

MACQUARIE UNIVERSITY

**Faculty of Science and Engineering**

**Department of Computing**

**ISYS224/ITEC624 Database Systems 2019 (Semester 2)**

**Assignment 2 (Report)**

**Database Programming and Implementation (worth 15%)**

**Student Name:**

**Student Number:**

**Student Declaration:**

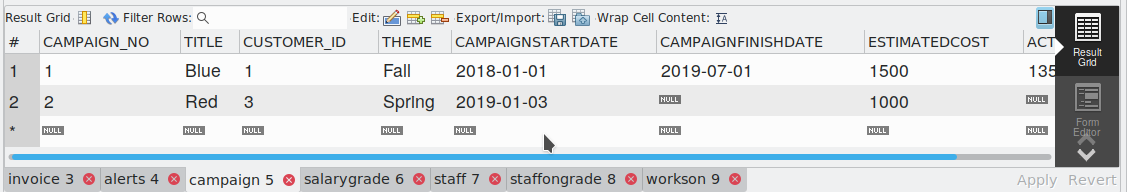
*I declare that the work reported here is my own. Any help received, from any person, through discussion or other means, has been acknowledged in the last section of this report.*

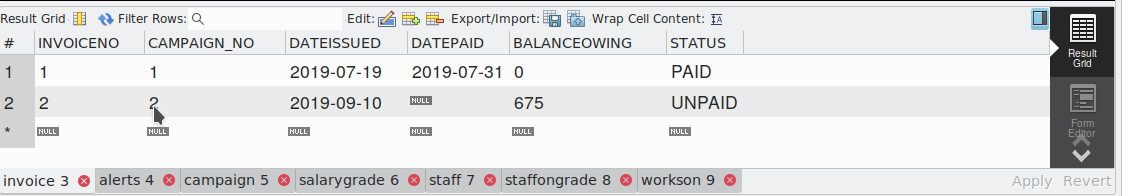
Student Signature:

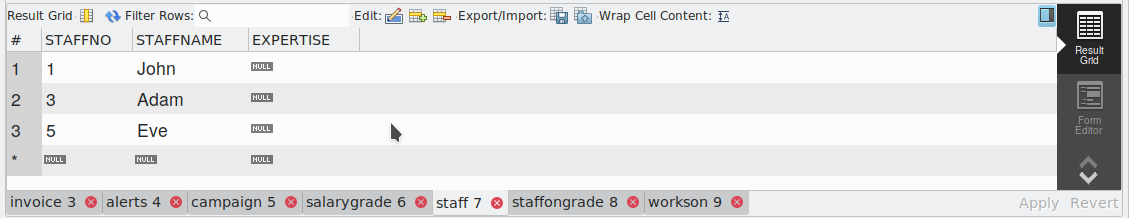
Student Name and Date:

