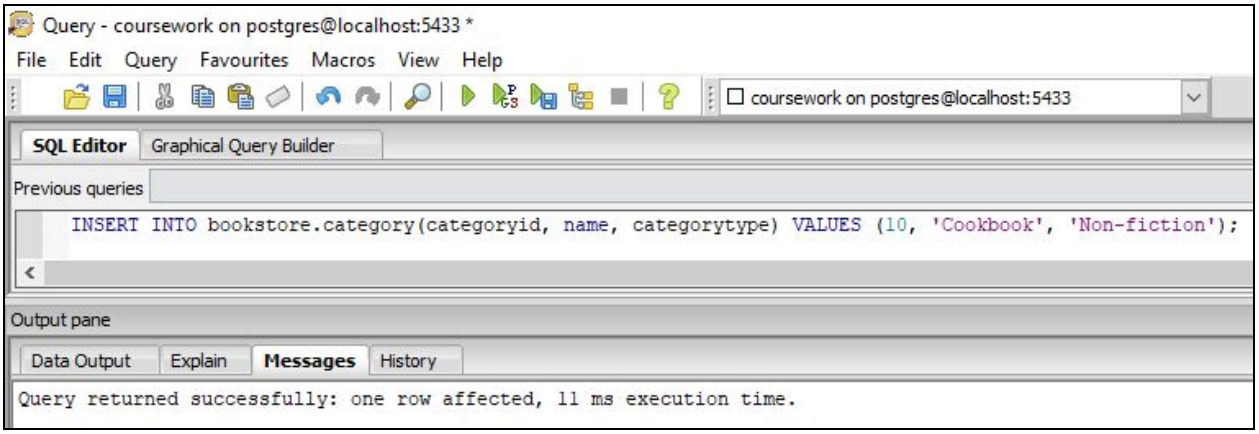
SQL Statements: Testing Evidence

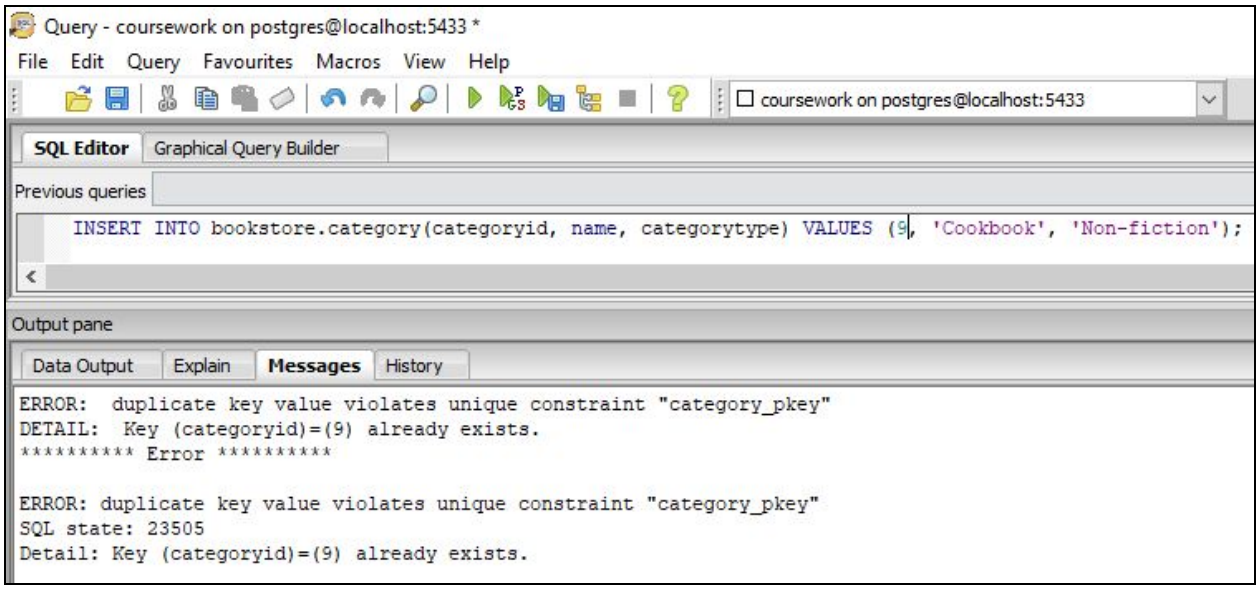
Task 1

"Given a category ID, name and type, create a new category".

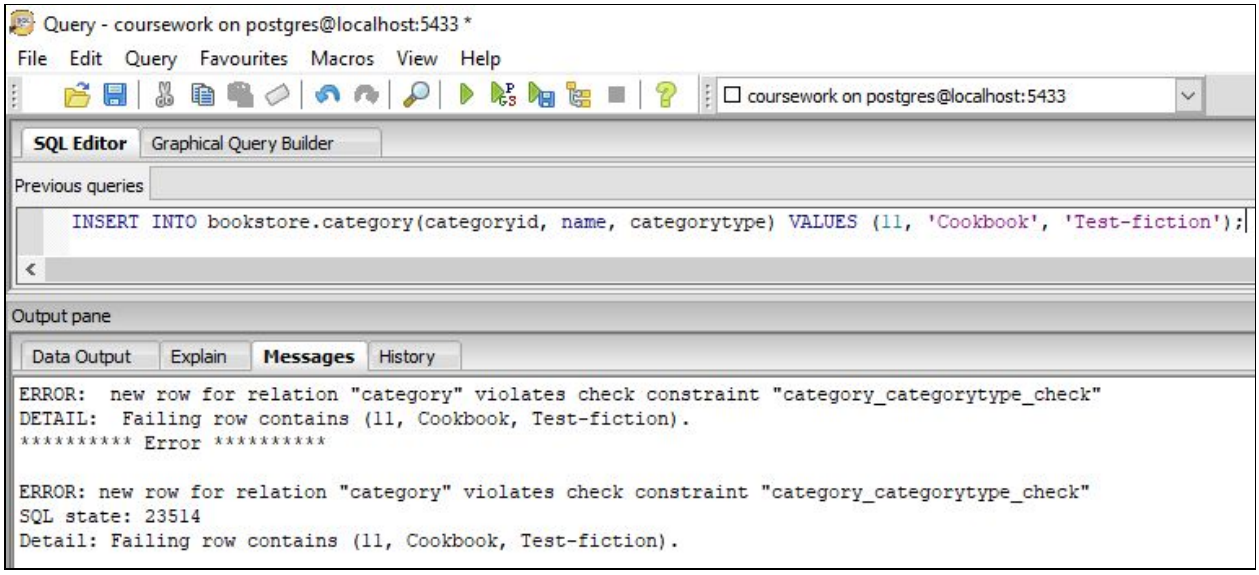
INSERT INTO bookstore.category(categoryid, name, categorytype) VALUES (10, 'Cookbook', 'Non-fiction');



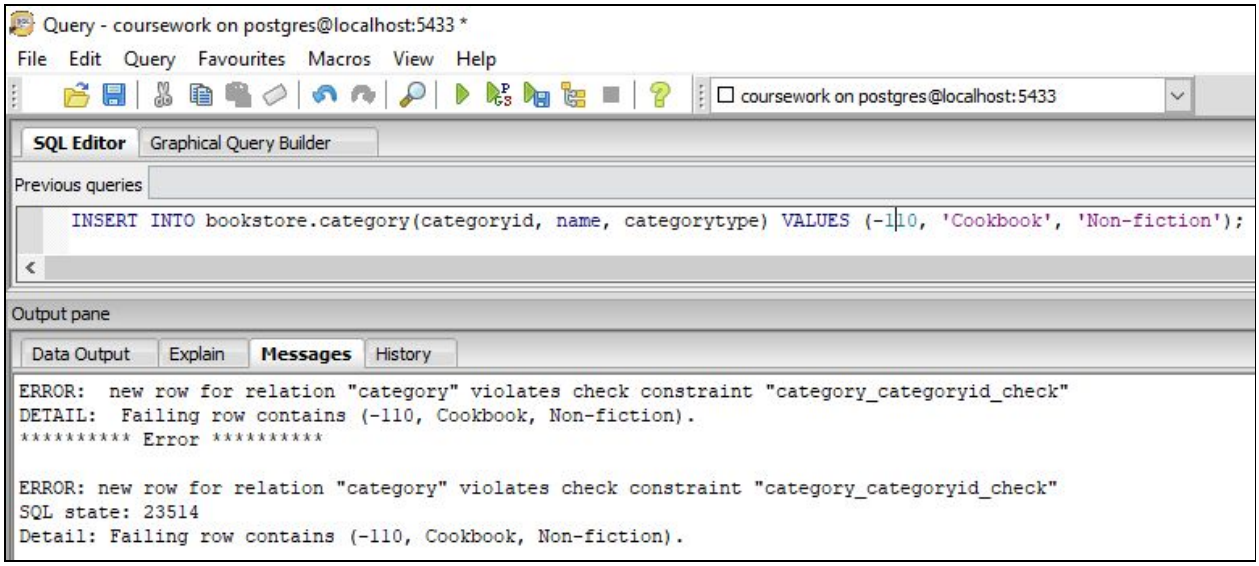
INSERT INTO bookstore.category(categoryid, name, categorytype) VALUES (9, 'Cookbook', 'Non-fiction');



INSERT INTO bookstore.category(categoryid, name, categorytype) VALUES (11, 'Cookbook', 'Test-fiction');



