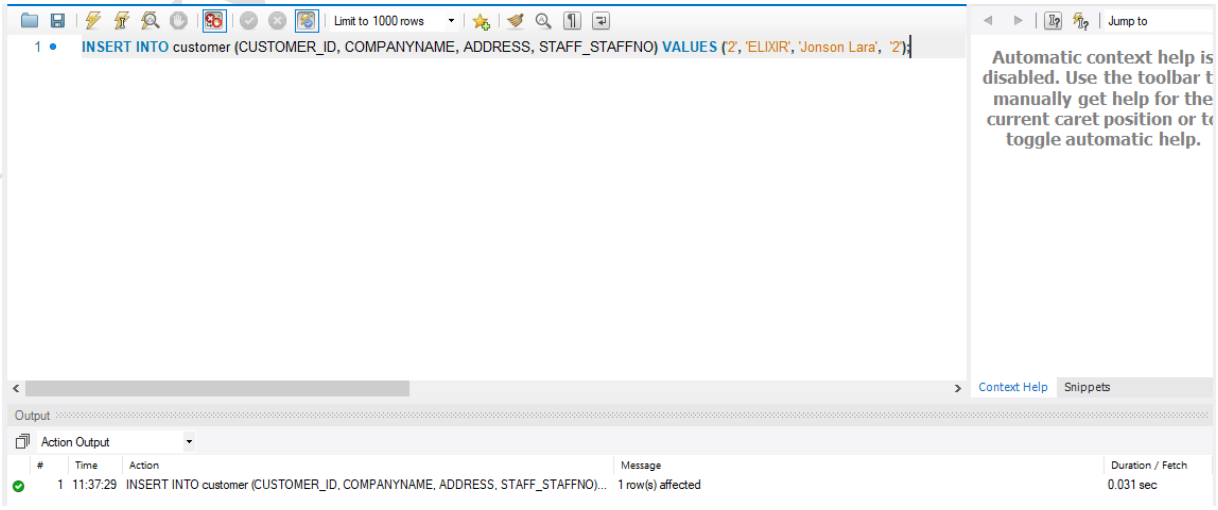


UPDATE staffongrade SET FINISHDATE='2019-12-31' WHERE STAFFNO=1 AND FINISHDATE IS NULL;



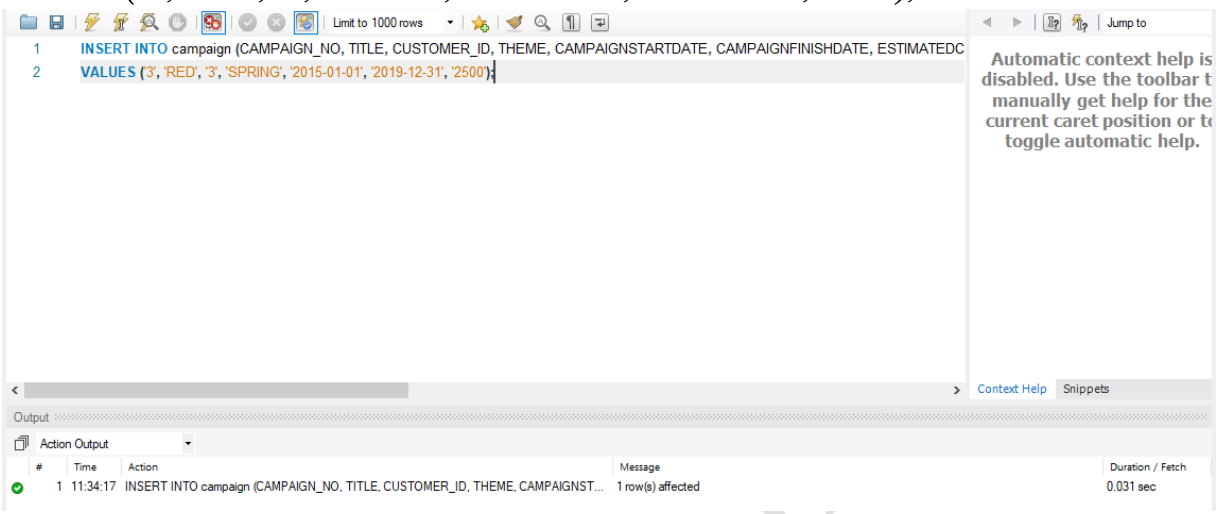
ADDING A NEW CUSTOMER

INSERT INTO customer (CUSTOMER_ID, COMPANYNAME, ADDRESS, STAFF_STAFFNO) VALUES ('2', 'ELIXIR', 'Jonson Lara', '2');

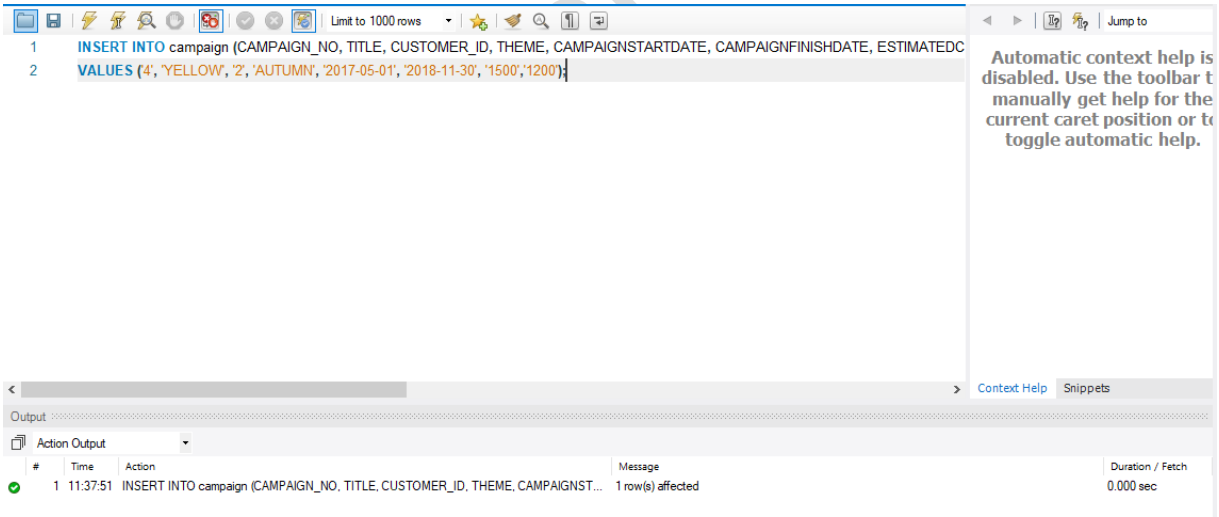


ADDING TWO MORE COMPAING

INSERT INTO campaign (CAMPAIGN_NO, TITLE, CUSTOMER_ID, THEME, CAMPAIGNSTARTDATE, CAMPAIGNFINISHDATE, ESTIMATEDCOST)
VALUES ('3', 'RED', '3', 'SPRING', '2015-01-01', '2019-12-31', '2500');



INSERT INTO campaign (CAMPAIGN_NO, TITLE, CUSTOMER_ID, THEME, CAMPAIGNSTARTDATE, CAMPAIGNFINISHDATE, ESTIMATEDCOST,ACTUALCOST)
VALUES ('4', 'YELLOW', '2', 'AUTUMN', '2017-05-01', '2018-11-30', '1500','1200');



INSERT data to workson table for new as well as old compaings.

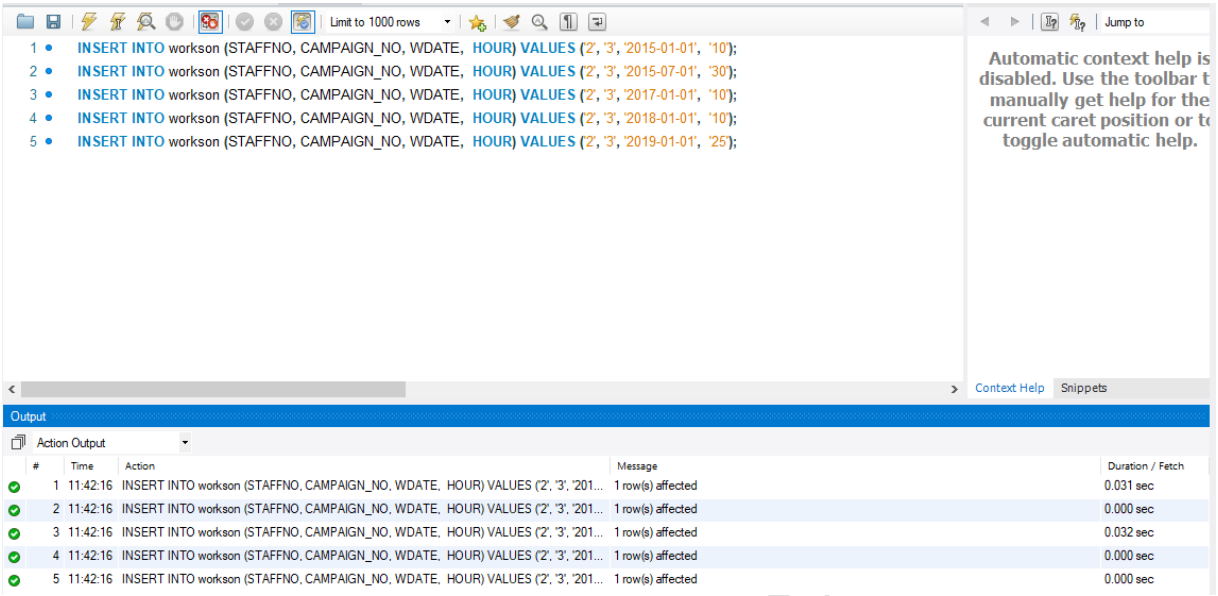
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('2', '3', '2015-01-01', '10');

INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('2', '3', '2015-07-01', '30');

