

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: Matthew De Masi**

**Student Number: 45585342**

**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.*

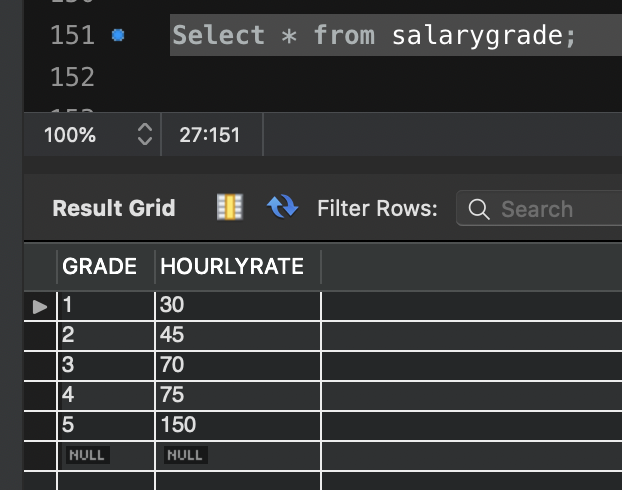
****

Student Signature:

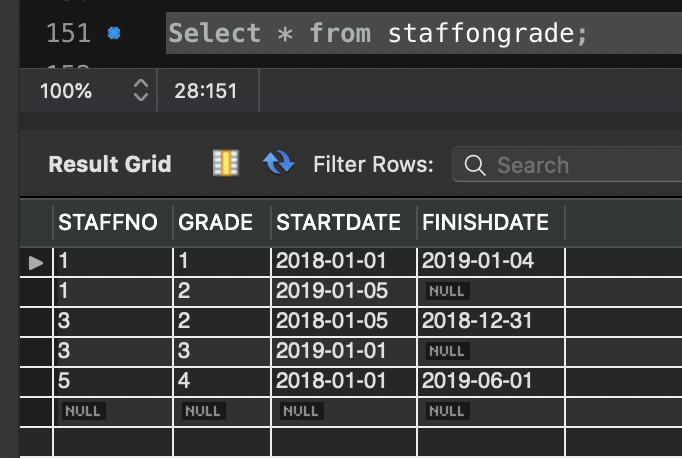
Student Name and Date: Matthew De Masi 23/10/2019

