**aProgramming Lab 3**

In this programming lab, we consider a Mail Order Database with 8 tables.

ZIPCODES(zip, city)

EMPLOYEES(eno, ename, zip, hdate)

PARTS(pno, pname, qoh, price, olevel)

CUSTOMERS(cno, cname, street, zip, phone)

ORDERS(ono, cno, eno, received, shipped)

ODETAILS(ono, pno, qty)

RESTOCK(res\_date, pno)

ORDERS\_ERRORS(Error\_Date, ono, Error\_Msg)

The **eno, pno, cno, ono** are the employee number, part number, customer number, and order number respectively.

**hdate** of in **EMPLOYEES** table means the hiring date of an employee.

**qoh** in the **PARTS** table is the quantity the part on hold.

**olevel** in the **PARTS** table is the number that determines when to restock a part. For example, given **olevel**=20, we need to restock this part to 2\*olevel when its qoh < 20.

The **received** and **shipped** are the dates that the order is received and shipped.

**qty** in the **ODETAILS** table is the quantity for the product for an order.

In the **RESTOCK** table, **res\_date** is the date when a restock is requested.

**ORDERS\_ERRORS** is a table to log order related error messages.

**Programming Tasks**

**Task1(**5 points**):** Create tables with scripts provided in ‘Create-Table.sql’ and load data into these tables with scripts provided in ‘Load-Data.sql’. Please read these provided scripts carefully before executing them.

**What to submit**: No submission is required

**Task2(**15 points**)**: There is a procedure “**add\_order**” in the file **p2\_task2.sql**. However, this procedure has syntax errors. In this task, you need to find and fix these syntax errors.

**What to submit**:

1. Point out where are errors for the procedure, and your SQL scripts for the correct procedure.
2. Call the “**add\_order**” procedure with the following parameters (666,1,3,null,null) as input. Submit your screenshot for query select \* from ORDERS after executing this procedure call.

Delimiter $$

create procedure add\_order(

IN onum int,

IN cnum int,

IN receive date

)

begin

DECLARE employee\_name varchar(255);

if (receive is null)

insert into ORDERS values (onum,cnum,enum,CURDATE(),null);

else then

insert into ORDERS values (onum,cnum,enum,receive,null);

end if;

end;

$$

delimiter ||

CREATE PROCEDURE add\_order(

IN onum INT,

IN cnum INT,

IN enum INT,

IN receive DATE

)

BEGIN

DECLARE employee\_name VARCHAR(255);

IF (receive IS NULL) THEN

INSERT INTO ORDERS VALUES (onum, cnum, enum, CURDATE(), null);

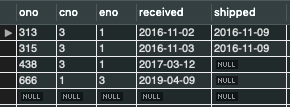
ELSE

INSERT INTO ORDERS VALUES (onum, cnum, enum, CURDATE(), null);

END IF;

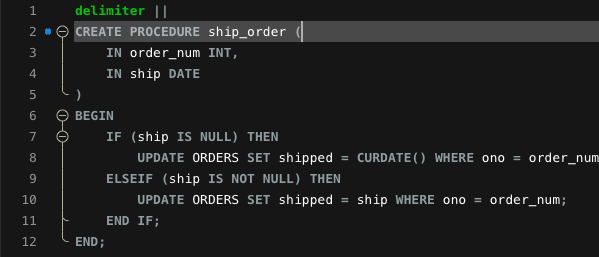
END;

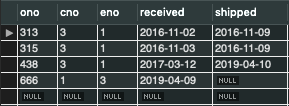
||



**cTask3(**10 points**)**: Create a procedure “**ship\_order**” with order number and shipping date as input. If the shipping date in the procedure call is null, use the current date as the shipping date.

**What to submit**: Your scripts for the “**ship\_order**” procedure. The screen of select \* from ORDERS after executing the procedure with (438, null) as input.





**Task4(**35 points**)**: Create a new **function** “**cancel\_order**”. The **cancel\_order** procedure will take the order number as input. In this function, we will delete rows in tables **ORDERS** and **ODETAILS** with ono equal to the order number input by the function caller. The function should return a message to indicate the order cancellation status as follows:

1. The order is removed, but order details do not exist, return “Order Details Do Not Exist! Order Is Cancelled Successfully!”
2. Both order and order details do not exist, return “Order Details Do Not Exist! Order Does Not Exist!”
3. Both order and order details are removed, return “Order Details Removed Successfully! Order Is Cancelled Successfully!”

**Note that:** You can use CONCAT() function for the concatenation of multiple strings. Pay attention to the Foreign key restriction exist between table **ORDERS** and **ODETIALS**.

**What to submit**:

1. Your scripts of “**cancel\_order**” function.
2. Screenshot of function calls for “**cancel\_order**” as

SET @x=cancel\_order(315);

SELECT @x;

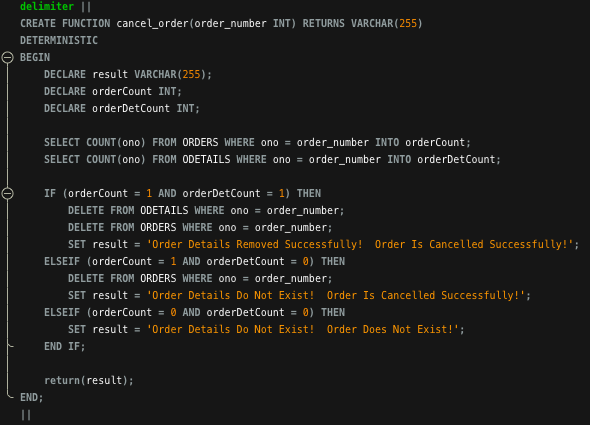
SET @x=cancel\_order(666);

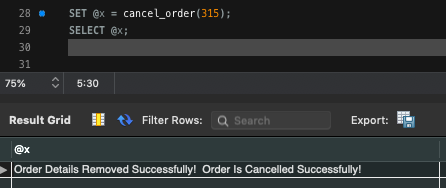
SELECT @x;

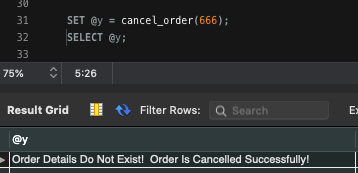
1. The screenshot for the following query after the above two function calls.

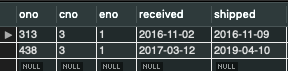
SELECT \* FROM ORDERS;

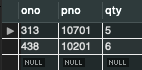
SELECT \* FROM ODETAILS;











**Task5**(35 points) Create a procedure “**add\_order\_details**” with order number, part number, and quantity as input. The procedure shall satisfy the following requirements:

1. If the quantity required by the order is greater than quantity on hold, insert an error message into the **ORDERS\_ERRORS** table with an error message as “Do not have enough quantity to sell!”. At the same time, the order shall be cancelled by calling the “**cancel\_order**” you created in task 2. Note that, “**cancel\_order**” shall be invoked directly inside the body of “**add\_order\_details**”.
2. If the quantity required by the order is less than quantity on hold, reduce the quantity on hold for the part in the PARTS table, and insert a record into the **ODETAILS** table. Then, if the remained quantity on hold for the part is less than the **olevel**, insert a restock record into the **RESTOCK** table, and update the quantity of the product to the double of the value of olevel this product.

**What to submit**:

Your scripts of “**add\_order\_details**” procedure

