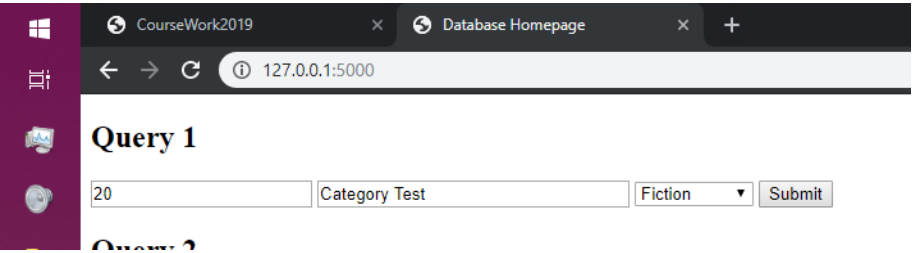
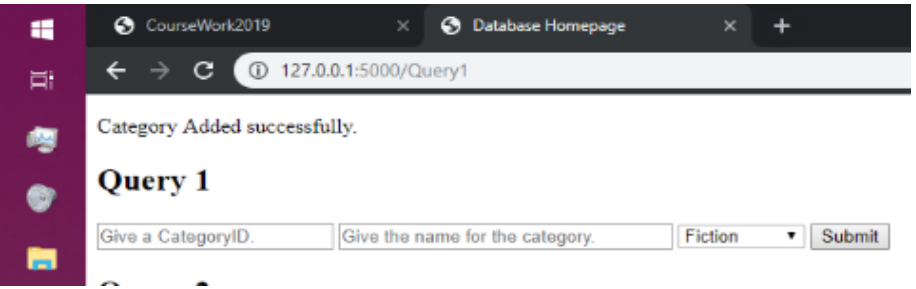
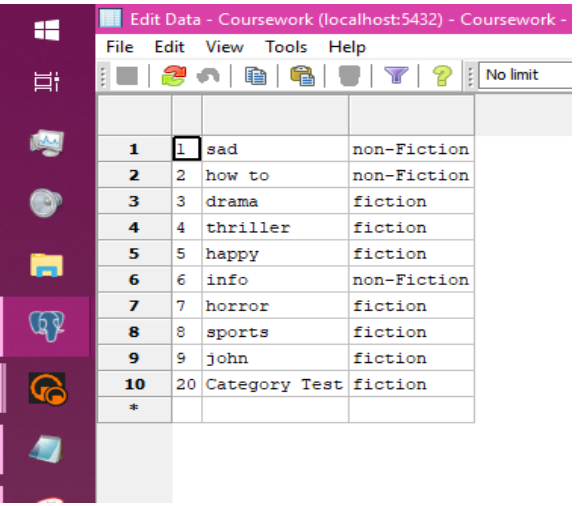
Query 1

Normal use, confirmation and data

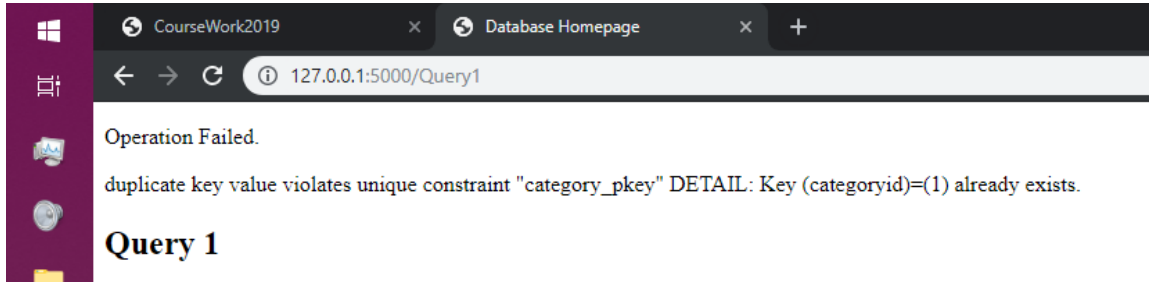




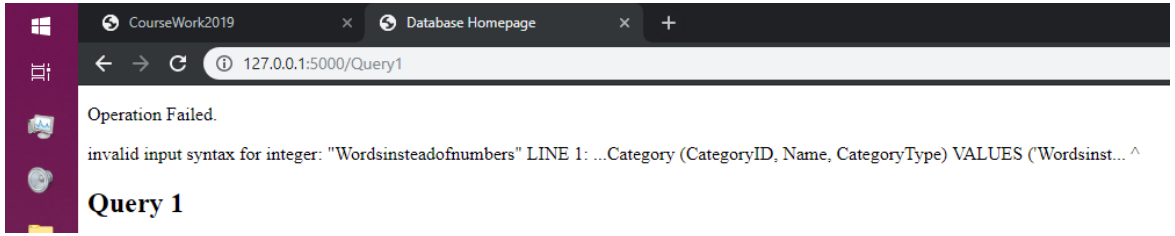


1	1	sad	non-Fiction
2	2	how to	non-Fiction
3	3	drama	fiction
4	4	thriller	fiction
5	5	happy	fiction
6	6	info	non-Fiction
7	7	horror	fiction
8	8	sports	fiction
9	9	john	fiction
10	20	Category Test	fiction
*			

Failure when CategoryID already exists



Failure when non integers is used as input



Query 1 PostgreSQL

Data Output | Explain | Messages | History

	categoryid integer	name character varying(50)	categorytype character varying(20)
1	1	sad	non-Fiction
2	2	how to	non-Fiction
3	3	drama	fiction
4	4	thriller	fiction
5	5	happy	fiction
6	6	info	non-Fiction
7	7	horror	fiction
8	8	sports	fiction
9	9	john	fiction

Before

Query 1*

```
Select * from Category;

INSERT INTO Category (CategoryID, name, CategoryType) VALUES ('20','Test Category','FICTION');

Select * from Category;
```

Query

Data Output | Explain | Messages | History

	categoryid integer	name character varying(50)	categorytype character varying(20)
1	1	sad	non-Fiction
2	2	how to	non-Fiction
3	3	drama	fiction
4	4	thriller	fiction
5	5	happy	fiction
6	6	info	non-Fiction
7	7	horror	fiction
8	8	sports	fiction
9	9	john	fiction
10	20	Test Category	fiction

After

```
Select * from Category;

INSERT INTO Category (CategoryID, name, CategoryType) VALUES ('1','CategoryIDthatalreadyexists','FICTION');

Select * from Category;
```

Output pane [Query 1]

Data Output | Explain | Messages | History

```
ERROR:  duplicate key value violates unique constraint "category_pkey"
DETAIL:  Key (categoryid)=(1) already exists.
********** Error **********

ERROR: duplicate key value violates unique constraint "category_pkey"
SQL state: 23505
Detail: Key (categoryid)=(1) already exists.
```