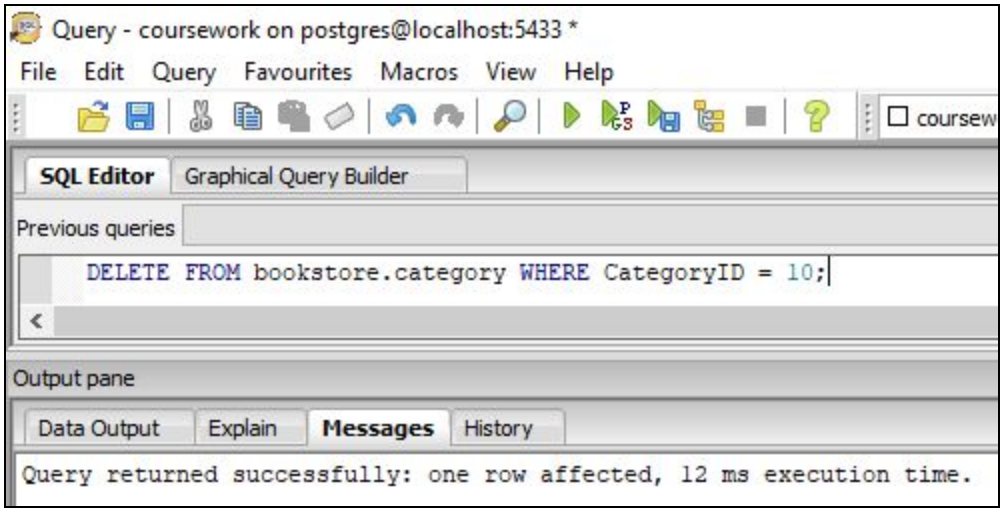
INSERT INTO bookstore.category(categoryid, name, categorytype) VALUES (-110, 'Cookbook', 'Non-fiction');



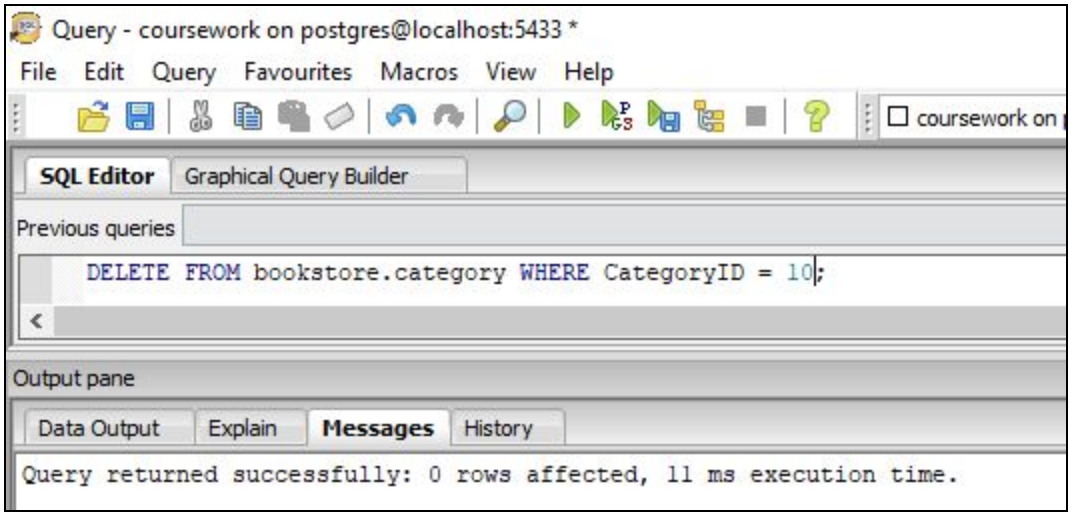
Task 2

“Given a category ID, remove the record for that category.”

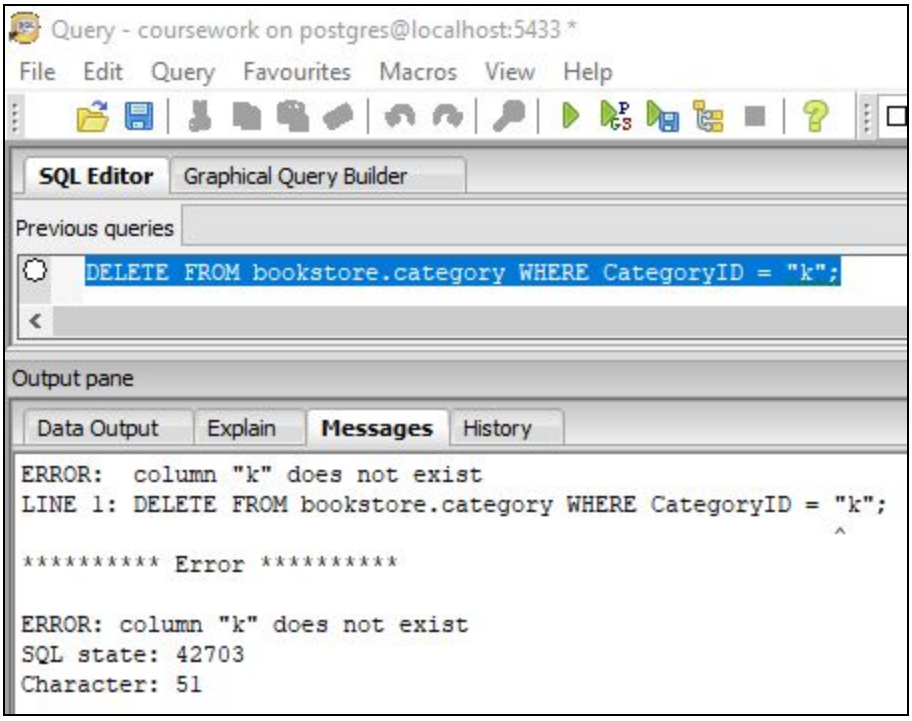
DELETE FROM bookstore.category WHERE CategoryID = 10;



DELETE FROM bookstore.category WHERE CategoryID = 10;



DELETE FROM bookstore.category WHERE CategoryID = "k";

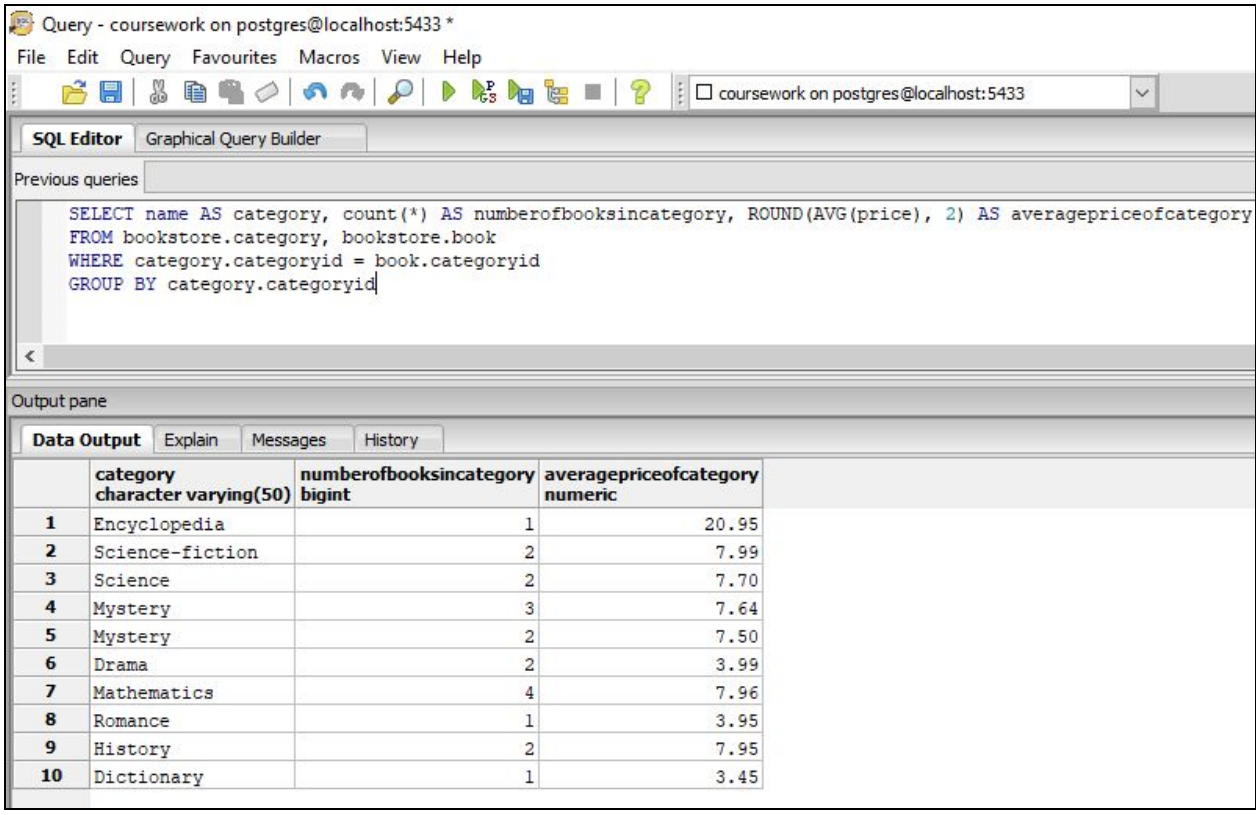


Task 3

"Produce a summary report of books available in each category. The report should include the number of book titles and the average price in each category as well as an appropriate report header and a summary line with totals (hint: summary line may be produced by a separate query). Format your field values appropriately."