1. **Initial State of the database**

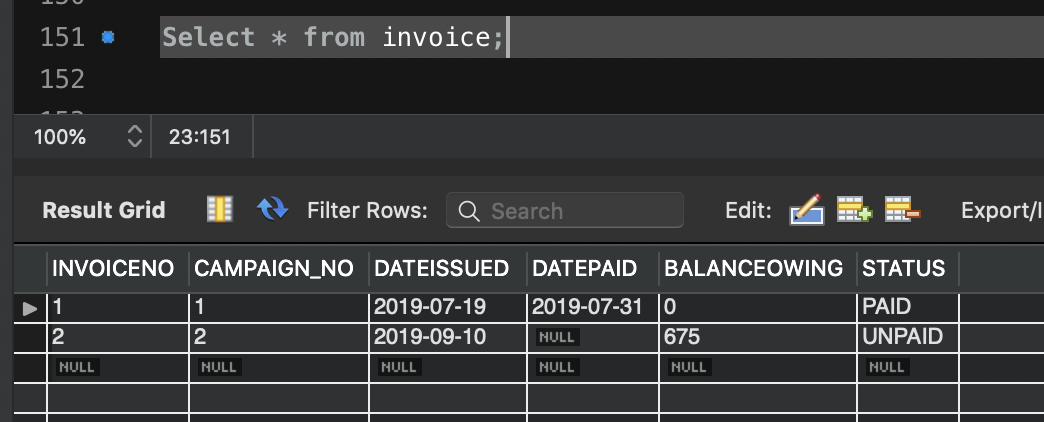
**Salary Table**



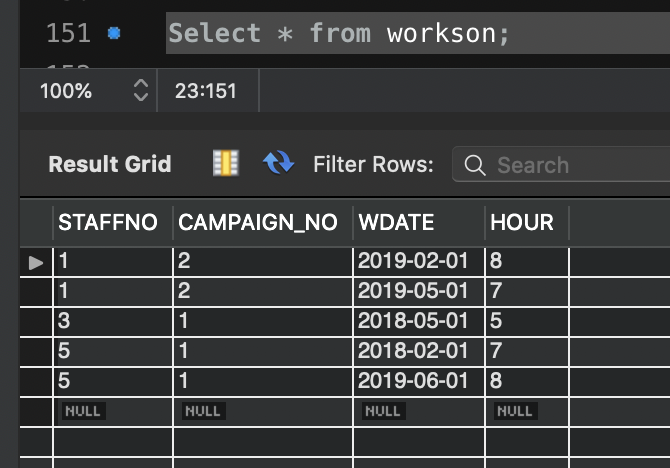
**Staffongrade Table**



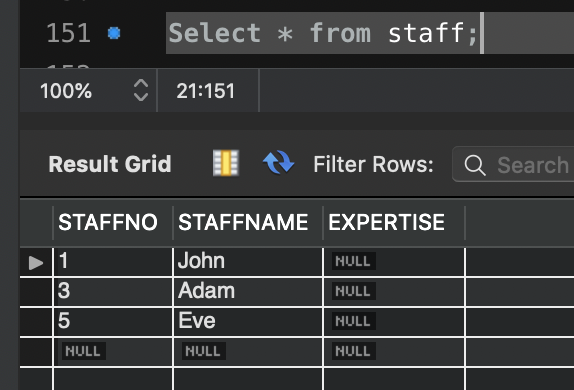
**Salary Table**



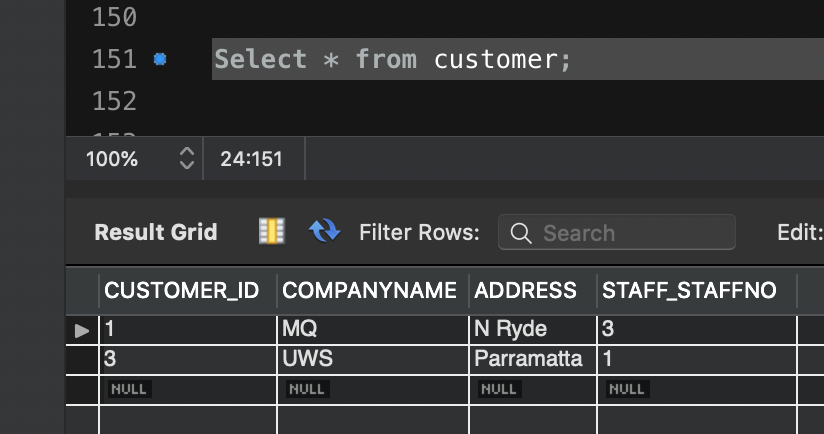
**Workson Table**



**Staff Table**



**Salary Table**



**Campaign Table**



1. **Stored Programs.**

**Trigger Overdue**

delimiter //

drop trigger if exists tr\_overdue

//

create trigger tr\_overdue

-- type of trigger, etc

AFTER UPDATE ON invoice

FOR EACH ROW

begin

-- implementation goes here

IF NEW.status = 'OVERDUE' THEN

INSERT INTO

alerts (message\_no,message\_date,orign,message)

values (current\_date(), current\_user(),CONCAT('Invoice with number: ', OLD.INVOICENO ,'is now overdue!'));

