Introduction to Databases - CECS535
Course Project - Fall 2012

WHAT TO TURN IN: a printout with all your SQL statement
(CREATE, SELECT, triggers, procedures, etc.).

1. Create the following tables:
   - **Customers**, with attributes cid, cname, address, age, income-level, username, password, where cid is an identification number and primary key. cid should be generated by the system.
   - **Publisher**, with attributes publisherid, name, address, discount, where publisherid is an identification number (key).
   - Books, with attributes isbn, title, author, qty in stock, price, cost, year published, publisherid, where isbn is the key and publisherid is a foreign key.
   - **Orders**, with attributes ordernum, cid, cardnum, cardmonth, cardyear, order date, ship date, where ordernum is a key and should be system generated, and cid is a foreign key.
   - **OrderList**, with attributes ordernum, isbn, quantity, where (ordernum, isbn) is the primary key and both ordernum and isbn are foreign keys.
   - **StockManage**r with attributes isbn, quantity, where isbn is both a primary key and a foreign key.

2. Add as much as you can of the following information to the database schema using checks, assertions or triggers (you can modify tables if already created to add a check).
   (a) All customer ages should be between 18 and 100.
   (b) All publisher discounts should be between 1.00 and 10.00
   (c) All cardyear values should be greater than 2010.
   (d) Each ship date should be later (temporarily) than the respective order date.
   (e) Each book price should be higher than the respective cost. Both price and cost should be greater than zero.

Sql queries for 1 & 2 on the following page.

```
CREATE TABLE wfmoor01_Customers (
      cid SERIAL PRIMARY KEY,
      cname VARCHAR(40),
      address VARCHAR(80),
      age INTEGER CHECK (age >= 18 and age <= 100),
      income_level NUMERIC(10,2),
      username VARCHAR(20),
      pass_word VARCHAR(20)
);

CREATE TABLE wfmoor01_Publisher (
      publisherid VARCHAR(40) PRIMARY KEY,
      name VARCHAR(40),
      address VARCHAR(40),
      discount NUMERIC(4,2) CHECK (discount >= 1.00 and discount <= 10.00)
);

CREATE TABLE wfmoor01_Books (
      isbn VARCHAR(40) PRIMARY KEY,
      title VARCHAR(40),
      author VARCHAR(40),
      qty_in_stock INTEGER,
      price NUMERIC(10,2) CHECK (price > 0),
      cost NUMERIC(10,2) CHECK (price > 0),
      year_published NUMERIC(4),
      publisherid VARCHAR(40) REFERENCES wfmoor01_Publisher(publisherid)
      CONSTRAINT con2 CHECK (price > cost)

);

CREATE TABLE wfmoor01_Orders (
      ordernum SERIAL PRIMARY KEY,
      cid INTEGER REFERENCES wfmoor01_Customers(cid),
      cardnum varchar(40),
      cardmonth NUMERIC(2) CHECK (cardmonth >= 1 and cardmonth <= 12),
      cardyear NUMERIC(4) CHECK (cardyear > 2010),
      order_date DATE,
      ship_date DATE
      CONSTRAINT con1 CHECK (ship_date > order_date)
);
```

```
CREATE TABLE wfmoor01_OrderList (
      ordernum INTEGER REFERENCES wfmoor01_Orders(ordernum),
      isbn VARCHAR(40) REFERENCES wfmoor01_Books(isbn),
      quantity INTEGER,
      CONSTRAINT ordernum_isbn PRIMARY KEY(ordernum, isbn)

);


CREATE TABLE wfmoor01_StockManager (
      isbn VARCHAR(40) PRIMARY KEY REFERENCES wfmoor01_Books(isbn),
      quantity INTEGER
);
```

3. Insert at least two tuples into each relation (you can make up the values, but they should be valid data, i.e. respecting all constraints).

Code and examples below and on the following pages.

```
insert into wfmoor01_customers (cname, address,  age, income_level, username, pass_word) values ('Wendell Moore',  'PO Box 384', 46, 60000.00, 'delmer0818', 'wfmoor01');

insert into wfmoor01_customers (cname, address,  age, income_level, username, pass_word) values ('Amanda Pierce',  '345 Elm St. #3', 35, 47000.00, 'naners0823', 'ap567890');

Select * from wfmoor01_customers;
```

SQL Editor | Graphical Query Builder

Previous queries

```
select * from wfmoor01_customers;
```

Output pane

Data Output | Explain | Messages | History

	cid integer	cname character varying(40)	address character varying(80)	age integer	income_level numeric(10,2)	username character varying(20)	pass_word character varying(20)
1	2	Wendell Moore	PO Box 384	46	60000.00	delmer0818	wfmoor01
2	3	Amanda Pierce	345 Elm St. #3	35	47000.00	naners0823	ap567890

```
insert into wfmoor01_publisher (publisherid, name, address, discount)
values ('RH-0001', 'Random House', '123 E. Main St.', 5.00);

insert into wfmoor01_publisher (publisherid, name, address, discount)
values ('SM-0099', 'SAMS', '800 E 96th St.', 7.50);

Select * from wfmoor01_publisher;
```