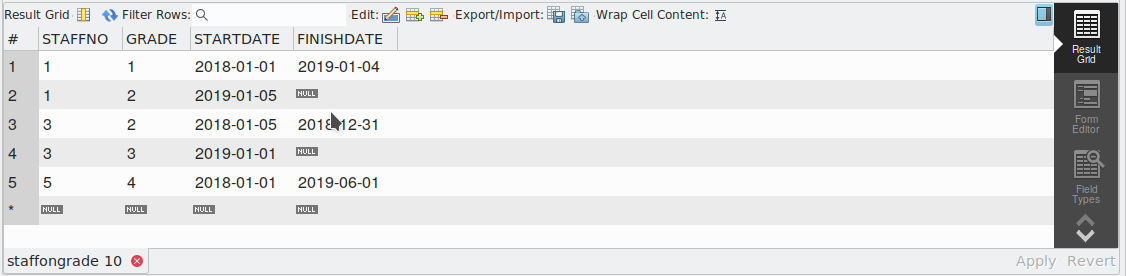
1. **Initial State of the database**

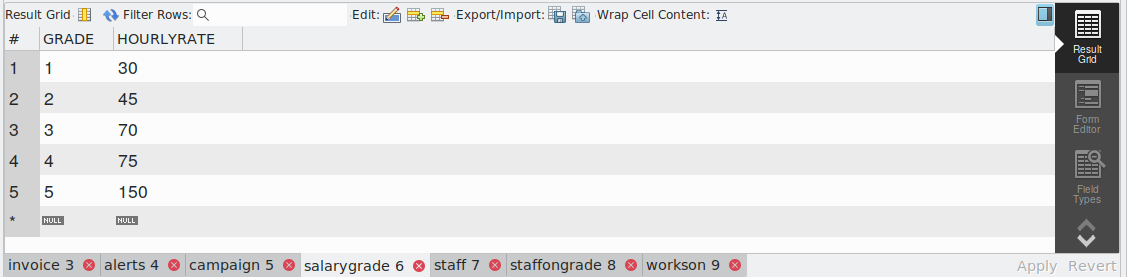
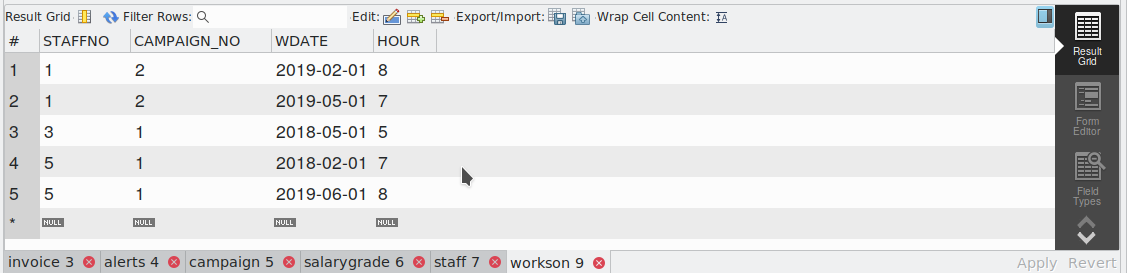
*Paste here the screen shots showing the provided sample data in the tables*.











1. **Stored Programs.**

*Paste your sql file <yourLastname\_yourFirstname>\_programs.sql here.*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*\*\*\*\* Stored Programs FOR Assn.2, 2019 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*\*\*\*\*\*\*\*\* Student Name and Number \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Date \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\* I DECLARE that the code provided below is my own work \*\**

*\*\*\*\*\*\*\* Any help received is duely acknowledged here \*\*\*\*\*\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\*\*\*\*\*\*\*\*\* Trigger TR\_OVERDUE \*\*\*\*\*\*\*\*\*\*\*\*/*

*delimiter //*

*DROP TRIGGER IF EXISTS tr\_overdue*

*//*

*CREATE TRIGGER tr\_overdue AFTER UPDATE ON invoice*

*FOR EACH ROW*

*BEGIN*

*IF old.STATUS <> "OVERDUE" AND new.STATUS = "OVERDUE" THEN*

*INSERT INTO alerts (message\_date,origin,message) values(CURRENT\_DATE,CURRENT\_USER,CONCAT( "Invoice with number: ",new.INVOICENO," is now overdue!"));*

*END IF;*

*END*

*//*

*/\*\*\*\*\*\*\*\*\*\*\*\*\* Helper FUNCTIONs/PROCEDUREs used, two FUNCTIONs FOR example \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*DROP FUNCTION IF EXISTS rate\_on\_date*

*//*

*CREATE FUNCTION rate\_on\_date(staff\_id INTEGER, given\_date DATE) returns FLOAT DETERMINISTIC*

*BEGIN*

*DECLARE total\_days\_pay FLOAT;*

*DECLARE hour\_rate FLOAT;*

*DECLARE finished INTEGER;*

*DECLARE cur1 CURSOR FOR SELECT HOURLYRATE FROM staffongrade,salarygrade,workson WHERE staffongrade.STAFFNO=staff\_id AND salarygrade.GRADE=staffongrade.GRADE AND workson.WDATE=given\_date AND workson.STAFFNO=staff\_id;*

*DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=1;*

*OPEN cur1;*

*read\_loop: LOOP*

*IF finished=1 THEN*

*LEAVE read\_loop;*

*END IF;*

*FETCH cur1 INTO hour\_rate;*

*END LOOP ;*

*CLOSE cur1;*

*return hour\_rate;*

*END*

*//*

*--*

*DROP FUNCTION IF EXISTS cost\_of\_campaign*

*//*

*CREATE FUNCTION cost\_of\_campaign (camp\_id INTEGER)*

*returns FLOAT*

*-- retuens the total cost incurred due to any given campaigh (camp\_id)*

*-- READS SQL DATA*

*-- DETERMINISTIC*

*BEGIN*

*-- code*

*DECLARE total\_cost FLOAT;*

*DECLARE finished INTEGER;*

*DECLARE cur1 CURSOR FOR SELECT SUM(HOURLYRATE\*HOUR) FROM staffongrade,salarygrade,workson,campaign WHERE staffongrade.STAFFNO=workson.STAFFNO AND salarygrade.GRADE=staffongrade.GRADE AND workson.CAMPAIGN\_NO=camp\_id AND staffongrade.STARTDATE >=campaign.CAMPAIGNSTARTDATE AND staffongrade.FINISHDATE <= campaign.CAMPAIGNFINISHDATE;*

*DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=1;*

*OPEN cur1;*

*read\_loop: LOOP*

*IF finished=1 THEN*

*LEAVE read\_loop;*

*END IF;*

*FETCH cur1 INTO total\_cost;*

*END LOOP ;*

*CLOSE cur1;*

*return total\_cost;*

*END*

*//*

*/\*\*\*\*\*\*\*\*\*\*\*\* PROCEDURE SP\_FINISH\_CAMPAIGN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*DROP PROCEDURE IF EXISTS sp\_finish\_campaign*

*//*

*CREATE PROCEDURE sp\_finish\_campaign (IN c\_title VARCHAR(30))*

*-- code*

*-- DECLARE CONTINUE HANDLER FOR NOT FOUND; \*/*

*BEGIN*

*-- do the error handling*

*DECLARE cost INTEGER;*

*DECLARE finished INTEGER;*

*DECLARE camp\_id INTEGER;*

*DECLARE result\_count INTEGER;*

*DECLARE error\_string varchar(255) DEFAULT "N";*

*DECLARE cs CURSOR FOR SELECT COUNT(CAMPAIGN\_NO) FROM campaign WHERE TITLE=c\_title;*

*DECLARE mycursor CURSOR FOR SELECT CAMPAIGN\_NO FROM campaign WHERE TITLE=c\_title;*

*DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=1;*

*OPEN cs;*

*FETCH cs INTO result\_count;*

*CLOSE cs;*

*OPEN mycursor;*

*read\_loop: LOOP*

*IF result\_count=0 THEN*

*SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT="ERROR! Campaign title does not exist";*

*LEAVE read\_loop;*

*END IF;*

*IF finished=1 THEN*

*LEAVE read\_loop;*

*END IF;*

*FETCH mycursor INTO camp\_id;*

*END LOOP ;*

*SET cost=cost\_of\_campaign(camp\_id);*

*UPDATE campaign SET CAMPAIGNFINISHDATE=CURRENT\_DATE , ACTUALCOST=cost WHERE TITLE=c\_title;*

*CLOSE mycursor;*

*END*

*//*

*/\*\*\*\*\*\*\*\*\*\*\*\* PROCEDURE SYNC\_INVOICE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*DROP PROCEDURE IF EXISTS sync\_invoice;*