END IF;

end

//

**Rate on date procedure**

drop function if exists rate\_on\_date //

create function rate\_on\_date(staff\_id int, given\_date DATE) returns float

DETERMINISTIC

begin

declare hour\_rate float;

declare start\_date date;

declare finish\_date date;

declare hourly\_rate int default 0;

declare v\_finished int default 0;

declare hourly\_work CURSOR FOR

select hourlyrate, startdate, finishdate

from salarygrade join staffongrade on staffongrade.grade = salarygrade.grade

where staffongrade.staffno = staff\_id;

declare continue handler for not found set v\_finished = 1;

open hourly\_work;

repeat fetch hourly\_work into hourly\_rate, start\_date, finish\_date;

if(given\_date between start\_date and finish\_date)

then set hour\_rate = hourly\_rate;

elseif given\_date >= start\_date and finish\_date is null

then set hour\_rate = hourly\_rate;

end if;

until v\_finished

end repeat;

close hourly\_work;

return hour\_rate;

end //

**Cost of Campaign procedure**

drop function if exists cost\_of\_campaign //

create function cost\_of\_campaign (camp\_id int) returns float

DETERMINISTIC

begin

declare total\_cost float;

select sum(rate\_on\_date(STAFFNO,WDATE)\*hour)

into total\_cost from workson

where camp\_id = CAMPAIGN\_NO;

return total\_cost;

end //

**Sp Finish Campaign procedure**

drop procedure if exists sp\_finish\_campaign //

create procedure sp\_finish\_campaign (in c\_title varchar(30))

begin

declare B varchar(30);

select count(c\_title) into B

from campaign where c\_title = TITLE;

if (B != 1) then signal sqlstate '45000'

set message\_text = 'ERROR! Campaign title does not exist!';

end if;

update campaign

set ACTUALCOST = cost\_of\_campaign (CAMPAIGN\_NO)

where c\_title = TITLE;

update campaign

set CAMPAIGNFINISHDATE = current\_date() where c\_title = TITLE;

end //

Delimiter ;

**Sync Invoice Table**

drop procedure if exists sync\_invoice //

CREATE procedure sync\_invoice()

begin

declare dDate date;

declare stat varchar(20);

declare d\_finished int default 0;

declare d\_array cursor for select DATEISSUED, STATUS from invoice;

declare continue handler for not found set d\_finished = 1;

open d\_array;

repeat

fetch d\_array into dDate, stat;

if (datediff(current\_date(), dDate)> 30 )then

Update invoice Set STATUS = 'OVERDUE' Where datediff(current\_date(), DATEISSUED) > 30 And STATUS = 'UNPAID';

end if;

until d\_finished

end repeat;

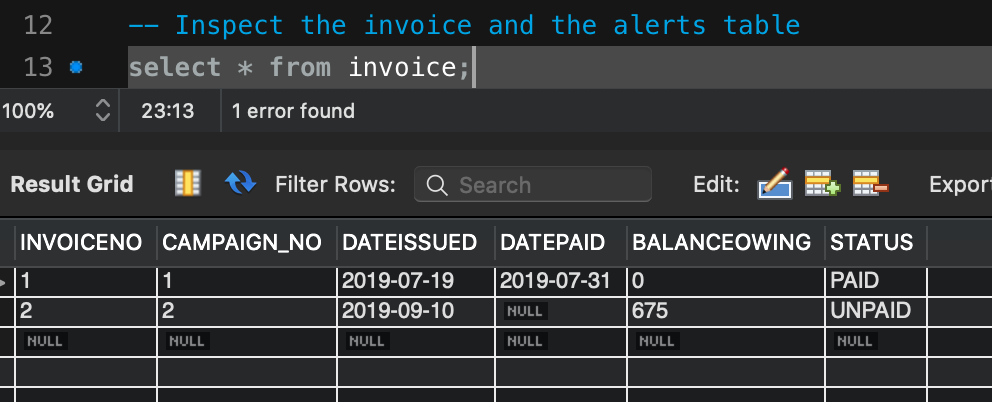
close d\_array;

