



JENSON USA





# ABOUT COMPANY

Jenson USA is a trusted American bicycle retailer founded in 1994, offering a wide range of bikes, parts, accessories, and gear both online and through physical stores. Known for its competitive pricing, fast shipping, and strong community involvement, Jenson USA has earned a solid reputation among cycling enthusiasts across the U.S.

CONTINUE





# PROJECT OVERVIEW

This project involves analyzing sales and customer data from Jenson USA using SQL to generate key business insights. By applying advanced SQL techniques such as JOINS to combine data from multiple tables, Common Table Expressions (CTEs) for better query organization, and RANK functions for identifying top-performing products and customers, the project aims to uncover patterns in purchasing behavior, product popularity, and regional sales performance. The goal is to support data-driven decision-making and improve customer engagement strategies.

Find the total number of products sold by each store along with the store name.

The screenshot shows a SQL query editor interface. At the top, there's a toolbar with various icons and a dropdown menu set to "Limit to 5000 rows". Below the toolbar, the SQL query is written in a monospaced font, numbered 1 through 9. The query is as follows:

```
1 • SELECT
2     stores.store_name, SUM(order_items.quantity)
3 FROM
4     order_items
5     LEFT JOIN
6     orders ON order_items.order_id = orders.order_id
7     LEFT JOIN
8     stores ON stores.store_id = orders.store_id
9 GROUP BY stores.store_name;
```

Below the query editor, there's a "Result Grid" section. It has a toolbar with icons for "Result Grid", "Filter Rows", "Export", and "Wrap Cell Content". The results are displayed in a table with two columns: "store\_name" and "sum(order\_items.quantity)".

store_name	sum(order_items.quantity)
Santa Cruz Bikes	1516
Baldwin Bikes	4779
Rowlett Bikes	783

