

**Q1) List all products along with their categories.**

```
SELECT Products.Name AS ProductName, Categories.CategoryName
```

```
FROM Products
```

```
JOIN Categories ON Products.CategoryID = Categories.CategoryID;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	ProductName	CategoryName
▶	ADHD Medication	Psychiatry
	Adrenaline Injectio...	Emergency_Medi...
	Amoxicillin	Internal_Medicine
	Anti-Psychotic Med...	Psychiatry
	Antibiotic Suspensi...	Pediatrics
	Antidepressant Ta...	Psychiatry
	Antihistamine Coug...	Pediatrics
	Antiseptic Wipes	Emergency_Medi...
	Anxiolytic Medication	Psychiatry
	Aspirin	Internal_Medicine

Result 63

×

Output

**Q2) List the names of the products for which orders have been received.**

```
SELECT DISTINCT Products.Name AS ProductName
```

```
FROM Products
```

```
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID;
```

Result Grid	
	Filter Rows
ProductName	
▶ Aspirin	
Metformin	
Lisinopril	
Simvastatin	
Amoxicillin	
Ibuprofen	
Omeprazole	
Hydrochlorothiazide	
Gabapentin	
Cetirizine	

Result 8 ×

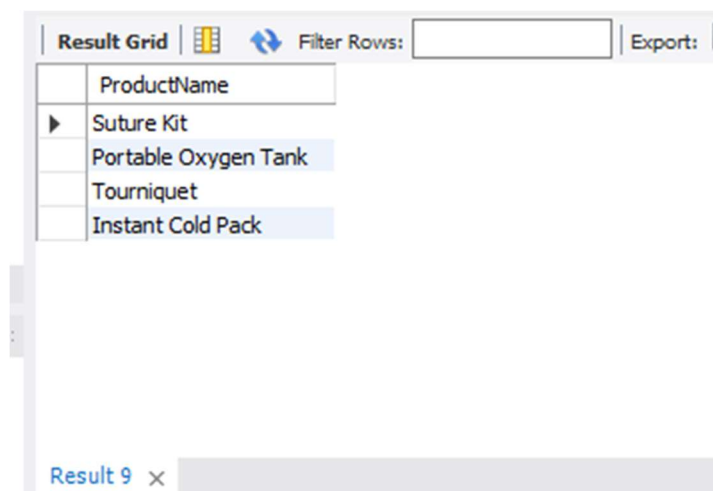
**Q3) List the names of the products for which orders have not been received.**

```
SELECT Products.Name AS ProductName
```

```
FROM Products
```

```
LEFT JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
```

```
WHERE OrderDetails.ProductID IS NULL;
```

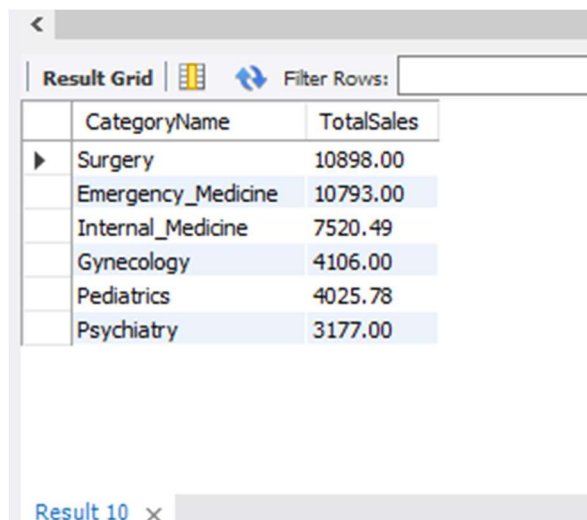


The screenshot shows a SQL query result grid. At the top, there is a toolbar with a 'Result Grid' button, a 'Filter Rows' button, a text input field for filtering, and an 'Export' button. Below the toolbar is a table with one column, 'ProductName'. The table contains four rows of data: 'Suture Kit', 'Portable Oxygen Tank', 'Tourniquet', and 'Instant Cold Pack'. The 'Portable Oxygen Tank' and 'Instant Cold Pack' rows are highlighted in blue. At the bottom of the window, there is a tab labeled 'Result 9' with a close button (X).

ProductName
Suture Kit
Portable Oxygen Tank
Tourniquet
Instant Cold Pack

**Q4) From the categories table, which categories have witnessed the maximum sales? Please show them in descending order.**

```
SELECT Categories.CategoryName, SUM(OrderDetails.Quantity*Products.Price) AS TotalSales  
  
FROM Categories  
  
JOIN Products ON Products.CategoryID = Categories.CategoryID  
  
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID  
  
GROUP BY Categories.CategoryName  
  
ORDER BY TotalSales DESC;
```



The screenshot shows a database query result grid. At the top, there is a toolbar with a back arrow, a 'Result Grid' button, a grid icon, a refresh icon, and a 'Filter Rows:' input field. Below the toolbar is a table with two columns: 'CategoryName' and 'TotalSales'. The table contains six rows of data, sorted in descending order of total sales. The first row is 'Surgery' with a total sales of 10898.00. The second row is 'Emergency\_Medicine' with a total sales of 10793.00. The third row is 'Internal\_Medicine' with a total sales of 7520.49. The fourth row is 'Gynecology' with a total sales of 4106.00. The fifth row is 'Pediatrics' with a total sales of 4025.78. The sixth row is 'Psychiatry' with a total sales of 3177.00. At the bottom of the grid, there is a tab labeled 'Result 10' with a close button (X).

CategoryName	TotalSales
Surgery	10898.00
Emergency_Medicine	10793.00
Internal_Medicine	7520.49
Gynecology	4106.00
Pediatrics	4025.78
Psychiatry	3177.00

**Q5) In terms of quantity, how many orders has each category received?**

```
SELECT Categories.CategoryName, SUM(OrderDetails.Quantity) AS QtySold
```

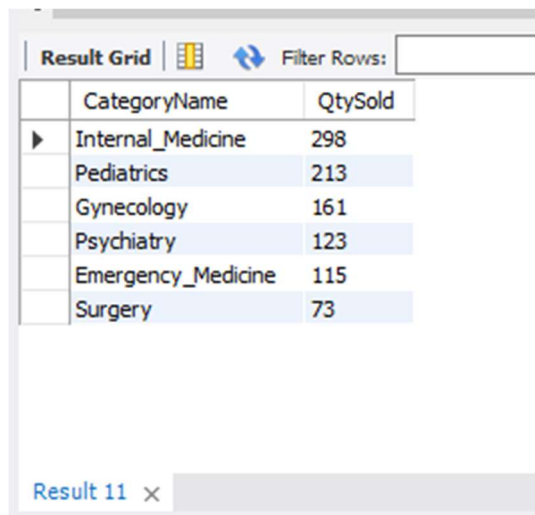
```
FROM Categories
```

```
JOIN Products ON Products.CategoryID = Categories.CategoryID
```

```
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
```

```
GROUP BY Categories.CategoryName
```

```
ORDER BY QtySold DESC;
```



The screenshot shows a SQL query result grid with the following data:

	CategoryName	QtySold
▶	Internal_Medicine	298
	Pediatrics	213
	Gynecology	161
	Psychiatry	123
	Emergency_Medicine	115
	Surgery	73

Result 11 x

**Q6) Now that the orders have been sold, find out the remaining stock of each product and its value.**

```
SELECT p.Name AS ProductName,  
       p.StockQuantity - COALESCE(SUM(od.Quantity),0) AS RemainingStock,  
       (p.StockQuantity - COALESCE(SUM(od.Quantity),0))*p.Price AS RemainingStockValue  
FROM Products p  
  
LEFT JOIN OrderDetails od ON od.ProductID = p.ProductID  
  
GROUP BY p.Name, p.StockQuantity, od.Quantity, p.price  
  
ORDER BY RemainingStockValue DESC;
```

Result Grid			
		Filter Rows:	Export:
	ProductName	RemainingStock	RemainingStockValue
►	Ibuprofen	441	28665.00
	Defibrillator Unit	15	22500.00
	Simvastatin	290	14790.00
	Kids' Cough Suppressant	372	11904.00
	Hydrochlorothiazide	180	10980.00
	Tramadol	290	9538.10
	LED Surgical Light	7	8750.00
	Pediatric Probiotic Powder	127	8382.00
	Digestive Aid for Kids	135	7425.00
	Cetirizine	463	5093.00
Result 12 x			

**Q7) Find the details of orders placed by 'Priya Singh' including product names and quantities.**