Failure from already existing category

Previous queries

Query 1*

```
Select * from Category;

INSERT INTO Category (CategoryID, name, CategoryType) VALUES ('21','Test Category','Not a category type');

Select * from Category;
```

Output pane [Query 1]

Data Output | Explain | Messages | History

```
ERROR:  new row for relation "category" violates check constraint "category_categorytype_check"
DETAIL:  Failing row contains (21, Test Category, Not a category type).

********** Error **********

ERROR: new row for relation "category" violates check constraint "category_categorytype_check"
SQL state: 23514
Detail: Failing row contains (21, Test Category, Not a category type).
```

Failure from invalid category type

Query 1*

```
Select * from Category;

INSERT INTO Category (CategoryID, name, CategoryType) VALUES ('not an integer','Test Category','FICTION');

Select * from Category;
```

Output pane [Query 1]

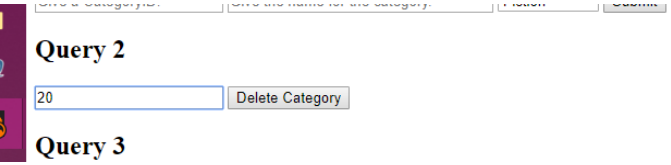
Data Output | Explain | Messages | History

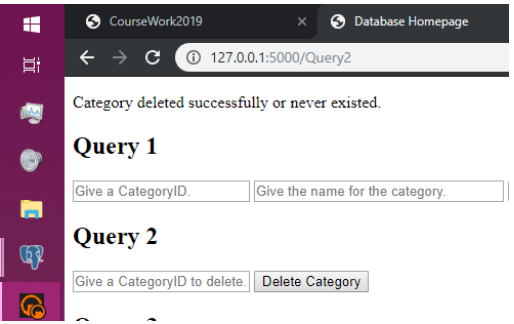
```
ERROR:  invalid input syntax for integer: "not an integer"
LINE 4: ...Category (CategoryID, name, CategoryType) VALUES ('not an in...
                                                             ^

********** Error **********

ERROR: invalid input syntax for integer: "not an integer"
SQL state: 22P02
Character: 92
```

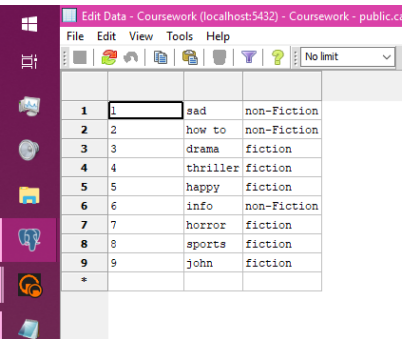
Failure from invalid input

Query 2

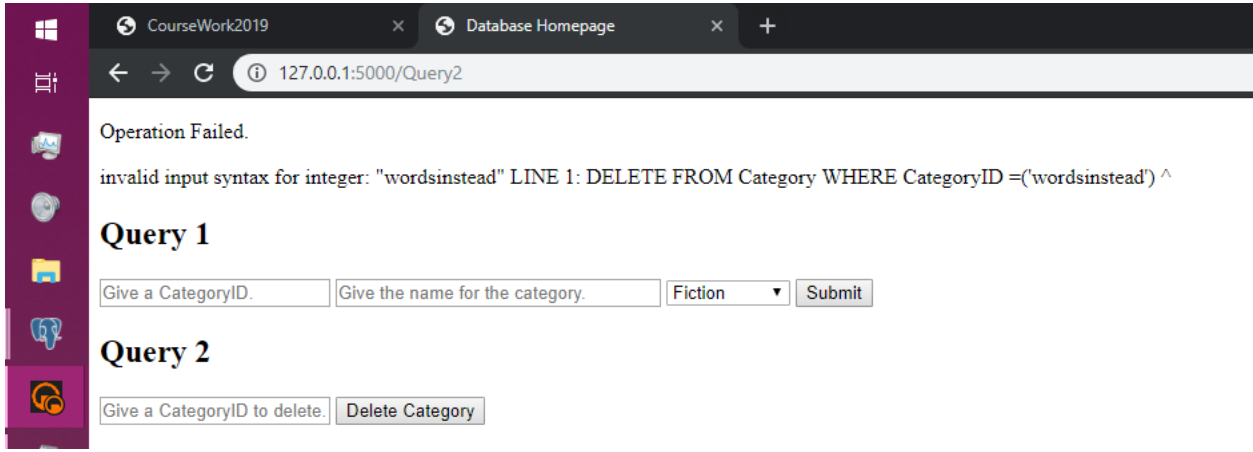
Normal use, confirmation, deletion of data







Failure when input is not valid



Query 2 Postgres

Output pane [Query 1]

Data Output | Explain | Messages | History

	categoryid integer	name character varying(50)	categorytype character varying(20)
1	1	sad	non-Fiction
2	2	how to	non-Fiction
3	3	drama	fiction
4	4	thriller	fiction
5	5	happy	fiction
6	6	info	non-Fiction
7	7	horror	fiction
8	8	sports	fiction
9	9	john	fiction
10	20	Test Category	fiction

Before and after deletion

```
Select * from Category;

DELETE FROM Category WHERE CategoryID = 20;

Select * from Category;
```

Output pane [Query 1]

Data Output | Explain | Messages | History

	categoryid integer	name character varying(50)	categorytype character varying(20)
1	1	sad	non-Fiction
2	2	how to	non-Fiction
3	3	drama	fiction
4	4	thriller	fiction
5	5	happy	fiction
6	6	info	non-Fiction
7	7	horror	fiction
8	8	sports	fiction
9	9	john	fiction

No change when deleting non existing category

Query 1*

```
Select * from Category;

DELETE FROM Category WHERE CategoryID = '20';

Select * from Category;
```

Output pane [Query 1]

Data Output | Explain | Messages | History

```
Query returned successfully: 0 rows affected, 12 msec execution time.
```

Failure from invalid input

```
Select * from Category;

DELETE FROM Category WHERE CategoryID = 'notanumber';

Select * from Category;
```

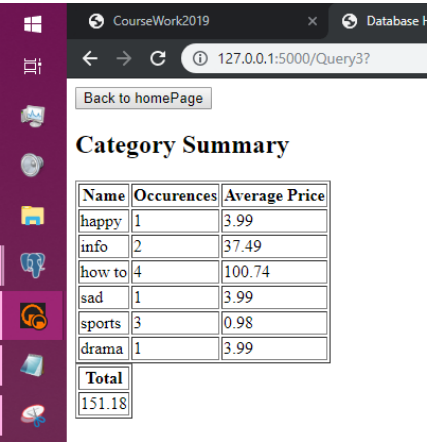
Output pane [Query 1]

