Information Technology in Business and Society - SQL Part II

Pearl Yu

Querying relational databases

SQL Tryit Editor v1.6

http://goo.gl/iBpPLO

Recap

SQL **SELECT FROM**

- All queries follow the same basic pattern:

```
SELECT [columns] FROM [table]
```

- E.g. What're in the Customers table?

```
SELECT * FROM Customers
```

- **Select particular columns:**
  For example, if we only want to see the list of product ID and price

```
SELECT ProductID, Price
FROM Products;
```

SQL

- **Adding in some conditions**

```
SELECT *
FROM Customers
WHERE [condition]
```

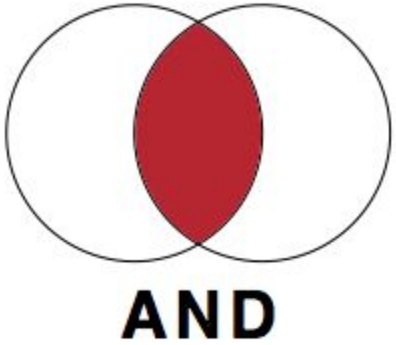
- Find the cheaper products (price < $10)

```
SELECT *
FROM Products
WHERE Price < 10;
```

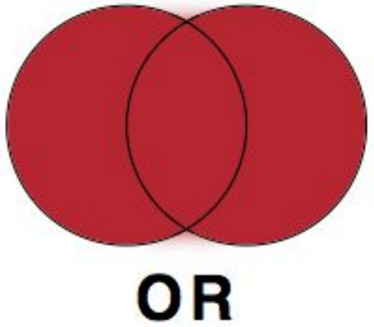
SQL **AND, OR**

- Now let’s **combine conditions.**

```
SELECT *
FROM Products
WHERE Price < 10 AND Price > 5;
```



```
SELECT *
FROM Products
WHERE Price < 10 OR Price > 100;
```



SQL **AND, OR**

- Now let’s **combine AND, OR operators.**
- E.g. Return the product records whose price is between 2 and 10, or the price is 97.

```
SELECT *
FROM Products
WHERE (Price >2 AND Price <10) OR Price = 97;
```

SQL **COUNT**

- Now let’s look at **functions.**
  - Function is a computational activity.
- E.g, how many products have a price less than $10?

```
SELECT COUNT(ProductId)
FROM Products
WHERE Price < 10;
```

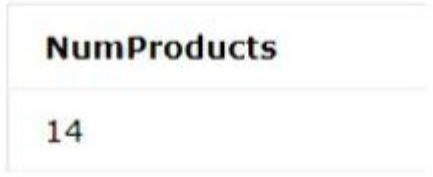
COUNT is a function that counts the number of rows that satisfy the criteria specified with the WHERE clause.

- Other functions: sum, avg, min, max…

SQL **AS**

- We can also **rename the column**.
- E.g, how many products have a price less than $10?

```
SELECT COUNT(ProductId) AS NumProducts
FROM Products
WHERE Price < 10;
```

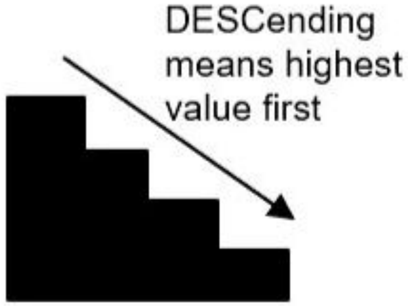


NumProducts
14

SQL **ORDER BY**

- How can we **sort** the products by price?

```
SELECT *
FROM Products
ORDER BY Price DESC;
```



- E.g. Order product name alphabetically (ASC order)?

```
SELECT *
FROM Products
ORDER BY ProductName ASC;
```

SQL **LIKE**

- How to **search** for the rows that contain particular words in a column?

```
SELECT * FROM [table]
WHERE [column] LIKE [pattern];
```

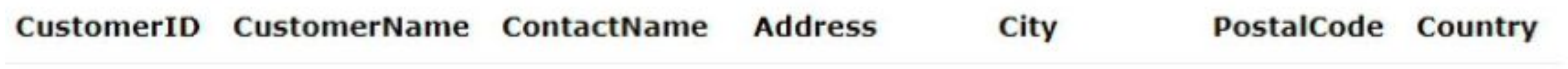
- E.g. Let’s find the rows that contains data ending with ‘bottles’ in Unit column.

```
SELECT *
FROM Products
WHERE Unit LIKE ’%bottles’;
```

% is a **wildcard** character, which can represent anything. It position matters!

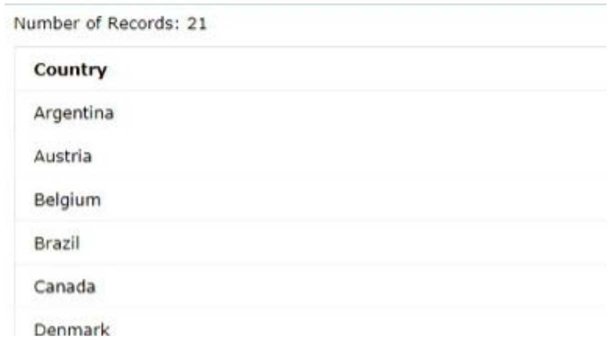
SQL **GROUP BY**

- GROUP BY groups rows with the same values into a summary row.
- What're the different countries where the customers are located?



CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country

```
SELECT Country
FROM Customers
GROUP BY Country;
```



Number of Records: 21

Country
Argentina
Austria
Belgium
Brazil
Canada
Denmark

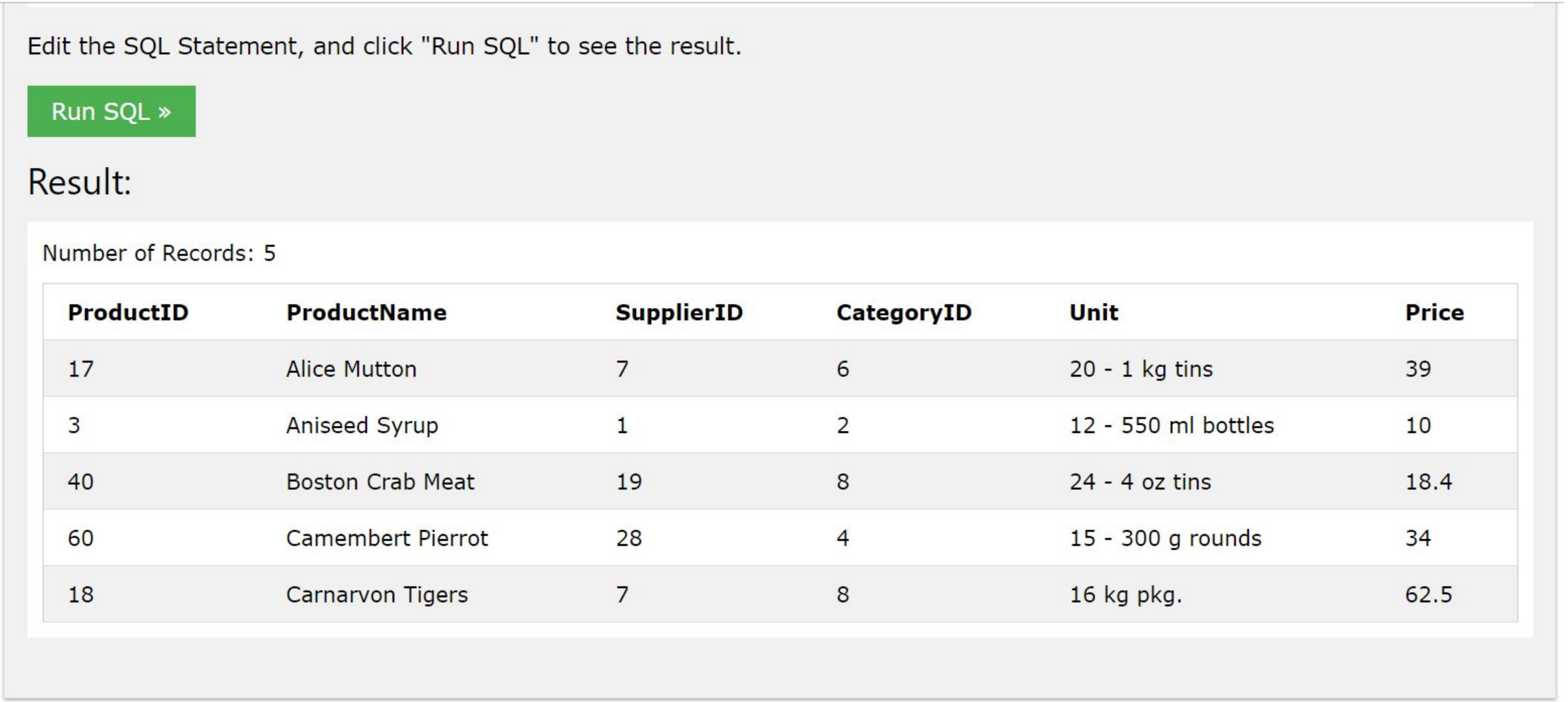
NEW STUFF

SQL **LIMIT**

- A way to get the “top N” rows from a query
- What is this query?

```
SELECT *
FROM Products
ORDER BY ProductName ASC
LIMIT 5;
```

LIMIT keyword will restrict the rows to just the first 5. - Only meaningful when you’ve ordered the rows by something with an ORDER BY.



Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 5

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
17	Alice Mutton	7	6	20 - 1 kg tins	39
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.4
60	Camembert Pierrot	28	4	15 - 300 g rounds	34
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5

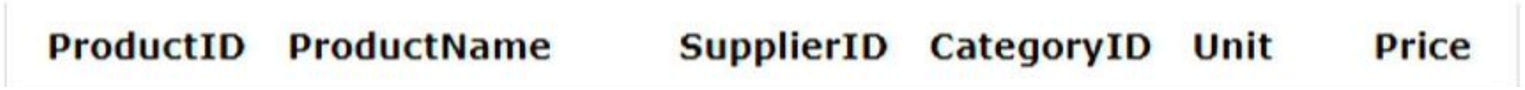
SQL **CASE- END**

- Goes through the condition(s), returns a value when the first condition is met

```
SELECT [field(s)],
 CASE WHEN [condition]
           THEN [output if true]
           ELSE [output if false]
 END
FROM [table];
```