```
SELECT Orders.OrderID, Products.Name AS ProductName, OrderDetails.Quantity
```

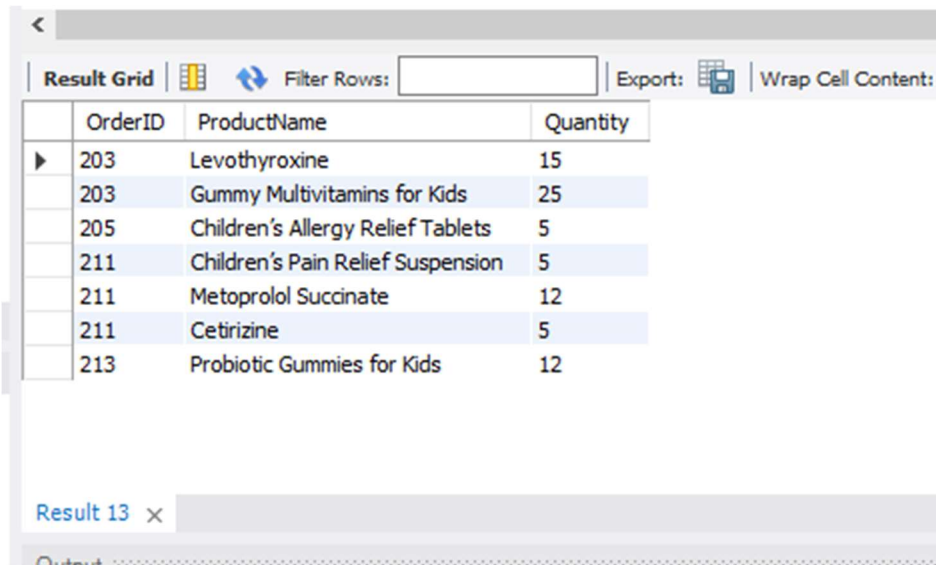
```
FROM Customers
```

```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
```

```
JOIN Products ON Products.ProductID = OrderDetails.ProductID
```

```
WHERE Customers.Name = 'Priya Singh';
```



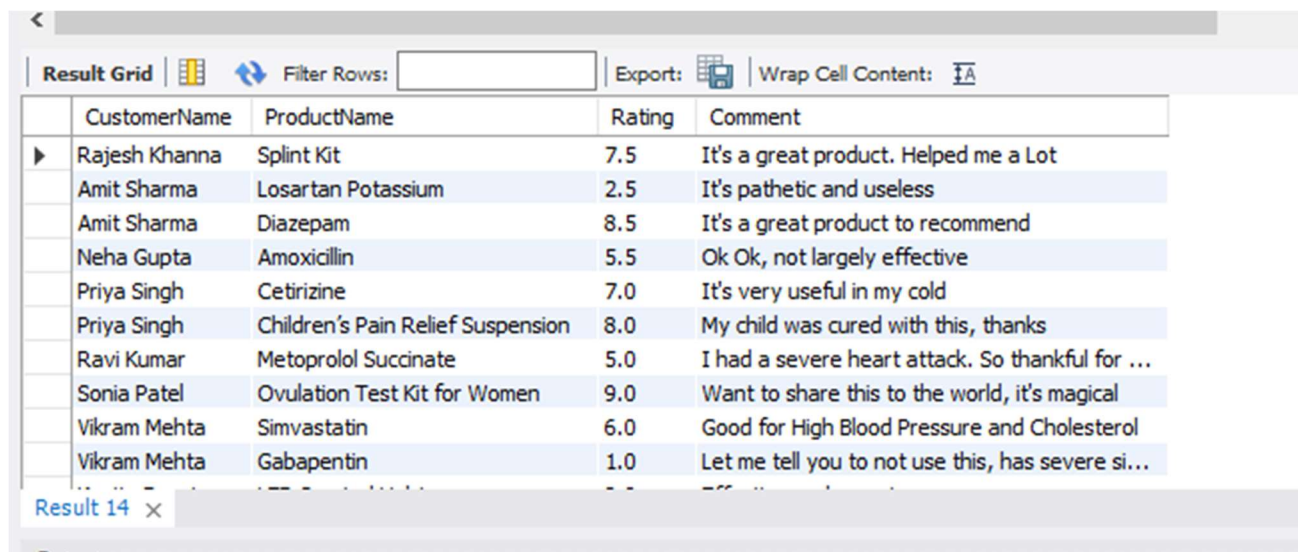
The screenshot shows a database query result grid with the following columns: OrderID, ProductName, and Quantity. The results are as follows:

OrderID	ProductName	Quantity
203	Levothyroxine	15
203	Gummy Multivitamins for Kids	25
205	Children's Allergy Relief Tablets	5
211	Children's Pain Relief Suspension	5
211	Metoprolol Succinate	12
211	Cetirizine	5
213	Probiotic Gummies for Kids	12

The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, and 'Export' and 'Wrap Cell Content' buttons. The result is labeled 'Result 13'.

Q8) Get all reviews along with product names and customer names.

```
SELECT Customers.Name AS CustomerName, Products.Name AS ProductName, Reviews.Rating, Reviews.Comment  
FROM Reviews  
JOIN Customers ON Customers.CustomerID = Reviews.CustomerID  
JOIN Products ON Products.ProductID = Reviews.ProductID;
```



The screenshot shows a database query result grid with 14 rows of data. The columns are CustomerName, ProductName, Rating, and Comment. The data is as follows:

CustomerName	ProductName	Rating	Comment
Rajesh Khanna	Splint Kit	7.5	It's a great product. Helped me a Lot
Amit Sharma	Losartan Potassium	2.5	It's pathetic and useless
Amit Sharma	Diazepam	8.5	It's a great product to recommend
Neha Gupta	Amoxicillin	5.5	Ok Ok, not largely effective
Priya Singh	Cetirizine	7.0	It's very useful in my cold
Priya Singh	Children's Pain Relief Suspension	8.0	My child was cured with this, thanks
Ravi Kumar	Metoprolol Succinate	5.0	I had a severe heart attack. So thankful for ...
Sonia Patel	Ovulation Test Kit for Women	9.0	Want to share this to the world, it's magical
Vikram Mehta	Simvastatin	6.0	Good for High Blood Pressure and Cholesterol
Vikram Mehta	Gabapentin	1.0	Let me tell you to not use this, has severe si...



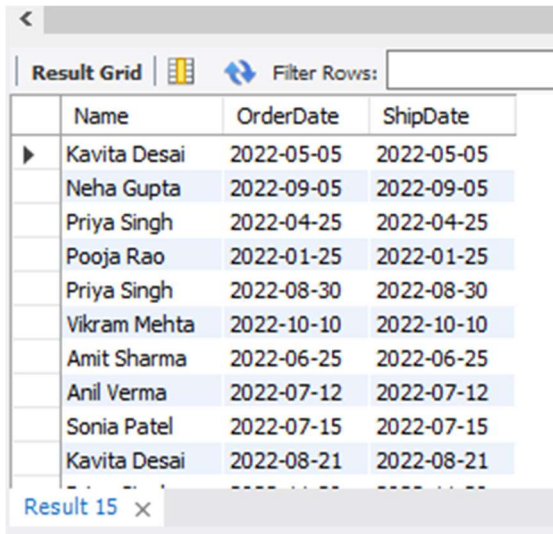
**Q9) List all customers who have placed orders along with the order dates and shipping dates.**

```
SELECT Customers.Name, Orders.OrderDate, Shipping.ShipDate
```

```
FROM Customers
```

```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
JOIN Shipping ON Shipping.OrderID = Orders.OrderID;
```



The screenshot shows a SQL query result grid with the following data:

	Name	OrderDate	ShipDate
▶	Kavita Desai	2022-05-05	2022-05-05
	Neha Gupta	2022-09-05	2022-09-05
	Priya Singh	2022-04-25	2022-04-25
	Pooja Rao	2022-01-25	2022-01-25
	Priya Singh	2022-08-30	2022-08-30
	Vikram Mehta	2022-10-10	2022-10-10
	Amit Sharma	2022-06-25	2022-06-25
	Anil Verma	2022-07-12	2022-07-12
	Sonia Patel	2022-07-15	2022-07-15
	Kavita Desai	2022-08-21	2022-08-21

Result 15 x

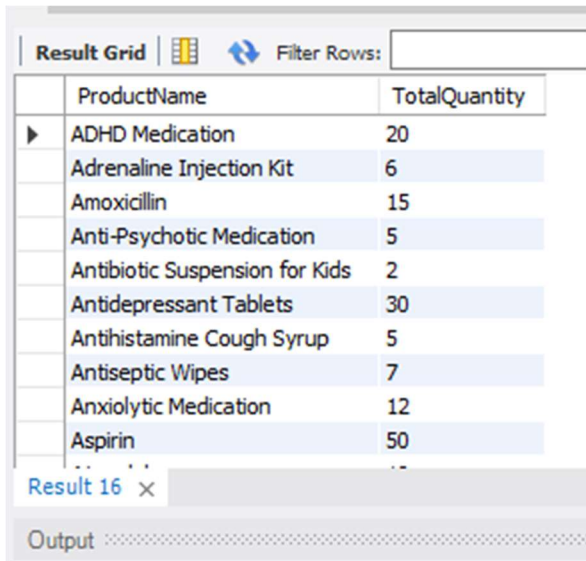
**Q10) Find the total quantity of each product ordered.**