SQL Editor | Graphical Query Builder

Previous queries

```
select * from wfmoor01_publisher;
```

Output pane

Data Output | Explain | Messages | History

	publisherid character varying(40)	name character varying(40)	address character varying(40)	discount numeric(4,2)
1	RH-0001	Random House	123 E. Main St.	5.00
2	SM-0099	SAMS	800 E 96th St.	7.50

```
insert into wfmoor01_books (isbn, title, author, qty_in_stock, price, cost, year_published, publisherid)
values ('0-672-32793-8', 'PSQL 101', 'John Smith', 23, 45, 40, 2002, 'RH-0001');

insert into wfmoor01_books (isbn, title, author, qty_in_stock, price, cost, year_published, publisherid)
values ('0-980-32627-0', 'PSQL in 24 Hours', Jane Doe', 3, 38.95, 19.67, 2012, 'SM-0099');

insert into wfmoor01_books (isbn, title, author, qty_in_stock, price, cost, year_published, publisherid)
values ('0-867-12345-7', 'PSQL FOR DUMMIES', 'John Doe', 5, 35, 30, 2012, 'RH-0001');

Select * from wfmoor01_books;
```

SQL Editor | Graphical Query Builder

Previous queries

```
Select * from wfmoor01_books;
```

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-980-32627-0	PSQL in 24 Hours	Jane Doe	3	38.95	31.45	2012	SM-0099
2	0-672-32793-8	PSQL 101	John Smith	23	45.00	40.00	2002	RH-0001
3	0-867-12345-7	PSQL FOR DUMMIES	John Doe	5	35.00	30.00	2012	RH-0001

```
insert into wfmoor01_orders (cid, cardnum, cardmonth, cardyear, order_date, ship_date) values (2,
'8888999911112222', 8, 2012, '8/18/2012', '12/25/2012');

insert into wfmoor01_orders (cid, cardnum, cardmonth, cardyear, order_date, ship_date) values (3,
'7777999966660000', 12, 2011, '9/9/2012', '1/1/2013');

Select * from wfmoor01_orders;
```

SQL Editor | Graphical Query Builder | Execute query, write result to file

Previous queries

```
select * from wfmoor01_orders;
```

Output pane

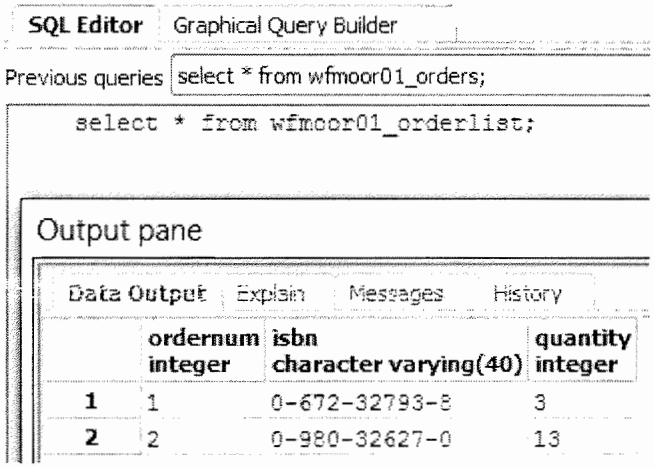
Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-18	2012-12-25
2	2	3	7777999966660000	12	2011	2012-09-09	2013-01-01

```
insert into wfmoor01_orderlist (ordernum, isbn, quantity) values (1, '0-672-32793-8', 3);

insert into wfmoor01_orderlist (ordernum, isbn, quantity) values (2, '0-980-32627-0', 13);

Select * from wfmoor01_orderlist;
```



```
insert into wfmoor01_stockmanager (isbn, quantity) values ('0-672-32793-8', 23);

insert into wfmoor01_stockmanager (isbn, quantity) values ('0-980-32627-0', 3);

Select * from wfmoor01_stockmanager;
```

SQL Editor | Graphical Query Builder

revious queries

```
select * from wfmoor01_stockmanager;
```

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	quantity integer
1	0-672-32793-8	23
2	0-980-32627-0	3

4. In the following, placing an order refers to a transaction where one row is added to the Orders table, together with one or more rows in the Orderlist table, specifying the components of that order. Note that all rows in Orderlist will have the same Orderlist.ordernum, which will be the value in Orders.ordernum.

(a) Create a trigger or stored procedure such that, when an order is placed and a book b is ordered, the quantity ordered is discounted from the quantity in stock for b in Books. If you end up with a negative number or zero,

i. if there is no tuple for this book in StockManager, insert there a tuple with values b's isbn and n, where n is 10 if the quantity in stock is zero, and 10 + (□ x), if the quantity in stock falls to x < 0.

ii. if there already is a tuple for this book in StockManager, with associated quantity m, modify the existing tuple so that m is changed to m + n, where n is as before.