Data Output | Explain | Messages | History

```
ERROR:  invalid input syntax for integer: "notanumber"
LINE 4: DELETE FROM Category WHERE CategoryID = 'notanumber';
                                                ^

********** Error **********

ERROR: invalid input syntax for integer: "notanumber"
SQL state: 22P02
Character: 70
```

Query 3

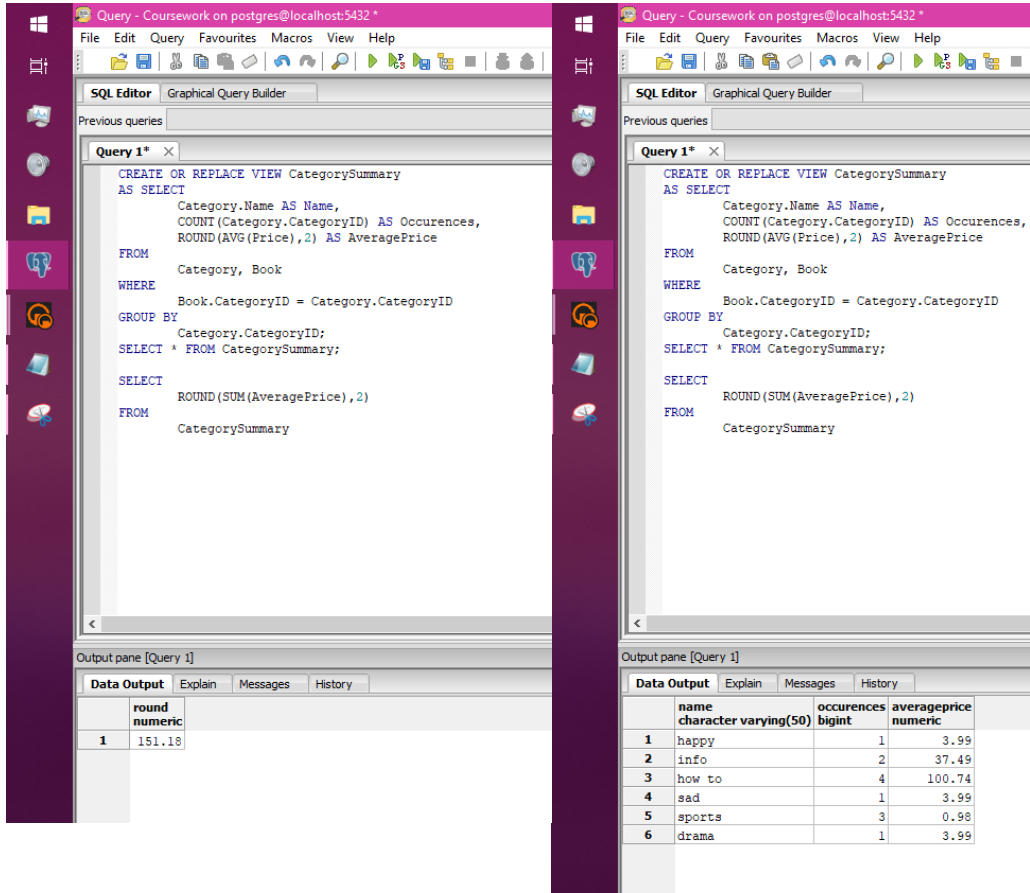
Normal use and result, has CREATE OR REPLACE so no issues if view already exists



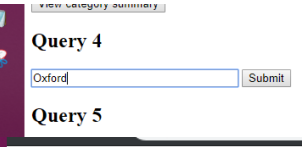
Name	Occurences	Average Price
happy	1	3.99
info	2	37.49
how to	4	100.74
sad	1	3.99
sports	3	0.98
drama	1	3.99

Total
151.18

Original queries in Postgresql



Query 4



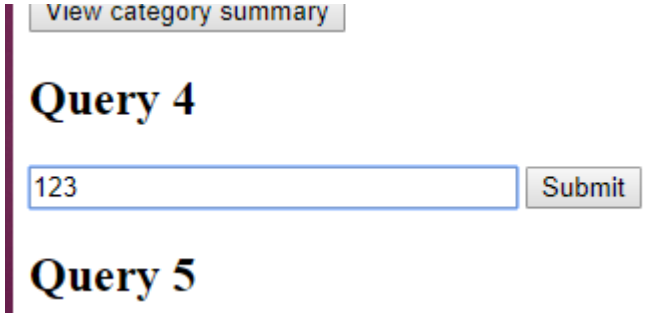
Input and result

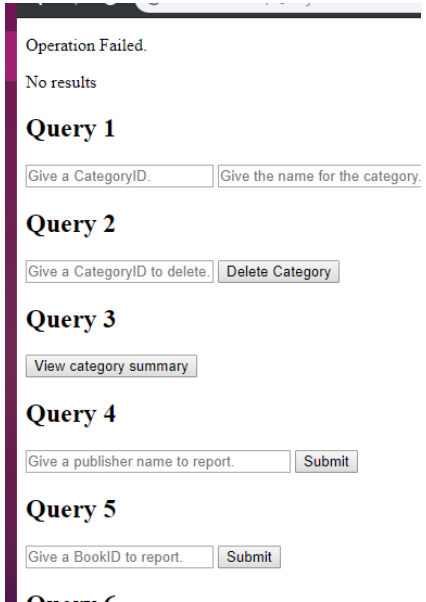
Back to homePage

Order Summary

BookID	Book Title	Total Orders Per Book	Total Quantity Per Book	Total Price	Total Unit Price	Month	Year
4	Dictionary	1	2	29.98	27.98	03	2019
1	How to not be stupid	1	1	2.99	1.99	03	2019
1	How to not be stupid	1	10	29.90	19.90	02	2017
3	John Cena AND how to find him	1	3	299.97	269.97	02	2017

Invalid input has no results and produces error





Query 4 Postgres

Normal use, case insensitive input

```
SELECT
        OrderLine.bookid AS "BookID",
        title AS "Book Title",
        count(OrderLine.bookid) AS "Total Orders",
        sum(quantity) AS "Total quantity",
        sum(price*quantity) AS "Total Price",
        sum(unitsellingprice*quantity) AS "Total Unit Selling Price",
        to_char(OrderDate, 'mm') AS "Month",
        to_char(OrderDate, 'yyyy') AS "Year"
FROM
        Publisher
        JOIN Book ON Book.PublisherID = Publisher.PublisherID
        JOIN OrderLine ON OrderLine.BookID = Book.bookID
        Join ShopOrder ON ShopOrder.ShopOrderID = OrderLine.ShopOrderID

WHERE
        LOWER(Publisher.name) = LOWER('oxford')
GROUP BY
        "Month",
        "Year",
        title,
        OrderLine.bookid
ORDER BY
        "Year" DESC,
        "Month"
```