Main Query

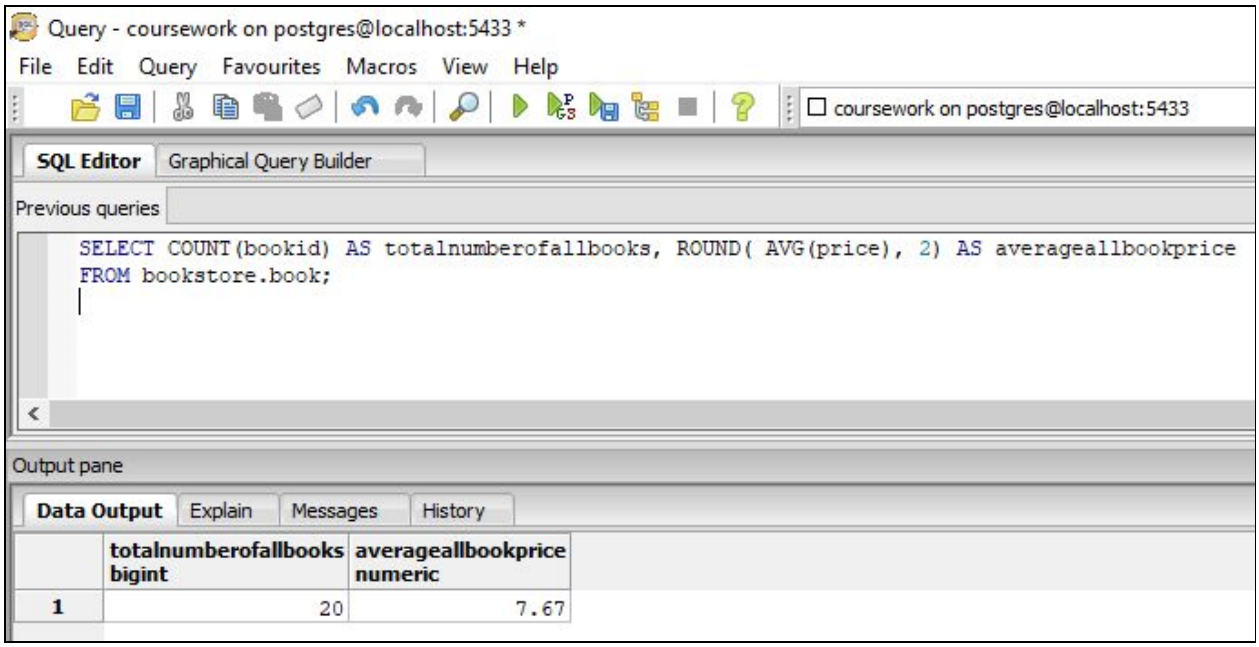
```
SELECT name AS category, count(*) AS numberofbooksincategory, ROUND(AVG(price), 2) AS averagepriceofcategory
FROM bookstore.category, bookstore.book
WHERE category.categoryid = book.categoryid
GROUP BY category.categoryid;
```



	category character varying(50)	numberofbooksincategory bigint	averagepriceofcategory numeric
1	Encyclopedia	1	20.95
2	Science-fiction	2	7.99
3	Science	2	7.70
4	Mystery	3	7.64
5	Mystery	2	7.50
6	Drama	2	3.99
7	Mathematics	4	7.96
8	Romance	1	3.95
9	History	2	7.95
10	Dictionary	1	3.45

Summary Line Query

SELECT COUNT(bookid) AS totalnumberofallbooks, ROUND(AVG(price), 2) AS averageallbookprice
FROM bookstore.book;



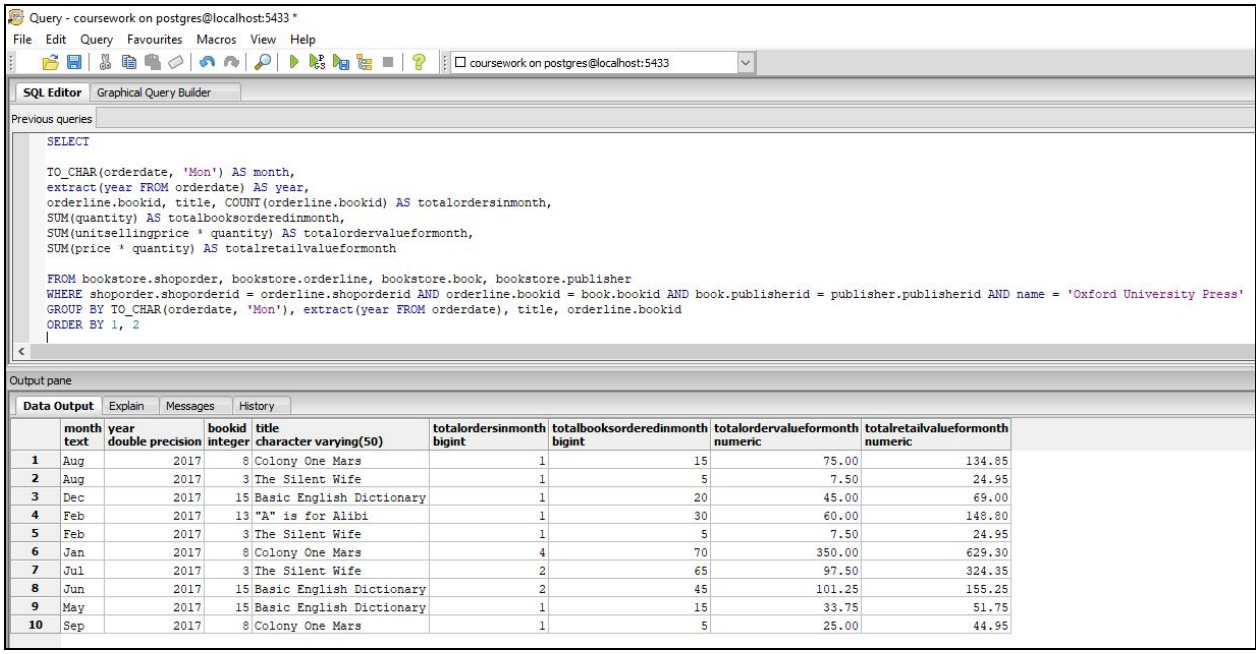
Task 4

“Given a publisher name, produce a report of books ordered by year and month. For each year and month the report should show bookid, title, total number of orders for the title, total quantity and total selling value (both order value and retail value).”

“SELECT

TO_CHAR(orderdate, 'Mon') AS month,
extract(year FROM orderdate) AS year,
orderline.bookid, title, COUNT(orderline.bookid) AS totalordersinmonth,
SUM(quantity) AS totalbooksorderedinmonth,
*SUM(unitsellingprice * quantity) AS totalordervalueformonth,*
*SUM(price * quantity) AS totalretailvalueformonth*

FROM bookstore.shoporder, bookstore.orderline, bookstore.book, bookstore.publisher
WHERE shoporder.shoporderid = orderline.shoporderid AND orderline.bookid = book.bookid AND book.publisherid = publisher.publisherid AND name = 'Oxford University Press'
GROUP BY TO_CHAR(orderdate, 'Mon'), extract(year FROM orderdate), title, orderline.bookid
ORDER BY 1, 2”

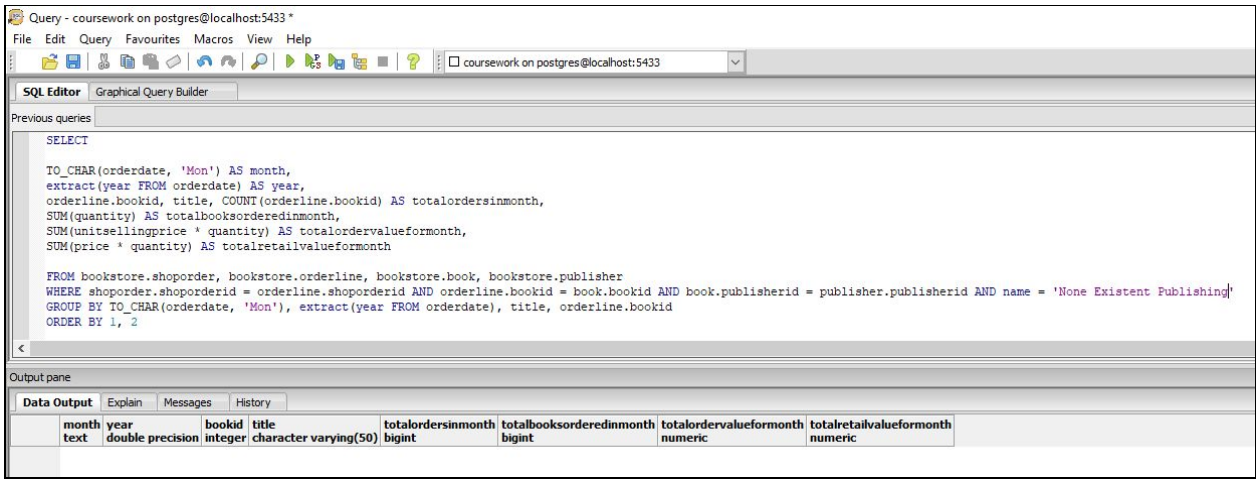


	month text	year double precision	bookid integer	title character varying(50)	totalordersinmonth bigint	totalbooksorderedinmonth bigint	totalordervalueformonth numeric	totalretailvalueformonth numeric
1	Aug	2017	8	Colony One Mars	1	15	75.00	134.85
2	Aug	2017	3	The Silent Wife	1	5	7.50	24.95
3	Dec	2017	15	Basic English Dictionary	1	20	45.00	69.00
4	Feb	2017	13	"A" is for Alibi	1	30	60.00	148.80
5	Feb	2017	3	The Silent Wife	1	5	7.50	24.95
6	Jan	2017	8	Colony One Mars	4	70	350.00	629.30
7	Jul	2017	3	The Silent Wife	2	65	97.50	324.35
8	Jun	2017	15	Basic English Dictionary	2	45	101.25	155.25
9	May	2017	15	Basic English Dictionary	1	15	33.75	51.75
10	Sep	2017	8	Colony One Mars	1	5	25.00	44.95