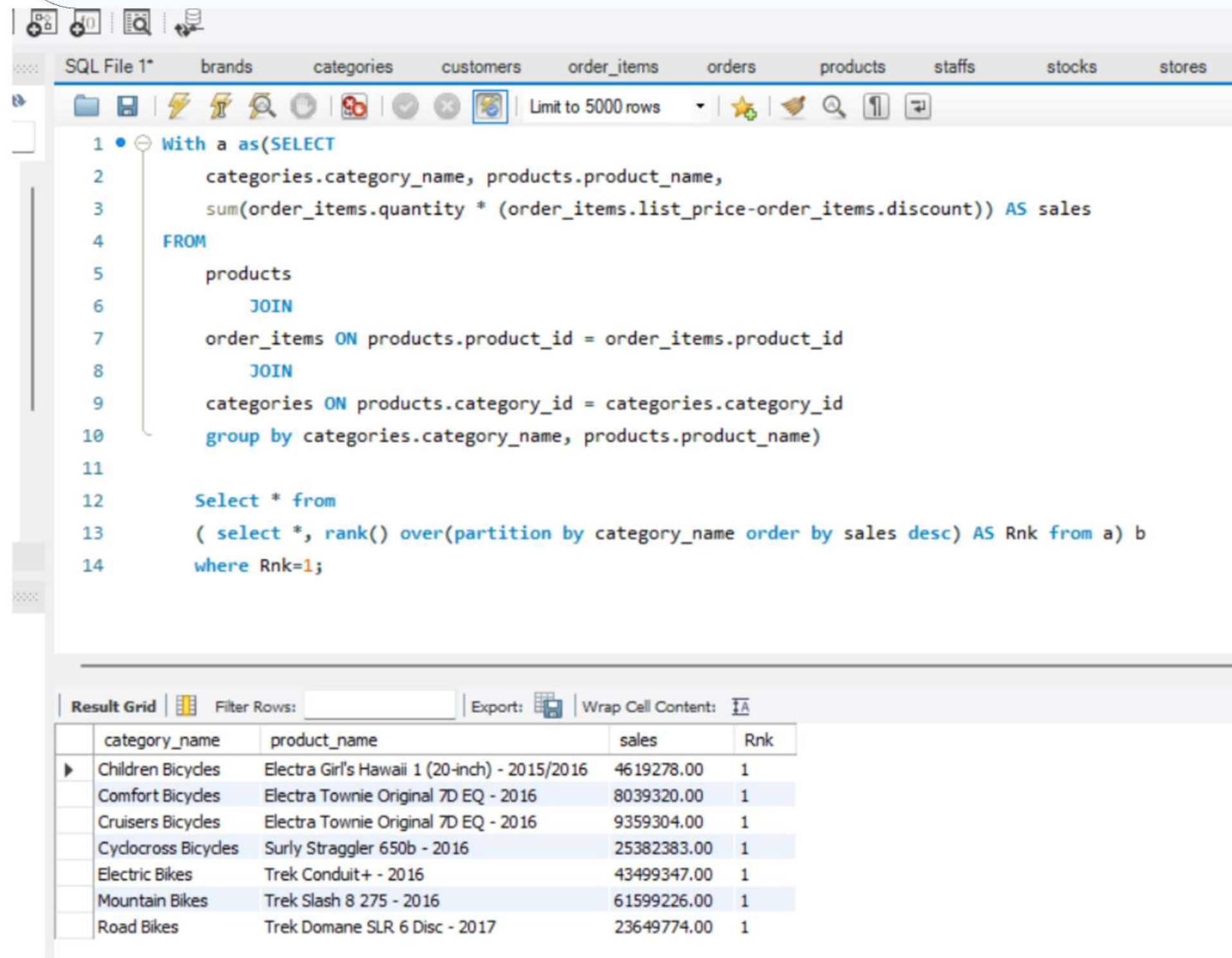


# Calculate the cumulative sum of quantities sold for each product over time.

```
1 • select orders.order_date, products.product_name, order_items.quantity,  
2    sum(order_items.quantity) over(partition by products.product_name order by orders.order_date)as Cumulative_sum  
3 from order_items join products on  
4    products.product_id=order_items.product_id  
5 join  
6    orders on  
7    orders.order_id=order_items.order_id;
```

Result Grid   Filter Rows:   Export:   Wrap Cell Content:				
	order_date	product_name	quantity	Cumulative_sum
▶	2018-01-01	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	1	1
	2018-01-21	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	2	3
	2018-04-30	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	2	5
	2017-01-29	Electra Amsterdam Fashion 7i Ladies' - 2017	2	2
	2017-02-28	Electra Amsterdam Fashion 7i Ladies' - 2017	1	3
	2017-03-03	Electra Amsterdam Fashion 7i Ladies' - 2017	1	4
	2017-03-09	Electra Amsterdam Fashion 7i Ladies' - 2017	2	6
	2017-04-06	Electra Amsterdam Fashion 7i Ladies' - 2017	1	7
	2017-04-15	Electra Amsterdam Fashion 7i Ladies' - 2017	2	9
	2017-04-16	Electra Amsterdam Fashion 7i Ladies' - 2017	1	10
	2017-06-27	Electra Amsterdam Fashion 7i Ladies' - 2017	2	14
	2017-06-27	Electra Amsterdam Fashion 7i Ladies' - 2017	2	14
	2017-07-15	Electra Amsterdam Fashion 7i Ladies' - 2017	2	16
	2017-07-19	Electra Amsterdam Fashion 7i Ladies' - 2017	2	18
	2017-08-18	Electra Amsterdam Fashion 7i Ladies' - 2017	1	19
	2017-08-21	Electra Amsterdam Fashion 7i Ladies' - 2017	2	21
	2017-09-14	Electra Amsterdam Fashion 7i Ladies' - 2017	2	23
	2017-10-04	Electra Amsterdam Fashion 7i Ladies' - 2017	2	27
	2017-10-04	Electra Amsterdam Fashion 7i Ladies' - 2017	2	27
	2017-10-31	Electra Amsterdam Fashion 7i Ladies' - 2017	2	29
	2017-11-04	Electra Amsterdam Fashion 7i Ladies' - 2017	1	30
	2017-11-28	Electra Amsterdam Fashion 7i Ladies' - 2017	1	31
	2017-12-04	Electra Amsterdam Fashion 7i Ladies' - 2017	2	34

Find the product with the highest total sales (quantity \* price) for each category.