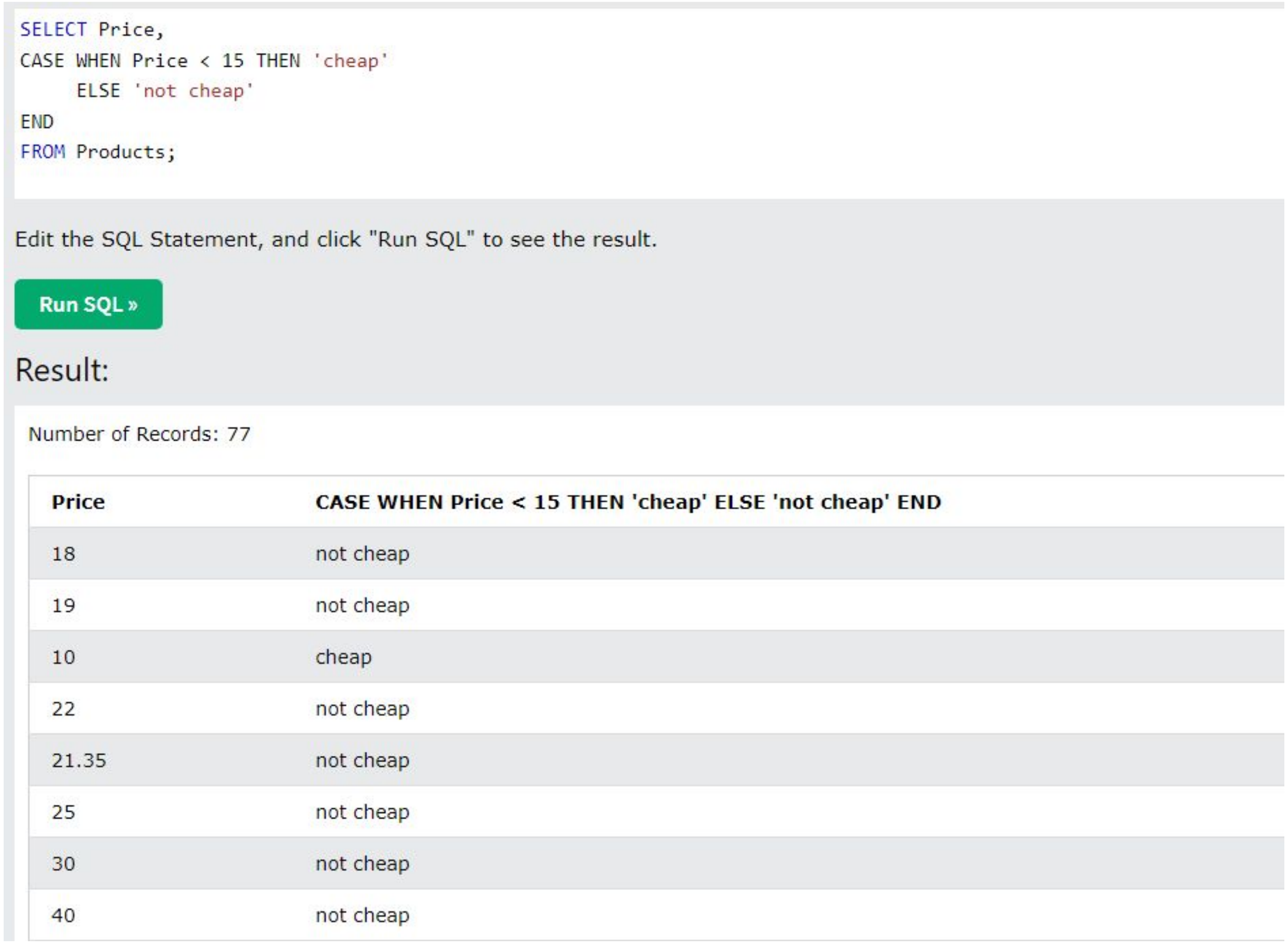
SQL CASE- END

- Let’s try it.
- Create a new column based on price that lists whether a product is cheap or not.



ProductID	ProductName	SupplierID	CategoryID	Unit	Price

```
SELECT Price,
 CASE WHEN Price < 15 THEN ‘cheap’
      ELSE ‘not cheap’
 END
FROM Products;
```


```sql
SELECT Price,
CASE WHEN Price < 15 THEN 'cheap'
     ELSE 'not cheap'
END
FROM Products;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

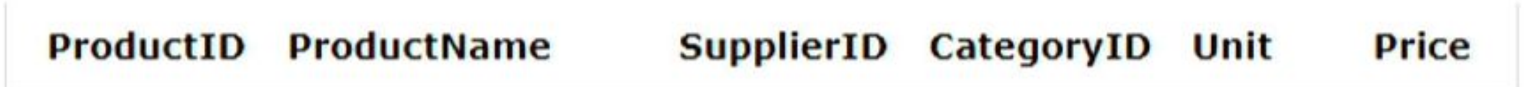
Result:

Number of Records: 77

Price	CASE WHEN Price < 15 THEN 'cheap' ELSE 'not cheap' END
18	not cheap
19	not cheap
10	cheap
22	not cheap
21.35	not cheap
25	not cheap
30	not cheap
40	not cheap

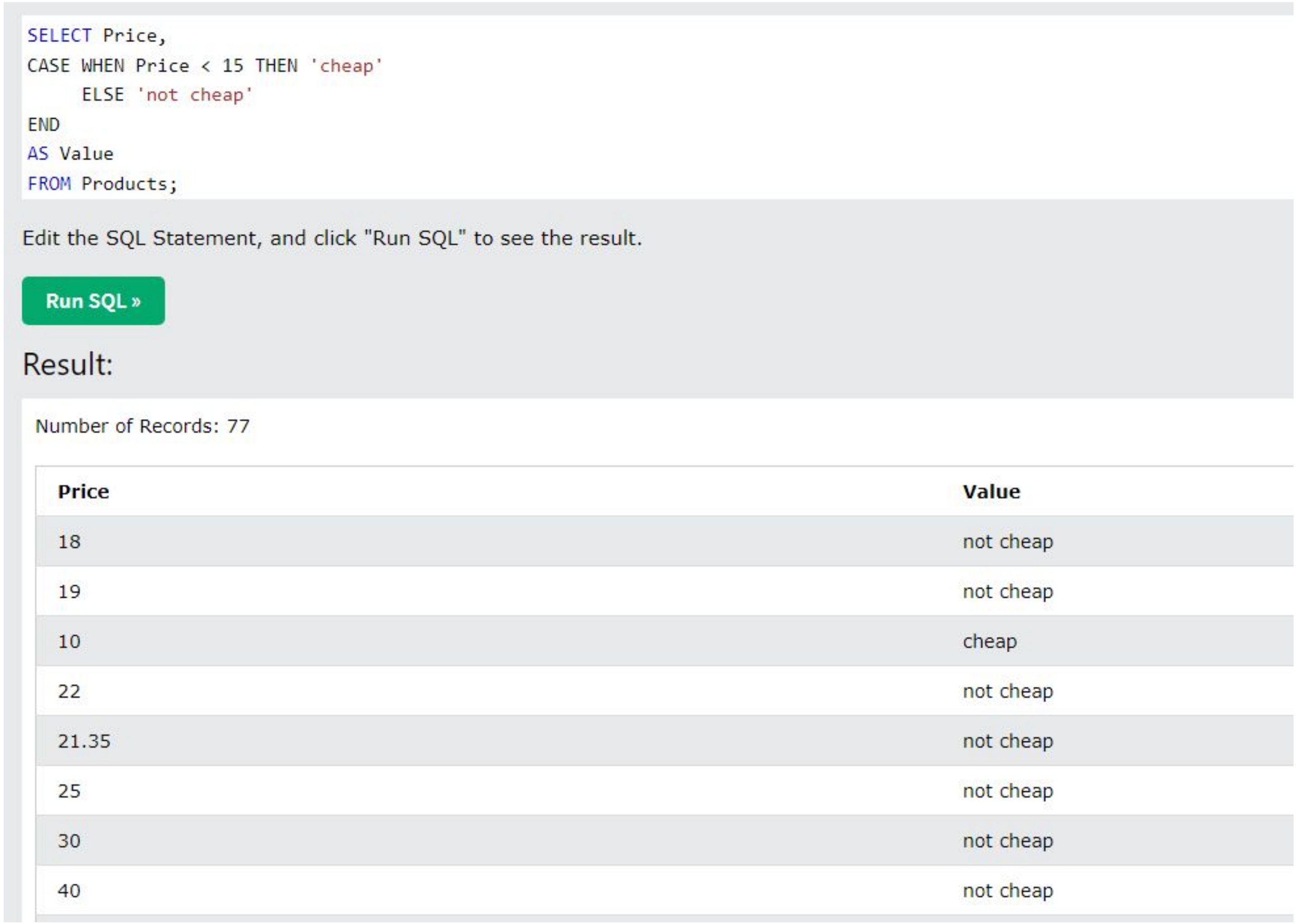
SQL **CASE- END**

- Create a new column based on price that lists whether a product is cheap or not.
- **Rename the new field** to ‘Value’



ProductID	ProductName	SupplierID	CategoryID	Unit	Price