*//*

*CREATE PROCEDURE sync\_invoice()*

*BEGIN*

*-- code*

*UPDATE invoice*

*SET STATUS="OVERDUE"*

*WHERE STATUS="UNPAID" AND DATEDIFF(CURRENT\_DATE,DATEISSUED) > 30;*

*END*

*//*

*delimiter ;*

1. **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.*

*/\*\*\*\*\*\*\*\*\*\*\*\* Student Name \*\*\*\*\*\*\*\*\*\*\*/*

*/\*\*\*\*\*\*\*\*\*\*\* Student Number \*\*\*\*\*\*\*\*\*\*/*

*-- Create the tables*

*-- Populate the tables*

*-- Create the stored objects (procedures/functions/triggers)*

*-- Then...*

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

*set autocommit = 0;*

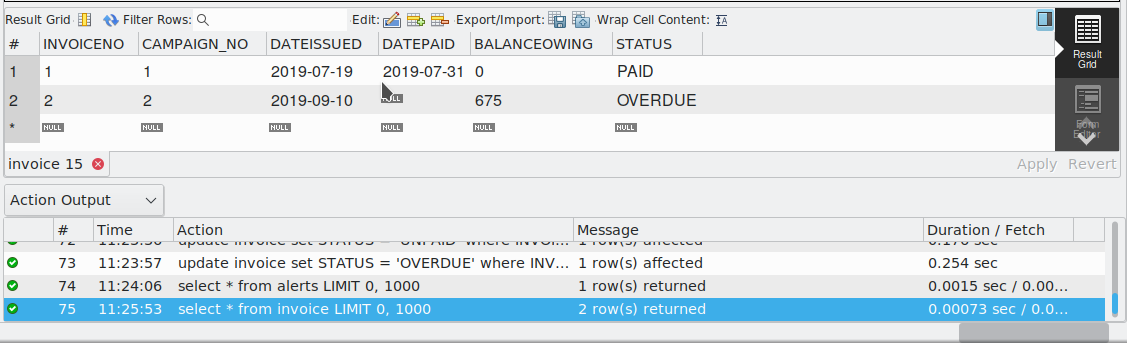
*-- Inspect the invoice and the alerts table*

*select \* from invoice;*

*select \* from alerts;*

*--Update the invoice table*

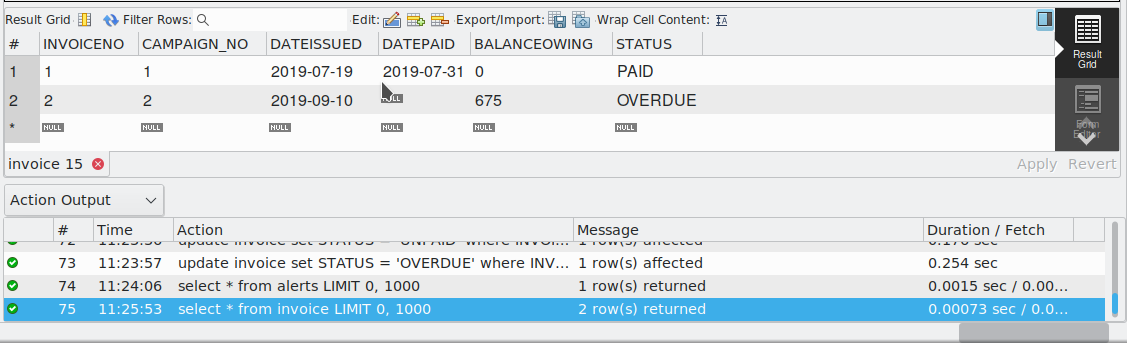
*update invoice set STATUS = 'OVERDUE' where INVOICENO = 2;*

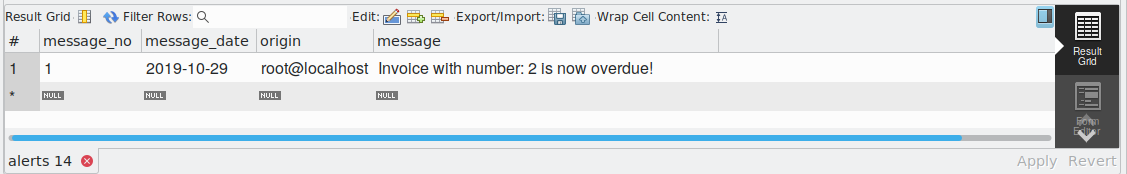


*-- Verify that the trigger you implemented works*

*select \* from invoice;*

*select \* from alerts;*





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

*rollback;*

*select \* from invoice;*

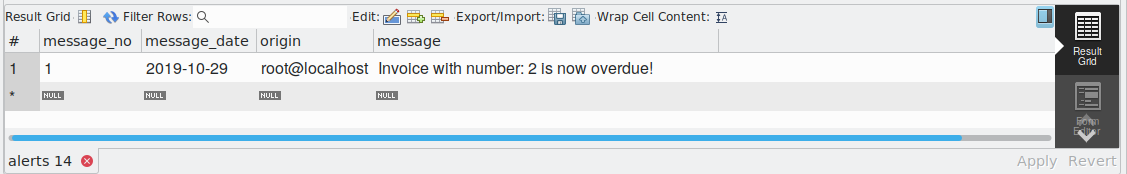
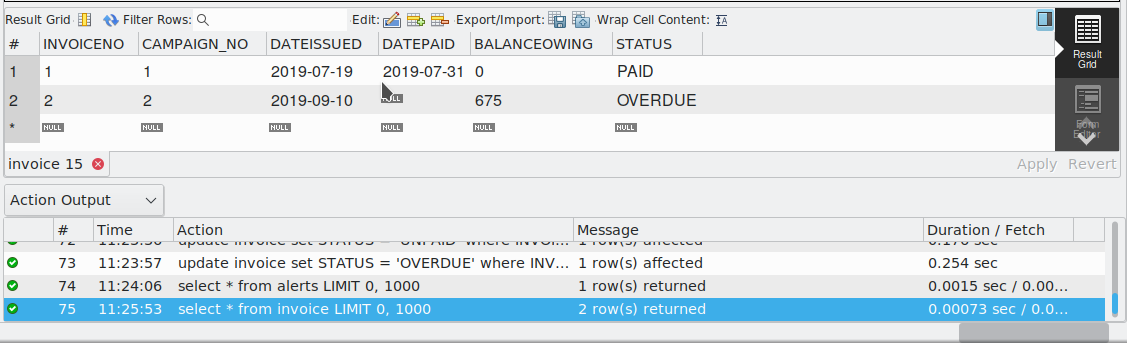
*select \* from alerts;*

