MACQUARIE
University
SYDNEY·AUSTRALIA

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:

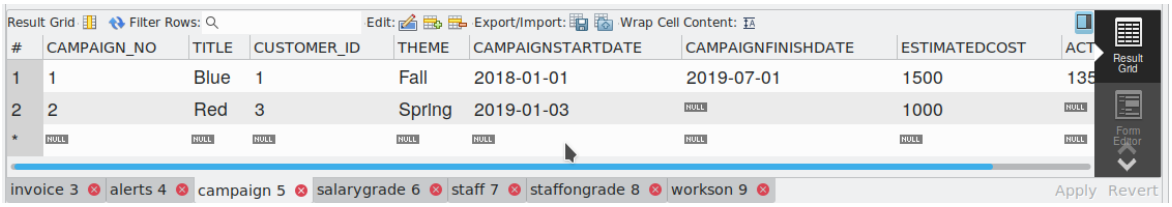
I declare that the work reported here is my own. Any help received, from any person, through discussion or other means, has been acknowledged in the last section of this report.

Student Signature:

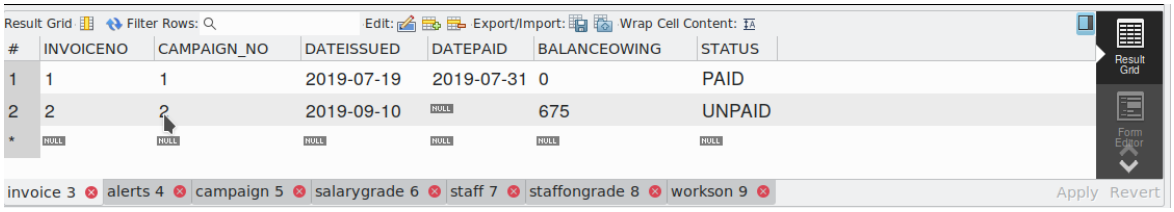
Student Name and Date:

1. Initial State of the database

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

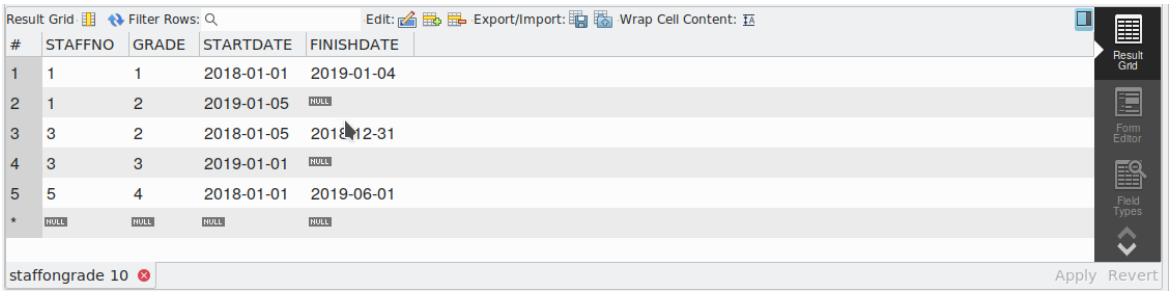


#	CAMPAIGN_NO	TITLE	CUSTOMER_ID	THEME	CAMPAIGNSTARTDATE	CAMPAIGNFINISHDATE	ESTIMATEDCOST	ACT
1	1	Blue	1	Fall	2018-01-01	2019-07-01	1500	135
2	2	Red	3	Spring	2019-01-03	NULL	1000	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