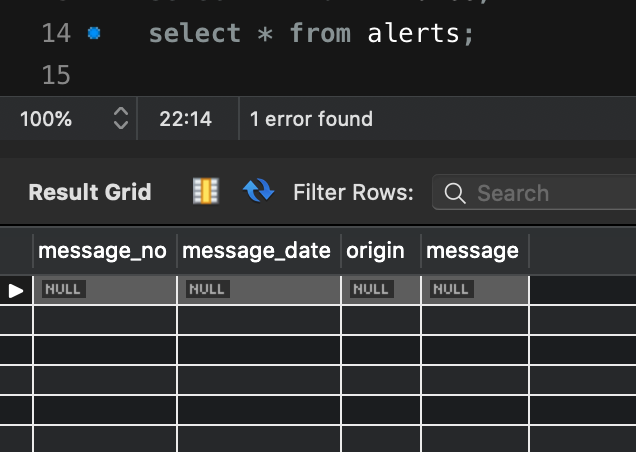
end

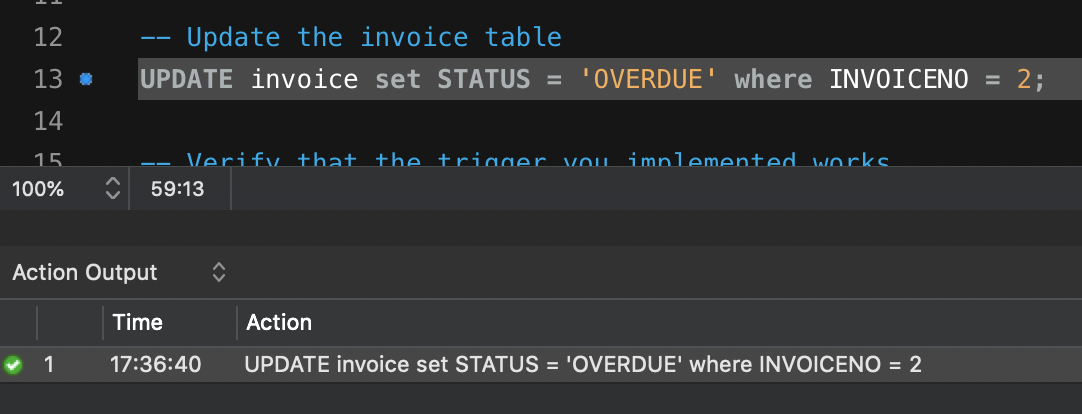
//

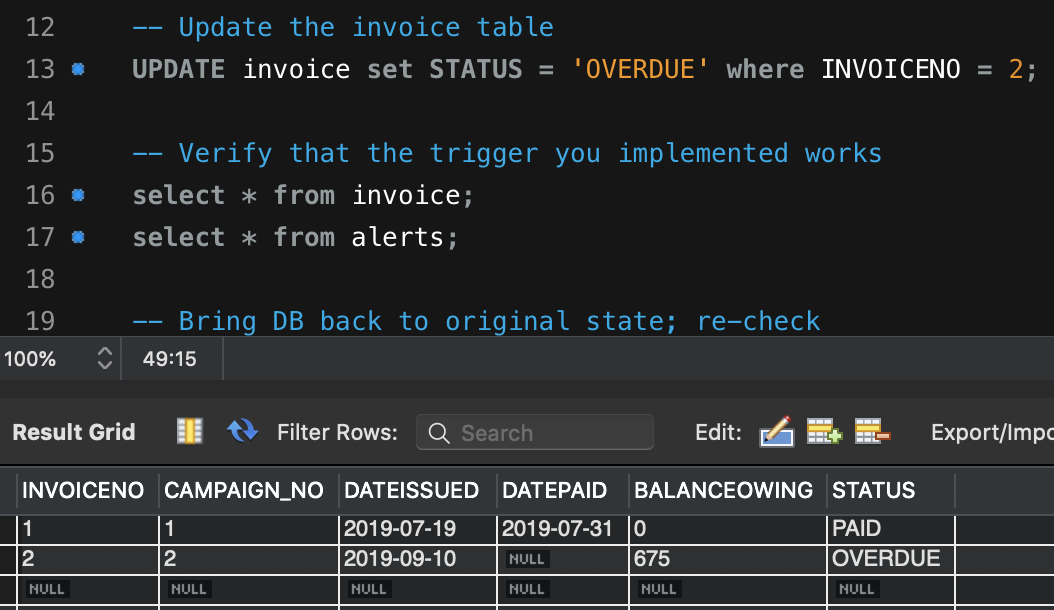
delimiter ;