*-- 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;*

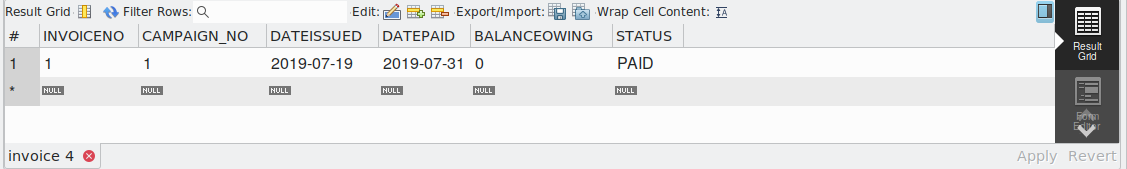
*select \* from campaign;*

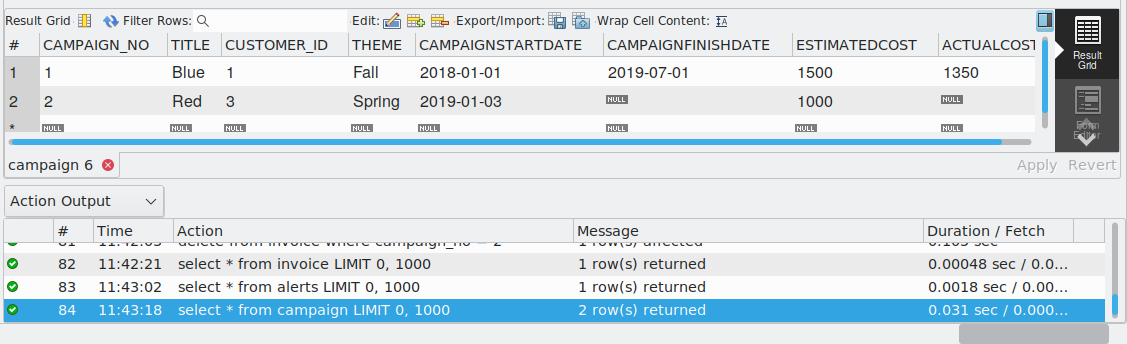
*select \* from salarygrade;*

*select \* from staff;*

*select \* from staffongrade;*

*select \* from workson;*



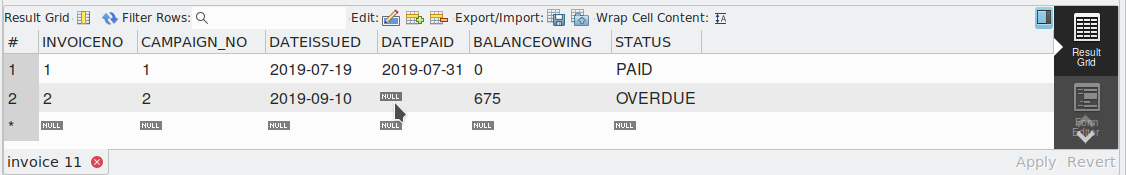


*-- Finish the campaign titled RED. Verify that it behaves as desired.*

*call sp\_finish\_campaign('RED');*

*select \* from campaign;*

*select \* from invoice;*



*call sp\_finish\_campaign('GREEN'); -- should SIGNAL error condition*



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

*call sync\_invoice;*

*select \* from alerts;*

*rollback;*

1. **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).*

*-- The test above is minimal with very little data.*

*-- Think of tests that you need to carry out to gain confidence cthat the programs you wrote does the right thing.*