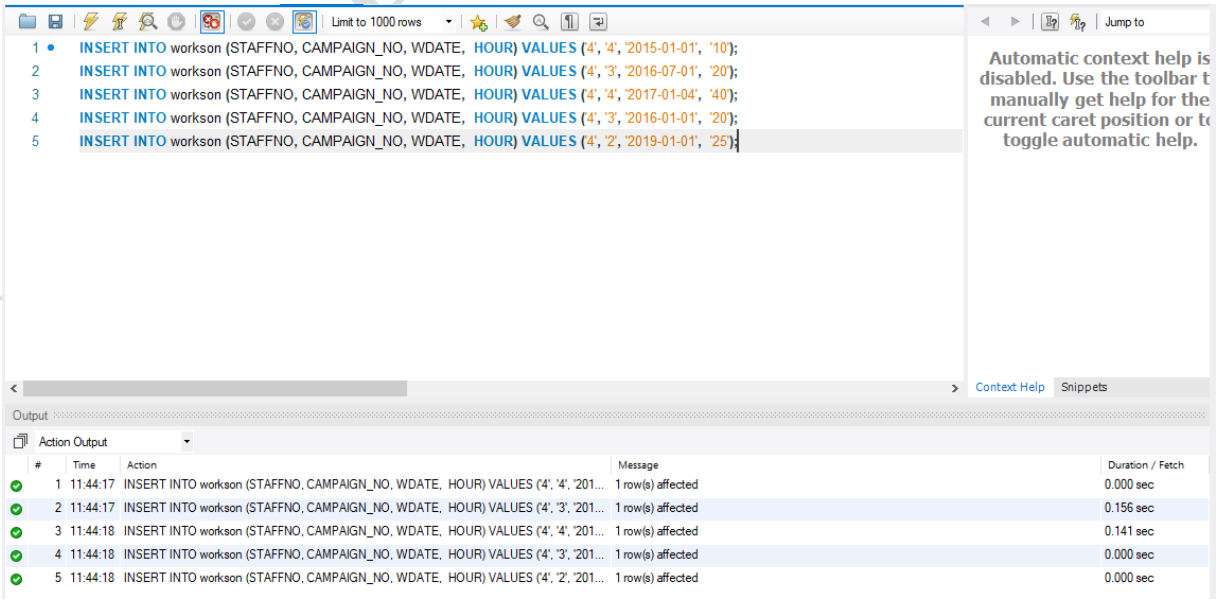
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('2', '3', '2017-01-01', '10');

INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('2', '3', '2018-01-01', '10');

INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('2', '3', '2019-01-01', '25');



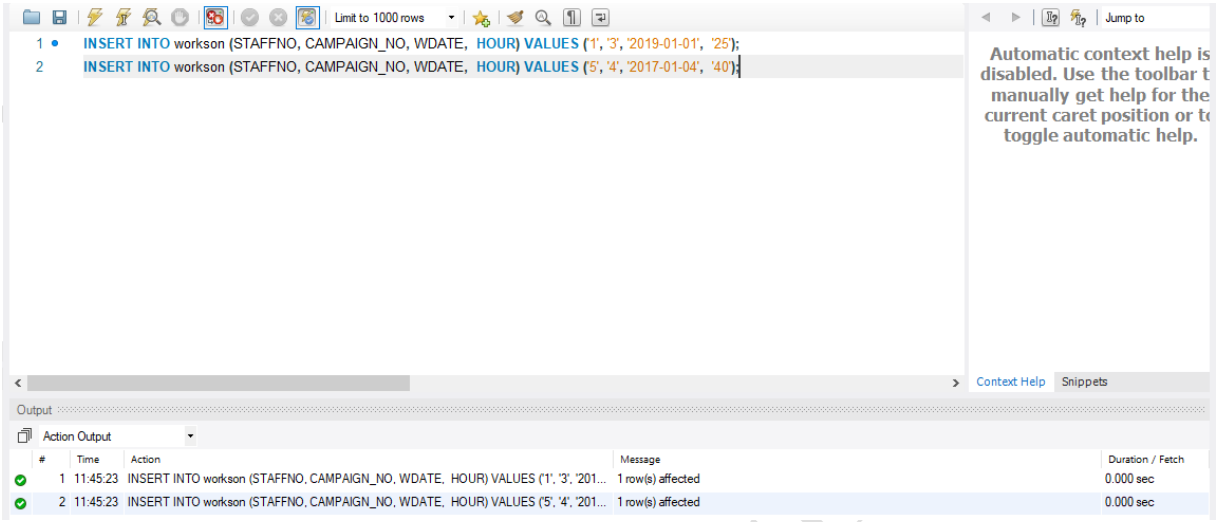
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('4', '4', '2015-01-01', '10');
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('4', '3', '2016-07-01', '20');
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('4', '4', '2017-01-04', '40');
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('4', '3', '2016-01-01', '20');
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('4', '2', '2019-01-01', '25');



INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('1',

'3', '2019-01-01', '25');

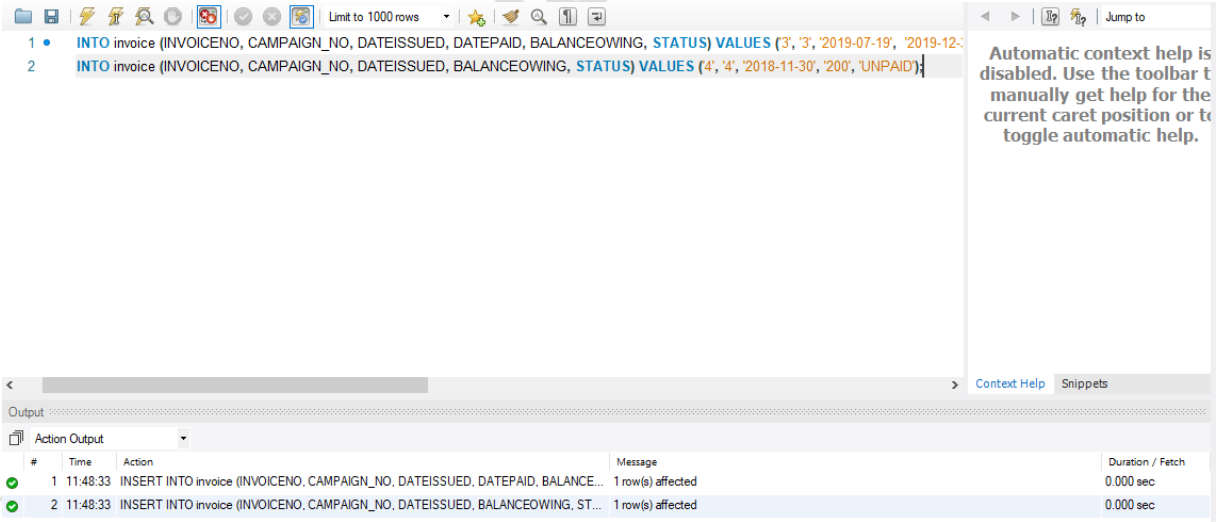
INSERT INTO workson (STAFFNO, CAMPAIGN_NO, WDATE, HOUR) VALUES ('5', '4', '2017-01-04', '40');



INSERT records on INVOICE table for new Campaings

INSERT INTO invoice (INVOICENO, CAMPAIGN_NO, DATEISSUED, DATEPAID, BALANCEOWING, STATUS) VALUES ('3', '3', '2019-07-19', '2019-12-31', '100', 'UNPAID');

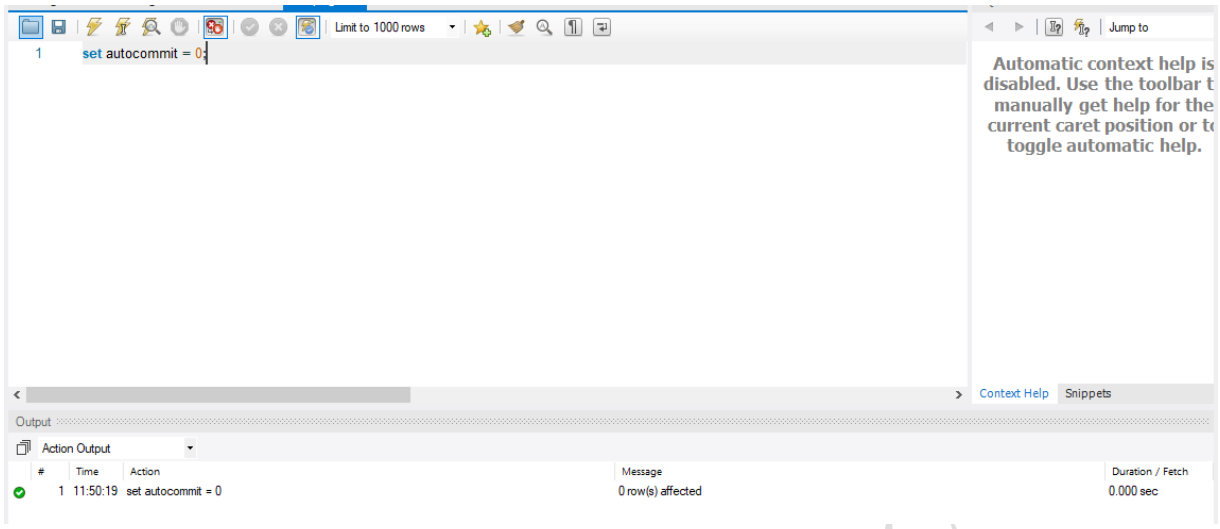
INSERT INTO invoice (INVOICENO, CAMPAIGN_NO, DATEISSUED, BALANCEOWING, STATUS) VALUES ('4', '4', '2018-11-30', '200', 'UNPAID');



Now run the test_script.sql file:

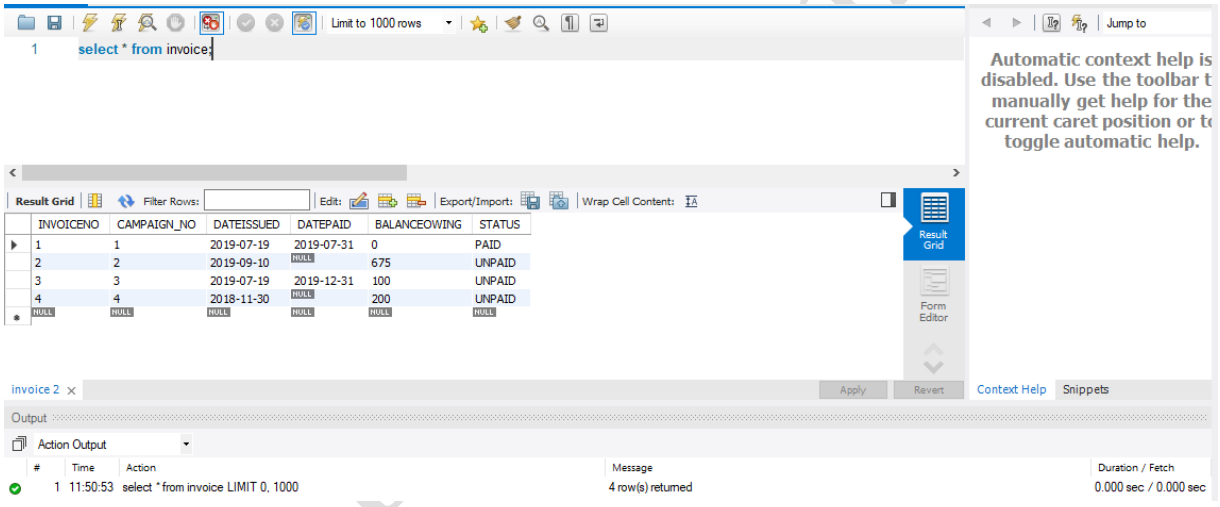
-- Turn off autocommit so you can have better control on what you are doing by rolling back transactions.

set autocommit = 0;

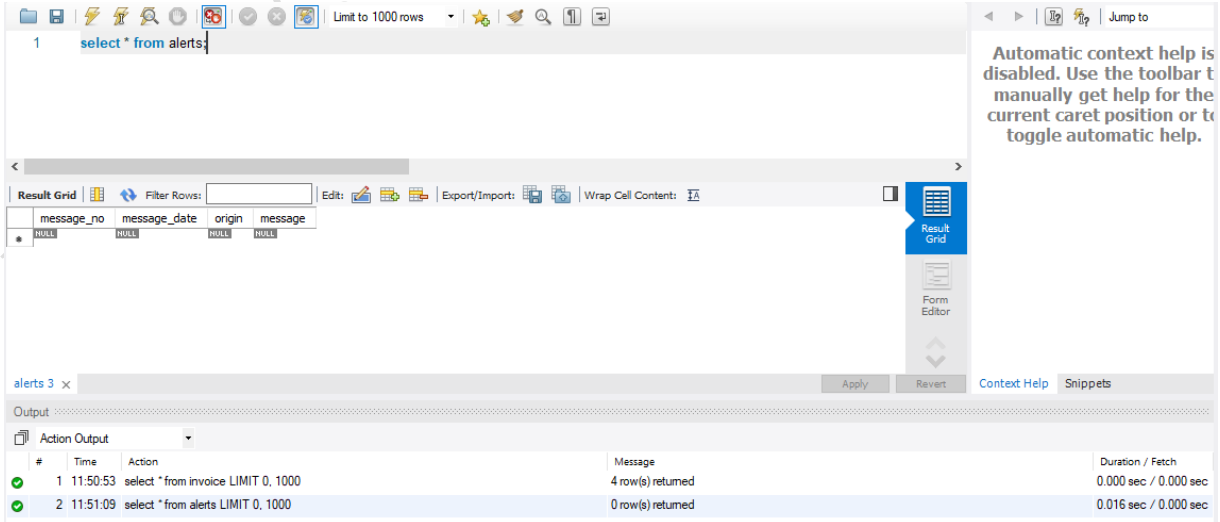


-- Inspect the invoice and the alerts table

*select * from invoice;*

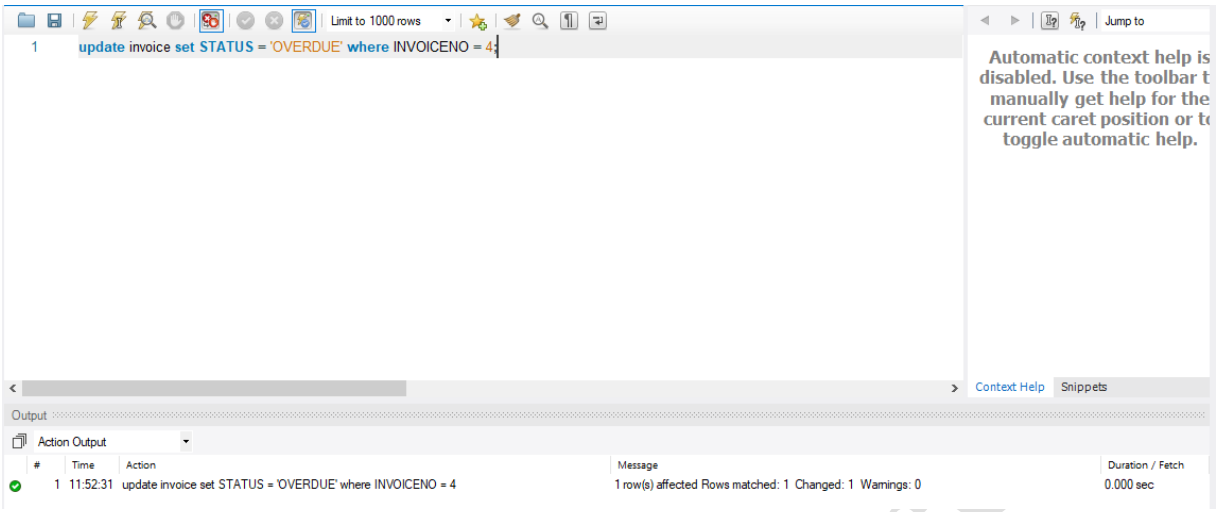


*select * from alerts;*

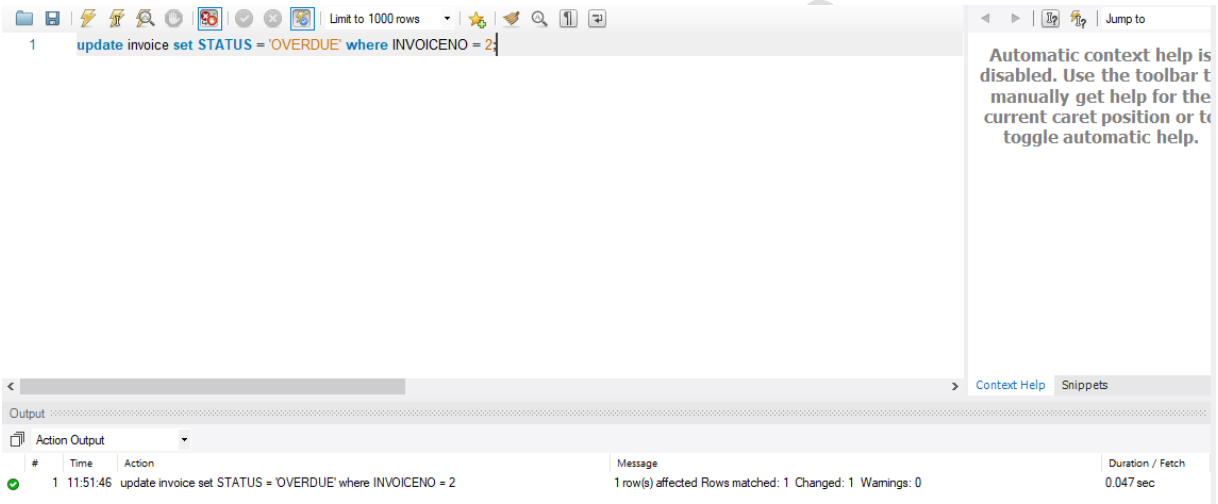


--Update the invoice table

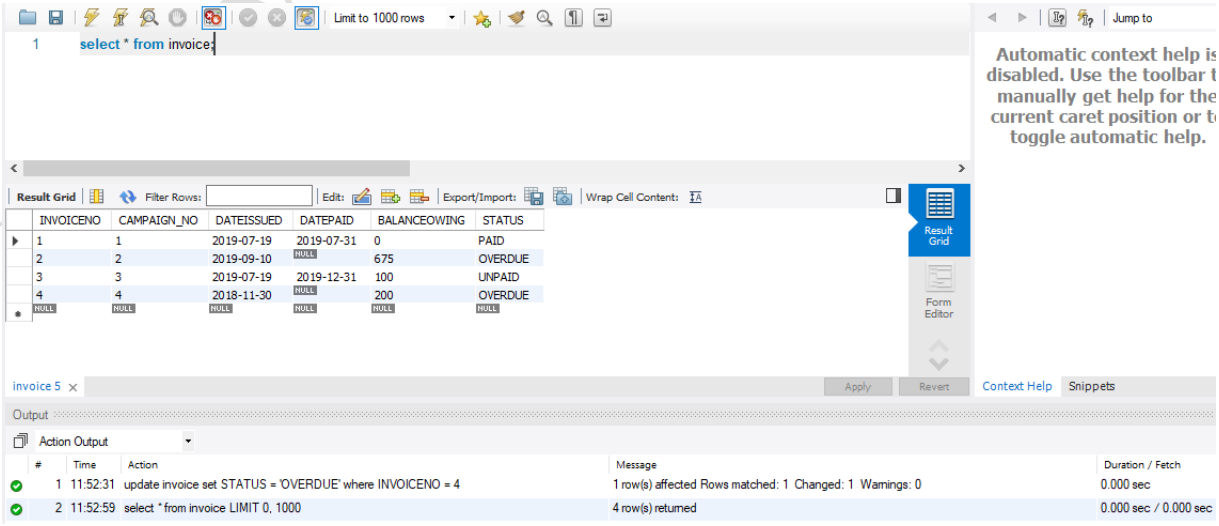
update invoice set STATUS = 'OVERDUE' where INVOICENO = 4;