Code for 4a on the following pages.

```
create or replace function get_order1 (book_isbn varchar(40), qty_ordered integer, custid integer, cardno
varchar(40), cardmnth numeric(2), cardyr numeric(4), date_order date, date_ship date)
returns text as $$
declare
      var_qty_in_stock integer;
      var_new_qty_in_stock integer;
      var_qty_ordered integer;
      var_price numeric(10,2);
      var_order_num integer;
      var_temp varchar(40);

BEGIN
      var_qty_ordered := $2;

      -- get book price and quantity in stock
      SELECT qty_in_stock, price INTO var_qty_in_stock, var_price FROM wfmoor01_books WHERE isbn = $1;

        var_new_qty_in_stock := var_qty_in_stock - var_qty_ordered;

        -- insert data into orders table
        insert into wfmoor01_orders (cid, cardnum, cardmonth, cardyear, order_date, ship_date) values ($3,
$4, $5, $6, $7, $8);

        -- insert data into orderlist
        insert into wfmoor01_orderlist (ordernum, isbn, quantity) values (lastval(), $1, $2);

      if var_new_qty_in_stock > 0 then  --quantity greater than 0.

            -- update books qty
            Update wfmoor01_books set qty_in_stock = var_new_qty_in_stock where isbn = $1;

            -- check for book in stockmanager
            select isbn into var_temp from wfmoor01_stockmanager where isbn = $1;

            if var_temp is not null then
                  -- if book exist, update stockmanager qty
                Update wfmoor01_stockmanager set quantity = var_new_qty_in_stock where isbn = $1;
            else
                  -- if book doesn't exist, insert a new book and qty in stockmanager
                insert into wfmoor01_stockmanager (isbn,quantity) values ($1, var_new_qty_in_stock);
            end if;
```

```
      else  --quantity less than or equal to 0.

            var_new_qty_in_stock := 10 + var_new_qty_in_stock;

            LOOP
                  IF var_new_qty_in_stock > 0 THEN
                        EXIT;  -- exit loop
                  else
                        var_new_qty_in_stock := 10 + var_new_qty_in_stock;
                  END IF;
            END LOOP;

            -- update books qty
            Update wfmoor01_books set qty_in_stock = var_new_qty_in_stock where isbn = $1;

            -- check for book in stockmanager
            select isbn into var_temp from wfmoor01_stockmanager where isbn = $1;

            if var_temp is not null then
                    -- if book exist, update stockmanager qty
                  Update wfmoor01_stockmanager set quantity = var_new_qty_in_stock where isbn = $1;
            else
                    -- if book doesn't exist, insert a new book and qty in stockmanager
                  insert into wfmoor01_stockmanager (isbn,quantity) values ($1, var_new_qty_in_stock);
            end if;

      end if;

        RETURN 'Order Complete';
END;

$$ language plpgsql;
```

Examples for 4a on the following pages.

Example1.

```
select get_order1 ('0-867-12345-7', 17, 3, '7777999966660000', 12, 2011, '12/2/2012', '1/1/2013');
```

For book 0-867-12345-7, qty odered is 17 by customer id 3. Card information, order and ship dates are included.

Original qty in stock for this book, per the "Books" table was 5. New qty in stock is now -12. Per stored procedure, 10 is added to the qty in stock until the amount is greater than 0. New qty in stock now equals 8. Also per the store procedure, since no row existed for this book in stock manager table a new row is added.

Tables after running stored procedure.

Wfmoor01_books:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-980-32627-0	PSQL in 24 Hours	Jane Doe	3	38.95	31.45	2012	SM-0099
2	0-672-32793-8	PSQL 101	John Smith	23	45.00	40.00	2002	RH-0001
3	0-867-12345-7	PSQL FOR DUMMIES	John Doe	8	35.00	30.00	2012	RH-0001

Wfmoor01_orders:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-1	2012-12-
2	2	3	7777999966660000	12	2011	2012-09-0	2013-01-
3	18	3	7777999966660000	12	2011	2012-12-0	2013-01-

Wfmoor01_orderlist:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	isbn character varying(40)	quantity integer
1	1	0-672-32793-8	3
2	2	0-980-32627-0	13
3	18	0-867-12345-7	17

Wfmoor01_stockmanager:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	quantity integer
1	0-672-32793-8	23
2	0-980-32627-0	3
3	0-867-12345-7	8

Example 2.

```
select get_order1 ('0-980-32627-0', 3, 2, '8888999911112222', 8, 2012, '11/302/2012', '12/31/2012');
```

For this book, qty odered is 3 and original qty in stock is 3. New qty in stock is now 0. Per stored procedure, 10 is added to the qty in stock until the amount is greater than 0. New qty in stock now equals 10.

Tables after running stored procedure.

Books:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-672-32793-8	PSQL 101	John Smith	23	45.00	40.00	2002	RH-0001
2	0-867-12345-7	PSQL FOR DUMMIES	John Doe	8	35.00	30.00	2012	RH-0001
3	0-980-32627-0	PSQL in 24 Hours	Jane Doe	10	38.95	31.45	2012	SM-0099

Orders:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-1	2012-12-
2	2	3	7777999966660000	12	2011	2012-09-0	2013-01-
3	18	3	7777999966660000	12	2011	2012-12-0	2013-01-
4	19	2	8888999911112222	8	2012	2012-11-3	2012-12-

Orderlist:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	isbn character varying(40)	quantity integer
1	1	0-672-32793-8	3
2	2	0-980-32627-0	13
3	18	0-867-12345-7	17
4	19	0-980-32627-0	3

Stockmanager:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	quantity integer
1	0-672-32793-8	23
2	0-867-12345-7	8
3	0-980-32627-0	10

Example 3.

```
select get_order1 ('0-980-32627-0', 3, 2, '8888999911112222', 8, 2012, '11/30/2012', '12/25/2012');
```

Qty odered is 3. Original qty in stock is 10. New qty in stock is now 7.