```
SELECT Price,
 CASE WHEN Price < 15 THEN ‘cheap’
      ELSE ‘not cheap’
 END AS ‘Value’
FROM Products;
```



```
SELECT Price,
CASE WHEN Price < 15 THEN 'cheap'
     ELSE 'not cheap'
END
AS Value
FROM Products;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

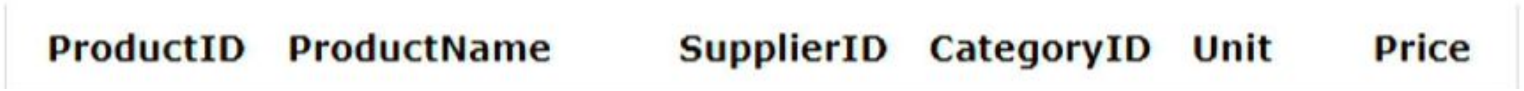
Result:

Number of Records: 77

Price	Value
18	not cheap
19	not cheap
10	cheap
22	not cheap
21.35	not cheap
25	not cheap
30	not cheap
40	not cheap

SQL **CASE- END**

- Create a new column based on price that lists whether a product is cheap or not. Rename the new field to ‘Value’.
- What if we’d like to **sort the output based on price**?



ProductID	ProductName	SupplierID	CategoryID	Unit	Price

```
SELECT Price,
 CASE WHEN Price < 15 THEN ‘cheap’
      ELSE ‘not cheap’
 END AS ‘Value’
FROM Products
ORDER BY Price DESC;
```

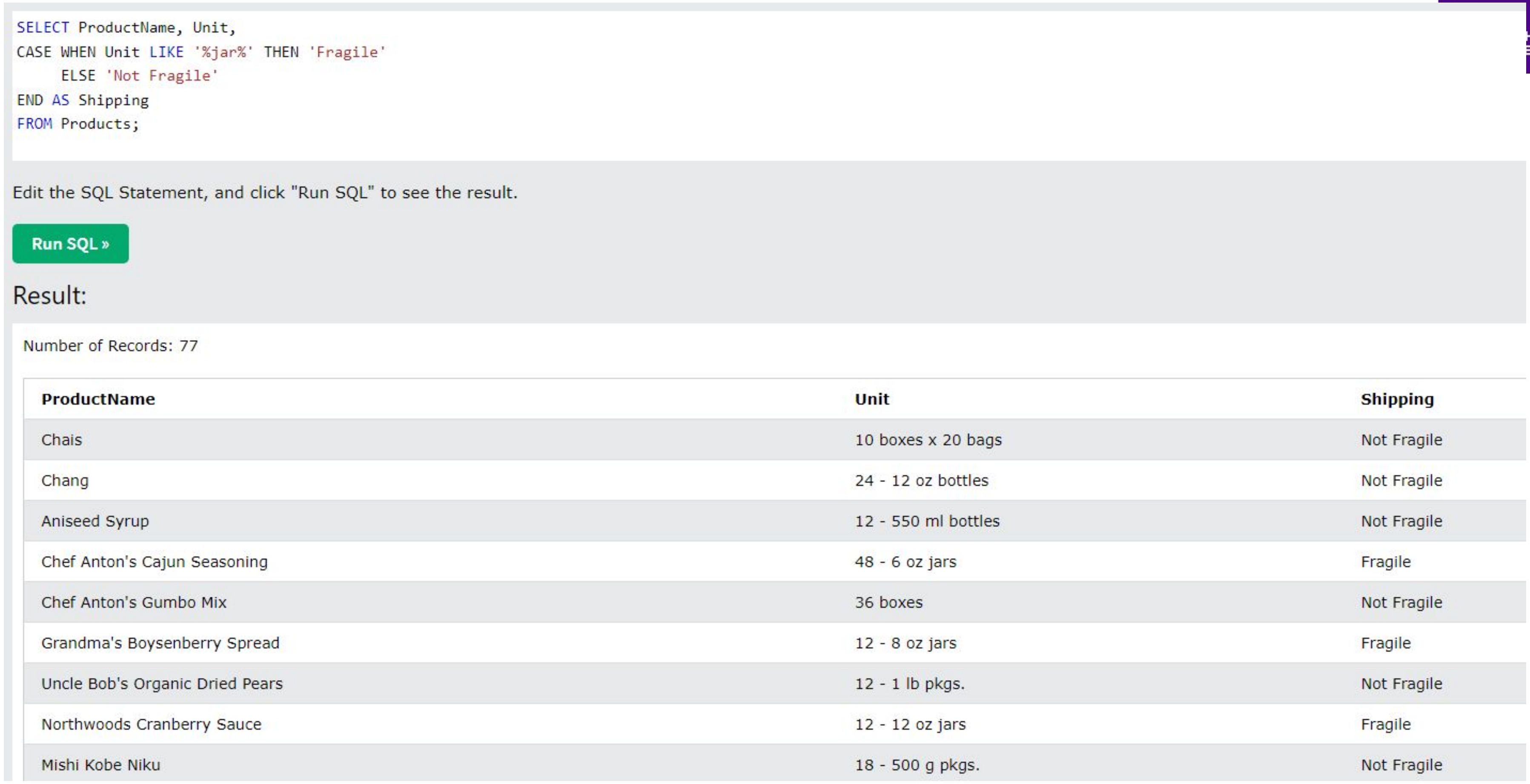
SQL Practice

- Let’s try **CASE - END**.
- List Products as ‘Fragile’ if Units are sold in jars, and ‘Not Fragile’ otherwise.