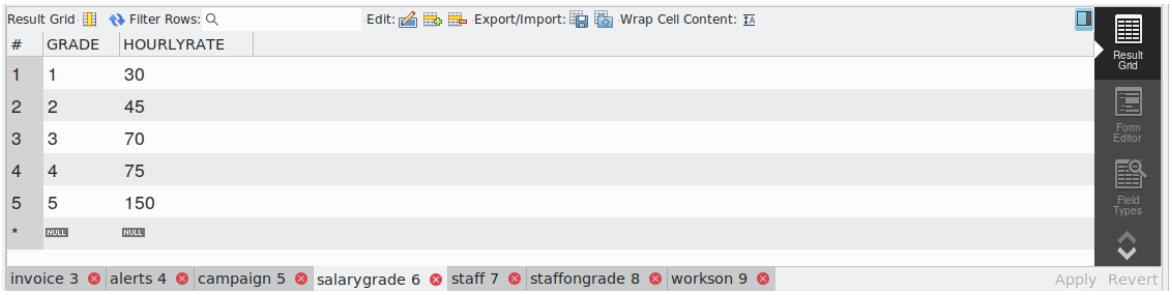


#	INVOICENO	CAMPAIGN_NO	DATEISSUED	DATEPAID	BALANCEOWING	STATUS
1	1	1	2019-07-19	2019-07-31	0	PAID
2	2	2	2019-09-10	NULL	675	UNPAID
*	NULL	NULL	NULL	NULL	NULL	NULL

#	STAFFNO	STAFFNAME	EXPERTISE
1	1	John	NULL
2	3	Adam	NULL
3	5	Eve	NULL
*	NULL	NULL	NULL



#	STAFFNO	GRADE	STARTDATE	FINISHDATE
1	1	1	2018-01-01	2019-01-04
2	1	2	2019-01-05	NULL
3	3	2	2018-01-05	2018-12-31
4	3	3	2019-01-01	NULL
5	5	4	2018-01-01	2019-06-01
*	NULL	NULL	NULL	NULL