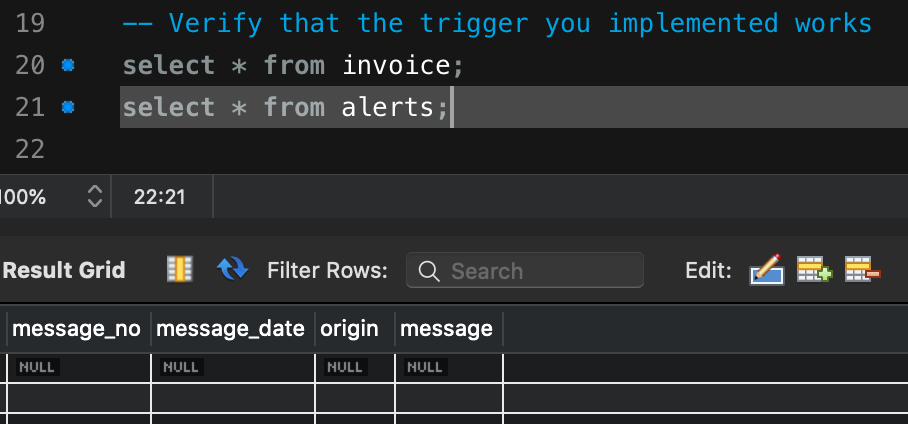
1. **Required Testing against Sample Database.**