Tables after running stored procedure.

Books:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-672-32793-8	PSQL 101	John Smith	23	45.00	40.00	2002	RH-0001
2	0-867-12345-7	PSQL FOR DUMMIES	John Doe	8	35.00	30.00	2012	RH-0001
3	0-980-32627-0	PSQL in 24 Hours	Jane Doe	7	38.95	31.46	2012	SM-0099

Orders:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-18	2012-12-25
2	2	3	7777999966660000	12	2011	2012-09-09	2013-01-01
3	18	3	7777999966660000	12	2011	2012-12-02	2013-01-01
4	19	2	8888999911112222	8	2012	2012-11-30	2012-12-31
5	20	2	8888999911112222	8	2012	2012-11-30	2012-12-25

Orderlist:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	isbn character varying(40)	quantity integer
1	1	0-672-32793-8	3
2	2	0-980-32627-0	13
3	18	0-867-12345-7	17
4	19	0-980-32627-0	3
5	20	0-980-32627-0	3

Stockmanager:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	quantity integer
1	0-672-32793-8	23
2	0-867-12345-7	8
3	0-980-32627-0	7

Example 4.

```
select get_order1 ('0-672-32793-8', 13, 2, '7777999966660000', 12, 2011, '10/31/2012', '11/01/2012');
```

Qty odered = 13. Original qty in stock = 23. New qty in stock is now 10.

Tables after running Stored Procedure.

Books:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-867-12345-7	PSQL FOR DUMMIES	John Doe	8	35.00	30.00	2012	RH-0001
2	0-980-32627-0	PSQL in 24 Hours	Jane Doe	7	38.95	31.45	2012	SM-0099
3	0-672-32793-8	PSQL 101	John Smith	10	45.00	40.00	2002	RH-0001

Orders:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-18	2012-12-25
2	2	3	7777999966660000	12	2011	2012-09-09	2013-01-01
3	18	3	7777999966660000	12	2011	2012-12-02	2013-01-01
4	19	2	8888999911112222	8	2012	2012-11-30	2012-12-31
5	20	2	8888999911112222	8	2012	2012-11-30	2012-12-25
6	21	2	7777999966660000	12	2011	2012-10-31	2012-11-01

Orderlist:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	isbn character varying(40)	quantity integer
1	1	0-672-32793-8	3
2	2	0-980-32627-0	13
3	18	0-867-12345-7	17
4	19	0-980-32627-0	3
5	20	0-980-32627-0	3
6	21	0-672-32793-8	13

Stockmanager:

Output pane

Data Output | Explain | Messages | History

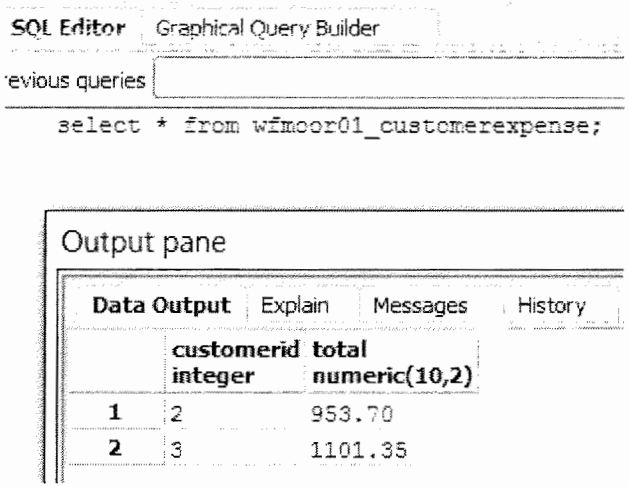
	isbn character varying(40)	quantity integer
1	0-867-12345-7	8
2	0-980-32627-0	7
3	0-672-32793-8	10

(b) Create a table CustomerExpense(customerid, total) with this information: for each customer, calculate how much money they have spent so far.

Code and example below.

```
create table wfmoor01_CustomerExpense(
      customerid  integer primary key references wfmoor01_customers(cid),
      total numeric(10,2)
      );
insert into wfmoor01_customerexpense (customerid, total) values (2, 953.70);
insert into wfmoor01_customerexpense (customerid, total) values (3, 1101.35);

select * from wfmoor01_customerexpense;
```



FYI - I added 2 rows to table based on the qty ordered for these customers in previous examples.

(c) Create a trigger or procedure to keep the table CustomerExpense updated as the database registers new orders.

(d) Create a trigger or procedure to keep the table CustomerExpense updated as the database registers cancellations of existing orders.

(e) Create a trigger or procedure to keep the table CustomerExpense updated as the database registers changes in existing orders.