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

2. 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 ;
```

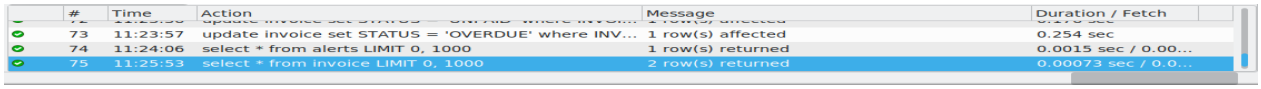
3. Required Testing against Sample Database.

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

-- Create the tables
-- Populate the tables
-- Create the stored objects (procedures/functions/triggers)
-- Then...

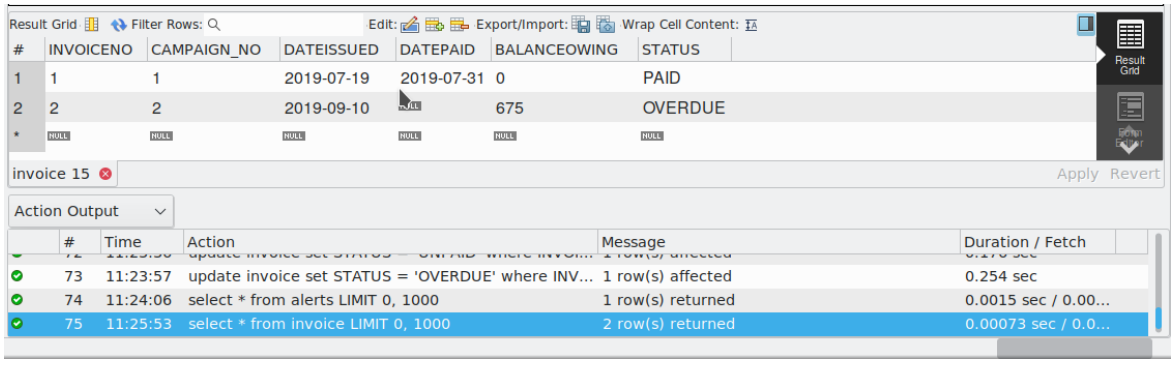
-- Turn off autocommit so you can have better control on what you are doing by rolling back transactions.
set autocommit = 0;

-- 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;



#	Time	Action	Message	Duration / Fetch
73	11:23:57	update invoice set STATUS = 'OVERDUE' where INV...	1 row(s) affected	0.254 sec
74	11:24:06	select * from alerts LIMIT 0, 1000	1 row(s) returned	0.0015 sec / 0.00...
75	11:25:53	select * from invoice LIMIT 0, 1000	2 row(s) returned	0.00073 sec / 0.0...

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



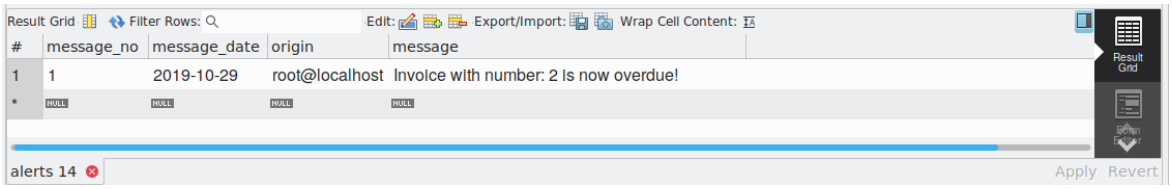
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	INVOICENO	CAMPAIGN_NO	DATEISSUED	DATEPAID	BALANCEOWING	STATUS
1	1	1	2019-07-19	2019-07-31	0	PAID
2	2	2	2019-09-10	NULL	675	OVERDUE
*	NULL	NULL	NULL	NULL	NULL	NULL

invoice 15 | Apply | Revert

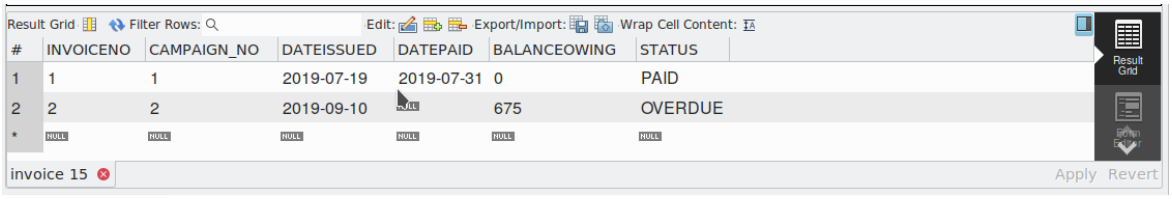
Action Output

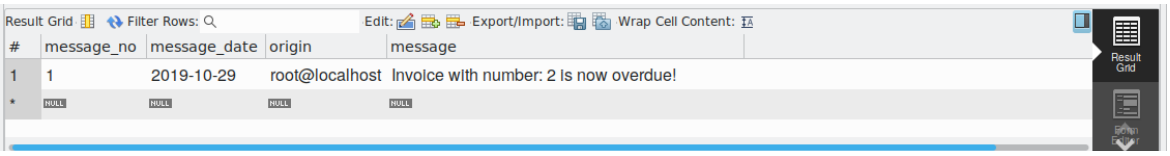