SELECT

TO_CHAR(orderdate, 'Mon') AS month,
extract(year FROM orderdate) AS year,
orderline.bookid, title, COUNT(orderline.bookid) AS totalordersinmonth,
SUM(quantity) AS totalbooksorderedinmonth,
*SUM(unitsellingprice * quantity) AS totalordervalueformonth,*
*SUM(price * quantity) AS totalretailvalueformonth*

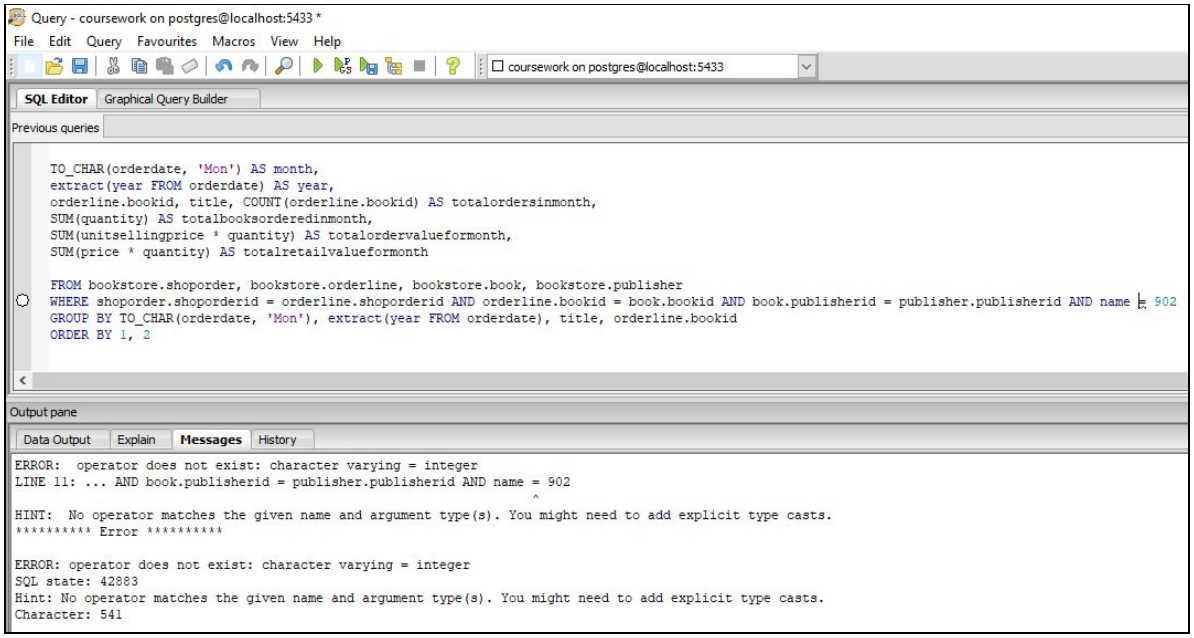
FROM bookstore.shoporder, bookstore.orderline, bookstore.book, bookstore.publisher
WHERE shoporder.shoporderid = orderline.shoporderid AND orderline.bookid = book.bookid
AND book.publisherid = publisher.publisherid AND name = 'None Existent Publishing'
GROUP BY TO_CHAR(orderdate, 'Mon'), extract(year FROM orderdate), title, orderline.bookid
ORDER BY 1, 2



SELECT

TO_CHAR(orderdate, 'Mon') AS month,
extract(year FROM orderdate) AS year,
orderline.bookid, title, COUNT(orderline.bookid) AS totalordersinmonth,
SUM(quantity) AS totalbooksorderedinmonth,
*SUM(unitsellingprice * quantity) AS totalordervalueformonth,*
*SUM(price * quantity) AS totalretailvalueformonth*

FROM bookstore.shoporder, bookstore.orderline, bookstore.book, bookstore.publisher
WHERE shoporder.shoporderid = orderline.shoporderid AND orderline.bookid = book.bookid
AND book.publisherid = publisher.publisherid AND name = 902
GROUP BY TO_CHAR(orderdate, 'Mon'), extract(year FROM orderdate), title, orderline.bookid
ORDER BY 1, 2

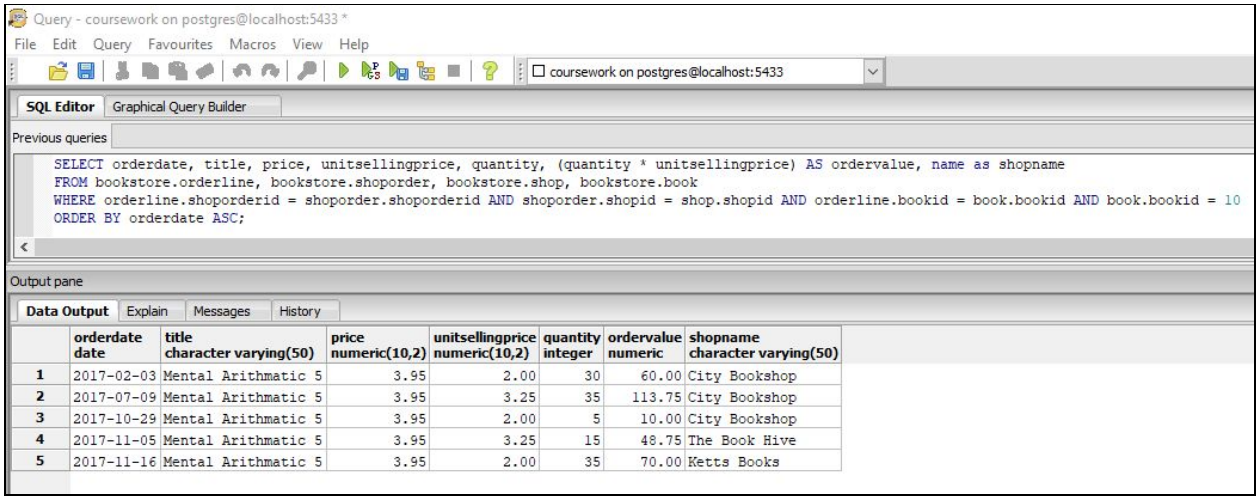

Query - coursework on postgres@localhost:5433 *
File Edit Query Favourites Macros View Help
coursework on postgres@localhost:5433
SQL Editor
Graphical Query Builder
Previous queries
TO_CHAR(orderdate, 'Mon') AS month,
extract(year FROM orderdate) AS year,
orderline.bookid, title, COUNT(orderline.bookid) AS totalordersinmonth,
SUM(quantity) AS totalbooksorderedinmonth,
SUM(unitsellingprice * quantity) AS totalordervalueformonth,
SUM(price * quantity) AS totalretailvalueformonth
FROM bookstore.shoporder, bookstore.orderline, bookstore.book, bookstore.publisher
WHERE shoporder.shoporderid = orderline.shoporderid AND orderline.bookid = book.bookid AND book.publisherid = publisher.publisherid AND name = 902
GROUP BY TO_CHAR(orderdate, 'Mon'), extract(year FROM orderdate), title, orderline.bookid
ORDER BY 1, 2
Output pane
Data Output
Explain
Messages
History
ERROR: operator does not exist: character varying = integer
LINE 11: ... AND book.publisherid = publisher.publisherid AND name = 902
^
HINT: No operator matches the given name and argument type(s). You might need to add explicit type casts.
********** Error **********
ERROR: operator does not exist: character varying = integer
SQL state: 42883
Hint: No operator matches the given name and argument type(s). You might need to add explicit type casts.
Character: 541

Task 5

"Given a book ID, produce the order history (i.e. all order lines) for that book. The query should include order date, order title, price, unitselling price, total quantity, order value and shop name. Include a summary line showing the total number of copies ordered and the total selling value (hint: summary line may be produced by a separate query)."

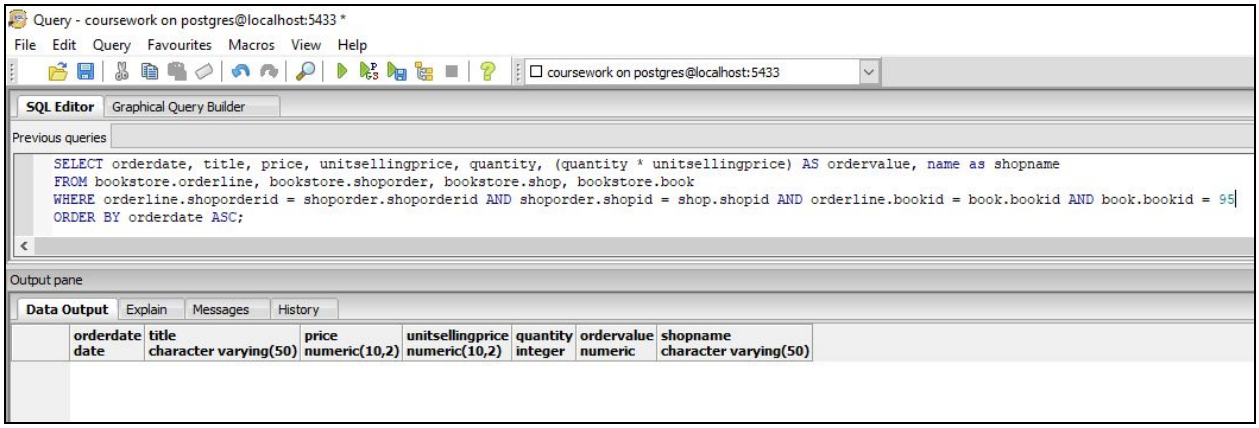
Main Query

```
SELECT orderdate, title, price, unitsellingprice, quantity, (quantity * unitsellingprice) AS ordervalue, name as shopname
FROM bookstore.orderline, bookstore.shoporder, bookstore.shop, bookstore.book
WHERE orderline.shoporderid = shoporder.shoporderid AND shoporder.shopid = shop.shopid
AND orderline.bookid = book.bookid AND book.bookid = 10
ORDER BY orderdate ASC;
```