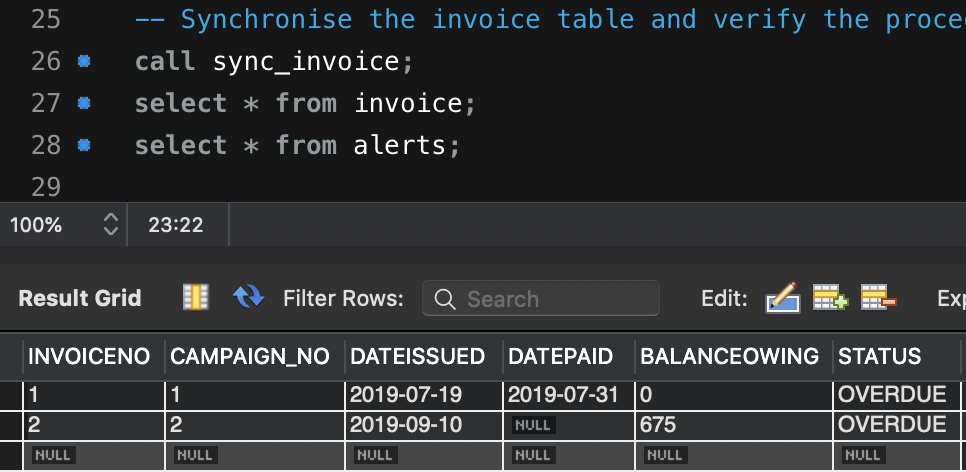
**

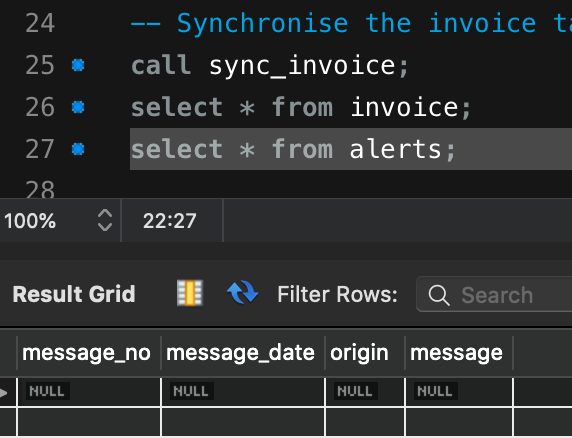
**

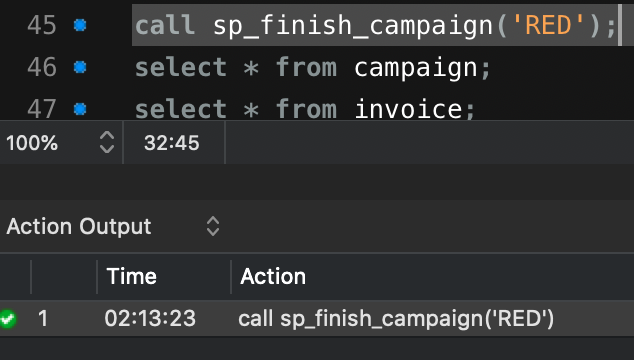
**

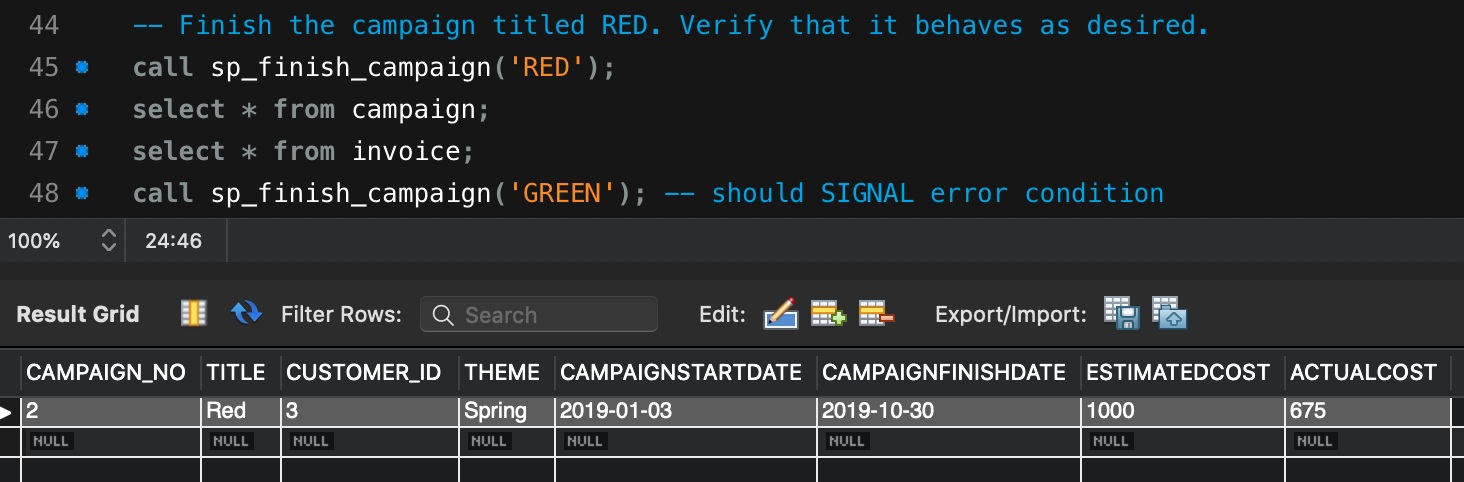
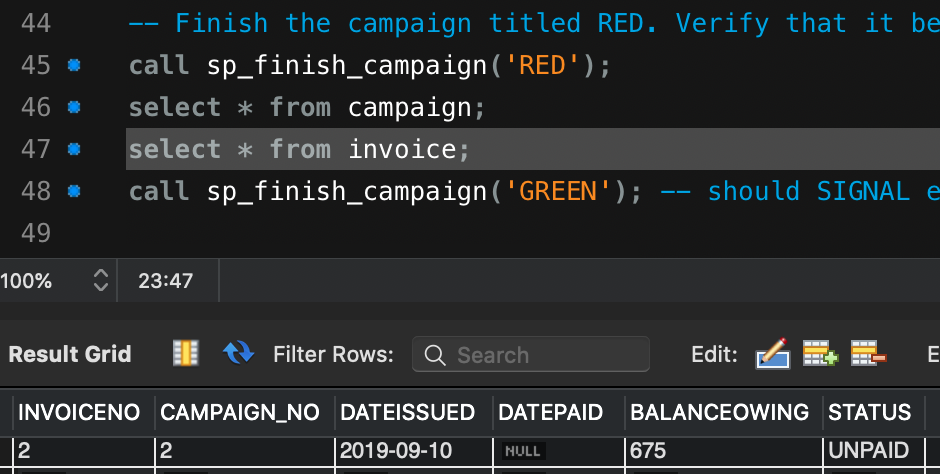
**