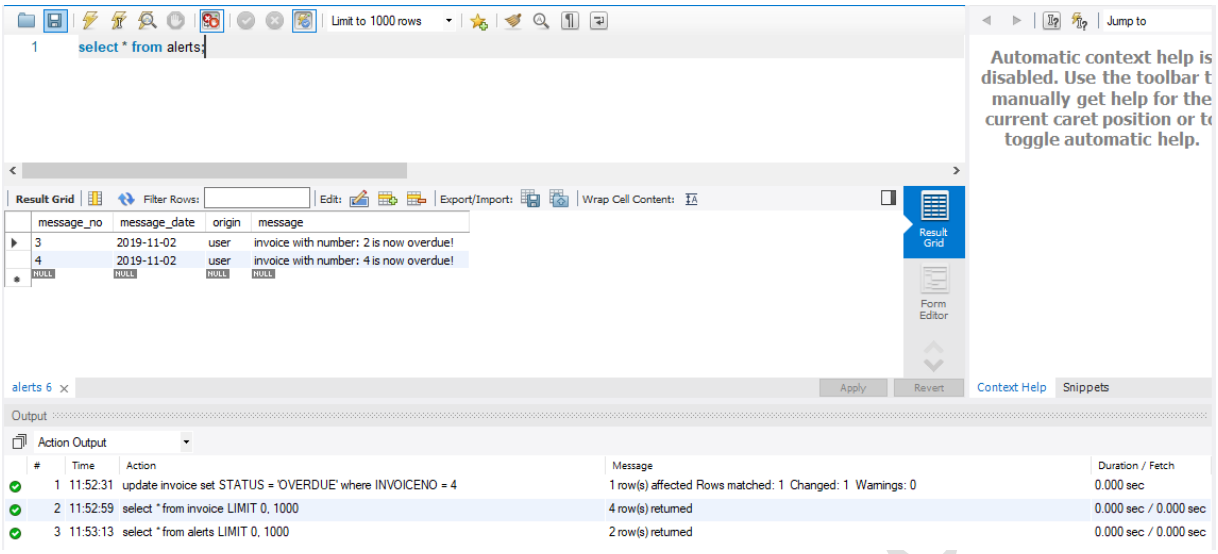
update invoice set STATUS = 'OVERDUE' where INVOICENO = 2;



-- Verify that the trigger you implemented works
*select * from invoice;*

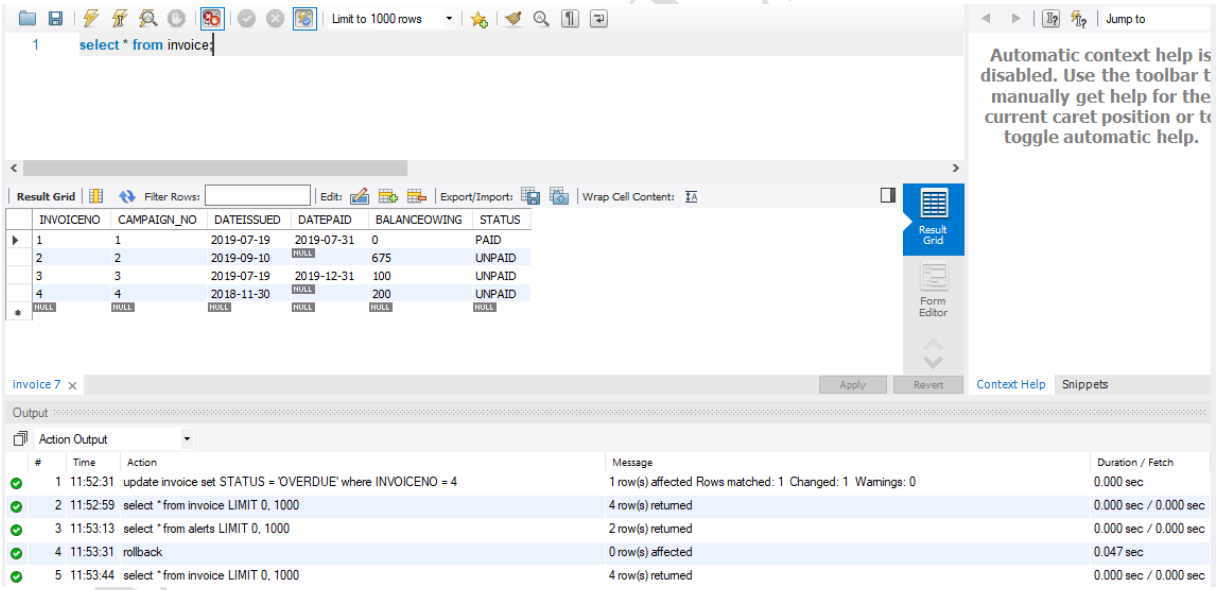


*select * from alerts;*

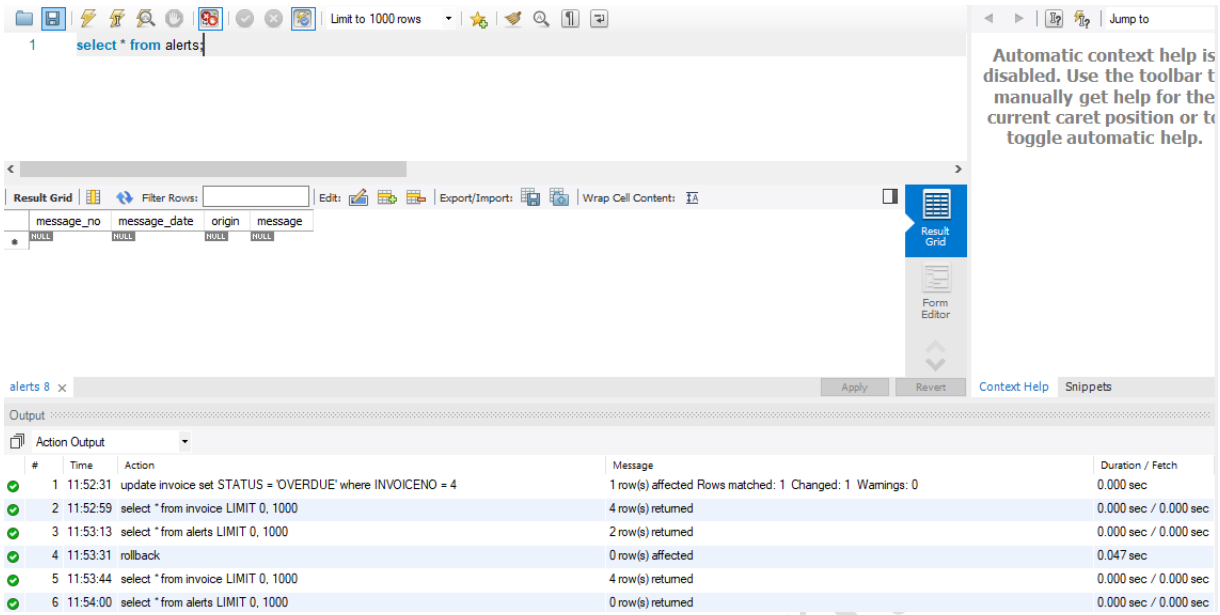


-- Bring DB back to original state; re-check rollback;