ProductID	ProductName	SupplierID	CategoryID	Unit	Price

```
SELECT ProductName, Unit,
 CASE WHEN Unit LIKE ‘%jar%’ THEN ’Fragile’
      ELSE ’Not Fragile’
 END
FROM Products;
```

Break till 10:35



```
SELECT ProductName, Unit,
CASE WHEN Unit LIKE '%jar%' THEN 'Fragile'
     ELSE 'Not Fragile'
END AS Shipping
FROM Products;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

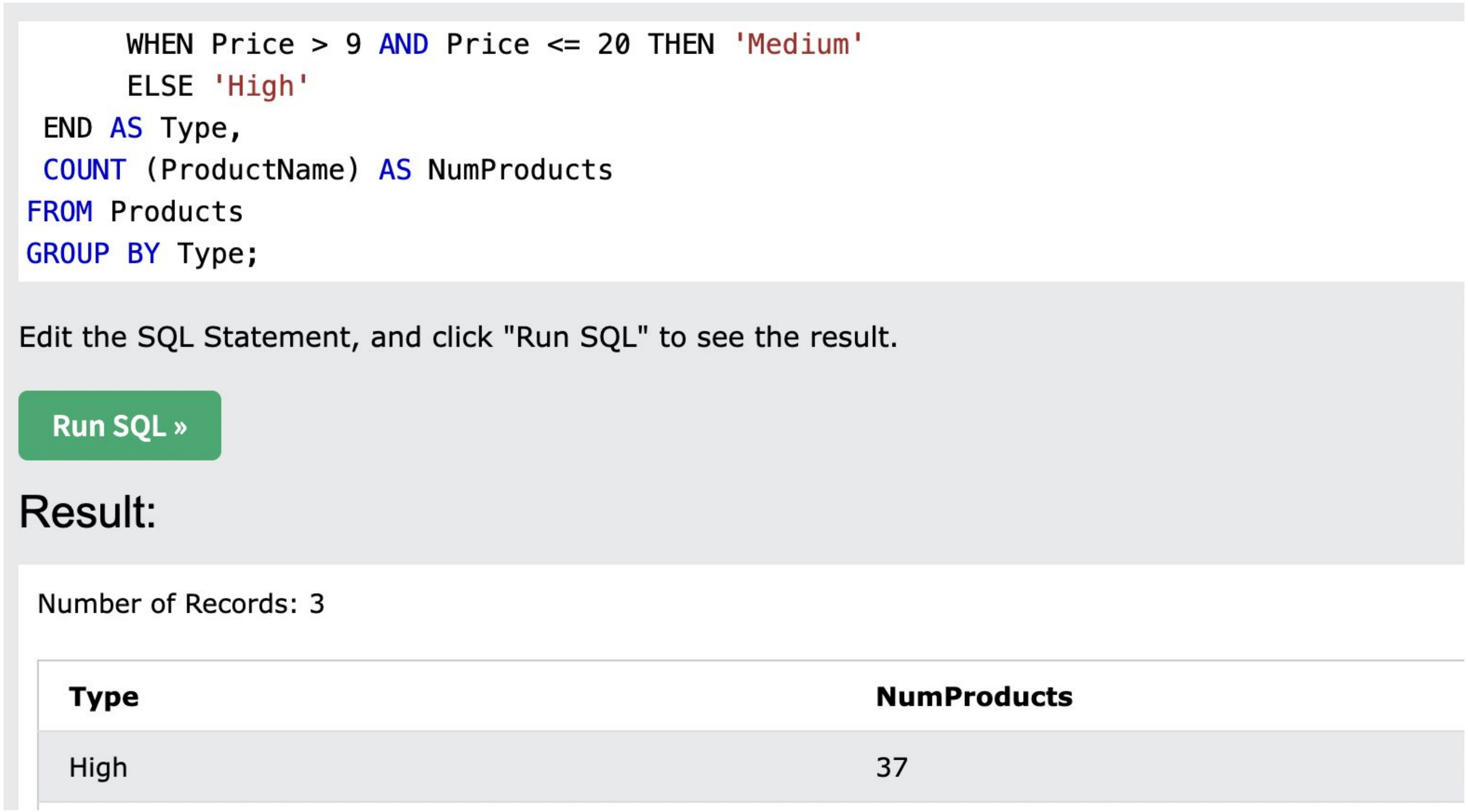
Number of Records: 77

ProductName	Unit	Shipping
Chais	10 boxes x 20 bags	Not Fragile
Chang	24 - 12 oz bottles	Not Fragile
Aniseed Syrup	12 - 550 ml bottles	Not Fragile
Chef Anton's Cajun Seasoning	48 - 6 oz jars	Fragile
Chef Anton's Gumbo Mix	36 boxes	Not Fragile
Grandma's Boysenberry Spread	12 - 8 oz jars	Fragile
Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	Not Fragile
Northwoods Cranberry Sauce	12 - 12 oz jars	Fragile
Mishi Kobe Niku	18 - 500 g pkgs.	Not Fragile

SQL Practice

- What is the function of the following query?

```
SELECT
 CASE WHEN Price <= 9 THEN 'Low'
      WHEN Price > 9 AND Price <= 20 THEN 'Medium'
      ELSE 'High'
 END AS Type,
 COUNT (ProductName) AS NumProducts
FROM Products
GROUP BY Type;
```



```
      WHEN Price > 9 AND Price <= 20 THEN 'Medium'
      ELSE 'High'
 END AS Type,
 COUNT (ProductName) AS NumProducts
FROM Products
GROUP BY Type;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