*-- Insert more data to the tables strategically to show quickly if there are semantic errors in your programs.*

*insert into customer(CUSTOMER\_ID,COMPANYNAME,ADDRESS,STAFF\_STAFFNO)*

*values*

*(9,"ABC LTD","Texas",1),*

*(5,"NOVATTA LTD","Ohio",5),*

*(4,"TRALAHTEK LTD","Home",1),*

*(8,"JKL LTD","GUAM",3),*

*(7,"XYZ LTD","CHICAGO",5);*

*commit;*

*insert into campaign (CAMPAIGN\_NO,TITLE,CUSTOMER\_ID,THEME,CAMPAIGNSTARTDATE,CAMPAIGNFINISHDATE,ESTIMATEDCOST,ACTUALCOST)*

*values*

*(6,"Marketing",9,"Dummy","2017-04-02","2017-08-09",25500,NULL),*

*(7,"promotion",5,"ANother","2018-06-24","2018-11-29",25900,NULL),*

*(9,"campaign",4,"broadcast","2019-10-23","2019-12-29",45700,NULL),*

*(8,"roadshow",8,"mysql","2018-04-23","2018-12-23",9500,NULL),*

*(10,"reseller",7,"technology","2019-04-13","2019-06-19",19000,NULL);*

*commit;*

*insert into workson (STAFFNO,CAMPAIGN\_NO,WDATE,HOUR)*

*values*

*(1,6,'2017-04-20',8),*

*(3,6,'2017-05-20',6),*

*(5,7,'2018-07-20',7),*

*(3,7,'2018-07-20',9),*

*(1,8,'2018-04-30',8),*

*(3,8,'2018-05-06',6),*

*(5,8,'2018-07-11',7),*