*select * from invoice;*

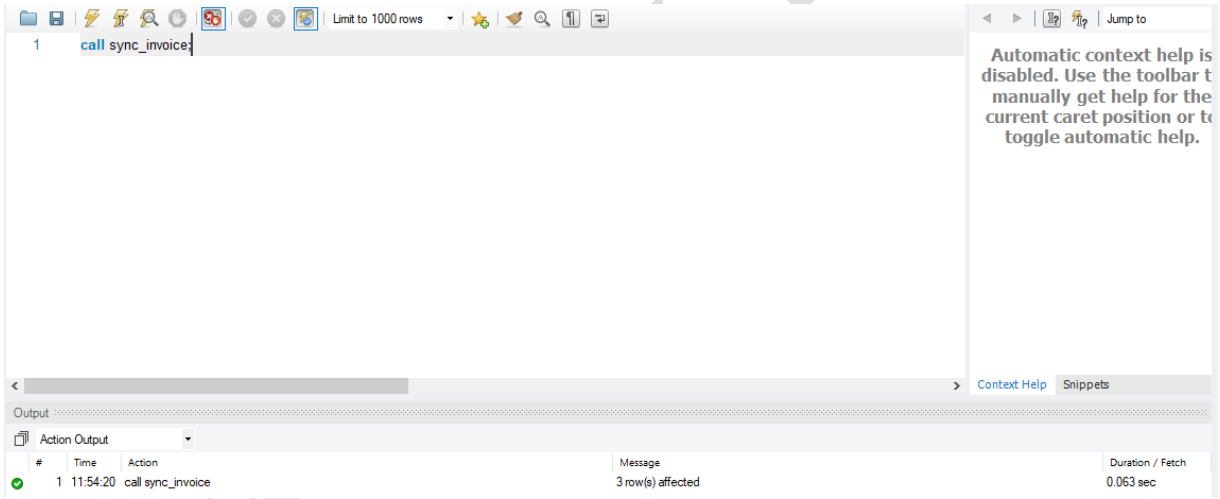


*select * from alerts;*

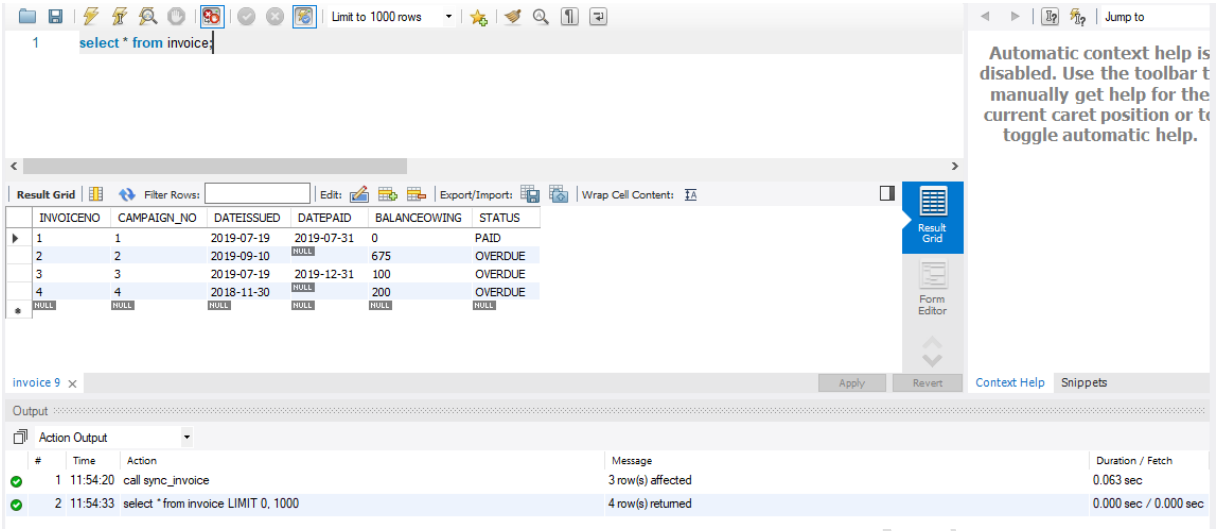


-- Synchronise the invoice table and verify the procedure behaves as desired

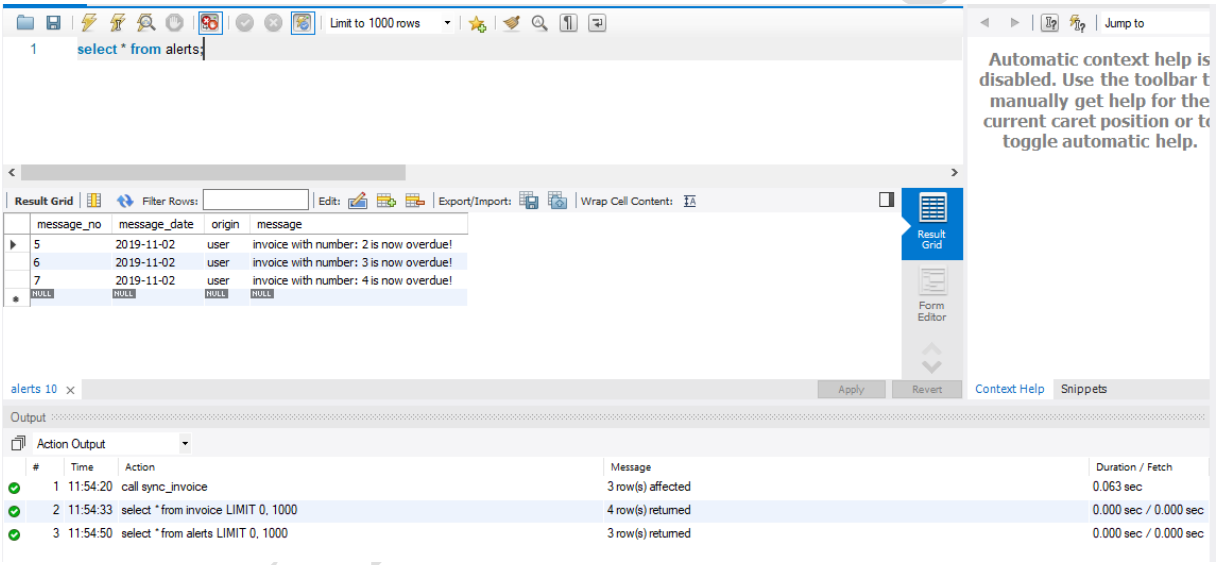
call sync_invoice;



*select * from invoice;*

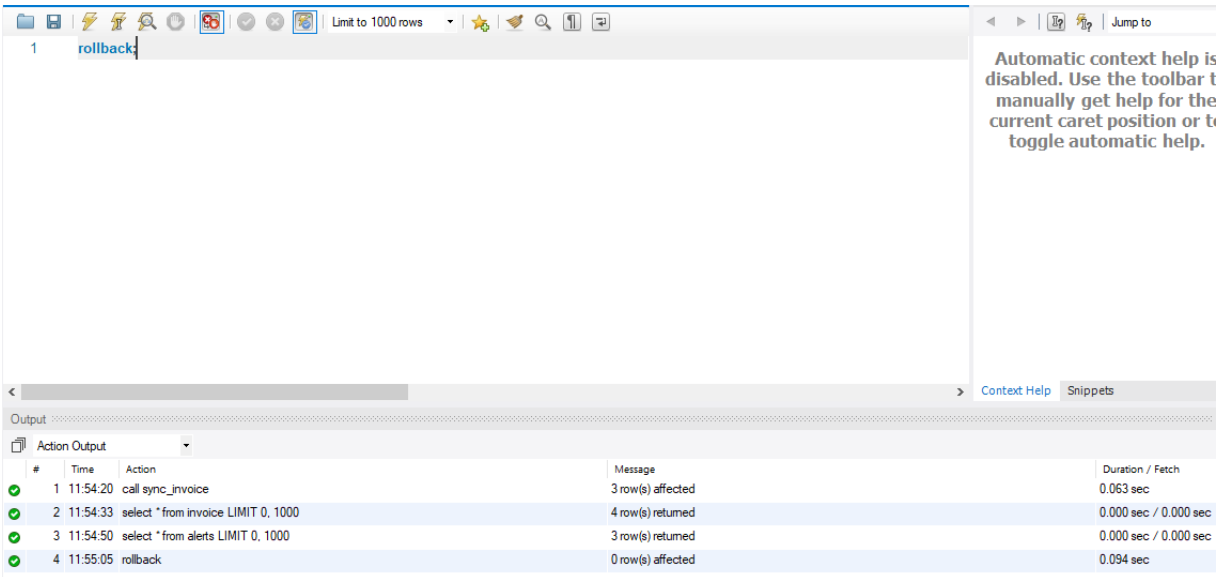


*select * from alerts;*

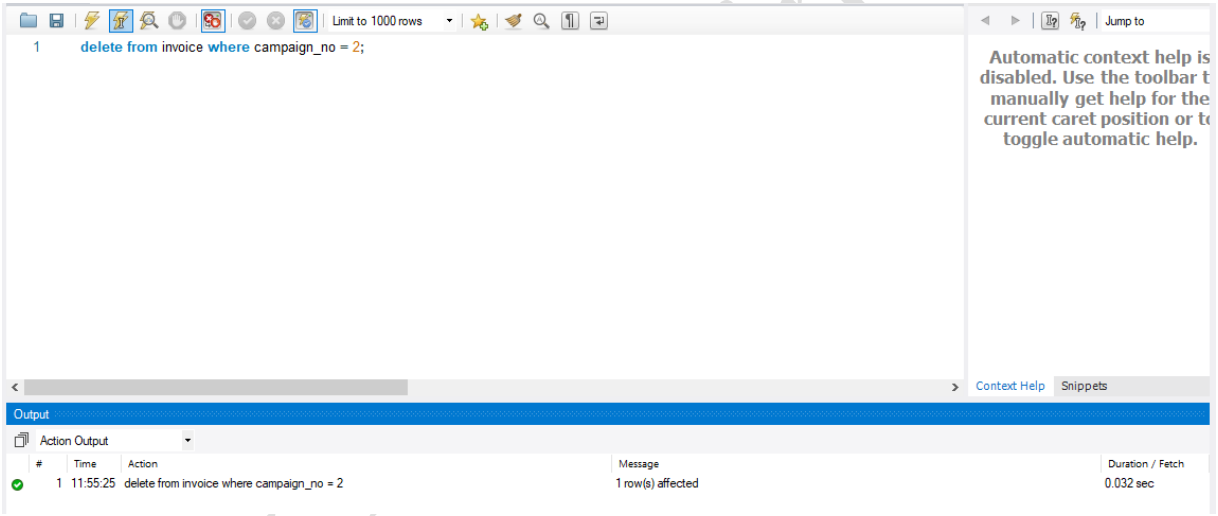


-- Bring DB back to original state; delete campaign# 2; check all relevant tables rollback;