#	Time	Action	Message	Duration / Fetch
73	11:23:57	update invoice set STATUS = 'OVERDUE' where INV...	1 row(s) affected	0.254 sec
74	11:24:06	select * from alerts LIMIT 0, 1000	1 row(s) returned	0.0015 sec / 0.00...
75	11:25:53	select * from invoice LIMIT 0, 1000	2 row(s) returned	0.00073 sec / 0.0...



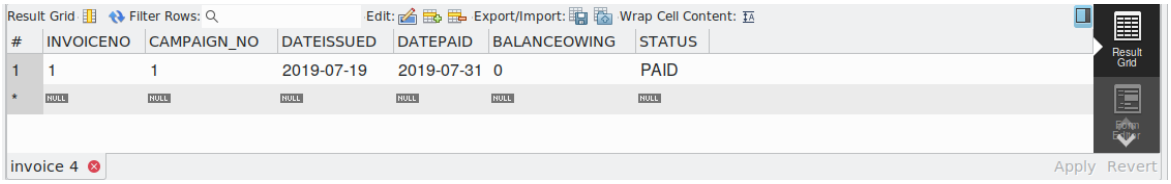
-- Bring DB back to original state; re-check
rollback;
*select * from invoice;*
*select * from alerts;*

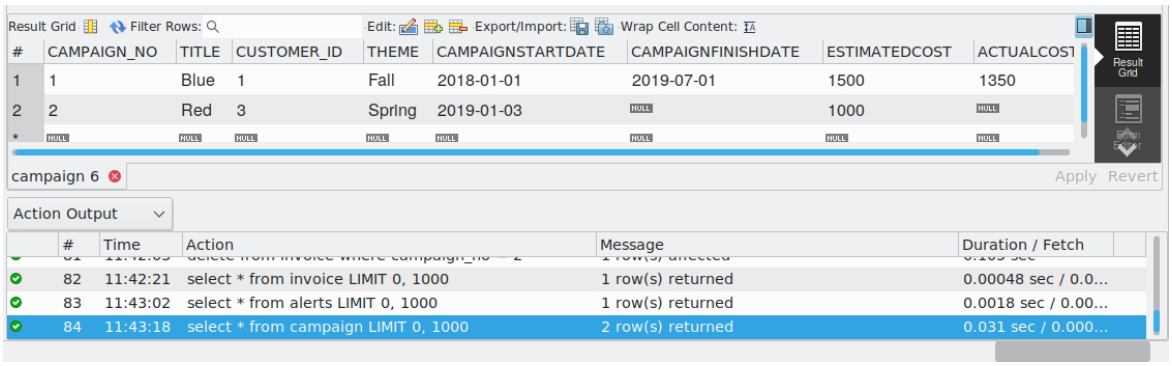
-- Synchronise the invoice table and verify the procedure behaves as desired
call sync_invoice;
*select * from invoice;*
*select * from alerts;*



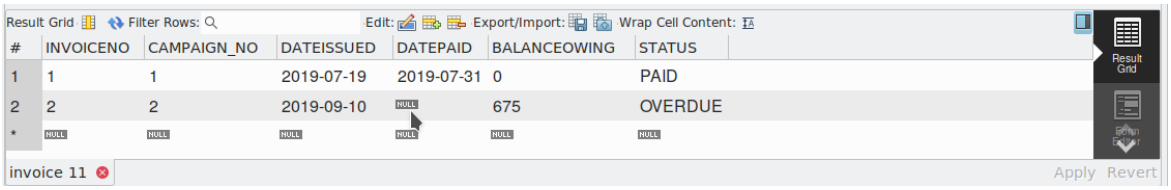


-- Bring DB back to original state; delete campaign# 2; check all relevant tables
rollback;
delete from invoice where campaign_no = 2;
*select * from invoice;*
*select * from alerts;*
*select * from campaign;*
*select * from salarygrade;*
*select * from staff;*
*select * from staffongrade;*
*select * 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;

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).*

-- 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.

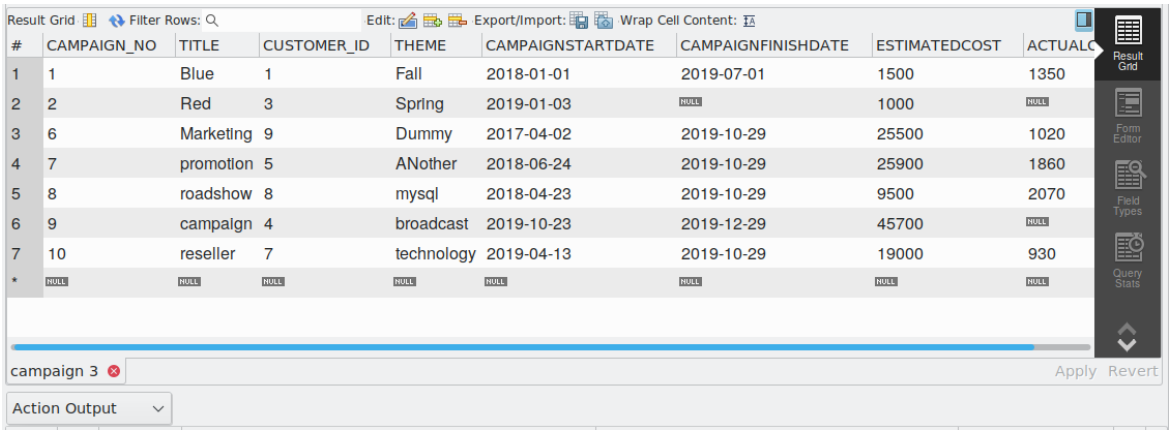
```
insert into customer(CUSTOMER_ID,COMPANYNAME,ADDRESS,STAFF_STAFFNO)
values
(9,"ABC LTD","Texas",1),
(5,"NOVATTA LTD","Ohio",5),
(4,"TRALAHTEK LTD","Home",1),
(8,"JKL LTD","GUAM",3),
(7,"XYZ LTD","CHICAGO",5);
commit;