*(1,10,'2019-04-20',5),*

*(3,10,'2019-04-20',7);*

*commit;*

*insert into invoice(CAMPAIGN\_NO,DATEISSUED,BALANCEOWING,STATUS)*

*values*

*(6,'2017-08-09',25500,"UNPAID"),*

*(7,'2018-11-29',25900,"UNPAID"),*

*(8,'2018-12-23',9500,"PAID"),*

*(10,'2019-06-19',19000,"UNPAID");*

*commit;*

*-- run the tests and display the results.*

*call sp\_finish\_campaign('promotion');*

*call sp\_finish\_campaign('roadshow');*

*call sp\_finish\_campaign('reseller');*

*call sp\_finish\_campaign('Marketing');*

*--- check error status*

*call sp\_finish\_campaign('ghadskbbcsjl');*

*call sync\_invoice;*

*commit;*

*select \* from campaign;*

*select \* from invoice;*

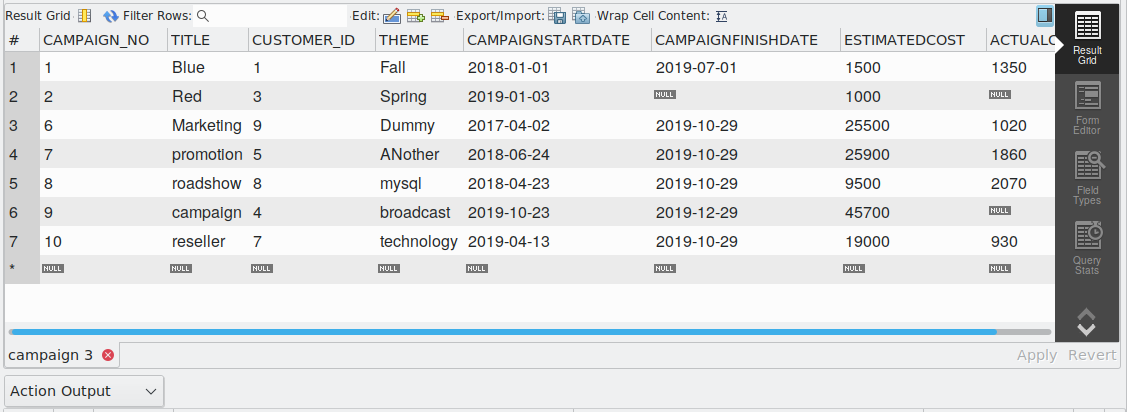
*select \* from workson;*

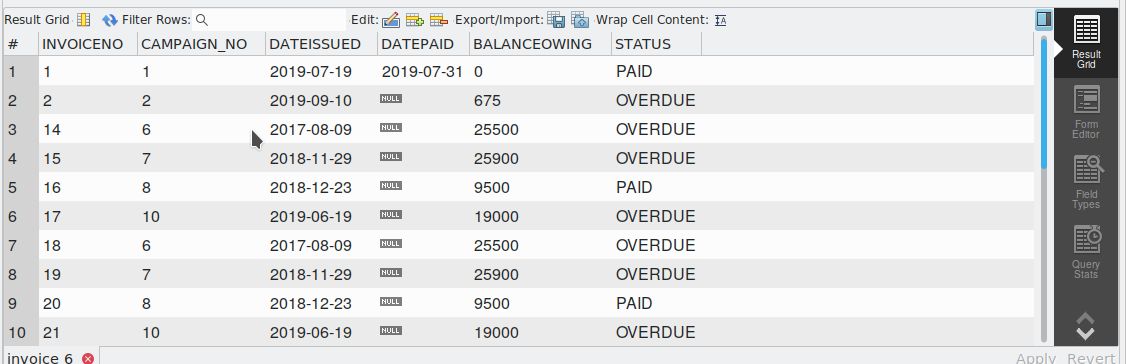
*select \* from customer;*

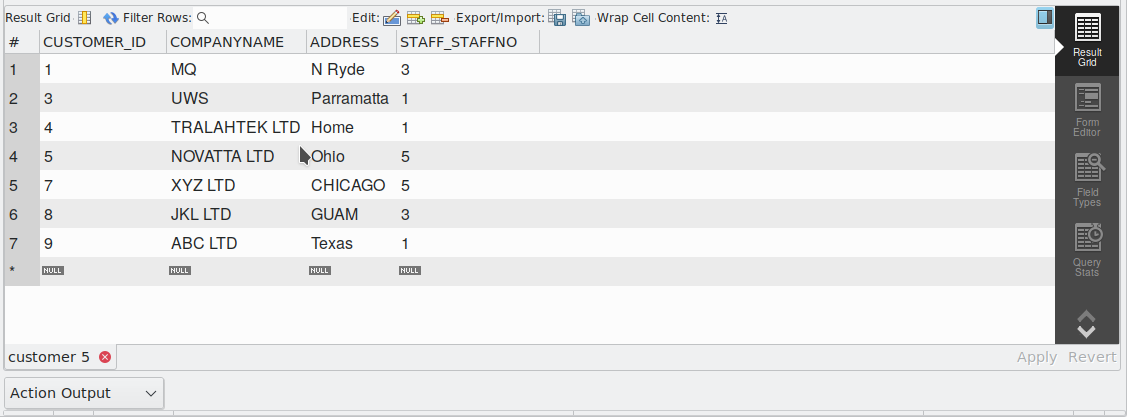
*select \* from alerts;*

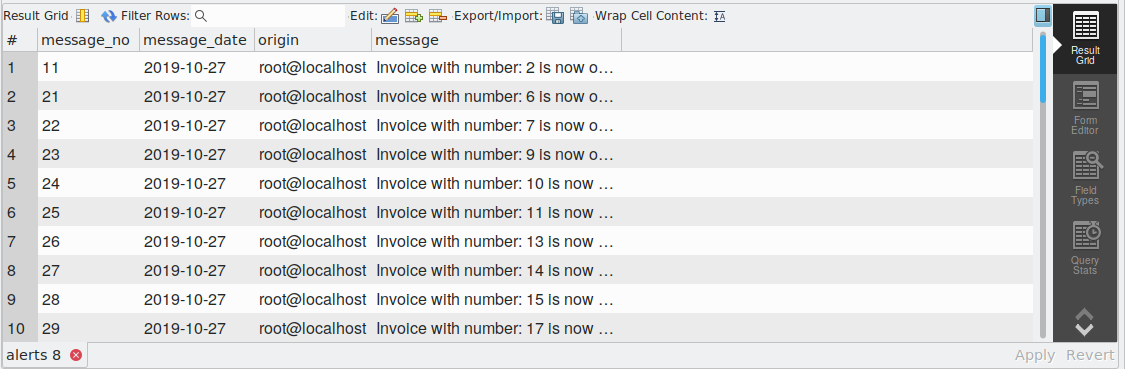
*/\* select \* from invoice; \*/*

*-- Finish with commiting work or rolling back.*



**

**

**

1. **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.*