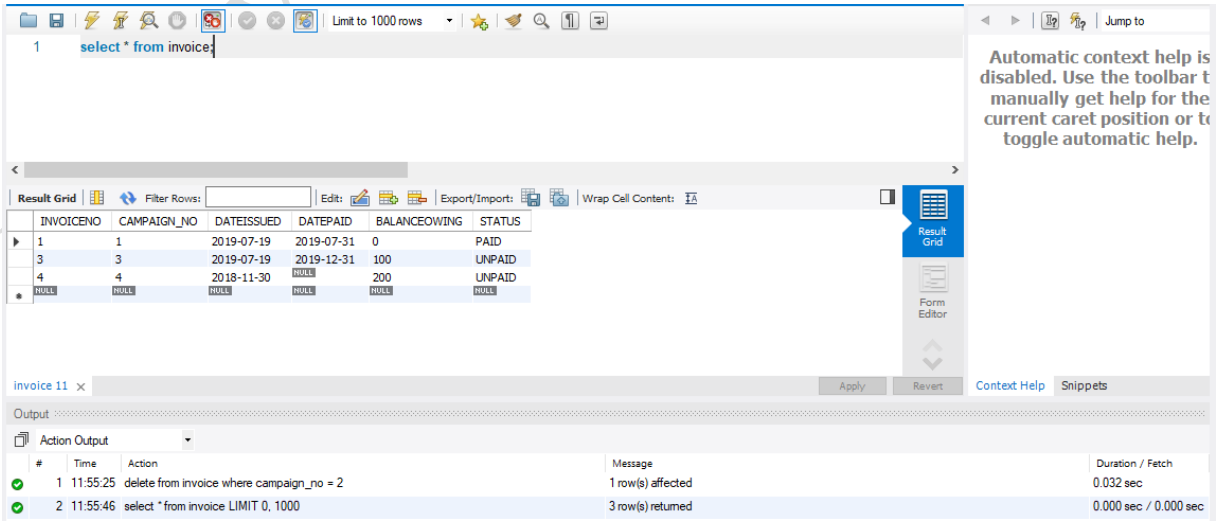
www



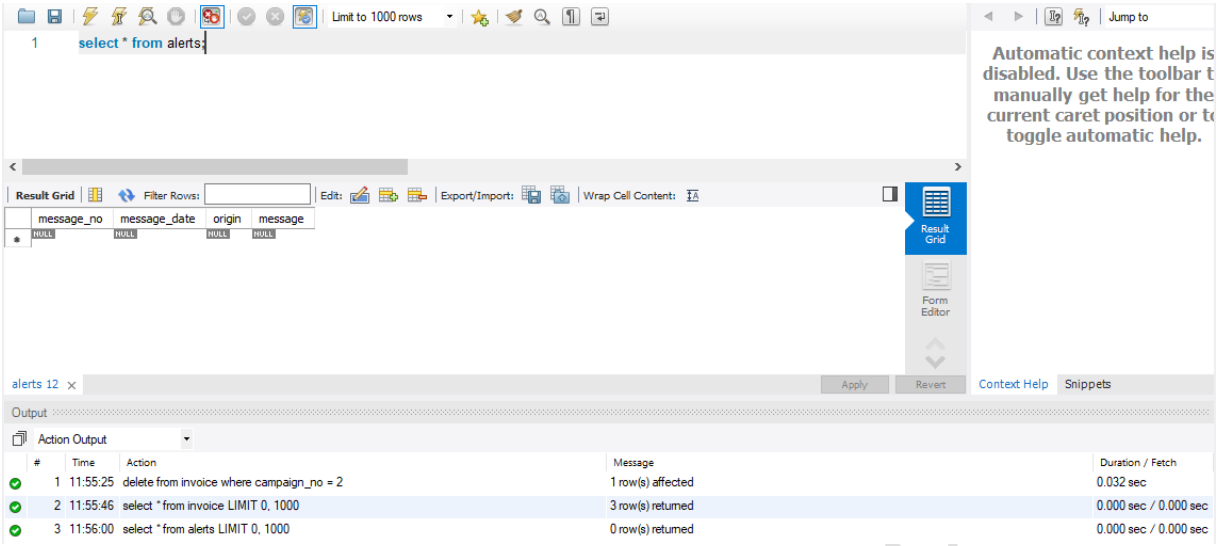
delete from invoice where campaign_no = 2;