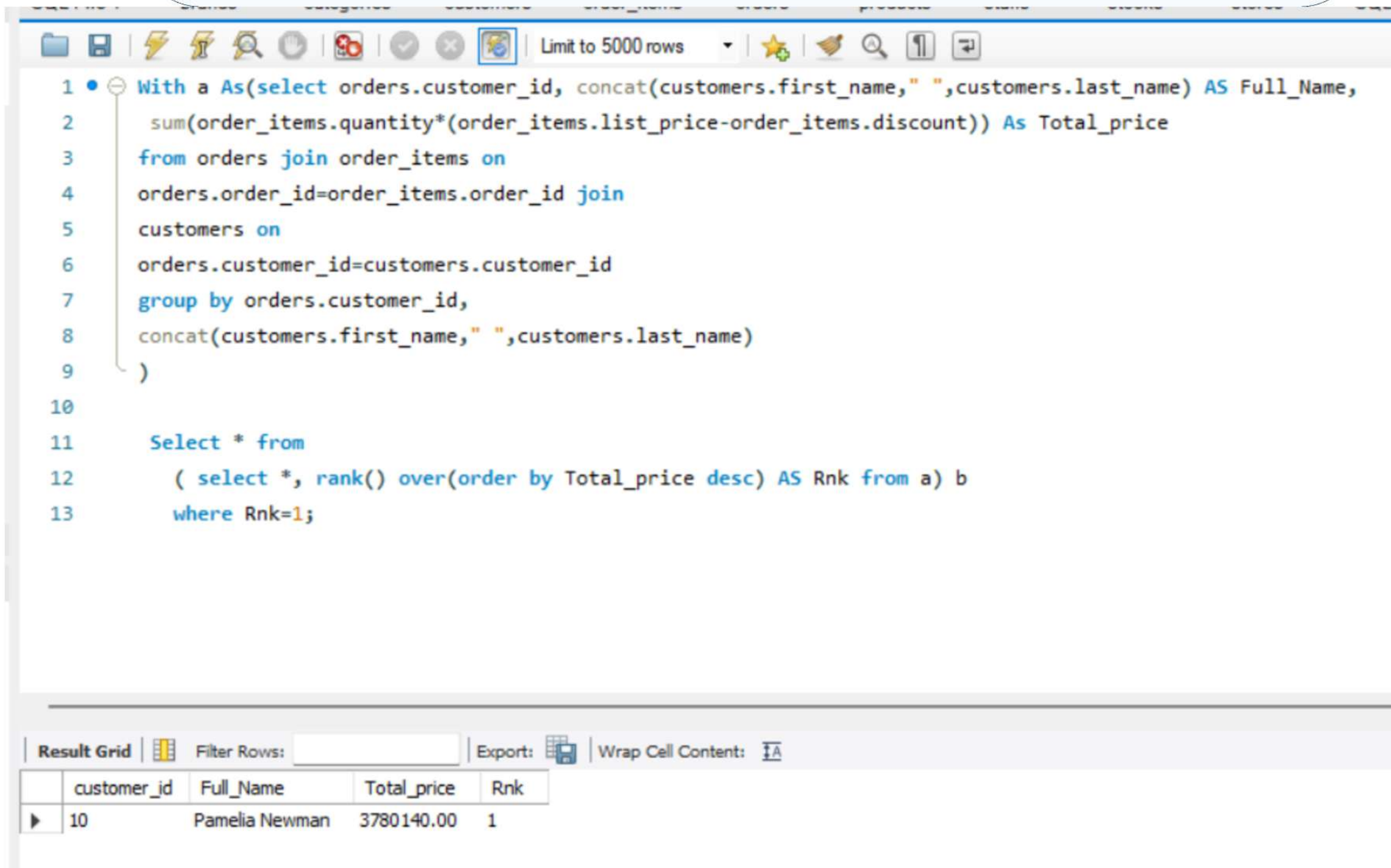
```
1 With a as(SELECT
2     categories.category_name, products.product_name,
3     sum(order_items.quantity * (order_items.list_price-order_items.discount)) AS sales
4 FROM
5     products
6     JOIN
7     order_items ON products.product_id = order_items.product_id
8     JOIN
9     categories ON products.category_id = categories.category_id
10    group by categories.category_name, products.product_name)
11
12    Select * from
13    ( select *, rank() over(partition by category_name order by sales desc) AS Rnk from a) b
14    where Rnk=1;
```

Result Grid

	category_name	product_name	sales	Rnk
▶	Children Bicycles	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	4619278.00	1
	Comfort Bicycles	Electra Townie Original 7D EQ - 2016	8039320.00	1
	Cruisers Bicycles	Electra Townie Original 7D EQ - 2016	9359304.00	1
	Cyclocross Bicycles	Surly Straggler 650b - 2016	25382383.00	1
	Electric Bikes	Trek Conduit+ - 2016	43499347.00	1
	Mountain Bikes	Trek Slash 8 275 - 2016	61599226.00	1
	Road Bikes	Trek Domane SLR 6 Disc - 2017	23649774.00	1



Find the customer who spent the most money on orders.