Code for stored procedure on the following pages.

```
create or replace function get_order2 (book_isbn varchar(40), qty_ordered integer, custid integer, cardno varchar(40),
cardmnth numeric(2), cardyr numeric(4), date_order date, date_ship date, order_type varchar(10), old_ordernum integer)
returns text as $$

declare
        var_qty_in_stock integer;
        var_new_qty_in_stock integer;
        var_qty_ordered integer;
        var_qty_canceled integer;
        var_qty_changed integer;
        var_price numeric(10,2);
        var_order_num integer;
        var_temp varchar(40);
        var_order_type varchar(10);
        var_old_ordernum integer;
        var_old_total numeric(10,2);
        var_total_price numeric(10,2);
        var_total_price2 numeric(10,2);
        var_new_ttl_price numeric(10,2);
        var_custid integer;

BEGIN
        var_qty_ordered := $2;
        var_order_type := $9;
        var_old_ordernum := $10;

-- if a new order
IF var_order_type = 'NEW' THEN

        -- get book price and quantity in stock
        SELECT qty_in_stock, price INTO var_qty_in_stock, var_price FROM wfmoor01_books WHERE isbn = $1;

        var_new_qty_in_stock := var_qty_in_stock - var_qty_ordered;
        var_total_price := var_qty_ordered * var_price;

        -- insert data into orders table
        insert into wfmoor01_orders (cid, cardnum, cardmonth, cardyear, order_date, ship_date) values ($3, $4, $5, $6, $7, $8);

        -- insert data into orderlist
        insert into wfmoor01_orderlist (ordernum, isbn, quantity) values (lastval(), $1, $2);

        -- UPDATE CUSTOMEREXPENSE
        -- get old total first
        select total into var_old_total from wfmoor01_customerexpense where customerid = $3;
        update wfmoor01_customerexpense set total = total + var_total_price where customerid = $3;

        if var_new_qty_in_stock > 0 then  --quantity greater than 0.

                -- update books qty
                Update wfmoor01_books set qty_in_stock = var_new_qty_in_stock where isbn = $1;

                -- check for book in stockmanager
```

```
                select isbn into var_temp from wfmoor01_stockmanager where isbn = $1;

                if var_temp is not null then
                        -- if book exist, update stockmanager qty
                        Update wfmoor01_stockmanager set quantity = var_new_qty_in_stock where isbn = $1;
                else
                        -- if book doesn't exist, insert a new book and qty in stockmanager
                        insert into wfmoor01_stockmanager (isbn,quantity) values ($1, var_new_qty_in_stock);
                end if;

        else  --quantity less than or equal to 0.

                var_new_qty_in_stock := 10 + var_new_qty_in_stock;

                LOOP
                        IF var_new_qty_in_stock > 0 THEN
                                EXIT;  -- exit loop
                        else
                                var_new_qty_in_stock := 10 + var_new_qty_in_stock;
                        END IF;
                END LOOP;

                -- update books qty
                Update wfmoor01_books set qty_in_stock = var_new_qty_in_stock where isbn = $1;

                -- check for book in stockmanager
                select isbn into var_temp from wfmoor01_stockmanager where isbn = $1;

                if var_temp is not null then
                        -- if book exist, update stockmanager qty
                        Update wfmoor01_stockmanager set quantity = var_new_qty_in_stock where isbn = $1;
                else
                        -- if book doesn't exist, insert a new book and qty in stockmanager
                        insert into wfmoor01_stockmanager (isbn,quantity) values ($1, var_new_qty_in_stock);
                end if;

        end if;
END IF;


-- if canceling order
IF var_order_type = 'CANCEL' THEN

    -- from orderlist get qty ordered and book number.
        select quantity, isbn into var_qty_canceled, var_temp from wfmoor01_orderlist where ordernum = var_old_ordernum;

        -- from order get custid
        select cid into var_custid from wfmoor01_orders where ordernum = var_old_ordernum;

        -- get book price from books table.
        select price into var_price from wfmoor01_books where isbn = var_temp;
```

```
        -- get the current customerexpense total amount.
        select total into var_old_total from wfmoor01_customerexpense where customerid = var_custid;

        -- get the current stockmanager quantity amount.
        select quantity into var_qty_in_stock from wfmoor01_stockmanager where isbn = var_temp;

    var_new_qty_in_stock := var_qty_in_stock + var_qty_canceled; -- add qty cancelled back into stock amount
        var_total_price := var_price * var_qty_canceled; -- calculate amount canceled
        var_new_ttl_price := var_old_total - var_total_price; -- subtract amount canceled from total expenses

        update wfmoor01_books set qty_in_stock = var_new_qty_in_stock where isbn = var_temp; -- update w/ new qty in stock
        update wfmoor01_stockmanager set quantity = var_new_qty_in_stock where isbn = var_temp; -- update w/ new qty in stock
        update wfmoor01_customerexpense set total = var_new_ttl_price where customerid = var_custid ; -- update w/ new total amount

        -- remove rec' from orders and orderlist tables
        delete from wfmoor01_orderlist where ordernum = var_old_ordernum;
        delete from wfmoor01_orders where ordernum = var_old_ordernum;

END IF;


-- if changing an existing order
IF var_order_type = 'CHANGE' THEN


        var_qty_changed := $2;

    -- from orderlist get qty originally ordered and book number.
        select quantity, isbn into var_qty_ordered, var_temp from wfmoor01_orderlist where ordernum = var_old_ordernum;

        -- from order get custid
        select cid into var_custid from wfmoor01_orders where ordernum = var_old_ordernum;

        -- get book price from books table.
        select price into var_price from wfmoor01_books where isbn = var_temp;

        -- get the current customerexpense total amount.
        select total into var_old_total from wfmoor01_customerexpense where customerid = var_custid;

        -- get the current stockmanager quantity amount.
        select quantity into var_qty_in_stock from wfmoor01_stockmanager where isbn = var_temp;

    var_new_qty_in_stock := (var_qty_in_stock + var_qty_ordered) - var_qty_changed; -- add qty from original order back into stock
amount and subtract new qty
        var_total_price := var_price * var_qty_ordered; -- calculate old order amt
        var_total_price2 := var_price * var_qty_changed; -- calculate new order amt
        var_new_ttl_price := (var_old_total + var_total_price) - var_total_price2; -- add amt from original order back in and subtract
new order amt

        update wfmoor01_books set qty_in_stock = var_new_qty_in_stock where isbn = var_temp; -- update w/ new qty in stock
        update wfmoor01_stockmanager set quantity = var_new_qty_in_stock where isbn = var_temp; -- update w/ new qty in stock
        update wfmoor01_customerexpense set total = var_new_ttl_price where customerid = var_custid ; -- update w/ new total amount
```

```
        -- update rec's from orderlist tables
        update wfmoor01_orderlist set quantity = var_new_qty_in_stock where ordernum = var_old_ordernum;


END IF;


        RETURN 'Order Complete';
END;

$$ language plpgsql;
```