```
SELECT Products.Name AS ProductName, SUM(OrderDetails.Quantity) AS TotalQuantity
```

```
FROM Products
```

```
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
```

```
GROUP BY Products.Name;
```



The screenshot shows a database query result grid. At the top, there is a tab labeled 'Result Grid' with a small icon of a grid. To the right of the tab is a 'Filter Rows:' input field. Below the tab, the results are displayed in a table with two columns: 'ProductName' and 'TotalQuantity'. The table contains 11 rows of data, each with a product name and its corresponding total quantity. The first row is 'ADHD Medication' with a quantity of 20. The second row is 'Adrenaline Injection Kit' with a quantity of 6. The third row is 'Amoxicillin' with a quantity of 15. The fourth row is 'Anti-Psychotic Medication' with a quantity of 5. The fifth row is 'Antibiotic Suspension for Kids' with a quantity of 2. The sixth row is 'Antidepressant Tablets' with a quantity of 30. The seventh row is 'Antihistamine Cough Syrup' with a quantity of 5. The eighth row is 'Antiseptic Wipes' with a quantity of 7. The ninth row is 'Anxiolytic Medication' with a quantity of 12. The tenth row is 'Aspirin' with a quantity of 50. The eleventh row is partially visible and shows a quantity of 12. Below the table, there is a tab labeled 'Result 16' with a close button (X). At the bottom, there is an 'Output' section with a dotted line indicating it is expandable.

ProductName	TotalQuantity
ADHD Medication	20
Adrenaline Injection Kit	6
Amoxicillin	15
Anti-Psychotic Medication	5
Antibiotic Suspension for Kids	2
Antidepressant Tablets	30
Antihistamine Cough Syrup	5
Antiseptic Wipes	7
Anxiolytic Medication	12
Aspirin	50
	12

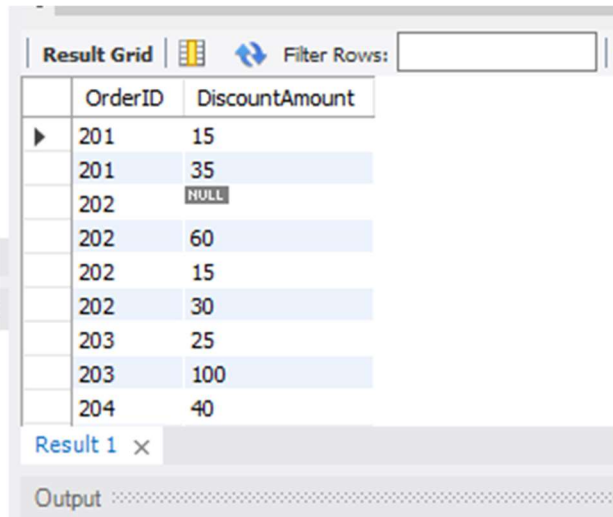
**Q11) Get all orders and their respective discount amounts if they have discounts.**

```
SELECT Orders.OrderID, Discounts.DiscountAmount
```

```
FROM Orders
```

```
LEFT JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
```

```
LEFT JOIN Discounts ON Discounts.ProductID = OrderDetails.ProductID;
```



	OrderID	DiscountAmount
▶	201	15
	201	35
	202	NULL
	202	60
	202	15
	202	30
	203	25
	203	100
	204	40

Result 1 ×

Output

**Q12) What is the total discount amount applied to each order?**

```
SELECT DISTINCT Orders.OrderID, COALESCE(SUM(Discounts.DiscountAmount),0) AS TotalDiscountAmount
FROM Orders
LEFT JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
LEFT JOIN Discounts ON OrderDetails.ProductID = Discounts.ProductID
GROUP BY Orders.OrderID;
```

Result Grid		Filter Rows:	Export:	Wrap
	OrderID	TotalDiscountAmount		
▶	201	50		
	202	105		
	203	125		
	204	110		
	205	20		
	206	55		
	207	65		
	208	50		
	209	35		
Result 2 x				
Output				

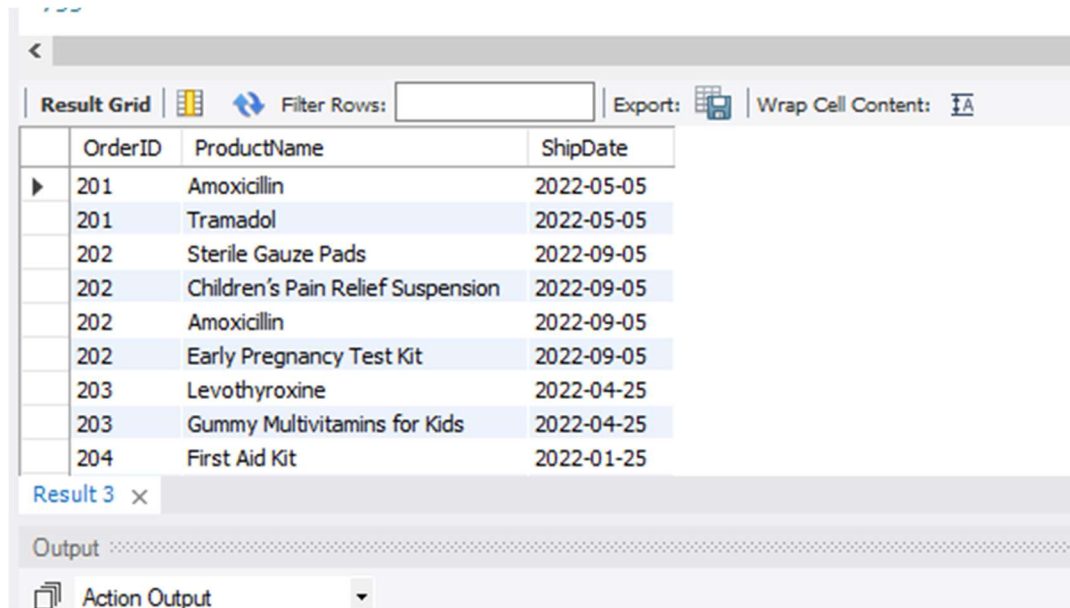
**Q13) List all orders with products and their respective shipping dates.**

```
SELECT OrderDetails.OrderID, Products.Name AS ProductName, Shipping.ShipDate
```

```
FROM Products
```

```
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
```

```
JOIN Shipping ON Shipping.OrderID = OrderDetails.OrderID;
```



The screenshot shows a database query result grid. The grid has a toolbar at the top with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with 10 rows and 3 columns: OrderID, ProductName, and ShipDate. The data is as follows:

OrderID	ProductName	ShipDate
201	Amoxicillin	2022-05-05
201	Tramadol	2022-05-05
202	Sterile Gauze Pads	2022-09-05
202	Children's Pain Relief Suspension	2022-09-05
202	Amoxicillin	2022-09-05
202	Early Pregnancy Test Kit	2022-09-05
203	Levothyroxine	2022-04-25
203	Gummy Multivitamins for Kids	2022-04-25
204	First Aid Kit	2022-01-25

Below the table, there is a tab labeled 'Result 3' and an 'Output' section with a dropdown menu set to 'Action Output'.

**14) Find customers who have not placed any orders.**

SELECT Customers.Name

FROM Customers

LEFT JOIN Orders ON Orders.CustomerID = Customers.CustomerID

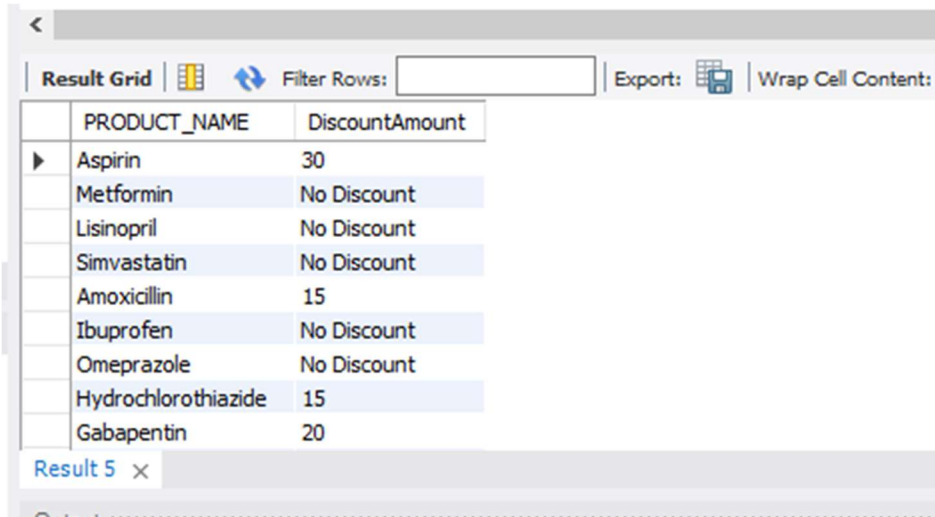
WHERE Orders.OrderID IS NULL;

Result Grid			Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 
Name					

Result 76 x

**Q15) Get products with their discounts if available.**