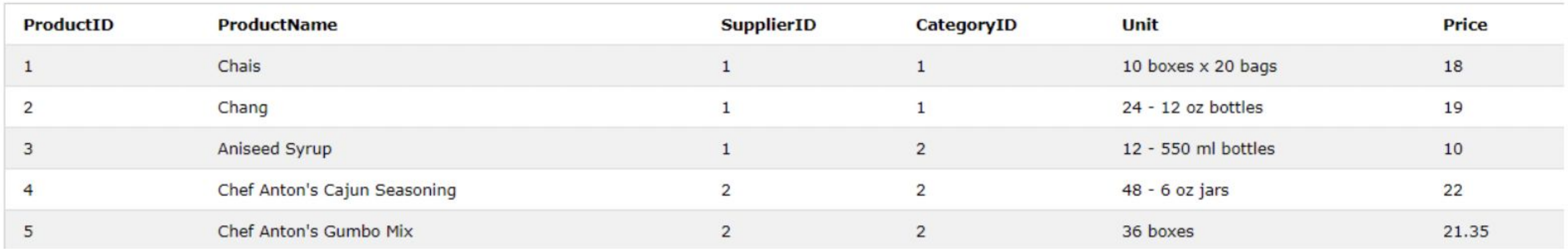
Result:

Number of Records: 3

Type	NumProducts
High	37

SQL Practice

- How to show the number of products based on the type of packaging?
- I.e., group products into ‘In Bags’, ‘In Bottles’, ‘In Jars’, and ‘Others’ as Packaging, and show the number of products within each type of packaging.

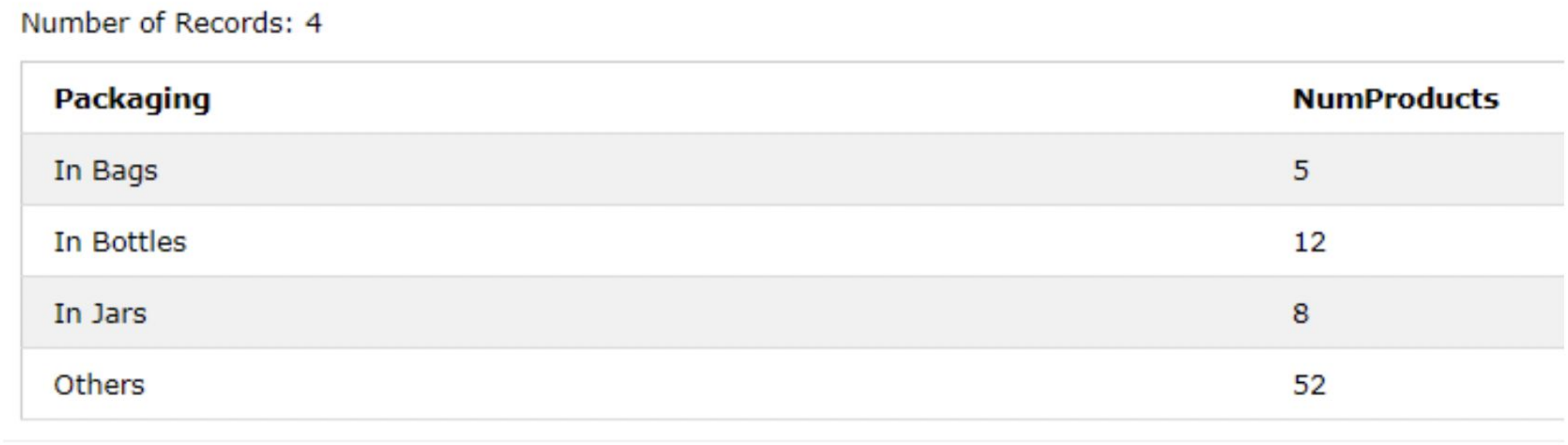


ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

SQL Practice

- How to show the number of products based on the type of packaging?
- I.e., group products into ‘In Bags’, ‘In Bottles’, ‘In Jars’, and ‘Others’ as Packaging, and show the number of products within each type of packaging.

```
SELECT
 CASE WHEN Unit LIKE ‘%bag%’ THEN 'In Bags'
      WHEN Unit LIKE ‘%bottle%’ THEN 'In Bottles'
      WHEN Unit LIKE ‘%jar%’ THEN 'In Jars'
 ELSE 'Others'
 END AS Packaging,
 COUNT (ProductName) AS NumProducts
FROM Products
GROUP BY Packaging;
```



Number of Records: 4

Packaging	NumProducts
In Bags	5
In Bottles	12
In Jars	8
Others	52

SQL JOIN

- How many products do we carry per category?

```
SELECT CategoryID,
   Count(ProductName)
FROM Products
GROUP BY CategoryID;
```

CategoryID	Count(Product Name)
1	12
2	12
3	13
4	10
5	7
6	6
7	5
8	12

Category ID “1” or “2” is not very informative. We would rather list the actual category, but it stored in a different table.

Querying relational databases

- Example: **Join (**Order and Book**) Math operation**

Order

Order#	Customer ID	ISBN	Payment
1	C1001	#0465039138	Credit
2	C1004	#1573928895	Credit
3	C1002	#0072952849	Cash
4	C1003	#0738206679	Cash
5	C1003	#0738206083	Cash
6	C1001	#0738206083	Credit
7	C1002	#1573928895	Credit
8	C1001	#0738206679	Credit