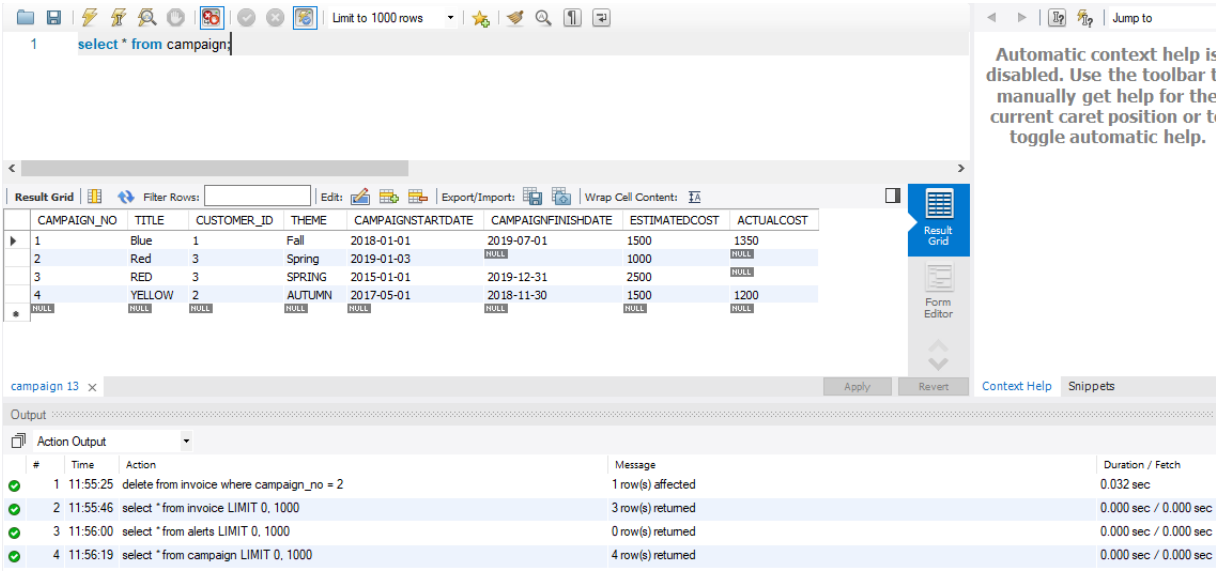
*select * from invoice;*



*select * from alerts;*

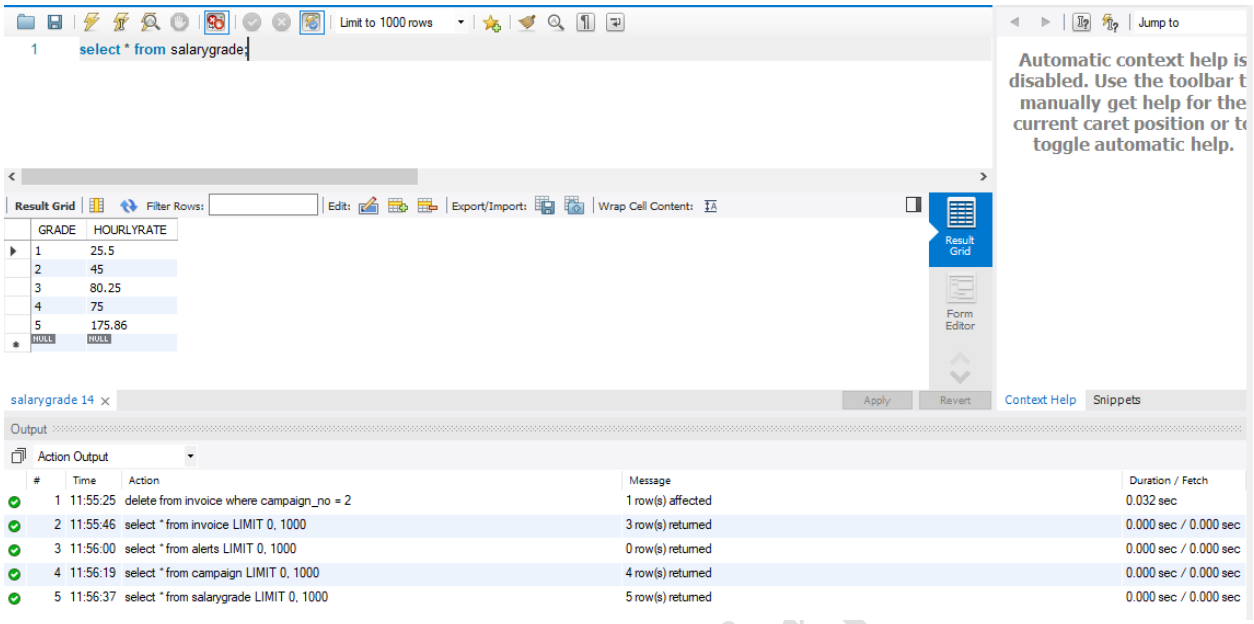


*select * from campaign;*

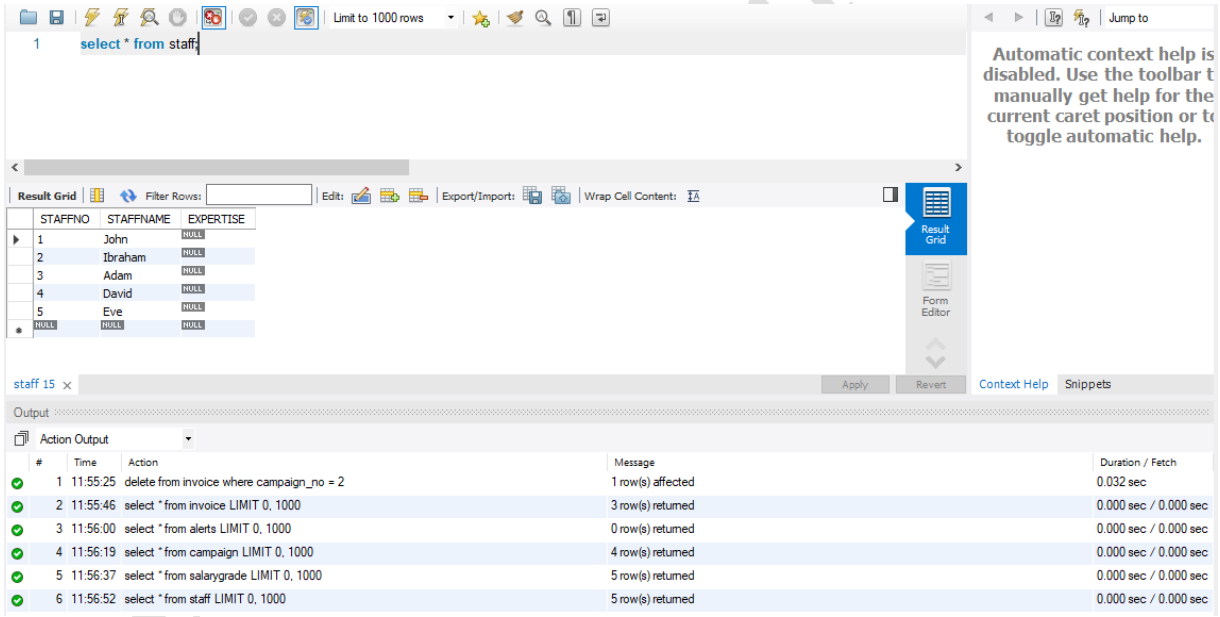


*select * from salarygrade;*

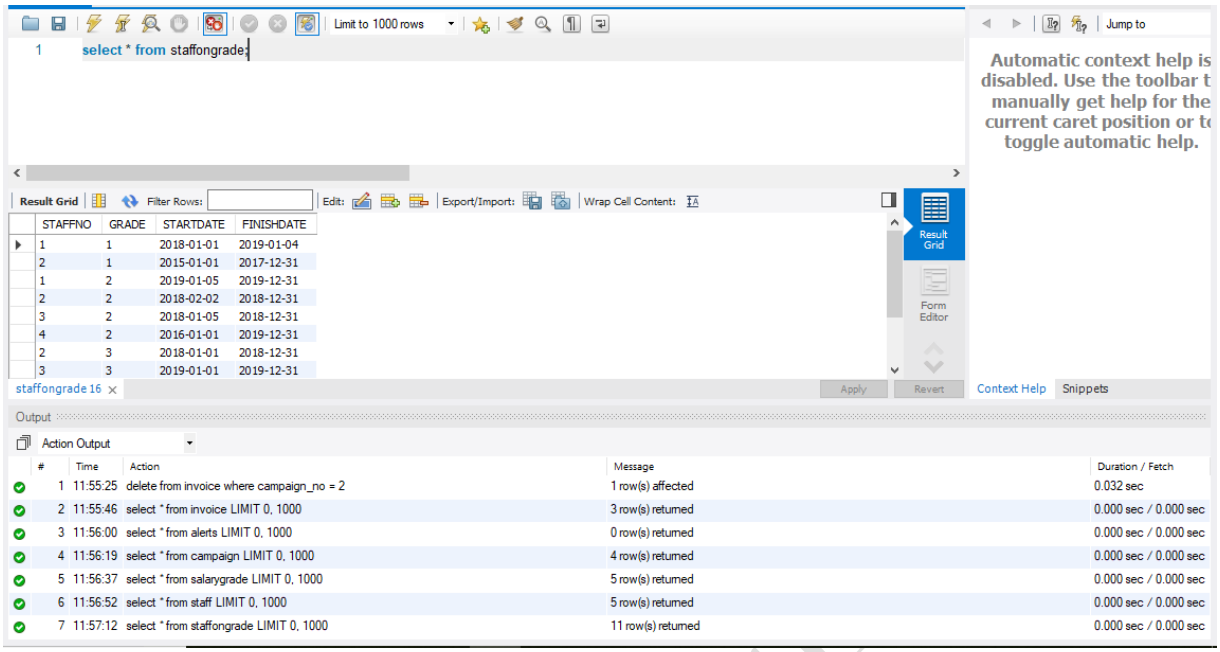
www.



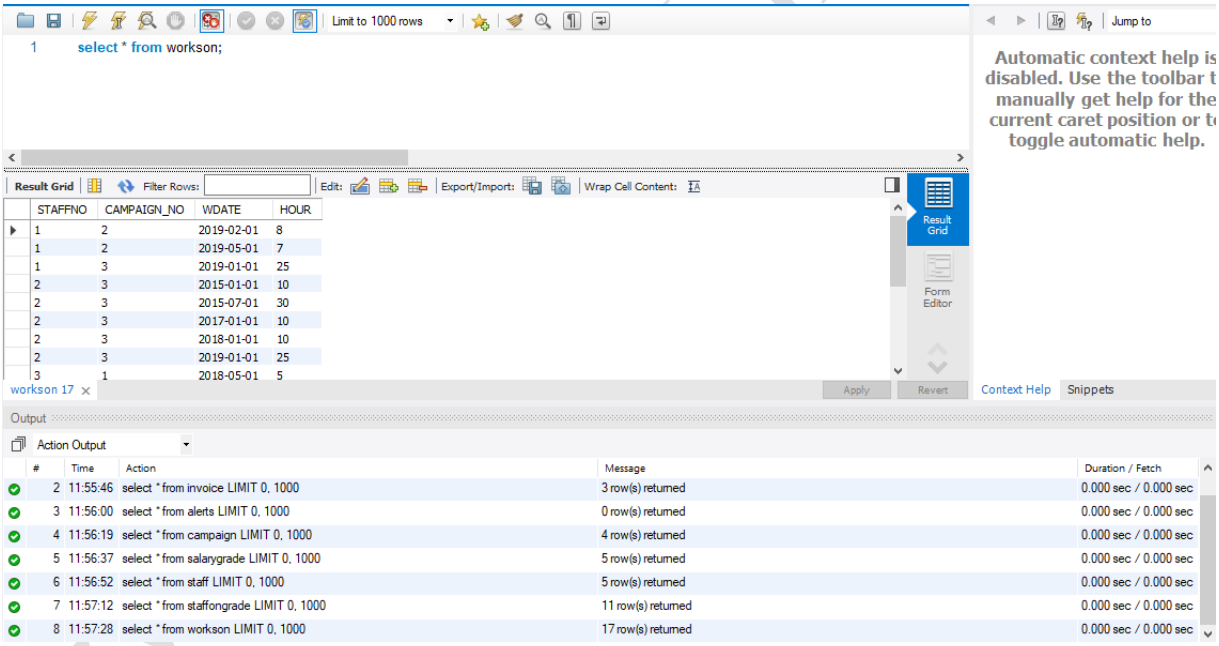
*select * from staff;*



*select * from staffongrade;*

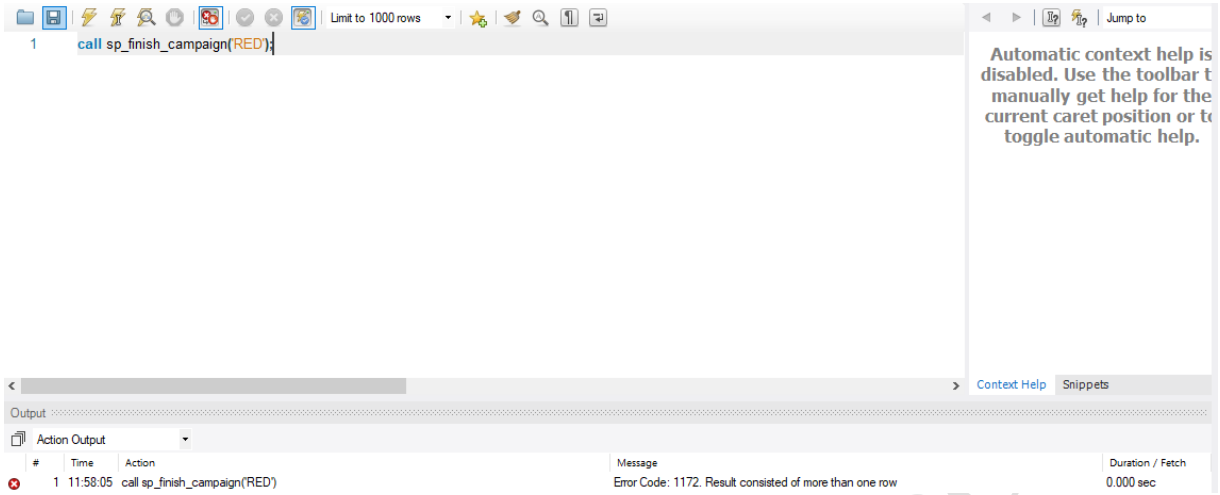


*select * from workson;*



-- Finish the campaign titled RED. Verify that it behaves as desired.

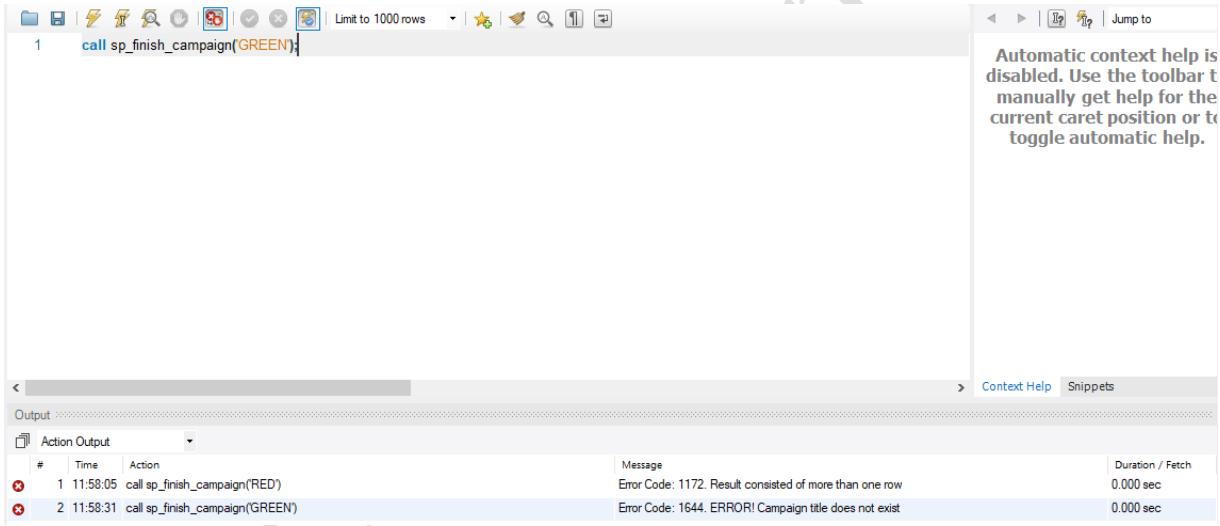
call sp_finish_campaign('RED');