Output pane [Query 1]

Data Output | Explain | Messages | History

	BookID integer	Book Title character varying(50)	Total Orders bigint	Total quantity bigint	Total Price numeric	Total Unit Selling Price numeric	Month text	Year text
1	4	Dictionary	1	2	29.98	27.98	03	2019
2	1	How to not be stupid	1	1	2.99	1.99	03	2019
3	1	How to not be stupid	1	10	29.90	19.90	02	2017
4	3	John Cena AND how to find him	1	3	299.97	269.97	03	2017

No result

```
SELECT
        OrderLine.bookid AS "BookID",
        title AS "Book Title",
        count(OrderLine.bookid) AS "Total Orders",
        sum(quantity) AS "Total quantity",
        sum(price*quantity) AS "Total Price",
        sum(unitsellingprice*quantity) AS "Total Unit Selling Price",
        to_char(OrderDate, 'mm') AS "Month",
        to_char(OrderDate, 'yyyy') AS "Year"
FROM
        Publisher
        JOIN Book ON Book.PublisherID = Publisher.PublisherID
        JOIN OrderLine ON OrderLine.BookID = Book.bookID
        Join ShopOrder ON ShopOrder.ShopOrderID = OrderLine.ShopOrderID

WHERE
        LOWER(Publisher.name) = LOWER('notfound')
GROUP BY
        "Month",
        "Year",
        title,
        OrderLine.bookid
ORDER BY
        "Year" DESC,
        "Month"
```

Output pane [Query 1]

Data Output | Explain | Messages | History

BookID integer	Book Title character varying(50)	Total Orders bigint	Total quantity bigint	Total Price numeric	Total Unit Selling Price numeric	Month text	Year text

Failure from invalid input

```
SELECT
        OrderLine.bookid AS "BookID",
        title AS "Book Title",
        count(OrderLine.bookid) AS "Total Orders",
        sum(quantity) AS "Total quantity",
        sum(price*quantity) AS "Total Price",
        sum(unitsellingprice*quantity) AS "Total Unit Selling Price",
        to_char(OrderDate, 'mm') AS "Month",
        to_char(OrderDate, 'yyyy') AS "Year"
FROM
        Publisher
        JOIN Book ON Book.PublisherID = Publisher.PublisherID
        JOIN OrderLine ON OrderLine.BookID = Book.bookID
        Join ShopOrder ON ShopOrder.ShopOrderID = OrderLine.ShopOrderID

WHERE
        LOWER(Publisher.name) = LOWER(20)
GROUP BY
        "Month",
        "Year",
        title,
        OrderLine.bookid
ORDER BY
        "Year" DESC,
        "Month"
```

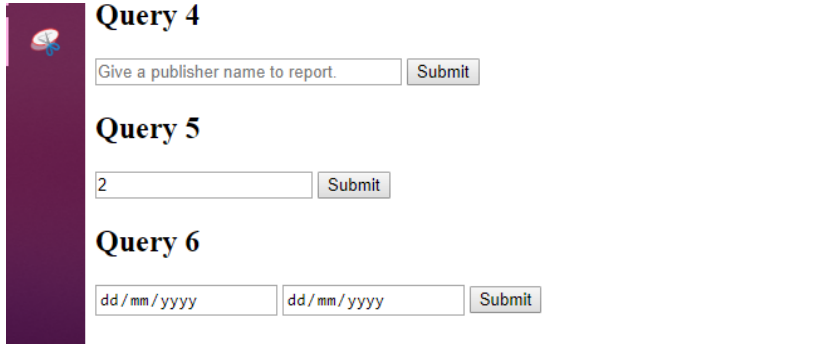
Output pane [Query 1]

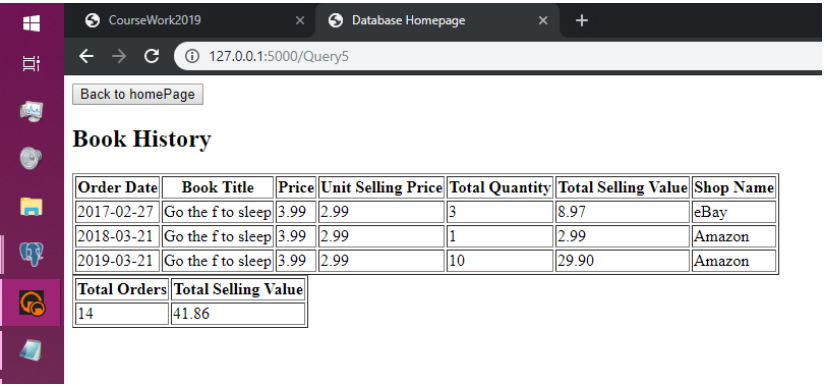
Data Output | Explain | Messages | History

```
ERROR:  function lower(integer) does not exist
LINE 17:  LOWER(Publisher.name) = LOWER(20)
                                  ^
HINT:  No function matches the given name and argument types. You might need to add explicit type casts.

********** Error **********

ERROR: function lower(integer) does not exist
SQL state: 42883
Hint: No function matches the given name and argument types. You might need to add explicit type casts.
Character: 555
```

Query 5

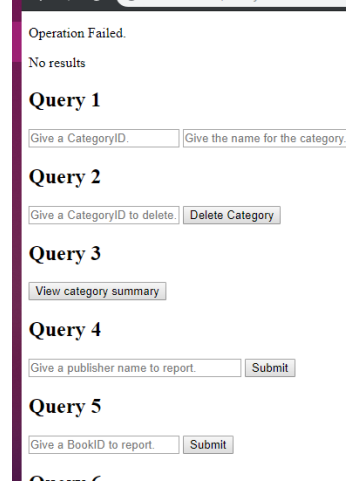
Normal use and result





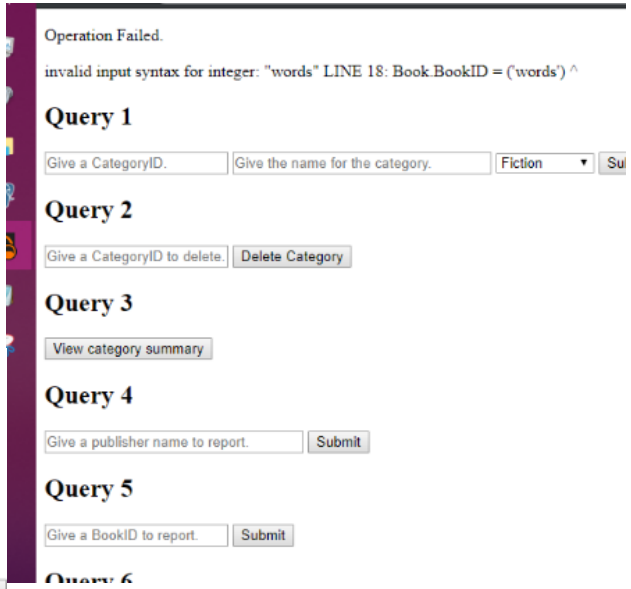
Order Date	Book Title	Price	Unit Selling Price	Total Quantity	Total Selling Value	Shop Name
2017-02-27	Go the f to sleep	3.99	2.99	3	8.97	eBay
2018-03-21	Go the f to sleep	3.99	2.99	1	2.99	Amazon
2019-03-21	Go the f to sleep	3.99	2.99	10	29.90	Amazon