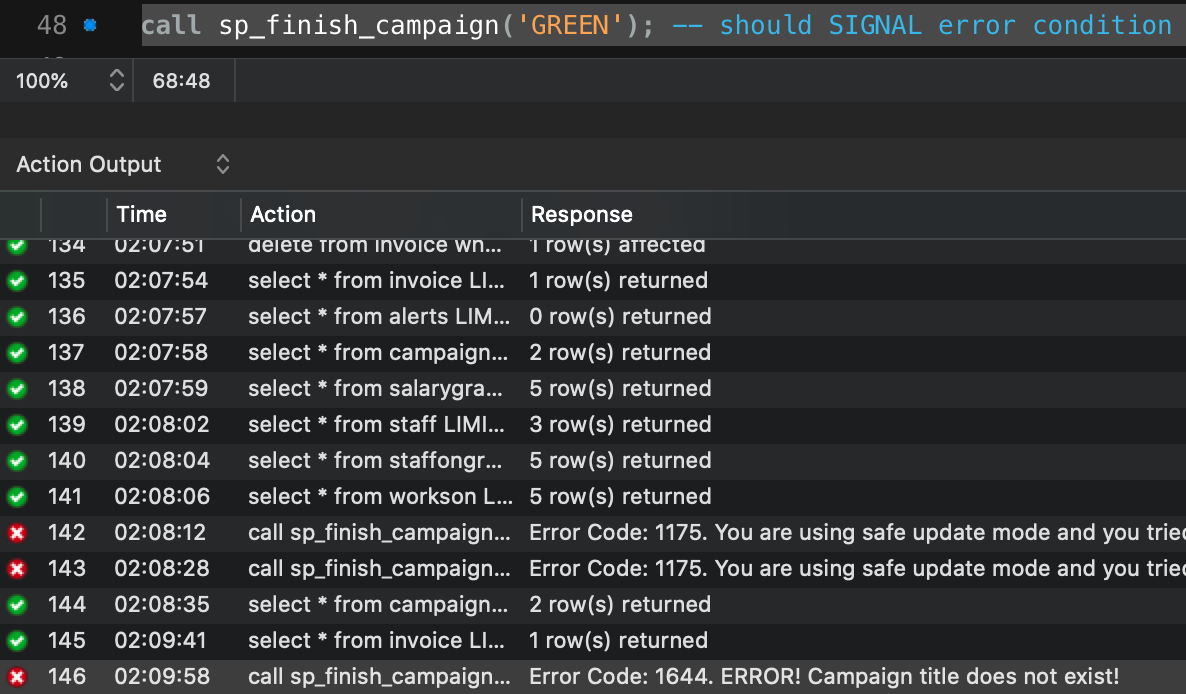


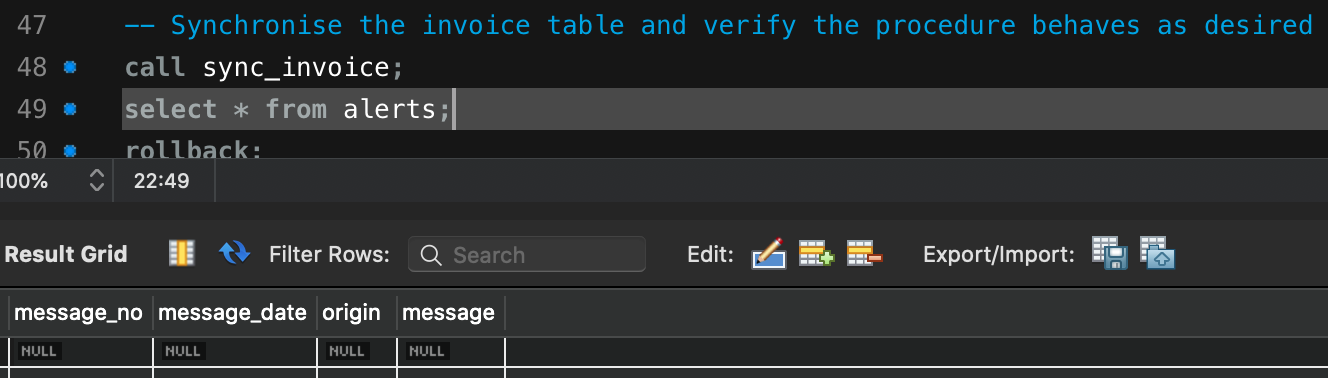
**

**

**

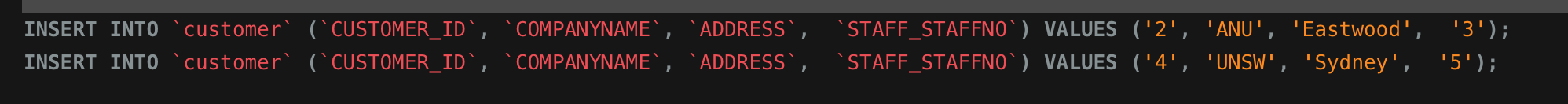
**

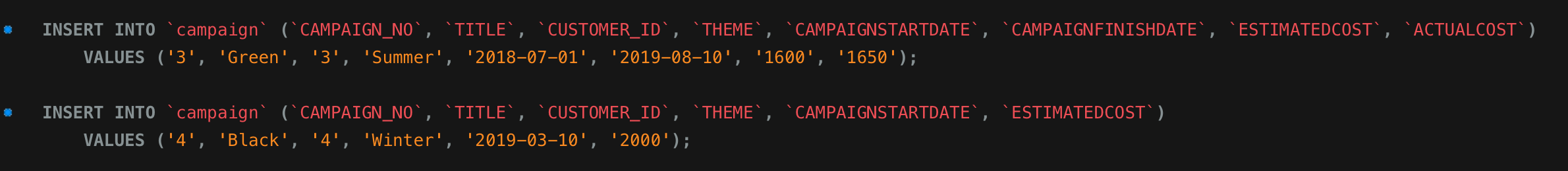