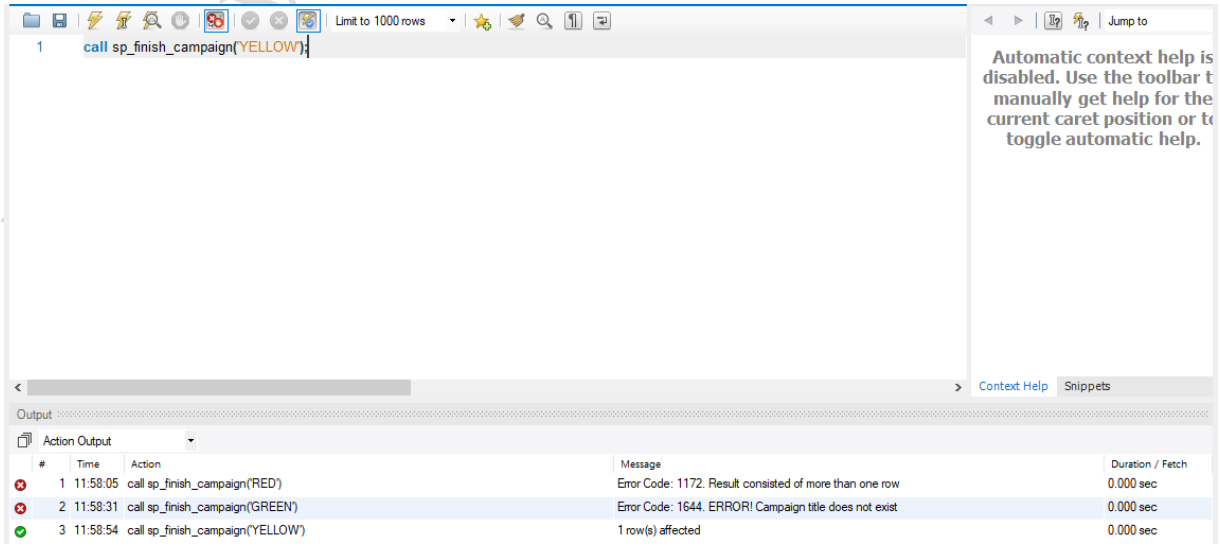
*select * from campaign;*

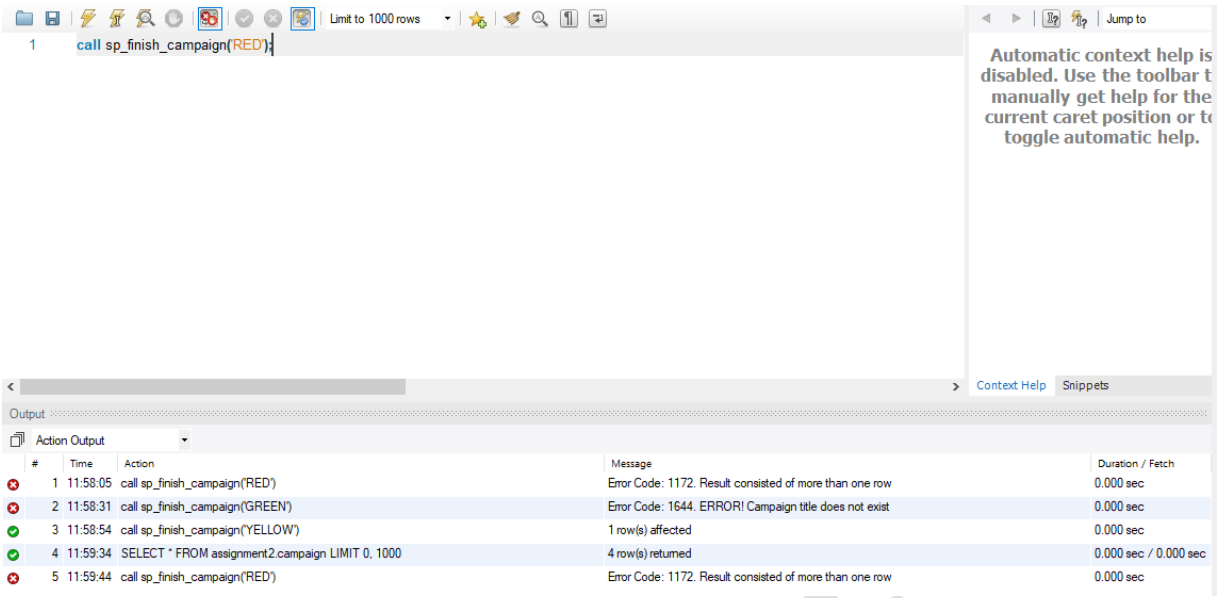
*select * from invoice;*

call sp_finish_campaign('GREEN'); -- should SIGNAL error condition



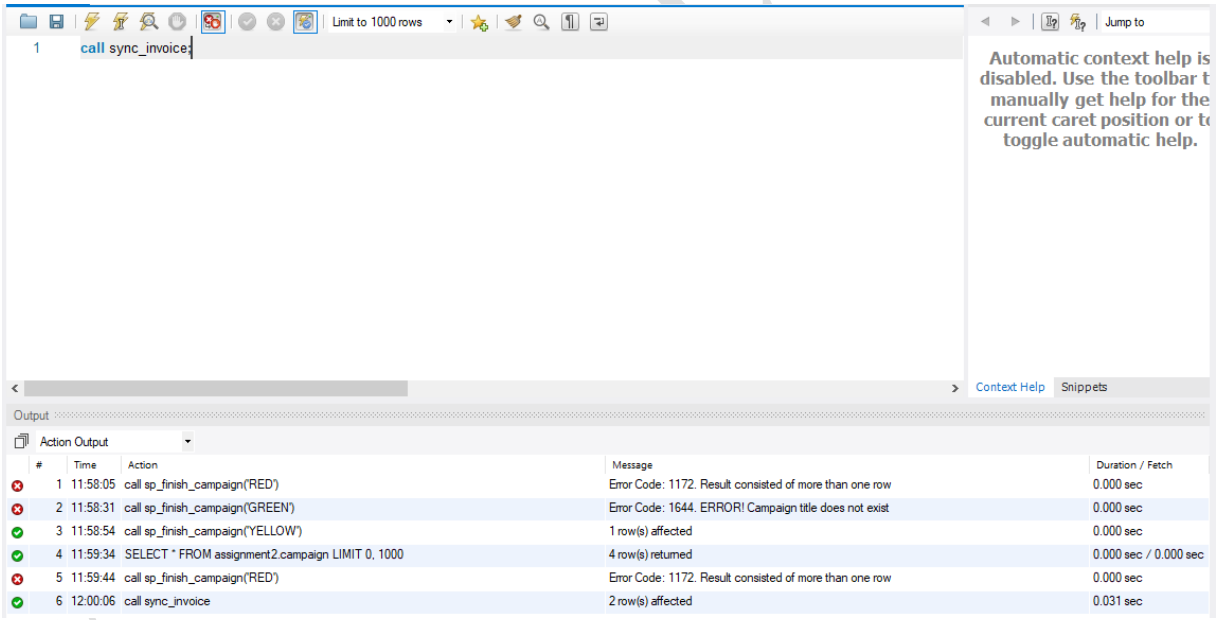
call sp_finish_campaign('YELLOW');



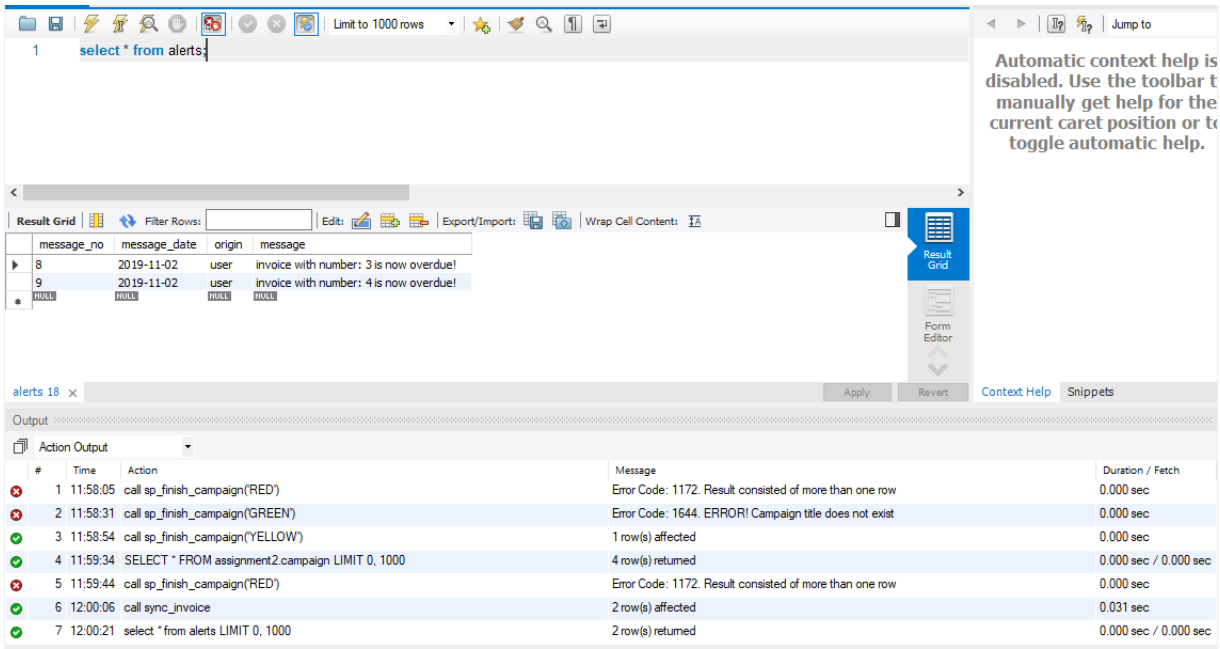


-- Synchronise the invoice table and verify the procedure behaves as desired

call sync_invoice;



*select * from alerts;*



rollback;

5. Notes (optional).

Mention here anything worth noting,e.g., whether you faced any particular difficulty in completing any of these tasks, the nature and extent of any help you received from anyone, and why.

When we call the stored procedure sp_finish_campaign for RED it throws an error message that Result consisted more than one row. Here our function failed because there are duplicate records inserted as mentioned in the STEP 4.

When I run the update statement it throws an error as you are working in safe mode so I searched on google and find the solution given below. After executing the below statement I was able to execute the update statement.

```
SET SQL_SAFE_UPDATES = 0;
```

www.ExpertLi