```
SELECT Products.Name AS PRODUCT_NAME, COALESCE(Discounts.DiscountAmount,"No Discount") AS DiscountAmount  
FROM Products  
  
LEFT JOIN Discounts ON Discounts.ProductID = Products.ProductID;
```



The screenshot shows a SQL query result grid with the following data:

	PRODUCT_NAME	DiscountAmount
▶	Aspirin	30
	Metformin	No Discount
	Lisinopril	No Discount
	Simvastatin	No Discount
	Amoxicillin	15
	Ibuprofen	No Discount
	Omeprazole	No Discount
	Hydrochlorothiazide	15
	Gabapentin	20

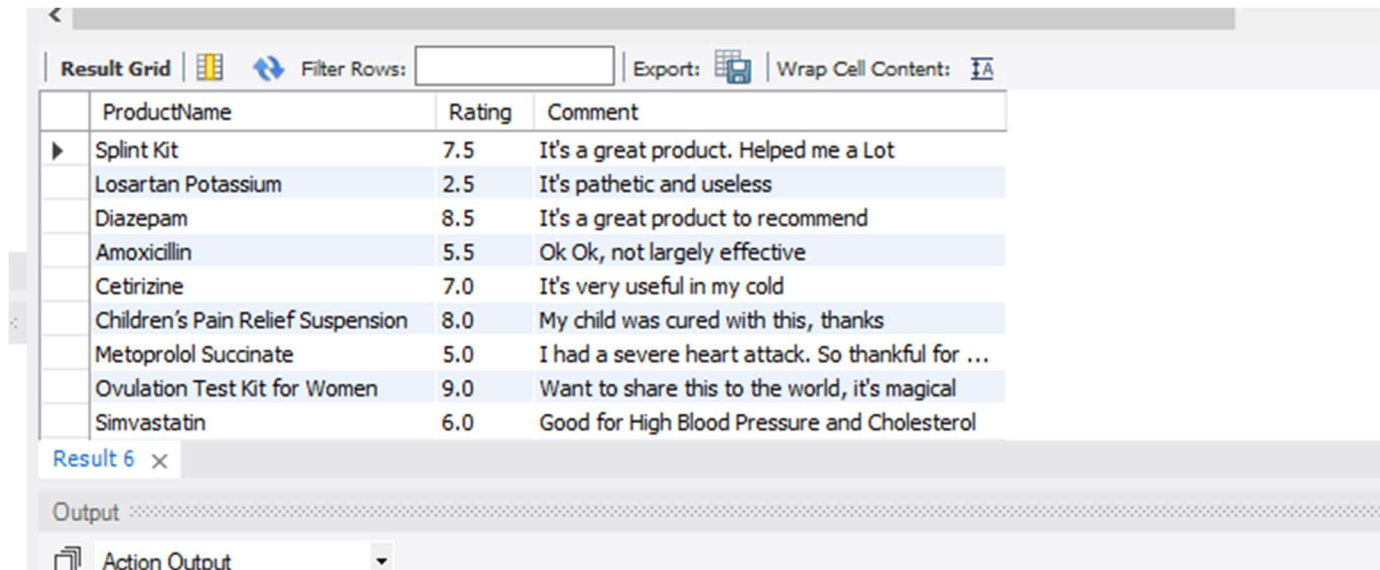
Result 5 x

**Q16) List all products that have been reviewed along with their review details.**

```
SELECT Products.Name AS ProductName, Reviews.Rating, Reviews.Comment
```

```
FROM Products
```

```
JOIN Reviews ON Reviews.ProductID = Products.ProductID;
```



The screenshot shows a database query result grid with the following data:

	ProductName	Rating	Comment
▶	Splint Kit	7.5	It's a great product. Helped me a Lot
	Losartan Potassium	2.5	It's pathetic and useless
	Diazepam	8.5	It's a great product to recommend
	Amoxicillin	5.5	Ok Ok, not largely effective
	Cetirizine	7.0	It's very useful in my cold
	Children's Pain Relief Suspension	8.0	My child was cured with this, thanks
	Metoprolol Succinate	5.0	I had a severe heart attack. So thankful for ...
	Ovulation Test Kit for Women	9.0	Want to share this to the world, it's magical
	Simvastatin	6.0	Good for High Blood Pressure and Cholesterol

Below the table, there is a tab labeled "Result 6" and a section labeled "Output" with a dropdown menu set to "Action Output".



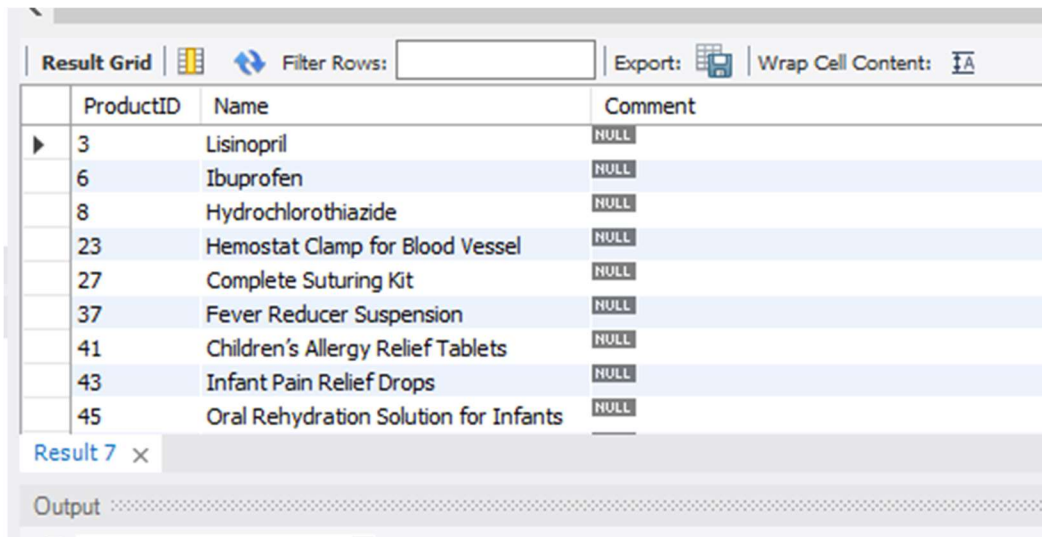
**Q17) List products that have never been reviewed.**

```
SELECT Products.ProductID, Products.Name, Reviews.Comment
```

```
FROM Products
```

```
LEFT JOIN Reviews ON Reviews.ProductID = Products.ProductID
```

```
WHERE Reviews.ProductID IS NULL;
```



The screenshot shows a SQL query result grid with the following columns: ProductID, Name, and Comment. The results list 10 products, all of which have a NULL value in the Comment column, indicating they have never been reviewed. The interface includes a 'Result Grid' tab, a 'Filter Rows' search bar, and 'Export' and 'Wrap Cell Content' options.

	ProductID	Name	Comment
▶	3	Lisinopril	NULL
	6	Ibuprofen	NULL
	8	Hydrochlorothiazide	NULL
	23	Hemostat Clamp for Blood Vessel	NULL
	27	Complete Suturing Kit	NULL
	37	Fever Reducer Suspension	NULL
	41	Children's Allergy Relief Tablets	NULL
	43	Infant Pain Relief Drops	NULL
	45	Oral Rehydration Solution for Infants	NULL

Result 7 x

Output

**Q18) Find customers who have ordered products in the "Pediatrics" category.**