Examples for 4c, 4d & 4e on the following pages:

Example 1. New Order.

```
select get_order2 ('0-672-32793-8', 8, 2, '7777999966660000', 12, 2011, '12/5/2012', '1/1/2013', 'NEW', 0);
```

Similar to the get_order1 stored procedure. Added two more fields (see the ones in red). One field for whether order is "NEW", "CANCEL" or "CHANGE" orders and a field for the original order number if transaction is a "CHANGE" or "CANCEL".

Store procedure will get the book price from the books table. And will calculate the total price and add it to the customer expense.

Price per book for '0-672-32793-8' = 45. Qty order = 8. Total expense is 8 * 45 = 360. 360 is added to the current total for customerid 2 which is 953.70. New total will be 1313.70. All other tables are updates accordingly.

Tables after running Stored Procedure.

Customerexpense:

Output pane

Data Output | Explain | Messages | History

	customerid integer	total numeric(10,2)
1	3	1101.35
2	2	1313.70

Books:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-867-12345-7	PSQL FOR DUMMIES	John Doe	8	35.00	30.00	2012	RH-0001
2	0-980-32627-0	PSQL in 24 Hours	Jane Doe	7	38.95	31.45	2012	SM-0099
3	0-672-32793-8	PSQL 101	John Smith	2	45.00	40.00	2002	RH-0001

Orders:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-18	2012-12-25
2	2	3	7777999966660000	12	2011	2012-09-09	2013-01-01
3	18	3	7777999966660000	12	2011	2012-12-02	2013-01-01
4	19	2	8888999911112222	8	2012	2012-11-30	2012-12-31
5	20	2	8888999911112222	8	2012	2012-11-30	2012-12-25
6	21	2	7777999966660000	12	2011	2012-10-31	2012-11-01
7	24	2	7777999966660000	12	2011	2012-12-05	2013-01-01

Orderlist:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	isbn character varying(40)	quantity integer
1	1	0-672-32793-8	3
2	2	0-980-32627-0	13
3	18	0-867-12345-7	17
4	19	0-980-32627-0	3
5	20	0-980-32627-0	3
6	21	0-672-32793-8	13
7	24	0-672-32793-8	8

Stockmanager:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	quantity integer
1	0-867-12345-7	8
2	0-980-32627-0	7
3	0-672-32793-8	2

Example 2. Cancel an order.

```
select get_order2 ('', 0, 0, ' ', 0, 0, '12/5/2012', '12/5/2012', 'CANCEL', 2);
```

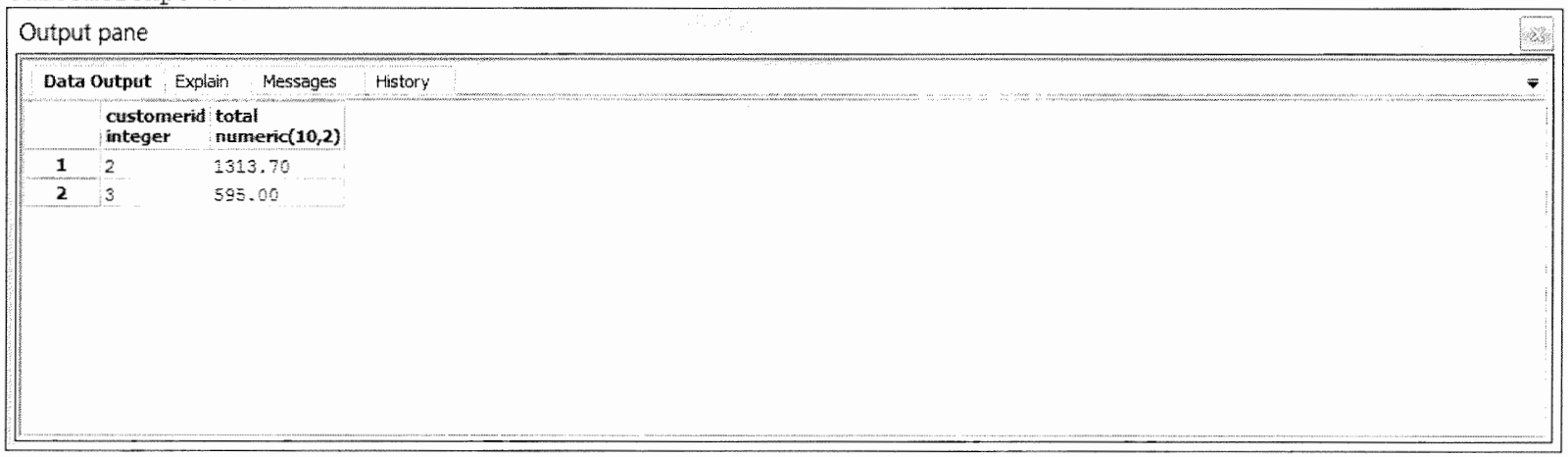
Order number is 2. This order will be deleted from the Orders and Orderlist tables and the expense and qty in stock will be adjusted accordingly for each table. Today's date is being passed in both fields as default parameters. Stored procedure will pull additional information (i.e. customer id) from tables based on the order number.