Book

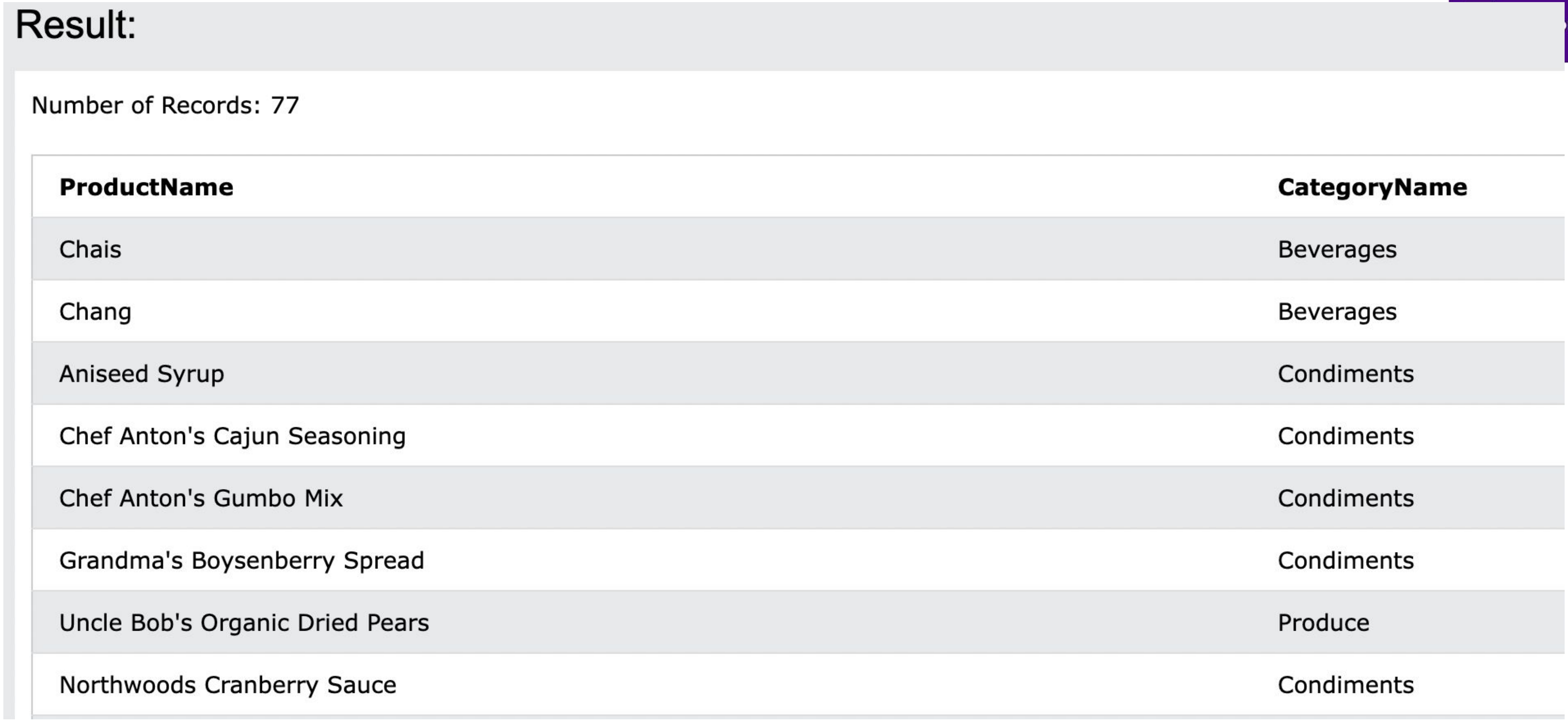
ISBN	Book Name	Author	Price
#0072952849	MIS in the Information Age	Haag, Stephen	$98.75
#0465039138	Inside Apple	Lessig, Lawrence	$25.00
#0738206083	Database Systems	Rheingold, How..	$29.95
#0738206679	Alibaba's World	Barabasi, Albert-	$34.95
#1234567890	Getting by at Stern	Author, Bookwri..	$25.00
#1573928895	Disruptive Innovation	Litman, Jessica	$55.00

SQL JOIN

- We can “link” tables by using a JOIN clause to display fields from multiple tables.
- Join Products and Categories tables using CategoryID

```
SELECT ProductName,CategoryName
FROM Products
JOIN Categories
ON Products.CategoryID = Categories.CategoryID;
```

Since “CategoryID” appears in both tables, you have to be more specific using this format: Table.Field = Table.Field



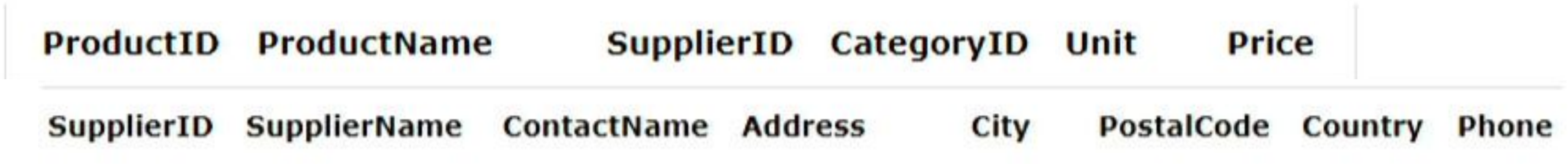
Result:

Number of Records: 77

ProductName	CategoryName
Chais	Beverages
Chang	Beverages
Aniseed Syrup	Condiments
Chef Anton's Cajun Seasoning	Condiments
Chef Anton's Gumbo Mix	Condiments
Grandma's Boysenberry Spread	Condiments
Uncle Bob's Organic Dried Pears	Produce
Northwoods Cranberry Sauce	Condiments

SQL Practice

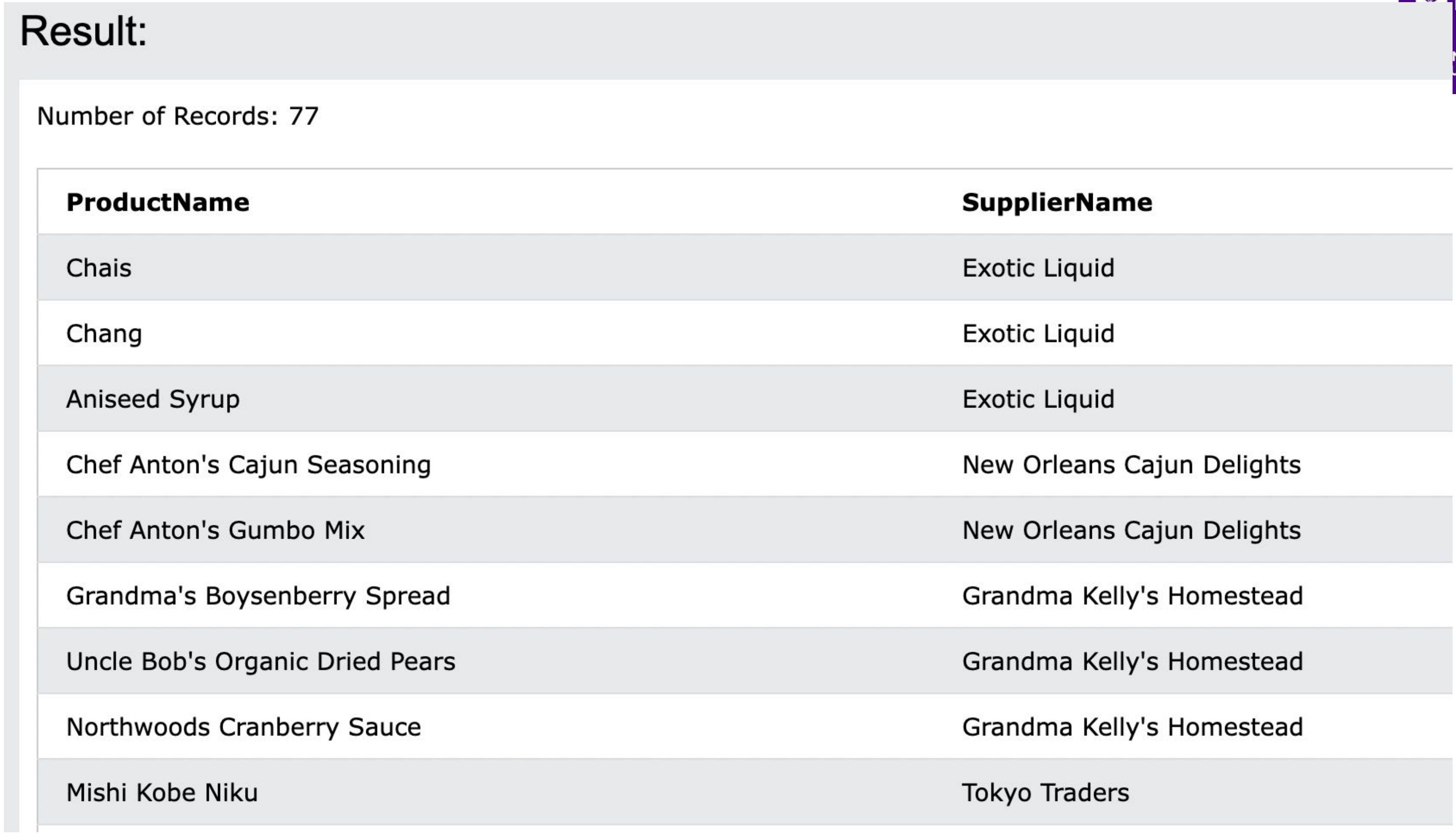
- Retrieve the name of each product and the name of the supplier.