```
SELECT DISTINCT Customers.Name AS CustomerName, Categories.CategoryName
```

```
FROM Customers
```

```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
```

```
JOIN Products ON Products.ProductID = OrderDetails.ProductID
```

```
JOIN Categories ON Categories.CategoryID = Products.CategoryID
```

```
WHERE Categories.CategoryName = "Pediatrics";
```

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

	CustomerName	CategoryName
▶	Akash Verma	Pediatrics
	Priya Singh	Pediatrics
	Anil Verma	Pediatrics
	Anita Rao	Pediatrics
	Preeti Desai	Pediatrics
	Neha Gupta	Pediatrics
	Amit Bansal	Pediatrics
	Pooja Rao	Pediatrics
	Ravi Kumar	Pediatrics

Result 8 x

Output : .....

Action Output ▼

#	Time	Action	Message
✓ 7	08:43:01	SELECT Products.ProductID, Products.Name, Reviews.Comment FROM Products LEFT JO...	25 row(s) returned
✓ 8	08:44:06	SELECT DISTINCT Customers.Name AS CustomerName, Categories.CategoryName FROM ...	15 row(s) returned

**Q19) List all customers who have reviewed a product they purchased.**

```
SELECT DISTINCT Customers.Name AS CustomerName
```

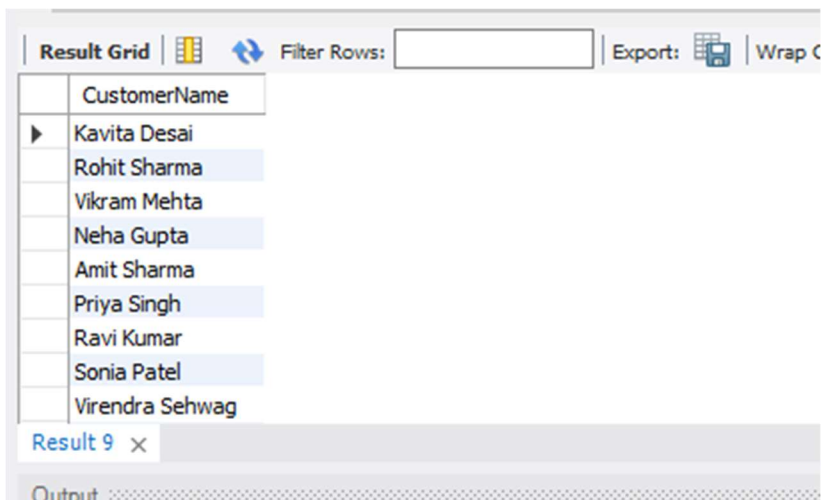
```
FROM Customers
```

```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
```

```
JOIN Reviews ON Reviews.ProductID = OrderDetails.ProductID
```

```
WHERE OrderDetails.ProductID = Reviews.ProductID;
```



The screenshot shows a SQL query result grid with a single column titled 'CustomerName'. The grid contains ten rows of customer names. The interface includes a 'Filter Rows' search bar, an 'Export' button, and a 'Wrap' option. Below the grid, there is a tab labeled 'Result 9' and an 'Output' section.

CustomerName
Kavita Desai
Rohit Sharma
Vikram Mehta
Neha Gupta
Amit Sharma
Priya Singh
Ravi Kumar
Sonia Patel
Virendra Sehwag

**Q20) Find the top 5 most expensive products and their categories.**

```
SELECT Products.Name AS ProductName, Products.Price, Categories.CategoryName
```

```
FROM Products
```

```
JOIN Categories ON Categories.CategoryID = Products.CategoryID
```

```
ORDER BY Products.Price DESC
```

```
LIMIT 5;
```

Result Grid	Filter Rows:	Export:
ProductName	Price	CategoryName
Defibrillator Unit	1500.00	Emergency_Medicine
LED Surgical Light	1250.00	Surgery
Portable Oxygen Tank	250.00	Emergency_Medicine
Adrenaline Injection Kit	70.00	Emergency_Medicine
Pediatric Probiotic Powder	66.00	Pediatrics

Result 82 ×

**Q21) List all orders where the total price exceeds Rs 500.**

```
SELECT Orders.OrderID, SUM(Products.Price*OrderDetails.Quantity) AS Total_Price
```

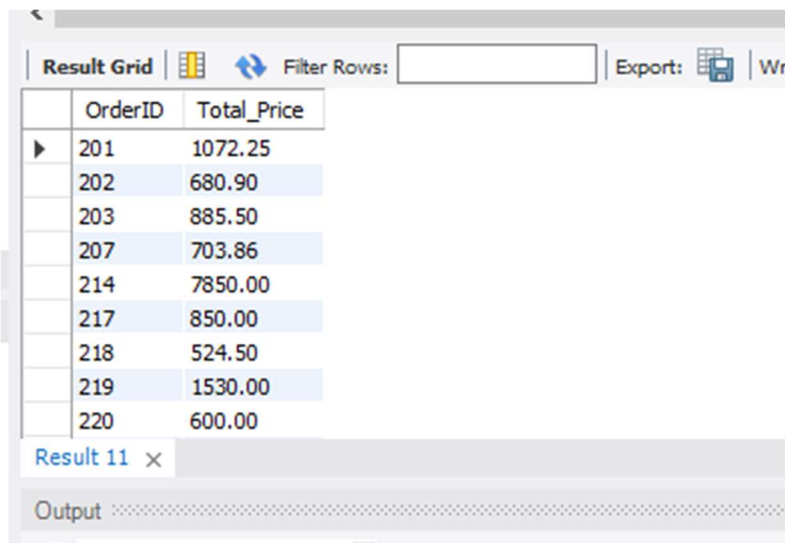
```
FROM Orders
```

```
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
```

```
JOIN Products ON Products.ProductID = OrderDetails.ProductID
```

```
GROUP BY Orders.OrderID
```

```
HAVING Total_Price > 500;
```



The screenshot shows a SQL query result grid with the following data:

OrderID	Total_Price
201	1072.25
202	680.90
203	885.50
207	703.86
214	7850.00
217	850.00
218	524.50
219	1530.00
220	600.00

The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, and an 'Export' button. The results are displayed in a table with columns 'OrderID' and 'Total\_Price'. The total price for each order is listed, and the results are filtered to show only orders where the total price is greater than 500.

**Q22) Find products in the "Gynecology" category with an average rating above 7.**

```
SELECT Products.Name AS ProductName, AVG(Reviews.Rating) AS AverageRating
```

```
FROM Products
```

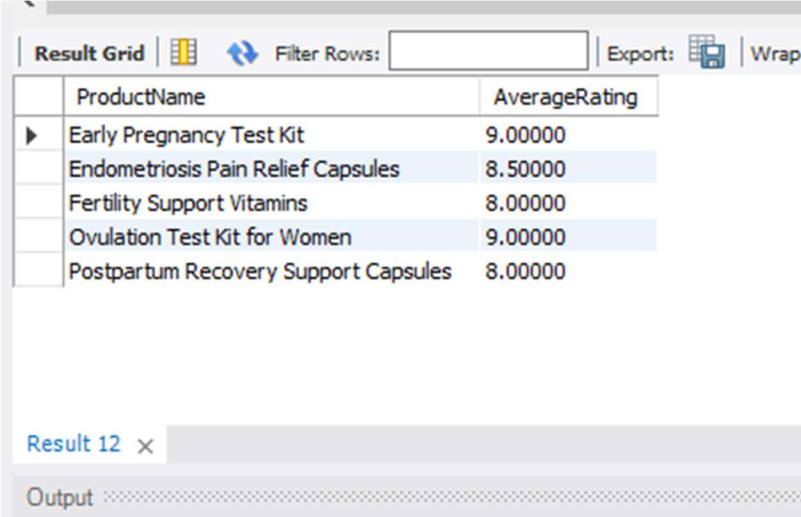
```
JOIN Reviews ON Reviews.ProductID = Products.ProductID
```