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.
```



#	CAMPAIGN_NO	TITLE	CUSTOMER_ID	THEME	CAMPAIGNSTARTDATE	CAMPAIGNFINISHDATE	ESTIMATEDCOST	ACTUALC
1	1	Blue	1	Fall	2018-01-01	2019-07-01	1500	1350
2	2	Red	3	Spring	2019-01-03	NULL	1000	NULL
3	6	Marketing	9	Dummy	2017-04-02	2019-10-29	25500	1020
4	7	promotion	5	ANother	2018-06-24	2019-10-29	25900	1860
5	8	roadshow	8	mysql	2018-04-23	2019-10-29	9500	2070
6	9	campaign	4	broadcast	2019-10-23	2019-12-29	45700	NULL
7	10	reseller	7	technology	2019-04-13	2019-10-29	19000	930
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	INVOICENO	CAMPAIGN_NO	DATEISSUED	DATEPAID	BALANCEOWING	STATUS
1	1	1	2019-07-19	2019-07-31	0	PAID
2	2	2	2019-09-10	NULL	675	OVERDUE
3	14	6	2017-08-09	NULL	25500	OVERDUE
4	15	7	2018-11-29	NULL	25900	OVERDUE
5	16	8	2018-12-23	NULL	9500	PAID
6	17	10	2019-06-19	NULL	19000	OVERDUE
7	18	6	2017-08-09	NULL	25500	OVERDUE
8	19	7	2018-11-29	NULL	25900	OVERDUE
9	20	8	2018-12-23	NULL	9500	PAID
10	21	10	2019-06-19	NULL	19000	OVERDUE

invoice 6 | Apply | Revert

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	CUSTOMER_ID	COMPANYNAME	ADDRESS	STAFF_STAFFNO
1	1	MQ	N Ryde	3
2	3	UWS	Parramatta	1
3	4	TRALAHTEK LTD	Home	1
4	5	NOVATTA LTD	Ohio	5
5	7	XYZ LTD	CHICAGO	5
6	8	JKL LTD	GUAM	3
7	9	ABC LTD	Texas	1
*	NULL	NULL	NULL	NULL

customer 5 | Apply | Revert

Action Output

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	message_no	message_date	origin	message
1	11	2019-10-27	root@localhost	Invoice with number: 2 is now o…
2	21	2019-10-27	root@localhost	Invoice with number: 6 is now o…
3	22	2019-10-27	root@localhost	Invoice with number: 7 is now o…
4	23	2019-10-27	root@localhost	Invoice with number: 9 is now o…
5	24	2019-10-27	root@localhost	Invoice with number: 10 is now …
6	25	2019-10-27	root@localhost	Invoice with number: 11 is now …
7	26	2019-10-27	root@localhost	Invoice with number: 13 is now …
8	27	2019-10-27	root@localhost	Invoice with number: 14 is now …
9	28	2019-10-27	root@localhost	Invoice with number: 15 is now …
10	29	2019-10-27	root@localhost	Invoice with number: 17 is now …

alerts 8 | Apply | Revert

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.