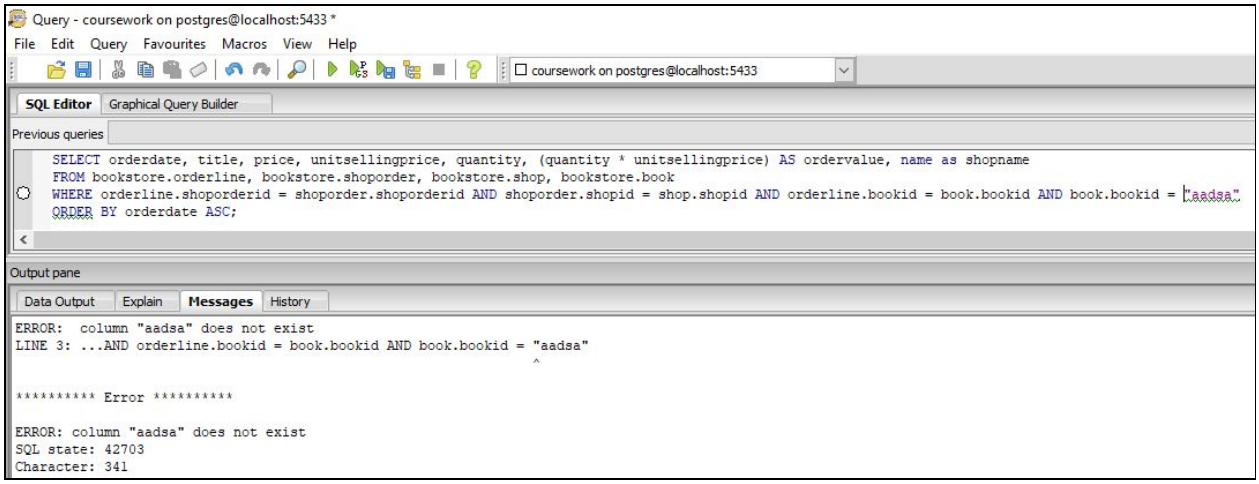


	orderdate date	title character varying(50)	price numeric(10,2)	unitsellingprice numeric(10,2)	quantity integer	ordervalue numeric	shopname character varying(50)
1	2017-02-03	Mental Arithmatic 5	3.95	2.00	30	60.00	City Bookshop
2	2017-07-09	Mental Arithmatic 5	3.95	3.25	35	113.75	City Bookshop
3	2017-10-29	Mental Arithmatic 5	3.95	2.00	5	10.00	City Bookshop
4	2017-11-05	Mental Arithmatic 5	3.95	3.25	15	48.75	The Book Hive
5	2017-11-16	Mental Arithmatic 5	3.95	2.00	35	70.00	Ketts Books

```
SELECT orderdate, title, price, unitsellingprice, quantity, (quantity * unitsellingprice) AS ordervalue, name as shopname
FROM bookstore.orderline, bookstore.shoporder, bookstore.shop, bookstore.book
WHERE orderline.shoporderid = shoporder.shoporderid AND shoporder.shopid = shop.shopid
AND orderline.bookid = book.bookid AND book.bookid = 95
ORDER BY orderdate ASC;
```

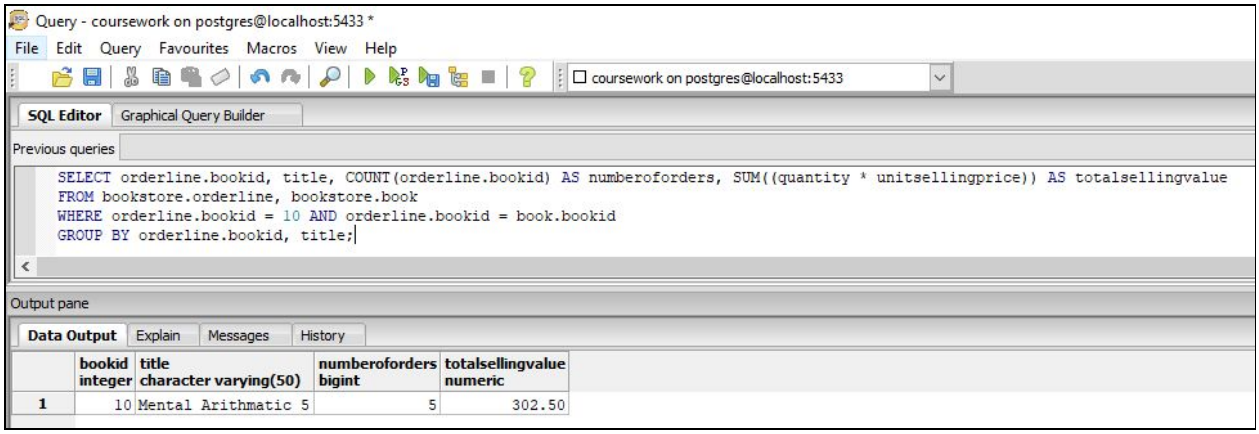


*SELECT orderdate, title, price, unitsellingprice, quantity, (quantity * unitsellingprice) AS ordervalue, name as shopname*
FROM bookstore.orderline, bookstore.shoporder, bookstore.shop, bookstore.book
WHERE orderline.shoporderid = shoporder.shoporderid AND shoporder.shopid = shop.shopid AND orderline.bookid = book.bookid AND book.bookid = "aadsa"
ORDER BY orderdate ASC;

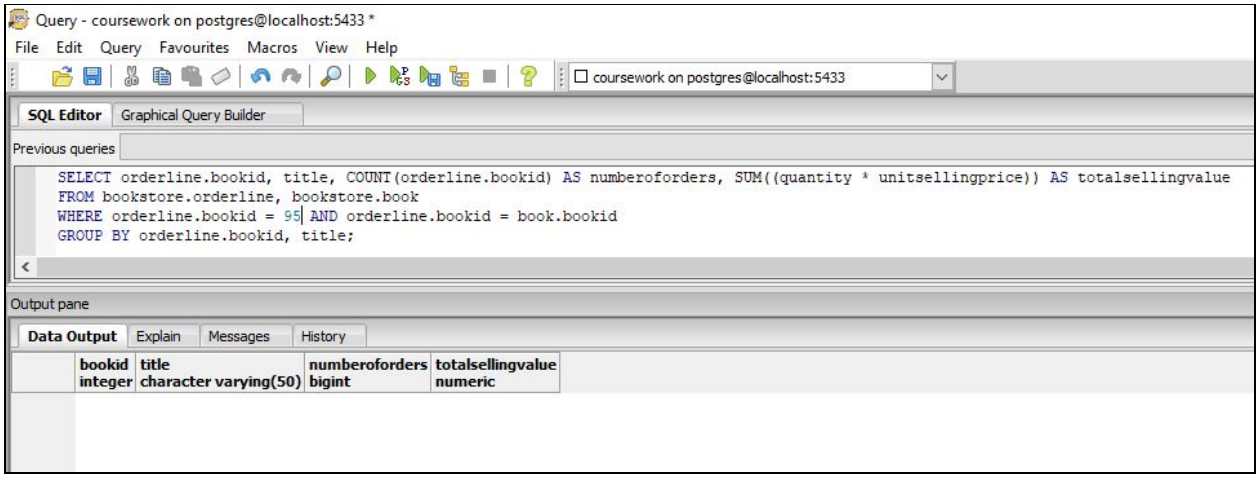


Summary Line Query

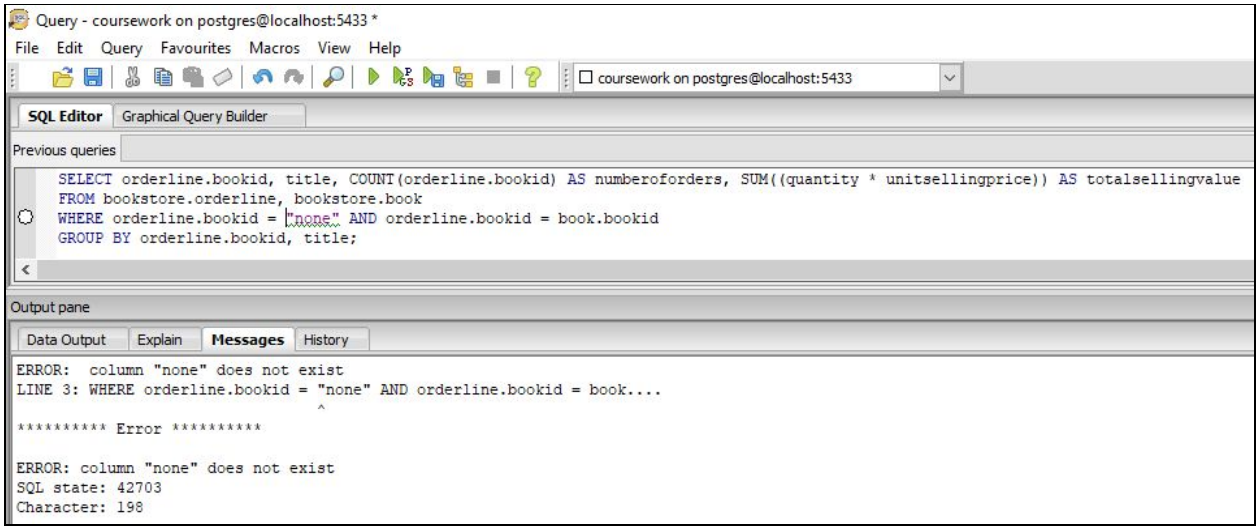
*SELECT orderline.bookid, title, COUNT(orderline.bookid) AS numberoforders, SUM((quantity * unitsellingprice)) AS totalsellingvalue*
FROM bookstore.orderline, bookstore.book
WHERE orderline.bookid = 10 AND orderline.bookid = book.bookid
GROUP BY orderline.bookid, title;