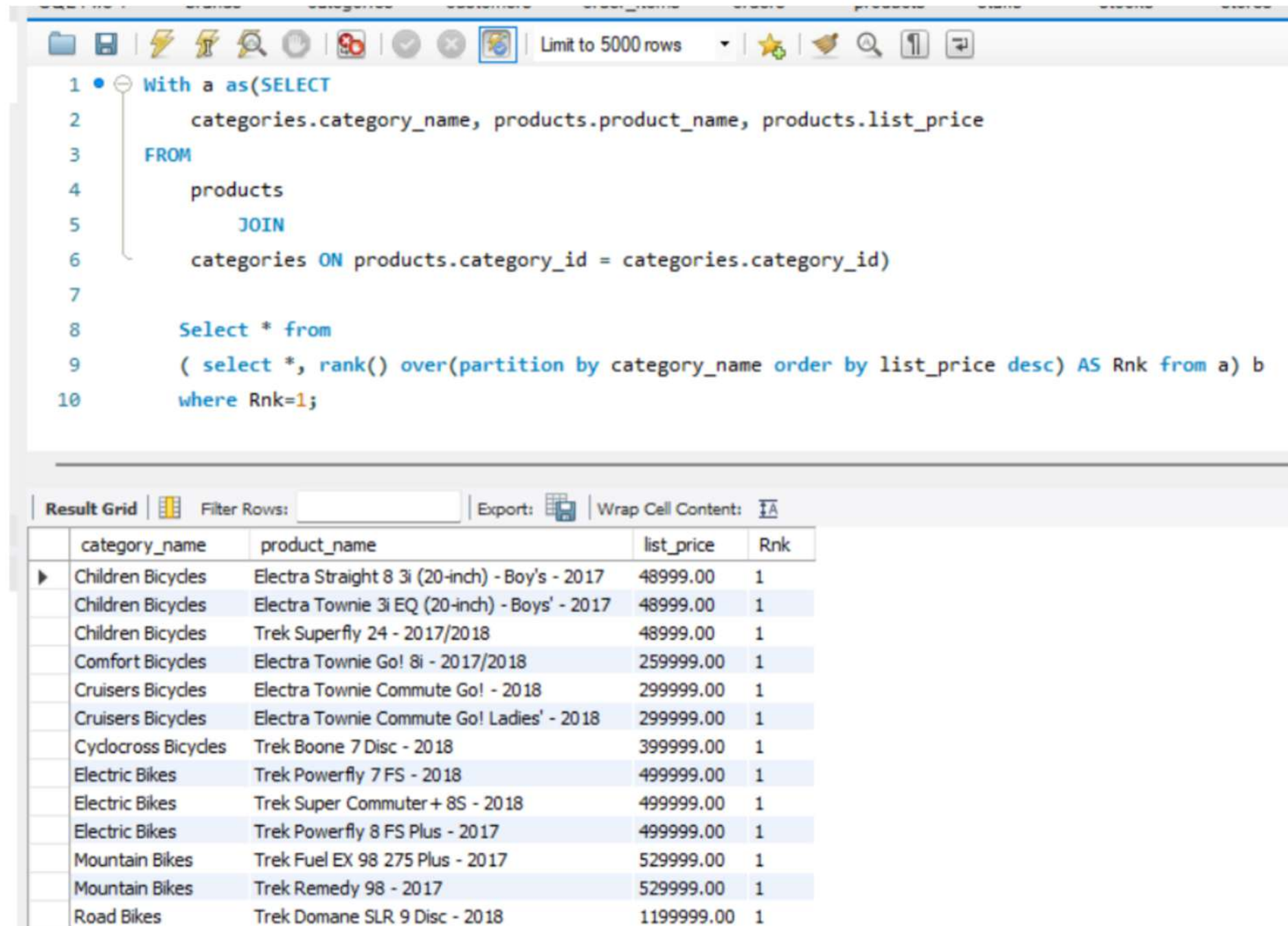


```
1 With a As(select orders.customer_id, concat(customers.first_name," ",customers.last_name) AS Full_Name,
2      sum(order_items.quantity*(order_items.list_price-order_items.discount)) As Total_price
3      from orders join order_items on
4      orders.order_id=order_items.order_id join
5      customers on
6      orders.customer_id=customers.customer_id
7      group by orders.customer_id,
8      concat(customers.first_name," ",customers.last_name)
9  )
10
11  Select * from
12      ( select *, rank() over(order by Total_price desc) AS Rnk from a) b
13  where Rnk=1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	customer_id	Full_Name	Total_price	Rnk
▶	10	Pamelia Newman	3780140.00	1

Find the highest-priced product for each category name.



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, a search icon, and a 'Limit to 5000 rows' dropdown. The SQL editor contains the following query:

```
1 With a as(SELECT
2     categories.category_name, products.product_name, products.list_price
3 FROM
4     products
5 JOIN
6     categories ON products.category_id = categories.category_id)
7
8 Select * from
9     ( select *, rank() over(partition by category_name order by list_price desc) AS Rnk from a) b
10 where Rnk=1;
```

Below the editor is the 'Result Grid' section, which includes a 'Filter Rows' input, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with the following data:

	category_name	product_name	list_price	Rnk
▶	Children Bicydes	Electra Straight 8 3i (20-inch) - Boy's - 2017	48999.00	1
	Children Bicydes	Electra Townie 3i EQ (20-inch) - Boys' - 2017	48999.00	1
	Children Bicydes	Trek Superfly 24 - 2017/2018	48999.00	1
	Comfort Bicydes	Electra Townie Go! 8i - 2017/2018	259999.00	1
	Cruisers Bicydes	Electra Townie Commute Go! - 2018	299999.00	1
	Cruisers Bicydes	Electra Townie Commute Go! Ladies' - 2018	299999.00	1
	Cyclocross Bicydes	Trek Boone 7 Disc - 2018	399999.00	1
	Electric Bikes	Trek Powerfly 7 FS - 2018	499999.00	1
	Electric Bikes	Trek Super Commuter + 8S - 2018	499999.00	1
	Electric Bikes	Trek Powerfly 8 FS Plus - 2017	499999.00	1
	Mountain Bikes	Trek Fuel EX 98 275 Plus - 2017	529999.00	1
	Mountain Bikes	Trek Remedy 98 - 2017	529999.00	1
	Road Bikes	Trek Domane SLR 9 Disc - 2018	1199999.00	1



Find the total number of orders placed by each customer per store.

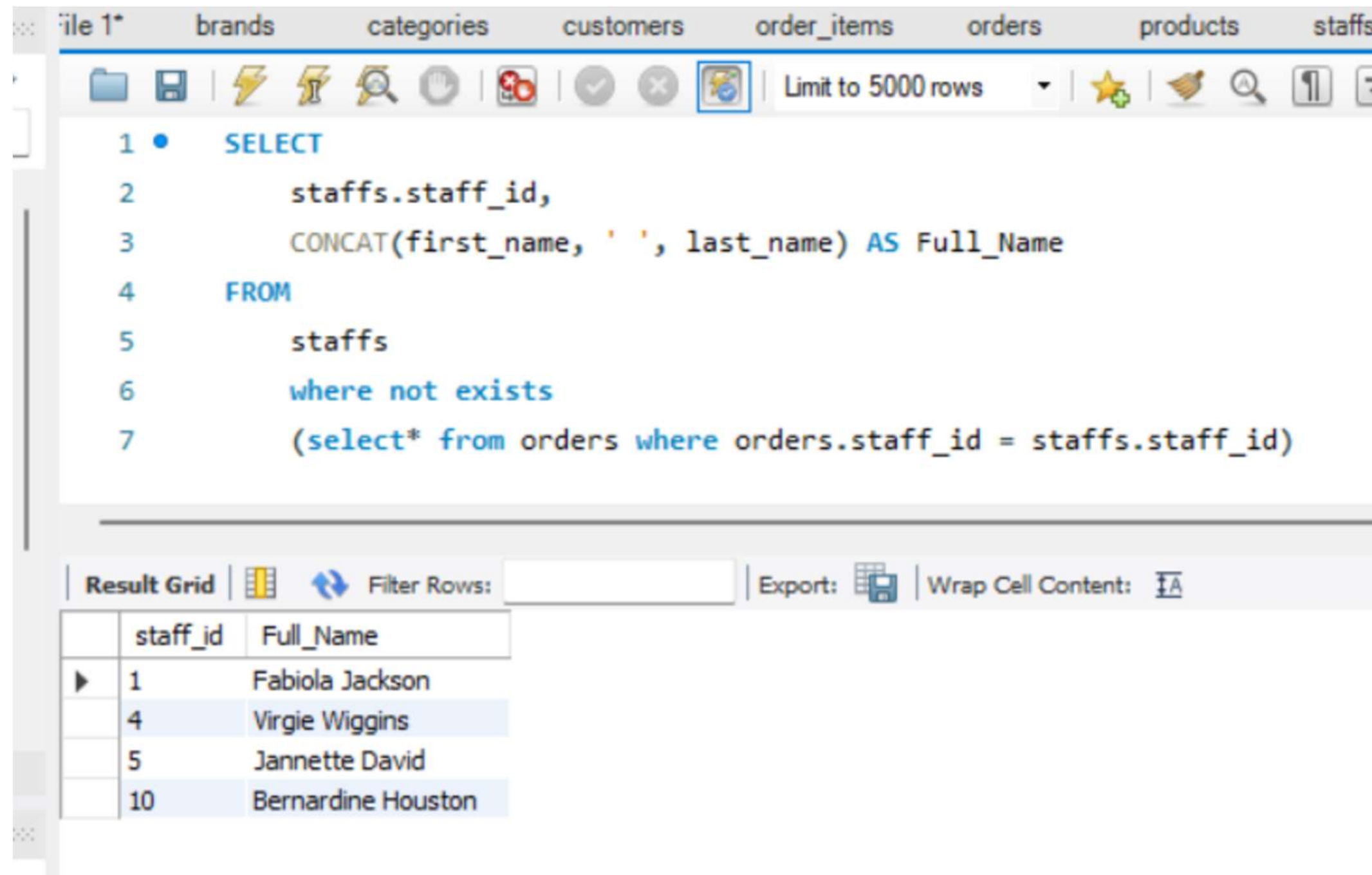
```
1 • SELECT
2     stores.store_name,
3     CONCAT(customers.first_name,
4             ',
5             customers.last_name) AS Customer_name,