```
JOIN Categories ON Categories.CategoryID = Products.CategoryID
```

```
WHERE Categories.CategoryName = "Gynecology"
```

```
GROUP BY Products.Name
```

```
HAVING AverageRating > 7;
```



The screenshot shows a database query result grid. At the top, there is a toolbar with a 'Result Grid' button, a 'Filter Rows' input field, an 'Export' button, and a 'Wrap' button. Below the toolbar is a table with two columns: 'ProductName' and 'AverageRating'. The table contains five rows of data. The first row is 'Early Pregnancy Test Kit' with an 'AverageRating' of 9.00000. The second row is 'Endometriosis Pain Relief Capsules' with an 'AverageRating' of 8.50000. The third row is 'Fertility Support Vitamins' with an 'AverageRating' of 8.00000. The fourth row is 'Ovulation Test Kit for Women' with an 'AverageRating' of 9.00000. The fifth row is 'Postpartum Recovery Support Capsules' with an 'AverageRating' of 8.00000. Below the table, there is a 'Result 12' tab and an 'Output' section.

ProductName	AverageRating
Early Pregnancy Test Kit	9.00000
Endometriosis Pain Relief Capsules	8.50000
Fertility Support Vitamins	8.00000
Ovulation Test Kit for Women	9.00000
Postpartum Recovery Support Capsules	8.00000

**Q23) List customers who have placed more than 3 orders.**

```
SELECT Customers.Name, COUNT(Orders.OrderID) AS OrderCount
```

```
FROM Customers
```

```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
GROUP BY Customers.Name
```

```
HAVING OrderCount > 3;
```

Result Grid			Filter Rows:
	Name	OrderCount	
▶	Priya Singh	4	

Result 85 ×

**Q24) Find all orders that include products with a discount.**

SELECT DISTINCT Orders.OrderID

FROM Orders

JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID

JOIN Discounts ON Discounts.ProductID = OrderDetails.ProductID

Result Grid		Filter Rows:	Export:	Wra
	OrderID			
▶	218			
	201			
	202			
	242			
	212			
	203			
	211			
	216			
	217			
Result 14		x		



**Q25) List products with a price higher than the average price of all products.**

SELECT Name, Price

FROM Products

WHERE Price > (SELECT AVG(Price) FROM Products);

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	Name	Price			
▶	Ibuprofen	65.00			
	Hydrochlorothiazide	61.00			
	LED Surgical Light	1250.00			
	Pediatric Probiotic Powder	66.00			
	Adrenaline Injection Kit	70.00			
	Defibrillator Unit	1500.00			
	Portable Oxygen Tank	250.00			

Products 87 ×



**Q26) Find the total number of products in each category.**

```
SELECT Categories.CategoryName, COUNT(Products.Name) AS ProductCount
```

```
FROM Categories
```

```
JOIN Products ON Products.CategoryID = Categories.CategoryID
```

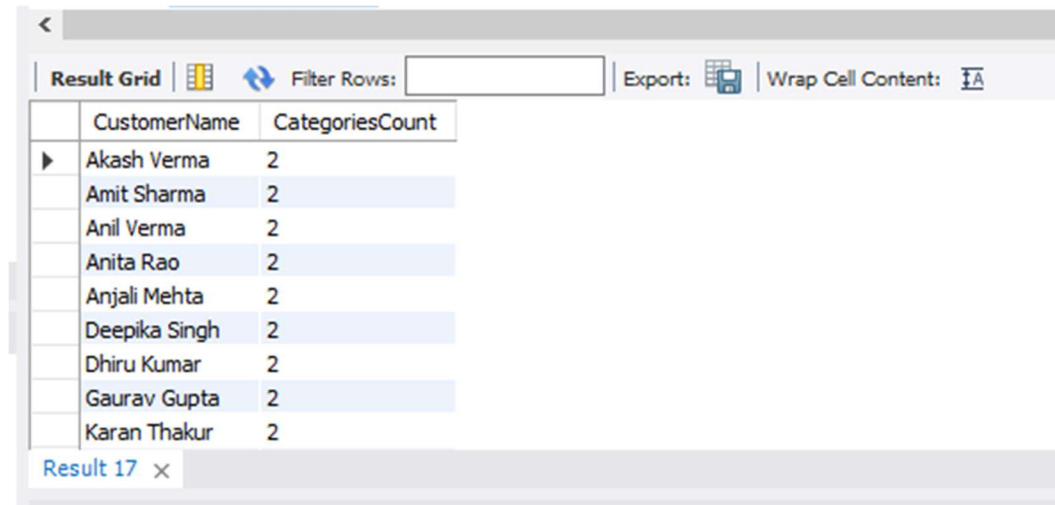
```
GROUP BY CategoryName;
```

Result Grid   Filter Rows: <input type="text"/>		
	CategoryName	ProductCount
▶	Emergency_Medicine	15
	Gynecology	12
	Internal_Medicine	20
	Pediatrics	18
	Psychiatry	10
	Surgery	10

Result 88  

**Q27) Find customers who have made purchases in multiple categories.**

```
SELECT Customers.Name AS CustomerName, COUNT(DISTINCT Products.CategoryID) AS CategoriesCount  
FROM Customers  
  
JOIN Orders ON Orders.CustomerID = Customers.CustomerID  
  
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID  
  
JOIN Products ON Products.ProductID = OrderDetails.ProductID  
  
GROUP BY Customers.Name  
  
HAVING CategoriesCount > 1;
```



The screenshot shows a database query result grid. The grid has two columns: 'CustomerName' and 'CategoriesCount'. There are 10 rows of data, each representing a customer who has purchased products from exactly two different categories. The interface includes a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. The result is labeled 'Result 17'.

CustomerName	CategoriesCount
Akash Verma	2
Amit Sharma	2
Anil Verma	2
Anita Rao	2
Anjali Mehta	2
Deepika Singh	2
Dhiru Kumar	2
Gaurav Gupta	2
Karan Thakur	2

**Q28) List the total sales value of each customer.**

```
SELECT Customers.Name AS CustomerName, SUM(OrderDetails.Quantity * Products.Price) AS TotalSales
```

```
FROM Customers
```

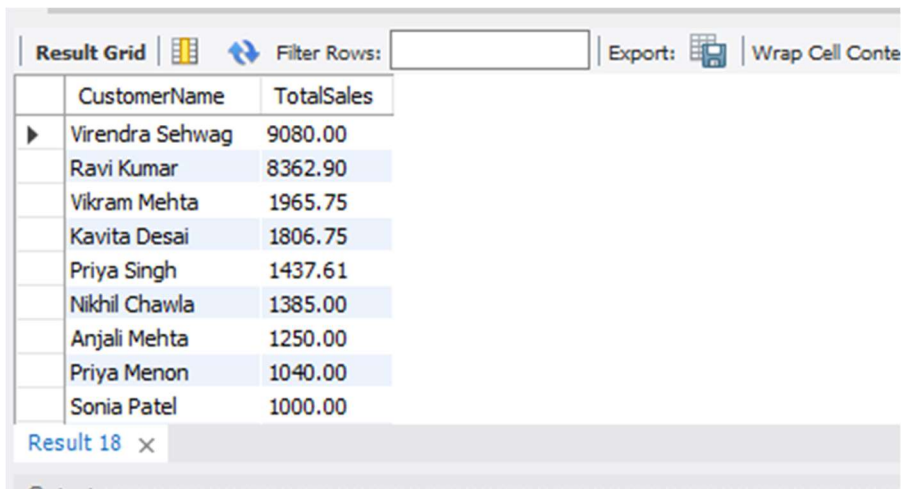
```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
```

```
JOIN Products ON Products.ProductID = OrderDetails.ProductID
```

```
GROUP BY CustomerName
```

```
ORDER BY TotalSales DESC;
```



The screenshot shows a SQL query result grid with two columns: CustomerName and TotalSales. The results are ordered by TotalSales in descending order. The grid includes a toolbar with options like 'Filter Rows', 'Export', and 'Wrap Cell Content'. A tab at the bottom indicates 'Result 18'.

CustomerName	TotalSales
Virendra Sehwag	9080.00
Ravi Kumar	8362.90
Vikram Mehta	1965.75
Kavita Desai	1806.75
Priya Singh	1437.61
Nikhil Chawla	1385.00
Anjali Mehta	1250.00
Priya Menon	1040.00
Sonia Patel	1000.00

**Q29) Find the product that has been ordered the most by quantity.**

```
SELECT Products.Name AS ProductName, SUM(OrderDetails.Quantity) AS TotalQuantity
```