Total Orders	Total Selling Value
14	41.86



Returns failure if there is no results

Failure when non integer inputted



Query 5 Postgres

```
CREATE OR REPLACE VIEW bookHistory
AS SELECT
        OrderDate AS OrderDate,
        Title AS BookTitle,
        Price AS Price,
        UnitSellingPrice AS UnitSellingPrice,
        SUM(Quantity) AS TotalQuantity,
        Quantity*UnitSellingPrice AS TotalSellingValue,
        Shop.Name AS ShopName
FROM
        Shop
JOIN ShopOrder ON ShopOrder.ShopID=Shop.ShopID
JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
JOIN Book ON Book.BookID=OrderLine.BookID
JOIN Publisher ON Publisher.PublisherID=Book.PublisherID
JOIN SalesRep ON SalesRep.SalesRepID=ShopOrder.SalesRepID
WHERE
        Book.BookID = 2
GROUP BY
        ShopOrder.OrderDate,
        Book.Title,
        Book.Price,
        OrderLine.UnitSellingPrice,
        OrderLine.Quantity,
        Shop.Name
ORDER BY
        OrderDate;

SELECT * FROM bookHistory;
```

Output pane [Query 1]

Data Output | Explain | Messages | History

	orderdate date	booktitle character varying(50)	price numeric(10,2)	unitsellingprice numeric(10,2)	totalquantity bigint	totalsellingvalue numeric	shopname character varying(50)
1	2017-02-27	Go the f to sleep	3.99	2.99	3	8.97	eBay
2	2018-03-21	Go the f to sleep	3.99	2.99	1	2.99	Amazon
3	2019-03-21	Go the f to sleep	3.99	2.99	10	29.90	Amazon

Both queries

```
        Shop.Name
ORDER BY
        OrderDate;

SELECT * FROM bookHistory;

SELECT
        SUM(TotalSellingValue) AS CompleteTotalValue,
        SUM(TotalQuantity) AS CompleteTotalQuantity
FROM
        bookHistory
```

Output pane [Query 1]

Data Output | Explain | Messages | History

	completetotalvalue numeric	completetotalquantity numeric
1	41.86	14

Failure from invalid input

```
CREATE OR REPLACE VIEW bookHistory
AS SELECT
        OrderDate AS OrderDate,
        Title AS BookTitle,
        Price AS Price,
        UnitSellingPrice AS UnitSellingPrice,
        SUM(Quantity) AS TotalQuantity,
        Quantity*UnitSellingPrice AS TotalSellingValue,
        Shop.Name AS ShopName
FROM
        Shop
JOIN ShopOrder ON ShopOrder.ShopID=Shop.ShopID
JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
JOIN Book ON Book.BookID=OrderLine.BookID
JOIN Publisher ON Publisher.PublisherID=Book.PublisherID
JOIN SalesRep ON SalesRep.SalesRepID=ShopOrder.SalesRepID
WHERE
        Book.BookID = 'notanumber'
GROUP BY
        ShopOrder.OrderDate,
        Book.Title,
        Book.Price,
        OrderLine.UnitSellingPrice,
        OrderLine.Quantity,
        Shop.Name
ORDER BY
        OrderDate;

SELECT
        SUM(TotalSellingValue) AS CompleteTotalValue,
        SUM(TotalQuantity) AS CompleteTotalQuantity
FROM
        bookHistory
```

Output pane [Query 1]

Data Output | Explain | Messages | History

```
ERROR:  invalid input syntax for integer: "notanumber"
LINE 18:  Book.BookID = 'notanumber'
                        ^
********** Error **********

ERROR: invalid input syntax for integer: "notanumber"
SQL state: 22P02
Character: 573
```

Returns nothing when there is no existing book

```
CREATE OR REPLACE VIEW bookHistory
AS SELECT
        OrderDate AS OrderDate,
        Title AS BookTitle,
        Price AS Price,
        UnitSellingPrice AS UnitSellingPrice,
        SUM(Quantity) AS TotalQuantity,
        Quantity*UnitSellingPrice AS TotalSellingValue,
        Shop.Name AS ShopName
FROM
        Shop
JOIN ShopOrder ON ShopOrder.ShopID=Shop.ShopID
JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
JOIN Book ON Book.BookID=OrderLine.BookID
JOIN Publisher ON Publisher.PublisherID=Book.PublisherID
JOIN SalesRep ON SalesRep.SalesRepID=ShopOrder.SalesRepID
WHERE
        Book.BookID = 0
GROUP BY
        ShopOrder.OrderDate,
        Book.Title,
        Book.Price,
        OrderLine.UnitSellingPrice,
        OrderLine.Quantity,
        Shop.Name
ORDER BY
        OrderDate;
SELECT * FROM Bookhistory;

SELECT
        SUM(TotalSellingValue) AS CompleteTotalValue,
        SUM(TotalQuantity) AS CompleteTotalQuantity
FROM
        bookHistory
```

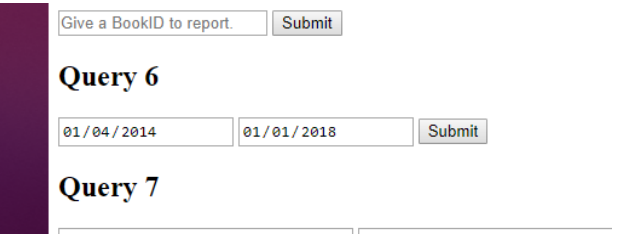
Output pane [Query 1]