*SELECT orderline.bookid, title, COUNT(orderline.bookid) AS numberoforders, SUM((quantity * unitsellingprice)) AS totalsellingvalue*
FROM bookstore.orderline, bookstore.book
WHERE orderline.bookid = 95 AND orderline.bookid = book.bookid
GROUP BY orderline.bookid, title;



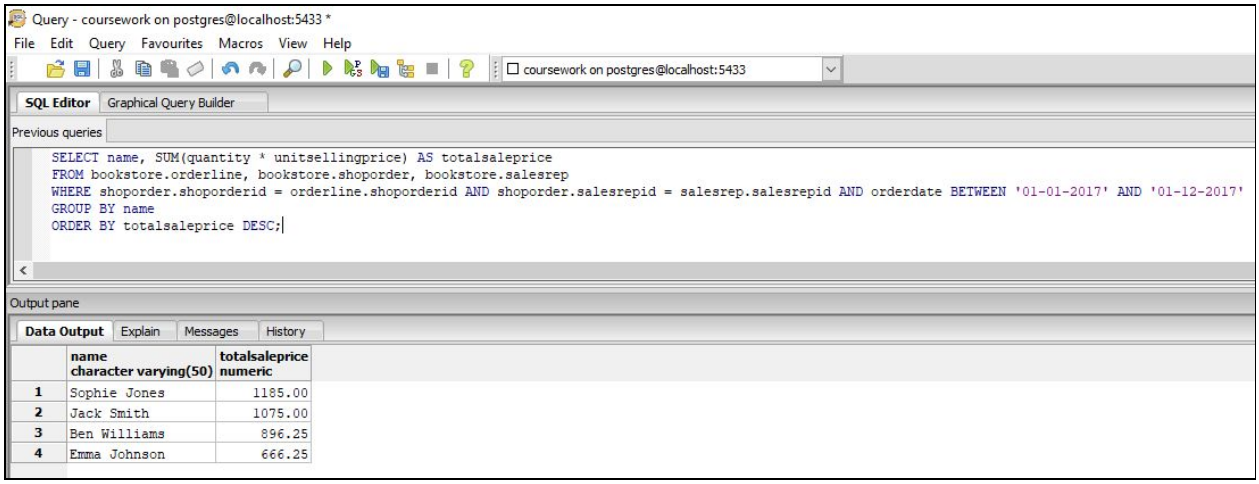
*SELECT orderline.bookid, title, COUNT(orderline.bookid) AS numberoforders, SUM((quantity * unitsellingprice)) AS totalsellingvalue*
FROM bookstore.orderline, bookstore.book
WHERE orderline.bookid = "none" AND orderline.bookid = book.bookid
GROUP BY orderline.bookid, title;


Query - coursework on postgres@localhost:5433 *
File Edit Query Favourites Macros View Help
coursework on postgres@localhost:5433
SQL Editor
Graphical Query Builder
Previous queries
SELECT orderline.bookid, title, COUNT(orderline.bookid) AS numberoforders, SUM((quantity * unitsellingprice)) AS totalsellingvalue
FROM bookstore.orderline, bookstore.book
WHERE orderline.bookid = "none" AND orderline.bookid = book.bookid
GROUP BY orderline.bookid, title;
Output pane
Data Output
Explain
Messages
History
ERROR: column "none" does not exist
LINE 3: WHERE orderline.bookid = "none" AND orderline.bookid = book....
^
********** Error **********
ERROR: column "none" does not exist
SQL state: 42703
Character: 198

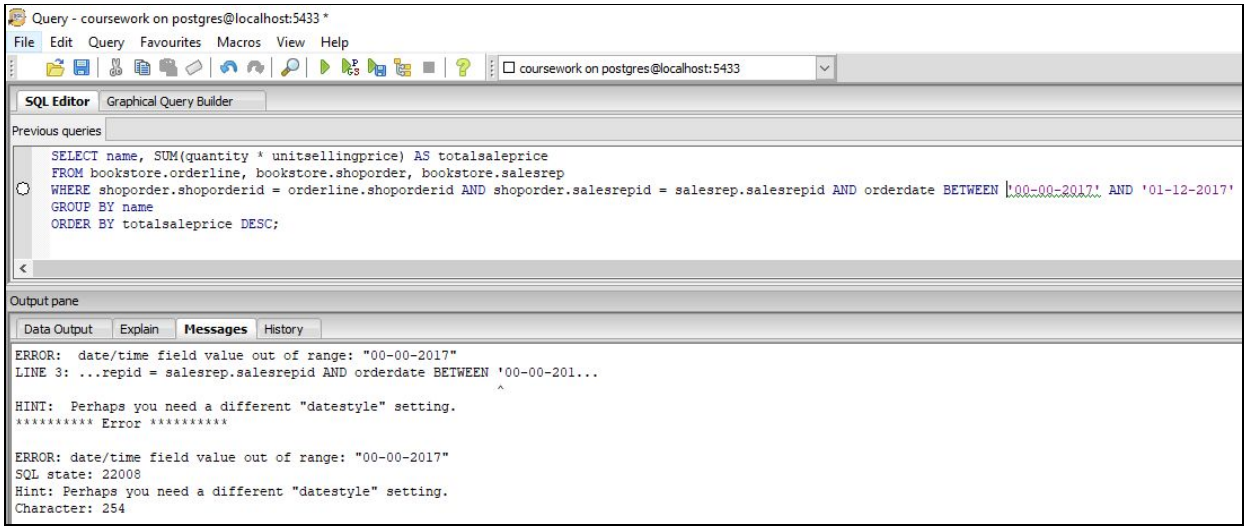
Task 6

“Given start and end dates, produce a report showing the performance of each sales representative over that period. The report should begin with the rep who generated most orders by value and include total units sold and total order value. It should include all sales reps.”

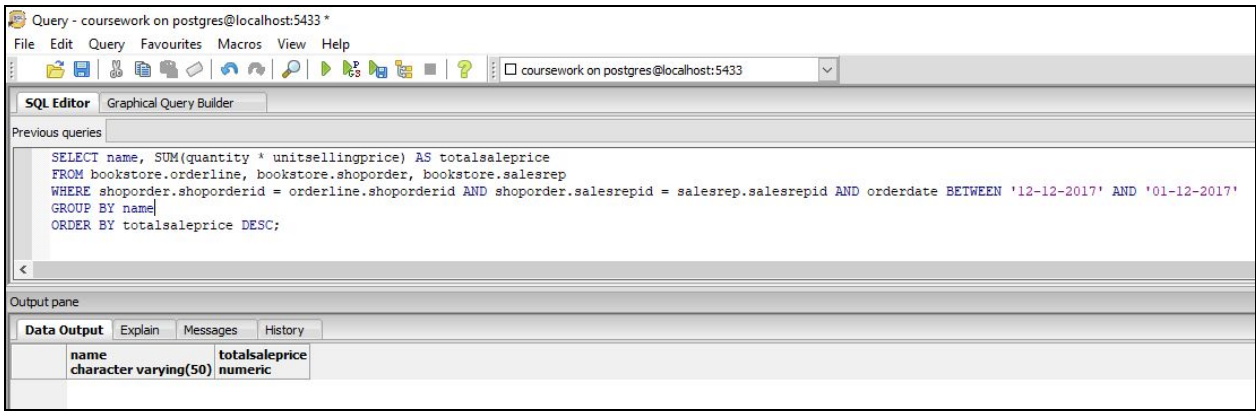
*“SELECT name, SUM(quantity * unitsellingprice) AS totalsaleprice*
FROM bookstore.orderline, bookstore.shoporder, bookstore.salesrep
WHERE shoporder.shoporderid = orderline.shoporderid AND shoporder.salesrepid = salesrep.salesrepid AND orderdate BETWEEN '01-01-2017' AND '01-12-2017'
GROUP BY name
ORDER BY totalsaleprice DESC;”



*SELECT name, SUM(quantity * unitsellingprice) AS totalsaleprice*
FROM bookstore.orderline, bookstore.shoporder, bookstore.salesrep
WHERE shoporder.shoporderid = orderline.shoporderid AND shoporder.salesrepid = salesrep.salesrepid AND orderdate BETWEEN '00-00-2017' AND '01-12-2017'
GROUP BY name
ORDER BY totalsaleprice DESC;