For order # 2, 13 books were order at a price of 38.95 each. Original expense was 506.35. 13 will be added back into the qty-in-stock of the book and 506.35 will be subtracted from customer's expense.

Original customer expense was 1101.35 new customer expense is now 595.00. Original qty in stock was 7, new qty in stock is now 20.

Tables after running stored procedure.

Customerexpense:



Books:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-867-12345-7	PSQL FOR DUMMIES	John Doe	8	35.00	30.00	2012	RH-0001
2	0-672-32793-8	PSQL 101	John Smith	2	45.00	40.00	2002	RH-0001
3	0-980-32627-0	PSQL in 24 Hours	Jane Doe	20	38.95	31.45	2012	SM-0099

Orders:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-18	2012-12-25
2	18	3	7777999966660000	12	2011	2012-12-02	2013-01-01
3	19	2	8888999911112222	8	2012	2012-11-30	2012-12-31
4	20	2	8888999911112222	8	2012	2012-11-30	2012-12-25
5	21	2	7777999966660000	12	2011	2012-10-31	2012-11-01
6	24	2	7777999966660000	12	2011	2012-12-05	2013-01-01

Orderlist:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	isbn character varying(40)	quantity integer
1	1	0-672-32793-8	3
2	18	0-867-12345-7	17
3	19	0-980-32627-0	3
4	20	0-980-32627-0	3
5	21	0-672-32793-8	13
6	24	0-672-32793-8	8

Stockmanager:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	quantity integer
1	0-867-12345-7	8
2	0-672-32793-8	2
3	0-980-32627-0	20

Example 3. Change an existing order.

```
select get_order2 ('', 4, 0, ' ', 0, 0, '12/5/2012', '12/5/2012', 'CHANGE', 1);
```

Order number to change is 1. New qty ordered is 4. Today's date is being passed in both fields as default parameters.

Stored procedure will pull additional information (i.e. customer id) from tables based on the order number.

For order # 1, 3 books were ordered at a price of 45.00 each. Original expense was 135.00. New expense is 4 * 45 = 180.

Current qty in stock for this book is 2. 3 will be added back in then the new qty ordered of 4 will be subtracted from that number giving a new qty in stock of 1.

Current expense for the customer who placed that order is 1313.70. The original expense of 135 is subtracted from this amount and the new expense of 180 will be added, giving a new customer expense of 1358.70.

Note: I should have caught this sooner. There is a bug in my code:

```
var_new_ttl_price := (var_old_total + var_total_price) - var_total_price2; -- add amt from original order back in and subtract new order amt
```

I should be subtracting the original order amount then adding the new order amount. I'll accept the penalty for this discrepancy.

Tables after running stored procedure.

Customerexpense;

Output pane

Data Output | Explain | Messages | History

	customerid integer	total numeric(10,2)
1	3	595.00
2	2	1268.70

books:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	title character varying(40)	author character varying(40)	qty_in_stock integer	price numeric(10,2)	cost numeric(10,2)	year_published numeric(4,0)	publisherid character varying(40)
1	0-867-12345-7	PSQL FOR DUMMIES	John Doe	8	35.00	30.00	2012	RH-0001
2	0-980-32627-0	PSQL in 24 Hours	Jane Doe	20	38.95	31.45	2012	SM-0099
3	0-672-32793-8	PSQL 101	John Smith	1	45.00	40.00	2002	RH-0001

Orders:

Output pane

Data Output | Explain | Messages | History

	ordernum integer	cid integer	cardnum character varying(40)	cardmonth numeric(2,0)	cardyear numeric(4,0)	order_date date	ship_date date
1	1	2	8888999911112222	8	2012	2012-08-18	2012-12-25
2	18	3	7777999966660000	12	2011	2012-12-02	2013-01-01
3	19	2	8888999911112222	8	2012	2012-11-30	2012-12-31
4	20	2	8888999911112222	8	2012	2012-11-30	2012-12-25
5	21	2	7777999966660000	12	2011	2012-10-31	2012-11-01
6	24	2	7777999966660000	12	2011	2012-12-05	2013-01-01

Orderlist;