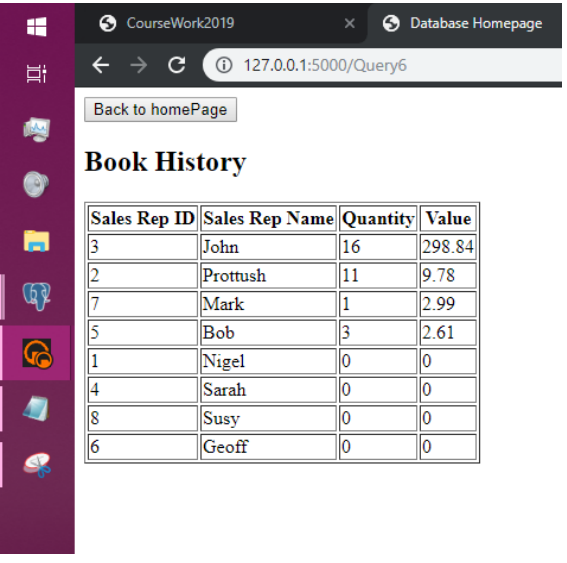
Data Output | Explain | Messages | History

orderdate date	booktitle character varying(50)	price numeric(10,2)	unitsellingprice numeric(10,2)	totalquantity bigint	totalsellingvalue numeric	shopname character varying(50)

Query 6

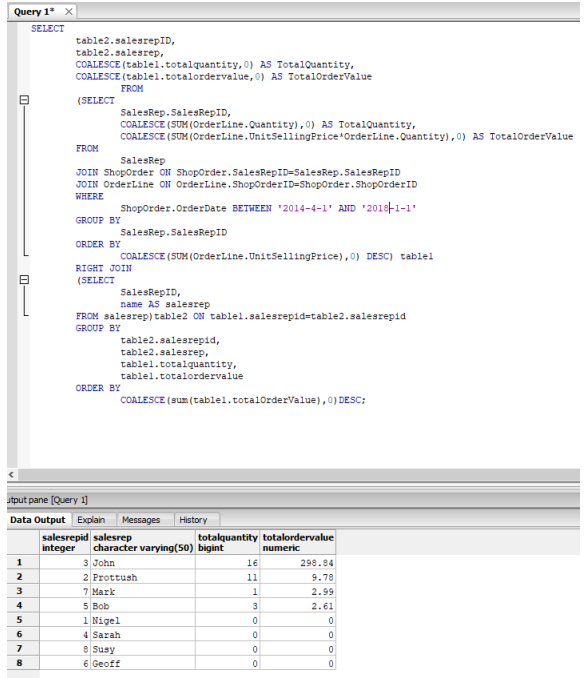
Regular use and results





Sales Rep ID	Sales Rep Name	Quantity	Value
3	John	16	298.84
2	Prottush	11	9.78
7	Mark	1	2.99
5	Bob	3	2.61
1	Nigel	0	0
4	Sarah	0	0
8	Susy	0	0
6	Geoff	0	0

Query in Postgresql

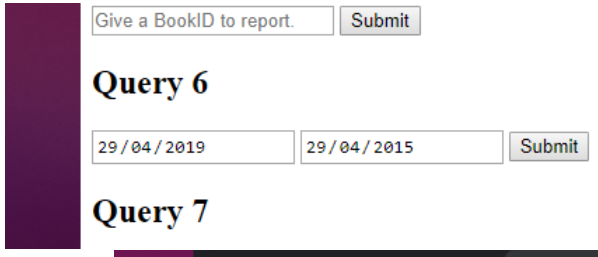


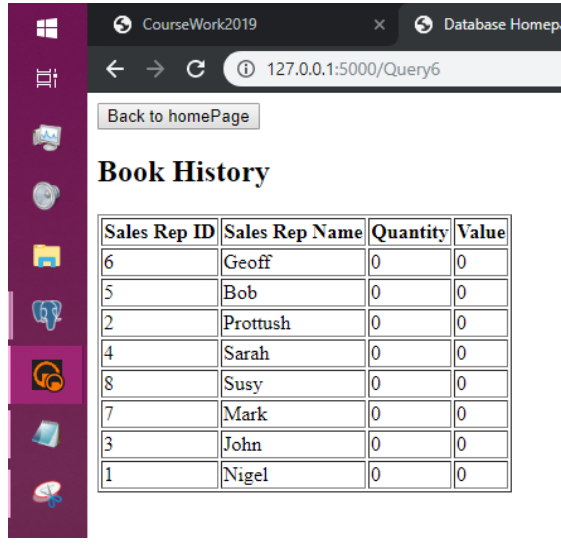
```
SELECT
        table2.salesrepID,
        table2.salesrep,
        COALESCE(table1.totalquantity,0) AS TotalQuantity,
        COALESCE(table1.totalordervalue,0) AS TotalOrderValue
                FROM
        (SELECT
                SalesRep.SalesRepID,
                COALESCE(SUM(OrderLine.Quantity),0) AS TotalQuantity,
                COALESCE(SUM(OrderLine.UnitSellingPrice*OrderLine.Quantity),0) AS TotalOrderValue
        FROM
                SalesRep
        JOIN ShopOrder ON ShopOrder.SalesRepID=SalesRep.SalesRepID
        JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
        WHERE
                ShopOrder.OrderDate BETWEEN '2014-4-1' AND '2018-1-1'
        GROUP BY
                SalesRep.SalesRepID
        ORDER BY
                COALESCE(SUM(OrderLine.UnitSellingPrice),0) DESC) table1
        RIGHT JOIN
        (SELECT
                SalesRepID,
                name AS salesrep
        FROM salesrep)table2 ON table1.salesrepid=table2.salesrepid
        GROUP BY
                table2.salesrepid,
                table2.salesrep,
                table1.totalquantity,
                table1.totalordervalue
        ORDER BY
                COALESCE(sum(table1.totalOrderValue),0)DESC;
```

Data Output | Explain | Messages | History

	salesrepid integer	salesrep character varying(50)	totalquantity bigint	totalordervalue numeric
1	3	John	16	298.84
2	2	Prottush	11	9.78
3	7	Mark	1	2.99
4	5	Bob	3	2.61
5	1	Nigel	0	0
6	4	Sarah	0	0
7	8	Susy	0	0
8	6	Geoff	0	0

Invalid input returns empty results





Sales Rep ID	Sales Rep Name	Quantity	Value
6	Geoff	0	0
5	Bob	0	0
2	Prottush	0	0
4	Sarah	0	0
8	Susy	0	0
7	Mark	0	0
3	John	0	0
1	Nigel	0	0

Query 6 Postgres

```
SELECT
        table2.salesrepID,
        table2.salesrep,
        COALESCE(table1.totalquantity,0) AS TotalQuantity,
        COALESCE(table1.totalordervalue,0) AS TotalOrderValue
                FROM
        (SELECT
                SalesRep.SalesRepID,
                COALESCE(SUM(OrderLine.Quantity),0) AS TotalQuantity,
                COALESCE(SUM(OrderLine.UnitSellingPrice*OrderLine.Quantity),0) AS TotalOrderValue
        FROM
                SalesRep
        JOIN ShopOrder ON ShopOrder.SalesRepID=SalesRep.SalesRepID
        JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
        WHERE
                ShopOrder.OrderDate BETWEEN '2014-4-1' AND '2018-1-1'
        GROUP BY
                SalesRep.SalesRepID
        ORDER BY
                COALESCE(SUM(OrderLine.UnitSellingPrice),0) DESC) table1
        RIGHT JOIN
        (SELECT
                SalesRepID,
                name AS salesrep
        FROM salesrep)table2 ON table1.salesrepid=table2.salesrepid
        GROUP BY
                table2.salesrepid,
                table2.salesrep,
                table1.totalquantity,
                table1.totalordervalue
        ORDER BY
                COALESCE(sum(table1.totalOrderValue),0)DESC;
```