6     COUNT(orders.order_id) Total_Orders
7 FROM
8     orders
9     JOIN
10    stores ON orders.store_id = stores.store_id
11    JOIN
12    customers ON orders.customer_id = customers.customer_id
13 GROUP BY stores.store_name , CONCAT(customers.first_name,
14         ',
15         customers.last_name);
```

Result Grid | Filter Rows: | Export:

	store_name	Customer_name	Total_Orders
	Baldwin Bikes	Trinidad Chapman	1
	Baldwin Bikes	Jeannie Wilcox	1
	Baldwin Bikes	Max Charles	1
	Baldwin Bikes	Bronwyn Vargas	1
	Baldwin Bikes	Christia Wilkins	1
	Baldwin Bikes	Aaron Knapp	1
	Baldwin Bikes	Lavette Wright	1
	Baldwin Bikes	Rosa Kinney	1
	Baldwin Bikes	Rodolfo Buck	1
	Baldwin Bikes	Romaine Haley	1
	Baldwin Bikes	Kimberli Cline	1
	Baldwin Bikes	Casey Gill	1
	Baldwin Bikes	Keitha Black	1

Result 12 x

Find the names of staff members who have not made any sales.



The screenshot shows a database management interface with a tabbed menu at the top containing 'file 1\*', 'brands', 'categories', 'customers', 'order\_items', 'orders', 'products', and 'staffs'. Below the menu is a toolbar with various icons, including a 'Limit to 5000 rows' dropdown. The main area displays a SQL query:

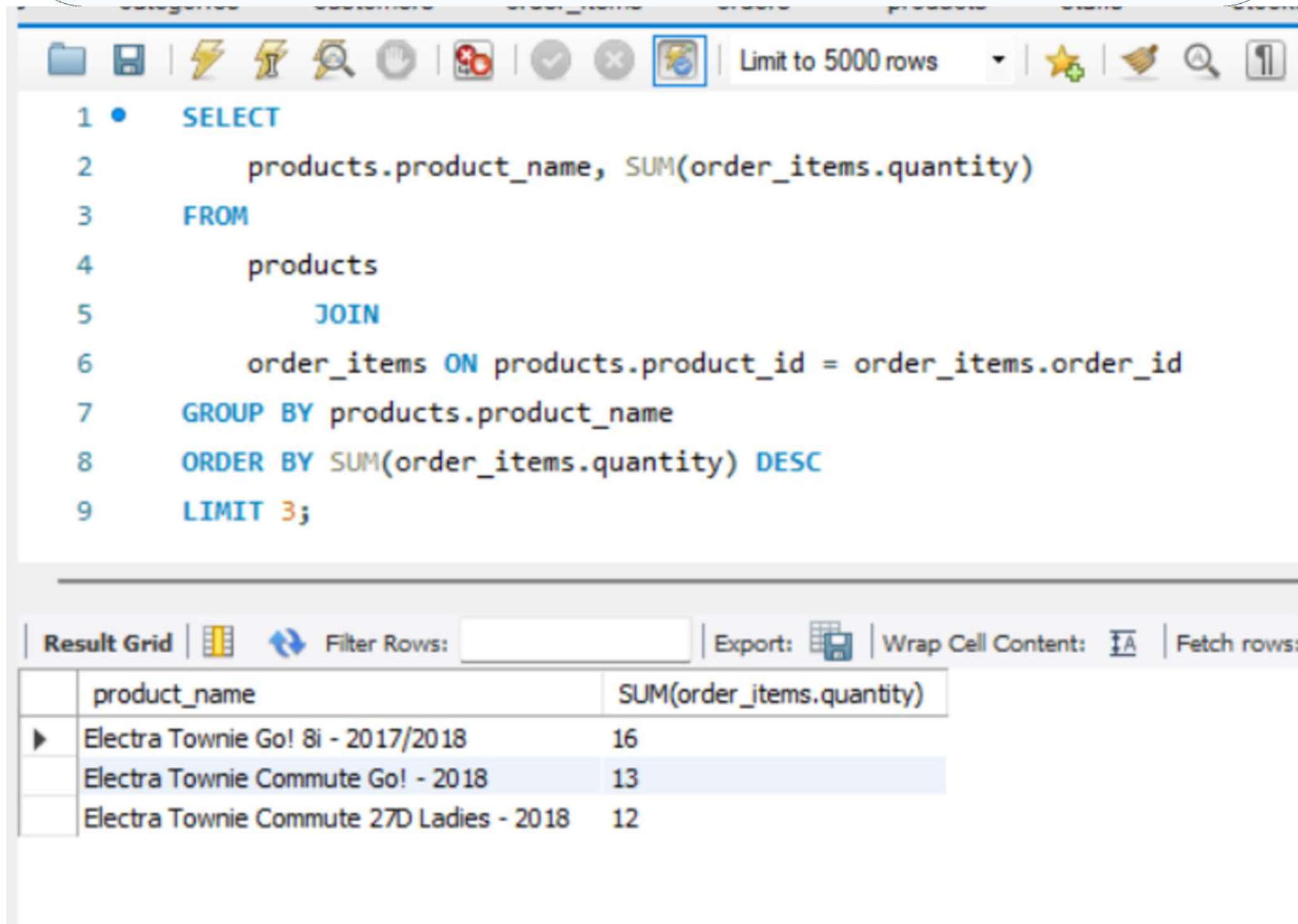
```
1 • SELECT
2     staffs.staff_id,
3     CONCAT(first_name, ' ', last_name) AS Full_Name
4 FROM
5     staffs
6 where not exists
7     (select* from orders where orders.staff_id = staffs.staff_id)
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows:' input field, an 'Export:' button, and a 'Wrap Cell Content:' checkbox. The grid displays the following data:

	staff_id	Full_Name
▶	1	Fabiola Jackson
	4	Virgie Wiggins
	5	Jannette David
	10	Bernardine Houston



Find the top 3 most sold products in terms of quantity.



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and settings. The query is as follows:

```
1 • SELECT
2     products.product_name, SUM(order_items.quantity)
3 FROM
4     products
5     JOIN
6     order_items ON products.product_id = order_items.order_id
7 GROUP BY products.product_name
8 ORDER BY SUM(order_items.quantity) DESC
9 LIMIT 3;
```

Below the query editor is a 'Result Grid' section with a toolbar for filtering, exporting, and formatting. The results are displayed in a table with two columns: 'product\_name' and 'SUM(order\_items.quantity)'.

	product_name	SUM(order_items.quantity)
▶	Electra Townie Go! 8i - 2017/2018	16
	Electra Townie Commute Go! - 2018	13
	Electra Townie Commute 27D Ladies - 2018	12

Find the median value of the price list.

The screenshot shows a SQL IDE interface with a query editor and a result grid. The query is as follows:

```
1 with a as (Select list_price, row_number() over(order by list_price) pos,  
2 count(*) over() n from order_items)  
3  
4 select case  
5 when n % 2=0 then (select avg(list_price) from a where pos in((n/2),(n/2)+1))  
6 else (select list_price from a where pos in ((n+1)/2))  
7 end as median from a limit 1;
```

The result grid shows the following data:

median
59999.000000

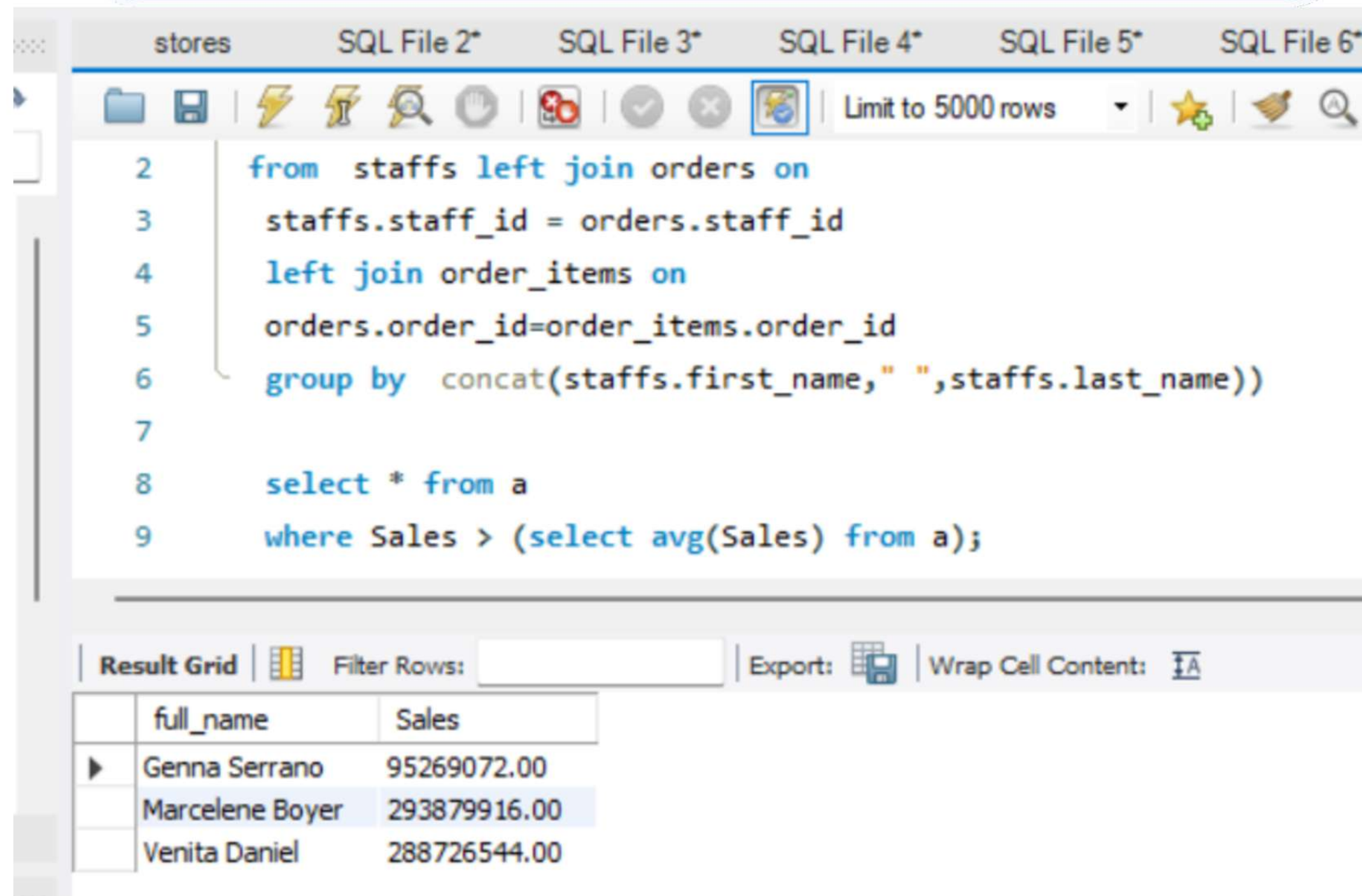


List all products that have never been ordered.(use Exists)

```
1 • SELECT
2     products.product_name
3 FROM
4     products
5 WHERE
6     NOT EXISTS( SELECT
7                 *
8                 FROM
9                     order_items
10                WHERE
11                    order_items.product_id = products.product_id)
```

Result Grid		Filter Rows:	Export
	product_name		
▶	Trek 820 - 2016		
	Surly Krampus Frameset - 2018		
	Trek Kids' Dual Sport - 2018		
	Trek Domane SLR 6 Disc Women's - 2018		
	Electra Townie Go! 8i Ladies' - 2018		
	Trek Precaliber 12 Girl's - 2018		
	Electra Savannah 1 (20-inch) - Girl's - 2018		
	Electra Sweet Ride 1 (20-inch) - Girl's - 2018		
	Trek Checkpoint ALR 4 Women's - 2019		
	Trek Checkpoint ALR 5 - 2019		
	Trek Checkpoint ALR 5 Women's - 2019		
	Trek Checkpoint SL 5 Women's - 2019		
	Trek Checkpoint SL 6 - 2019		
	Trek Checkpoint ALR Frameset - 2019		

List the names of staff members who have made more sales than the average number of sales by all staff members.



The screenshot shows a database query editor with a toolbar at the top containing icons for file operations, execution, and search. The toolbar also includes a dropdown menu set to "Limit to 5000 rows". The SQL query is as follows:

```
2 from staffs left join orders on
3     staffs.staff_id = orders.staff_id
4 left join order_items on
5     orders.order_id=order_items.order_id
6 group by concat(staffs.first_name," ",staffs.last_name))
7
8 select * from a
9 where Sales > (select avg(Sales) from a);
```

Below the query editor, the "Result Grid" tab is active, displaying a table with the following data:

	full_name	Sales
▶	Genna Serrano	95269072.00
	Marcelene Boyer	293879916.00
	Venita Daniel	288726544.00



Identify the customers who have ordered all types of products (i.e., from every category).

```
1 • SELECT
2     customers.customer_id, COUNT(DISTINCT products.category_id)
3 FROM
4     customers
5     JOIN
6     orders ON customers.customer_id = orders.customer_id
7     JOIN
8     order_items ON order_items.order_id = orders.order_id
9     JOIN
10    products ON products.product_id = order_items.product_id
11 GROUP BY customers.customer_id
12 HAVING COUNT(DISTINCT products.category_id) = (SELECT
13     COUNT(category_id)
14 FROM
15     categories);
16
```

Result Grid			Filter Rows:	Exp
	customer_id	count(distinct products.category_id)		
▶	9	7		



**Thank You**

Presented By-  
*Anshika Tejwani*