**

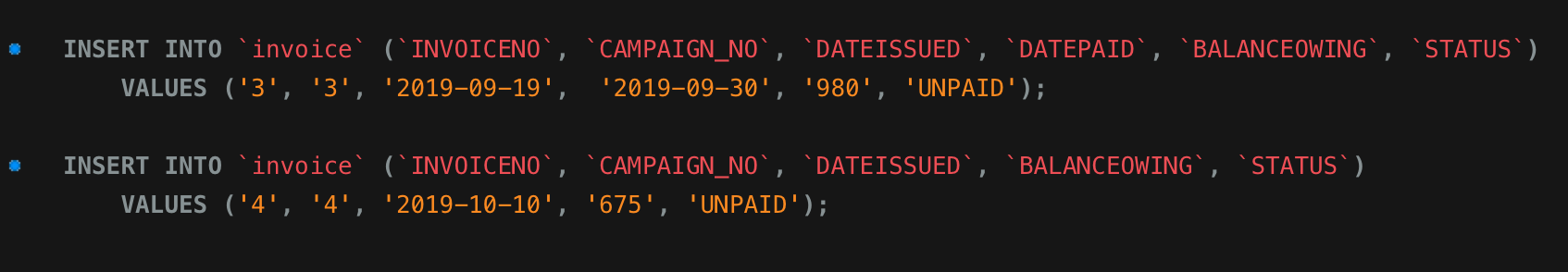
**

1. **More Extensive Testing.**

**Additional Records applied**







**Testing the additional records**