Data Output | Explain | Messages | History

	salesrepid integer	salesrep character varying(50)	totalquantity bigint	totalordervalue numeric
1	3	John	16	298.84
2	2	Prottush	11	9.78
3	7	Mark	1	2.99
4	5	Bob	3	2.61
5	1	Nigel	0	0
6	4	Sarah	0	0
7	8	Susy	0	0
8	6	Geoff	0	0

Query functions with different date formats

```
SELECT
        table2.salesrepID,
        table2.salesrep,
        COALESCE(table1.totalquantity,0) AS TotalQuantity,
        COALESCE(table1.totalordervalue,0) AS TotalOrderValue
                FROM
        (SELECT
                SalesRep.SalesRepID,
                COALESCE(SUM(OrderLine.Quantity),0) AS TotalQuantity,
                COALESCE(SUM(OrderLine.UnitSellingPrice*OrderLine.Quantity),0) AS TotalOrderValue
        FROM
                SalesRep
        JOIN ShopOrder ON ShopOrder.SalesRepID=SalesRep.SalesRepID
        JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
        WHERE
                ShopOrder.OrderDate BETWEEN '1-1-2014' AND '2019-1-1'
        GROUP BY
                SalesRep.SalesRepID
        ORDER BY
                COALESCE(SUM(OrderLine.UnitSellingPrice),0) DESC) table1
        RIGHT JOIN
        (SELECT
                SalesRepID,
                name AS salesrep
        FROM salesrep)table2 ON table1.salesrepid=table2.salesrepid
        GROUP BY
                table2.salesrepid,
                table2.salesrep,
                table1.totalquantity,
                table1.totalordervalue
        ORDER BY
                COALESCE(sum(table1.totalOrderValue),0)DESC;
```

Data Output | Explain | Messages | History

	salesrepid integer	salesrep character varying(50)	totalquantity bigint	totalordervalue numeric
1	2	Prottush	17	762.67
2	8	Susy	12	505.88
3	3	John	16	298.84
4	7	Mark	1	2.99
5	5	Bob	3	2.61
6	1	Nigel	0	0
7	4	Sarah	0	0
8	6	Geoff	0	0

Returns nothing from invalid range

```
SELECT
        table2.salesrepID,
        table2.salesrep,
        COALESCE(table1.totalquantity,0) AS TotalQuantity,
        COALESCE(table1.totalordervalue,0) AS TotalOrderValue
                FROM
        (SELECT
                SalesRep.SalesRepID,
                COALESCE(SUM(OrderLine.Quantity),0) AS TotalQuantity,
                COALESCE(SUM(OrderLine.UnitSellingPrice*OrderLine.Quantity),0) AS TotalOrderValue
        FROM
                SalesRep
        JOIN ShopOrder ON ShopOrder.SalesRepID=SalesRep.SalesRepID
        JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
        WHERE
                ShopOrder.OrderDate BETWEEN '2019-1-1' AND '2015-1-1'
        GROUP BY
                SalesRep.SalesRepID
        ORDER BY
                COALESCE(SUM(OrderLine.UnitSellingPrice),0) DESC) table1
        RIGHT JOIN
        (SELECT
                SalesRepID,
                name AS salesrep
        FROM salesrep)table2 ON table1.salesrepid=table2.salesrepid
        GROUP BY
                table2.salesrepid,
                table2.salesrep,
                table1.totalquantity,
                table1.totalordervalue
        ORDER BY
                COALESCE(sum(table1.totalOrderValue),0)DESC;
```

Data Output | Explain | Messages | History

	salesrepid integer	salesrep character varying(50)	totalquantity bigint	totalordervalue numeric
1	6	Geoff	0	0
2	5	Bob	0	0
3	2	Prottush	0	0
4	4	Sarah	0	0
5	8	Susy	0	0
6	7	Mark	0	0
7	3	John	0	0
8	1	Nigel	0	0

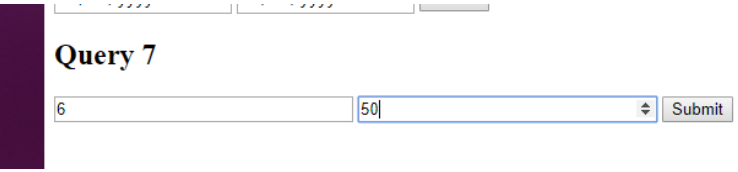
Failure from invalid input

```
SELECT
        table2.salesrepID,
        table2.salesrep,
        COALESCE(table1.totalquantity,0) AS TotalQuantity,
        COALESCE(table1.totalordervalue,0) AS TotalOrderValue
                FROM
        (SELECT
                SalesRep.SalesRepID,
                COALESCE(SUM(OrderLine.Quantity),0) AS TotalQuantity,
                COALESCE(SUM(OrderLine.UnitSellingPrice*OrderLine.Quantity),0) AS TotalOrderValue
        FROM
                SalesRep
        JOIN ShopOrder ON ShopOrder.SalesRepID=SalesRep.SalesRepID
        JOIN OrderLine ON OrderLine.ShopOrderID=ShopOrder.ShopOrderID
        WHERE
                ShopOrder.OrderDate BETWEEN '1' AND '2019-1-1'
        GROUP BY
                SalesRep.SalesRepID
        ORDER BY
                COALESCE(SUM(OrderLine.UnitSellingPrice),0) DESC) table1
        RIGHT JOIN
        (SELECT
                SalesRepID,
                name AS salesrep
        FROM salesrep)table2 ON table1.salesrepid=table2.salesrepid
        GROUP BY
                table2.salesrepid,
                table2.salesrep,
                table1.totalquantity,
                table1.totalordervalue
        ORDER BY
                COALESCE(sum(table1.totalOrderValue),0)DESC;
```