ProductID	ProductName	SupplierID	CategoryID	Unit	Price

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country	Phone

```
SELECT ProductName,SupplierName
FROM Products
JOIN Suppliers
ON Products.SupplierID =
Suppliers.SupplierID;
```



Result:

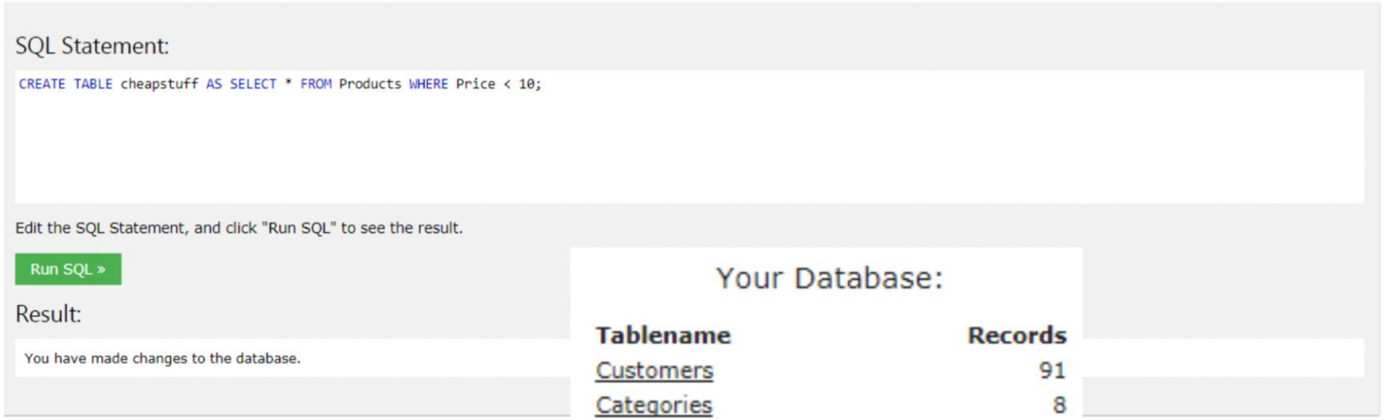
Number of Records: 77

ProductName	SupplierName
Chais	Exotic Liquid
Chang	Exotic Liquid
Aniseed Syrup	Exotic Liquid
Chef Anton's Cajun Seasoning	New Orleans Cajun Delights
Chef Anton's Gumbo Mix	New Orleans Cajun Delights
Grandma's Boysenberry Spread	Grandma Kelly's Homestead
Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead
Northwoods Cranberry Sauce	Grandma Kelly's Homestead
Mishi Kobe Niku	Tokyo Traders

SQL CREATE TABLE

- **CREATE TABLE** cheapstuff **AS**

  SELECT * FROM Products WHERE Price < 10



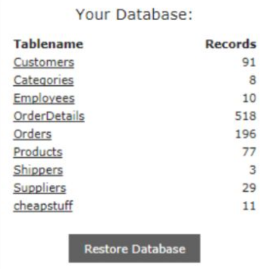
SQL Statement:

```
CREATE TABLE cheapstuff AS SELECT * FROM Products WHERE Price < 10;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

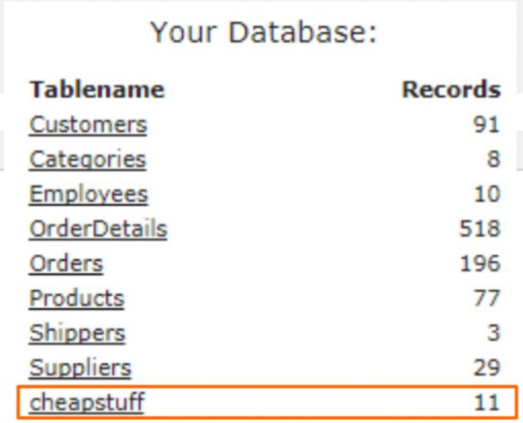
Result:

You have made changes to the database.



Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29
cheapstuff	11



Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29
cheapstuff	11

SQL CREATE TABLE

- Make a table of all expensive products (say Price > $50)
- The table consists of two columns: ProductName and Price

ProductID	ProductName	SupplierID	CategoryID	Unit	Price

```
CREATE TABLE ExpensiveProducts AS
    SELECT ProductName, Price
    FROM Products
    WHERE Price > 100;
```