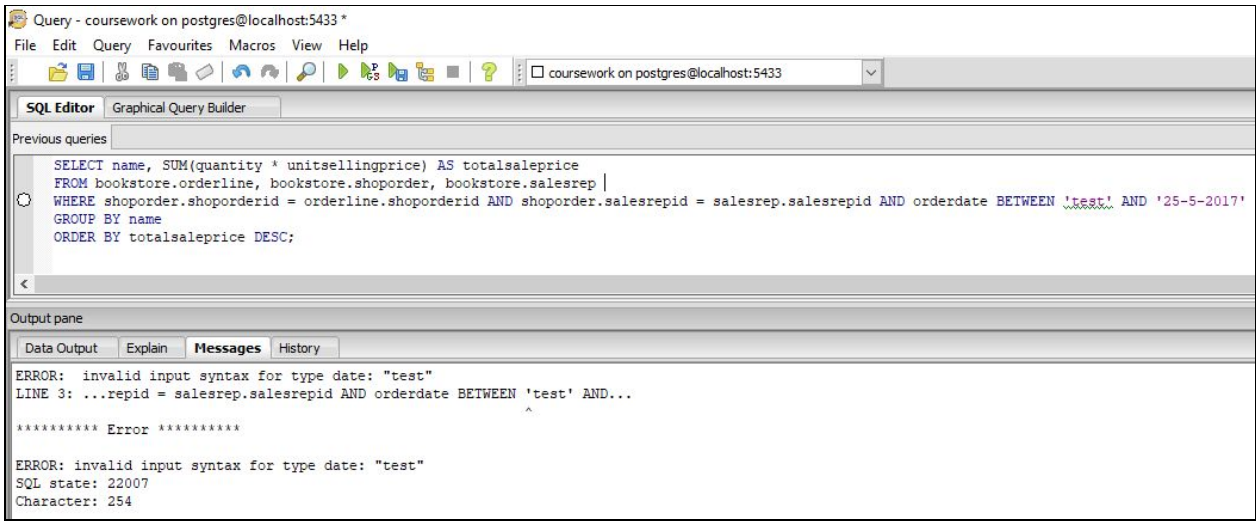


*SELECT name, SUM(quantity * unitsellingprice) AS totalsaleprice*
FROM bookstore.orderline, bookstore.shoporder, bookstore.salesrep
WHERE shoporder.shoporderid = orderline.shoporderid AND shoporder.salesrepid = salesrep.salesrepid AND orderdate BETWEEN '12-12-2017' AND '01-12-2017'
GROUP BY name
ORDER BY totalsaleprice DESC;

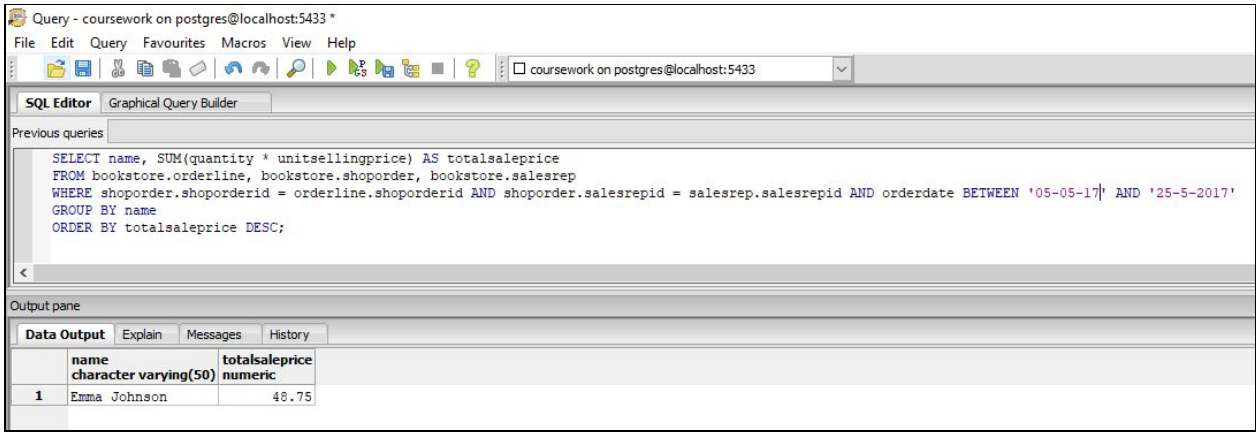


*SELECT name, SUM(quantity * unitsellingprice) AS totalsaleprice*
FROM bookstore.orderline, bookstore.shoporder, bookstore.salesrep
WHERE shoporder.shoporderid = orderline.shoporderid AND shoporder.salesrepid = salesrep.salesrepid AND orderdate BETWEEN 'test' AND '25-5-2017'
GROUP BY name
ORDER BY totalsaleprice DESC;

*SELECT name, SUM(quantity * unitsellingprice) AS totalsaleprice*

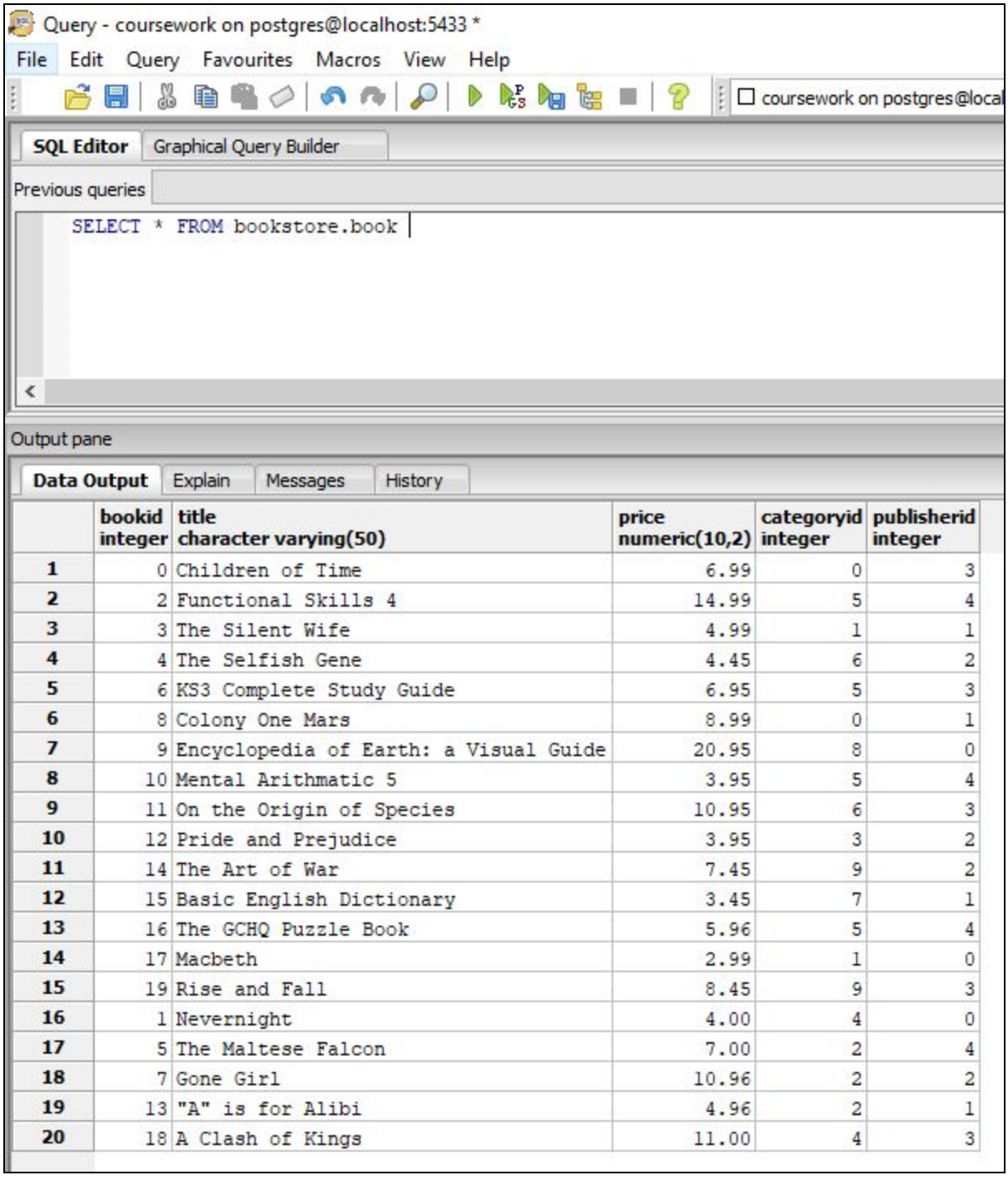


FROM bookstore.orderline, bookstore.shoporder, bookstore.salesrep
WHERE shoporder.shoporderid = orderline.shoporderid AND shoporder.salesrepid = salesrep.salesrepid AND orderdate BETWEEN '05-05-17' AND '25-5-2017'
GROUP BY name
ORDER BY totalsaleprice DESC;