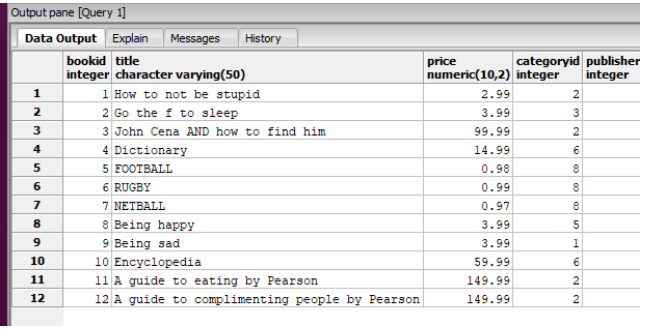
Data Output | Explain | Messages | History

```
ERROR:  invalid input syntax for type date: "1"
LINE 16:   ShopOrder.OrderDate BETWEEN '1' AND '2019-1-1'
                                       ^

******** Error **********

ERROR: invalid input syntax for type date: "1"
SQL state: 22007
Character: 527
```

Query 7

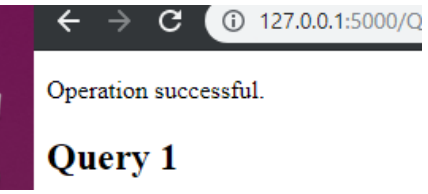


Regular use

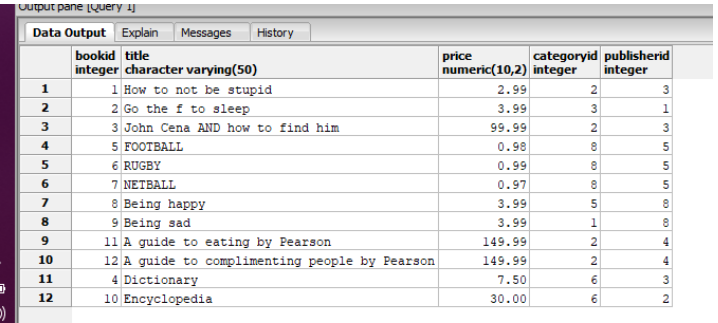


	bookid integer	title character varying(50)	price numeric(10,2)	categoryid integer	publisher integer
1	1	How to not be stupid	2.99	2	
2	2	Go the f to sleep	3.99	3	
3	3	John Cena AND how to find him	99.99	2	
4	4	Dictionary	14.99	6	
5	5	FOOTBALL	0.98	8	
6	6	RUGBY	0.99	8	
7	7	NETBALL	0.97	8	
8	8	Being happy	3.99	5	
9	9	Being sad	3.99	1	
10	10	Encyclopedia	59.99	6	
11	11	A guide to eating by Pearson	149.99	2	
12	12	A guide to complimenting people by Pearson	149.99	2	

Data before



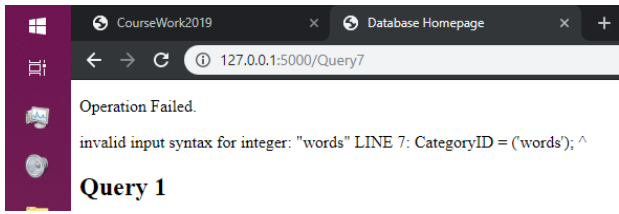
Confirmation

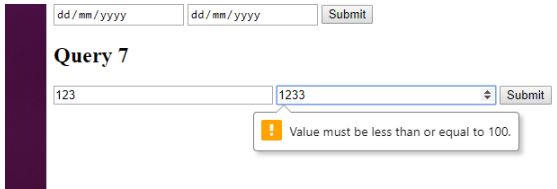


	bookid integer	title character varying(50)	price numeric(10,2)	categoryid integer	publisherid integer
1	1	How to not be stupid	2.99	2	3
2	2	Go the f to sleep	3.99	3	1
3	3	John Cena AND how to find him	99.99	2	3
4	5	FOOTBALL	0.98	8	5
5	6	RUGBY	0.99	8	5
6	7	NETBALL	0.97	8	5
7	8	Being happy	3.99	5	8
8	9	Being sad	3.99	1	8
9	11	A guide to eating by Pearson	149.99	2	4
10	12	A guide to complimenting people by Pearson	149.99	2	4
11	4	Dictionary	7.50	6	3
12	10	Encyclopedia	30.00	6	2

Data after

Failure when integer not given





Only accepts numbers from 1-100, no decimals.

Query 7 Postgres

Data Output | Explain | Messages | History

	bookid integer	title character varying(50)	price numeric(10,2)	categoryid integer	publisherid integer
1	1	How to not be stupid	2.99	2	3
2	2	Go the f to sleep	3.99	3	1
3	3	John Cena AND how to find him	99.99	2	3
4	5	FOOTBALL	0.98	8	5
5	6	RUGBY	0.99	8	5
6	7	NETBALL	0.97	8	5
7	8	Being happy	3.99	5	8
8	9	Being sad	3.99	1	8
9	11	A guide to eating by Pearson	149.99	2	4
10	12	A guide to complimenting people by Pearson	149.99	2	4
11	4	Dictionary	7.50	6	3
12	10	Encyclopedia	30.00	6	2

Data before

```
SELECT * FROM Book;

UPDATE
	book
SET
	Price = (Price-(Price*20)/100)
WHERE
	CategoryID = 2;

SELECT * FROM Book;
```

Query (accepts decimals)

ut pane [Query 1]

ata Output | Explain | Messages | History

	bookid integer	title character varying(50)	price numeric(10,2)	categoryid integer	publisherid integer
1	2	Go the f to sleep	3.99	3	1
2	5	FOOTBALL	0.98	8	5
3	6	RUGBY	0.99	8	5
4	7	NETBALL	0.97	8	5
5	8	Being happy	3.99	5	8
6	9	Being sad	3.99	1	8
7	4	Dictionary	7.50	6	3
8	10	Encyclopedia	30.00	6	2
9	1	How to not be stupid	2.39	2	3
10	3	John Cena AND how to find him	79.99	2	3
11	11	A guide to eating by Pearson	119.99	2	4
12	12	A guide to complimenting people by Pearson	119.99	2	4

Data after

Failure from invalid input

```
SELECT * FROM Book;

UPDATE
	book
SET
	Price = (Price-(Price*'not an int')/100)
WHERE
	CategoryID = 2;

SELECT * FROM Book;
```

Output pane [Query 1]

Data Output | Explain | Messages | History

```
ERROR:  invalid input syntax for type numeric: "not an int"
LINE 8:  Price = (Price-(Price*'not an int')/100)
                               ^

********** Error **********

ERROR: invalid input syntax for type numeric: "not an int"
SQL state: 22P02
Character: 71
```

```
SELECT * FROM Book;

UPDATE
	book
SET
	Price = (Price-(Price*20)/100)
WHERE
	CategoryID = 'not an int';

SELECT * FROM Book;
```

Output pane [Query 1]

Data Output | Explain | Messages | History

```
ERROR:  invalid input syntax for integer: "not an int"
LINE 10:  CategoryID = 'not an int';
                       ^

********** Error **********

ERROR: invalid input syntax for integer: "not an int"
SQL state: 22P02
Character: 102
```