```
FROM Products
```

```
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
```

```
GROUP BY ProductName
```

```
ORDER BY TotalQuantity DESC
```

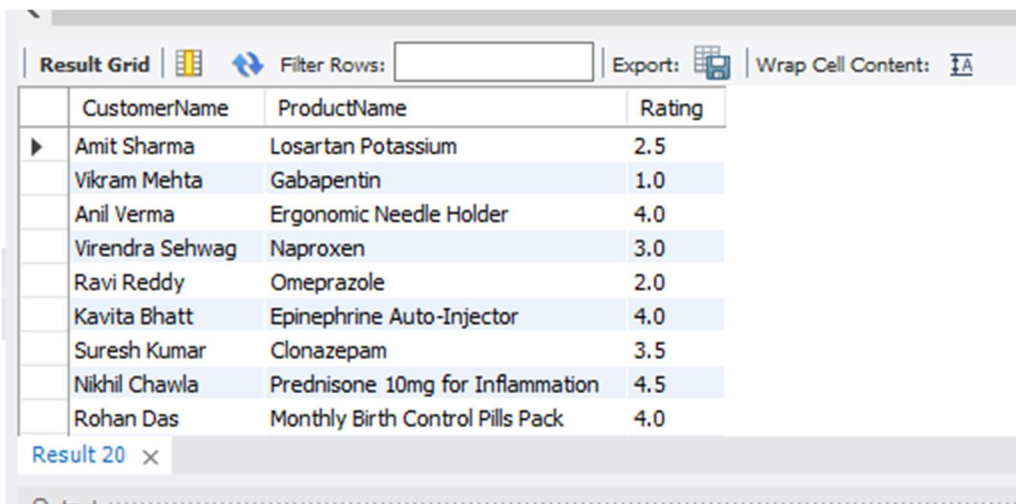
```
LIMIT 1;
```

Result Grid			Filter Rows:
	ProductName	TotalQuantity	
▶	Aspirin	50	

Result 91 ×

**Q30) Identify customers who have purchased a product that they later reviewed poorly (rating < 5).**

```
SELECT DISTINCT Customers.Name AS CustomerName, Products.Name AS ProductName, Reviews.Rating
FROM Customers
JOIN Reviews ON Reviews.CustomerID = Customers.CustomerID
JOIN Products ON Products.ProductID = Reviews.ProductID
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
WHERE Reviews.Rating < 5;
```



The screenshot shows a SQL query result grid with the following columns: CustomerName, ProductName, and Rating. The grid displays 10 rows of data, with the first row highlighted. The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' button. The bottom of the grid shows 'Result 20' and a close button.

	CustomerName	ProductName	Rating
▶	Amit Sharma	Losartan Potassium	2.5
	Vikram Mehta	Gabapentin	1.0
	Anil Verma	Ergonomic Needle Holder	4.0
	Virendra Sehwal	Naproxen	3.0
	Ravi Reddy	Omeprazole	2.0
	Kavita Bhatt	Epinephrine Auto-Injector	4.0
	Suresh Kumar	Clonazepam	3.5
	Nikhil Chawla	Prednisone 10mg for Inflammation	4.5
	Rohan Das	Monthly Birth Control Pills Pack	4.0

**Q31) List the orders that have the highest total discount applied.**

```
SELECT Orders.OrderID, SUM(Discounts.DiscountAmount) AS Total_Discount
```

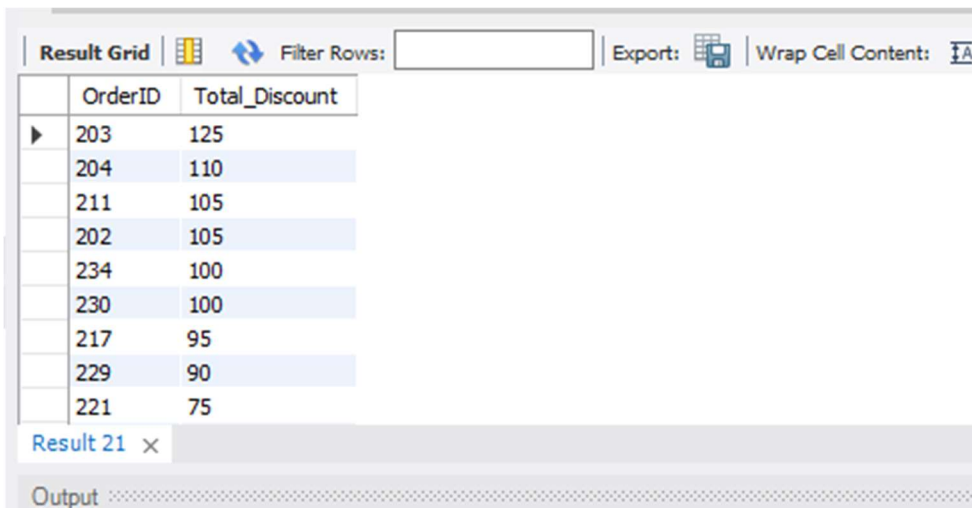
```
FROM Orders
```

```
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
```

```
JOIN Discounts ON Discounts.ProductID = OrderDetails.ProductID
```

```
GROUP BY Orders.OrderID
```

```
ORDER BY Total_Discount DESC;
```



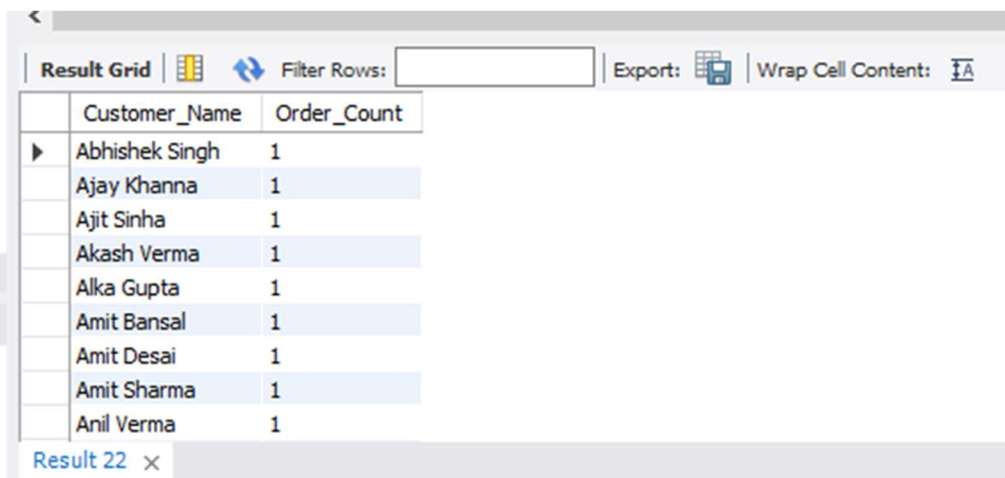
The screenshot shows a SQL query result grid with the following data:

	OrderID	Total_Discount
▶	203	125
	204	110
	211	105
	202	105
	234	100
	230	100
	217	95
	229	90
	221	75

Below the table, there is a tab labeled "Result 21" and an "Output" section.

**Q32) Find customers who have only placed one order.**

```
SELECT Customers.Name AS Customer_Name, COUNT(Orders.OrderID) AS Order_Count  
  
FROM Customers  
  
JOIN Orders ON Orders.CustomerID = Customers.CustomerID  
  
GROUP BY Customer_Name  
  
HAVING Order_Count = 1;
```



The screenshot shows a SQL query result grid with the following columns: Customer\_Name and Order\_Count. The results list 10 customers, each with an order count of 1. The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, and 'Export' and 'Wrap Cell Content' buttons.

Customer_Name	Order_Count
Abhishek Singh	1
Ajay Khanna	1
Ajit Sinha	1
Akash Verma	1
Alka Gupta	1
Amit Bansal	1
Amit Desai	1
Amit Sharma	1
Anil Verma	1

Result 22 x



**Q33) Get the average price of products in each category.**

```
SELECT Categories.CategoryName, AVG(Products.Price) AS Avg_Product_Price
```

```
FROM Products
```

```
JOIN Categories ON Categories.CategoryID = Products.CategoryID
```

```
GROUP BY CategoryName;
```

Result Grid			Filter Rows:	Export:
	CategoryName	Avg_Product_Price		
▶	Emergency_Medicine	143.266667		
	Gynecology	26.416667		
	Internal_Medicine	26.239000		
	Pediatrics	20.393333		
	Psychiatry	24.900000		
	Surgery	153.800000		

Result 95 ×

**Q34) List the products that have been ordered but never reviewed.**

```
SELECT Products.Name AS ProductName, Reviews.Comment
```

```
FROM Products
```

```
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
```

```
LEFT JOIN Reviews ON Reviews.ProductID = OrderDetails.ProductID
```

```
WHERE Reviews.ProductID IS NULL;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
ProductName	Comment			
▶ Lisinopril	NULL			
Ibuprofen	NULL			
Hydrochlorothiazide	NULL			
Hemostat Clamp for Blood Vessel	NULL			
Complete Suturing Kit	NULL			
Fever Reducer Suspension	NULL			
Children's Allergy Relief Tablets	NULL			
Children's Allergy Relief Tablets	NULL			
Infant Pain Relief Drops	NULL			