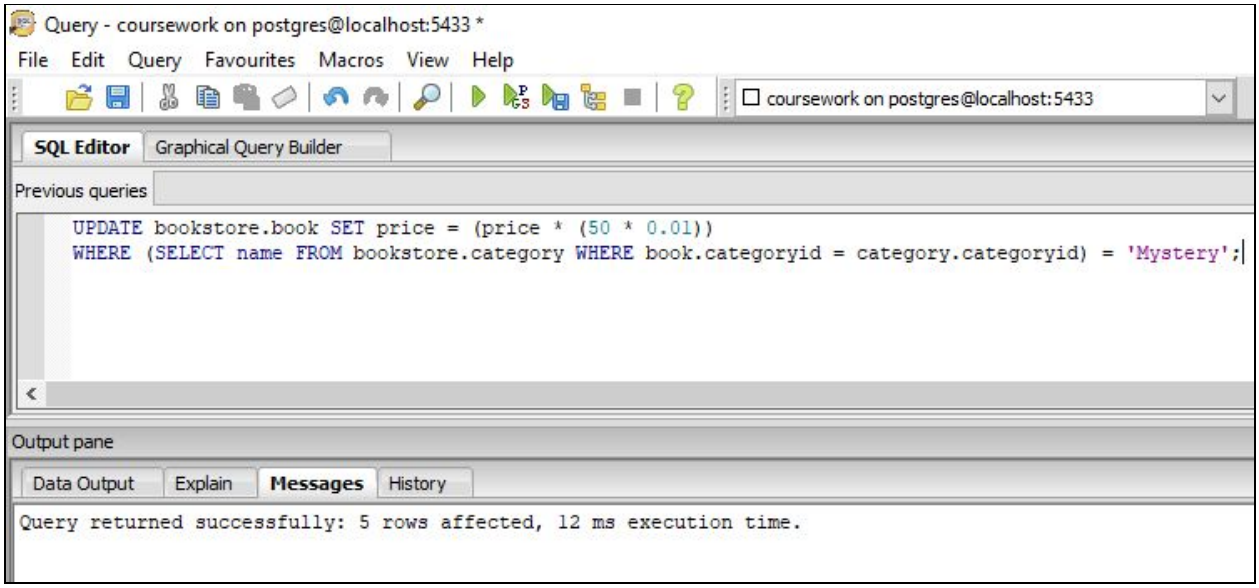
Task 7

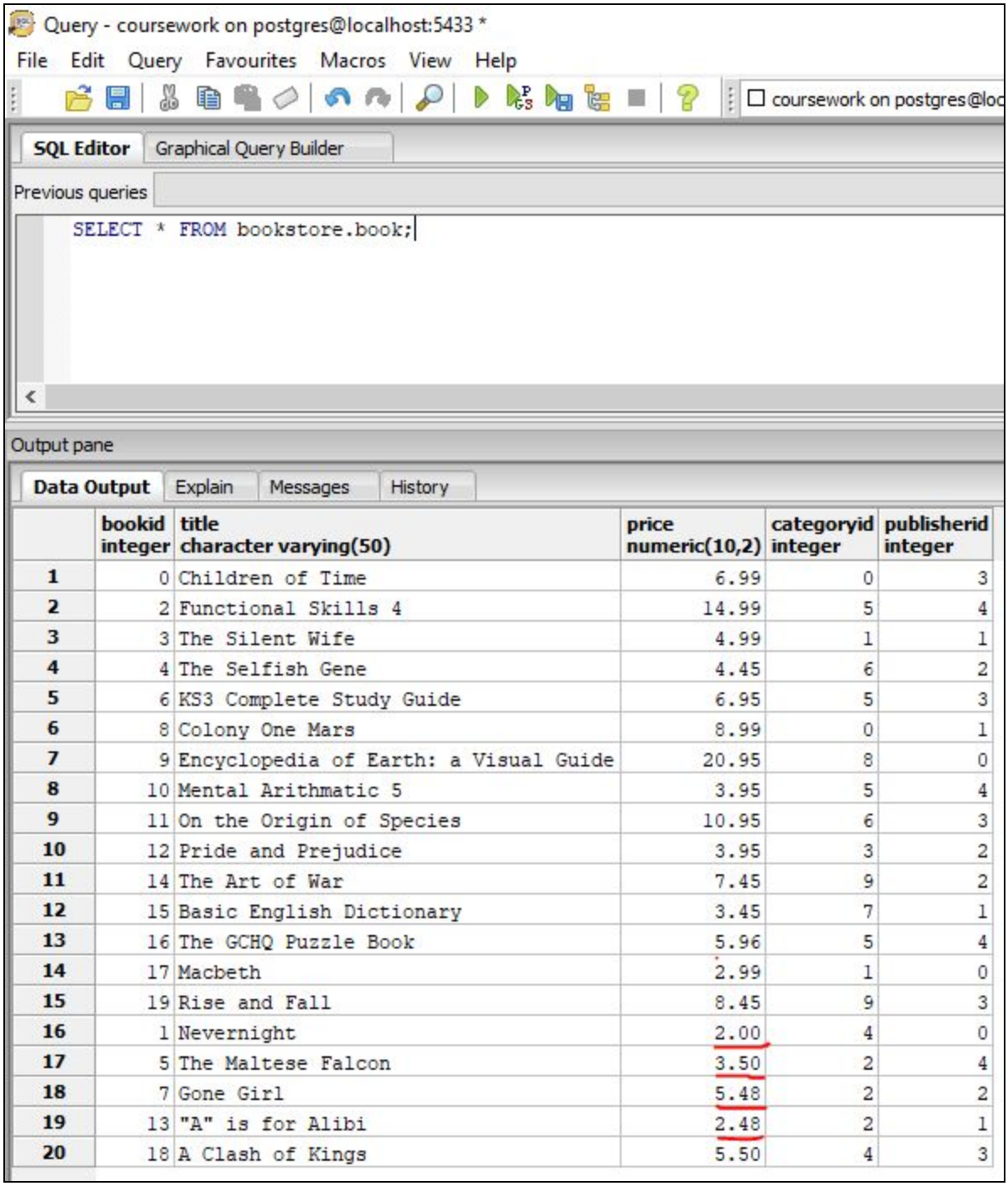
“Given a category ID and discount percentage, apply a discount to the standard price of all books in that category”



	bookid integer	title character varying(50)	price numeric(10,2)	categoryid integer	publisherid integer
1	0	Children of Time	6.99	0	3
2	2	Functional Skills 4	14.99	5	4
3	3	The Silent Wife	4.99	1	1
4	4	The Selfish Gene	4.45	6	2
5	6	KS3 Complete Study Guide	6.95	5	3
6	8	Colony One Mars	8.99	0	1
7	9	Encyclopedia of Earth: a Visual Guide	20.95	8	0
8	10	Mental Arithmatic 5	3.95	5	4
9	11	On the Origin of Species	10.95	6	3
10	12	Pride and Prejudice	3.95	3	2
11	14	The Art of War	7.45	9	2
12	15	Basic English Dictionary	3.45	7	1
13	16	The GCHQ Puzzle Book	5.96	5	4
14	17	Macbeth	2.99	1	0
15	19	Rise and Fall	8.45	9	3
16	1	Nevernight	4.00	4	0
17	5	The Maltese Falcon	7.00	2	4
18	7	Gone Girl	10.96	2	2
19	13	"A" is for Alibi	4.96	2	1
20	18	A Clash of Kings	11.00	4	3

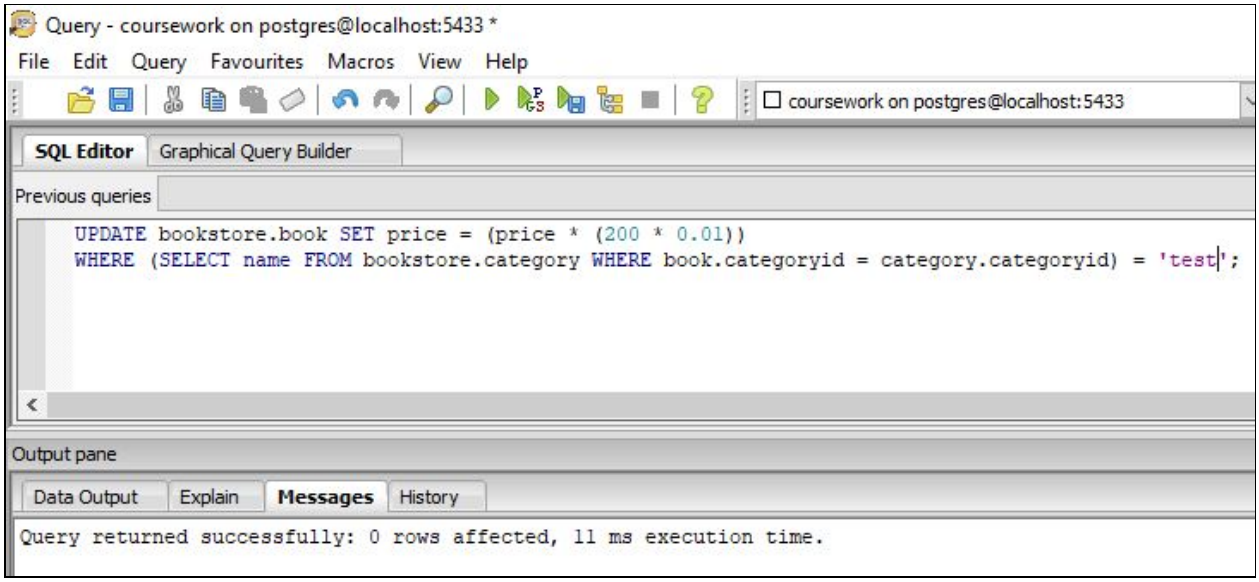
*UPDATE bookstore.book SET price = (price * (50 * 0.01))*
WHERE (SELECT name FROM bookstore.category WHERE book.categoryid = category.categoryid) = 'Mystery';





	bookid integer	title character varying(50)	price numeric(10,2)	categoryid integer	publisherid integer
1	0	Children of Time	6.99	0	3
2	2	Functional Skills 4	14.99	5	4
3	3	The Silent Wife	4.99	1	1
4	4	The Selfish Gene	4.45	6	2
5	6	KS3 Complete Study Guide	6.95	5	3
6	8	Colony One Mars	8.99	0	1
7	9	Encyclopedia of Earth: a Visual Guide	20.95	8	0
8	10	Mental Arithmatic 5	3.95	5	4
9	11	On the Origin of Species	10.95	6	3
10	12	Pride and Prejudice	3.95	3	2
11	14	The Art of War	7.45	9	2
12	15	Basic English Dictionary	3.45	7	1
13	16	The GCHQ Puzzle Book	5.96	5	4
14	17	Macbeth	2.99	1	0
15	19	Rise and Fall	8.45	9	3
16	1	Nevernight	2.00	4	0
17	5	The Maltese Falcon	3.50	2	4
18	7	Gone Girl	5.48	2	2
19	13	"A" is for Alibi	2.48	2	1
20	18	A Clash of Kings	5.50	4	3

*UPDATE bookstore.book SET price = (price * (200 * 0.01))*
WHERE (SELECT name FROM bookstore.category WHERE book.categoryid = category.categoryid) = 'test';


Query - coursework on postgres@localhost:5433 *
File Edit Query Favourites Macros View Help
coursework on postgres@localhost:5433
SQL Editor
Graphical Query Builder
Previous queries
UPDATE bookstore.book SET price = (price * (200 * 0.01))
WHERE (SELECT name FROM bookstore.category WHERE book.categoryid = category.categoryid) = 'test';
Output pane
Data Output
Explain
Messages
History
Query returned successfully: 0 rows affected, 11 ms execution time.