Output pane

Data Output | Explain | Messages | History

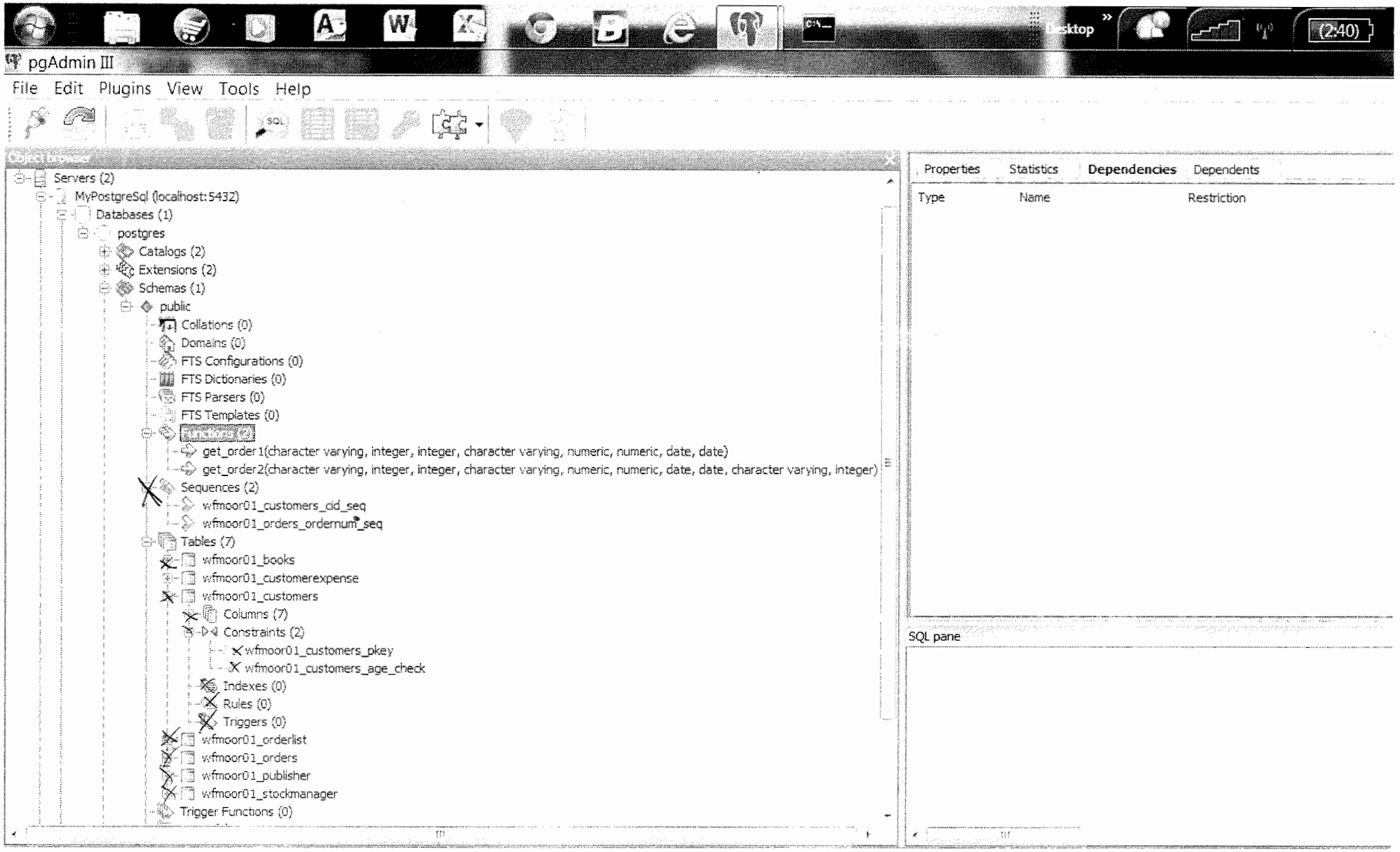
	ordernum integer	isbn character varying(40)	quantity integer
1	18	0-867-12345-7	17
2	19	0-980-32627-0	3
3	20	0-980-32627-0	3
4	21	0-672-32793-8	13
5	24	0-672-32793-8	8
6	1	0-672-32793-8	1

Stockmanager:

Output pane

Data Output | Explain | Messages | History

	isbn character varying(40)	quantity integer
1	0-867-12345-7	8
2	0-980-32627-0	20
3	0-672-32793-8	1


pgAdmin III
File Edit Plugins View Tools Help
Servers (2)
MyPostgreSql (localhost:5432)
Databases (1)
postgres
Catalogs (2)
Extensions (2)
Schemas (1)
public
Collations (0)
Domains (0)
FTS Configurations (0)
FTS Dictionaries (0)
FTS Parsers (0)
FTS Templates (0)
Functions (2)
get_order1(character varying, integer, integer, character varying, numeric, numeric, date, date)
get_order2(character varying, integer, integer, character varying, numeric, numeric, date, date, character varying, integer)
Sequences (2)
wfmoor01_customers_cid_seq
wfmoor01_orders_ordernum_seq
Tables (7)
wfmoor01_books
wfmoor01_customerexpense
wfmoor01_customers
Columns (7)
Constraints (2)
wfmoor01_customers_pkey
wfmoor01_customers_age_check
Indexes (0)
Rules (0)
Triggers (0)
wfmoor01_orderlist
wfmoor01_orders
wfmoor01_publisher
wfmoor01_stockmanager
Trigger Functions (0)
Properties
Statistics
Dependencies
Dependents
Type
Name
Restriction
SQL pane
Retrieving details on catalog information_schema... Done.

Python