Result 24 ×

**Q35) List all products with their categories that have a price higher than the average price in their category.**

```
SELECT Products.Name AS ProductName, Products.Price, Categories.CategoryName
```

```
FROM Products
```

```
JOIN Categories ON Categories.CategoryID = Products.CategoryID
```

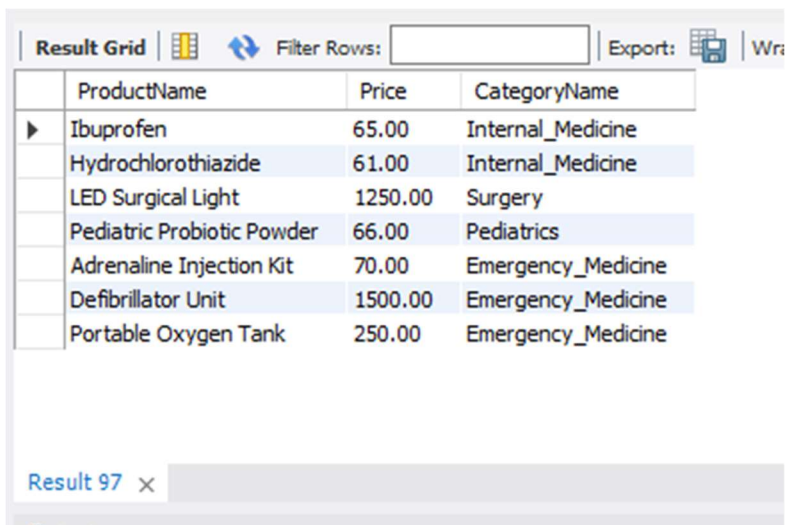
```
WHERE Products.Price > (
```

```
    SELECT AVG(Price)
```

```
FROM Products
```

```
WHERE CategoryID = Products.CategoryID
```

```
);
```



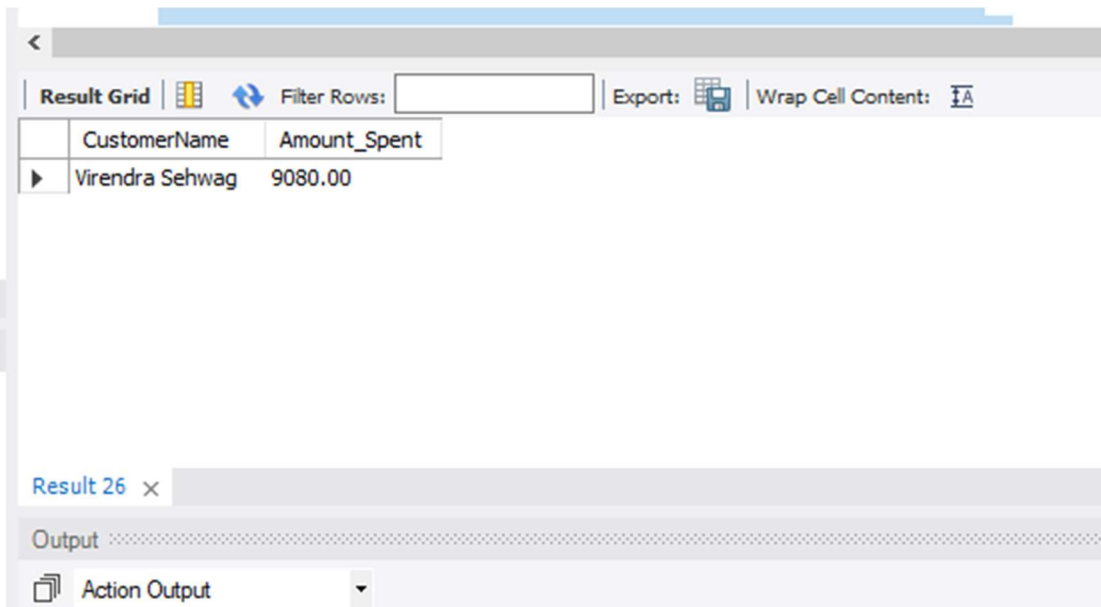
The screenshot shows a database query result grid with the following columns: ProductName, Price, and CategoryName. The results are as follows:

ProductName	Price	CategoryName
Ibuprofen	65.00	Internal_Medicine
Hydrochlorothiazide	61.00	Internal_Medicine
LED Surgical Light	1250.00	Surgery
Pediatric Probiotic Powder	66.00	Pediatrics
Adrenaline Injection Kit	70.00	Emergency_Medicine
Defibrillator Unit	1500.00	Emergency_Medicine
Portable Oxygen Tank	250.00	Emergency_Medicine

At the bottom of the screenshot, there is a tab labeled "Result 97" with a close button (X).

**Q36) Find the customer who has spent the most money in total.**

```
SELECT Customers.Name AS CustomerName, SUM(OrderDetails.Quantity*Products.Price) AS Amount_Spent  
FROM Customers  
  
JOIN Orders ON Orders.CustomerID = Customers.CustomerID  
  
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID  
  
JOIN Products ON Products.ProductID = OrderDetails.ProductID  
  
GROUP BY CustomerName  
  
ORDER BY Amount_Spent DESC  
  
LIMIT 1;
```



The screenshot shows a database query result grid. The grid has two columns: 'CustomerName' and 'Amount\_Spent'. The first row shows 'Virendra Sehwag' with an amount of '9080.00'. The interface includes a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the grid, there is a tab labeled 'Result 26' and an 'Output' section with a dropdown menu set to 'Action Output'.

CustomerName	Amount_Spent
Virendra Sehwag	9080.00

**Q37) Find all products with a rating below the average rating across all products.**

```
SELECT Products.ProductID, Products.Name AS ProductName, Reviews.Rating
```

```
FROM Products
```

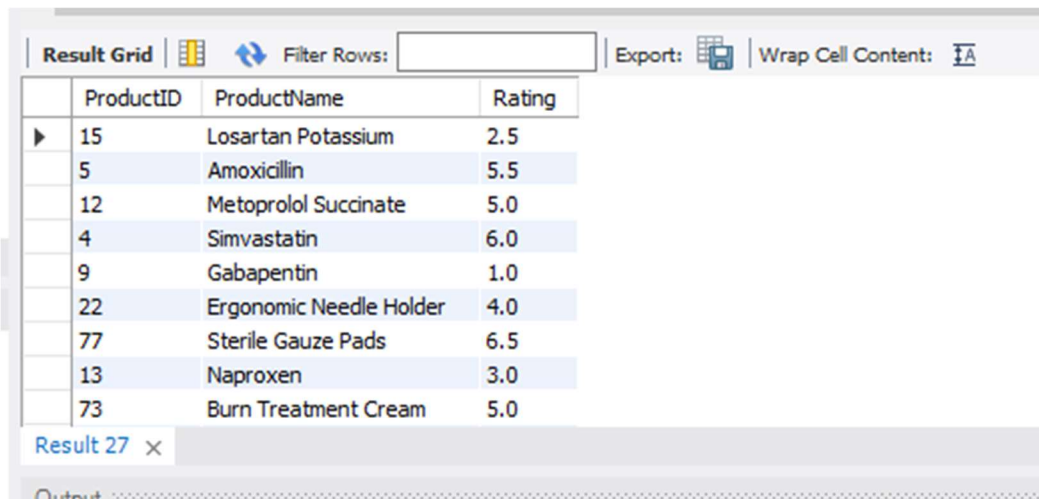
```
JOIN Reviews ON Reviews.ProductID = Products.ProductID
```

```
WHERE Reviews.Rating < (
```

```
    SELECT AVG(Reviews.Rating)
```

```
    FROM Reviews
```

```
);
```



The screenshot shows a SQL query result grid with the following columns: ProductID, ProductName, and Rating. The results are filtered to show products with a rating below the average rating across all products. The average rating is 4.0, so products with ratings less than 4.0 are displayed.

ProductID	ProductName	Rating
15	Losartan Potassium	2.5
5	Amoxicillin	5.5
12	Metoprolol Succinate	5.0
4	Simvastatin	6.0
9	Gabapentin	1.0
22	Ergonomic Needle Holder	4.0
77	Sterile Gauze Pads	6.5
13	Naproxen	3.0
73	Burn Treatment Cream	5.0

Result 27 x

**Q38) Get the total number of orders each customer has placed in each year.**

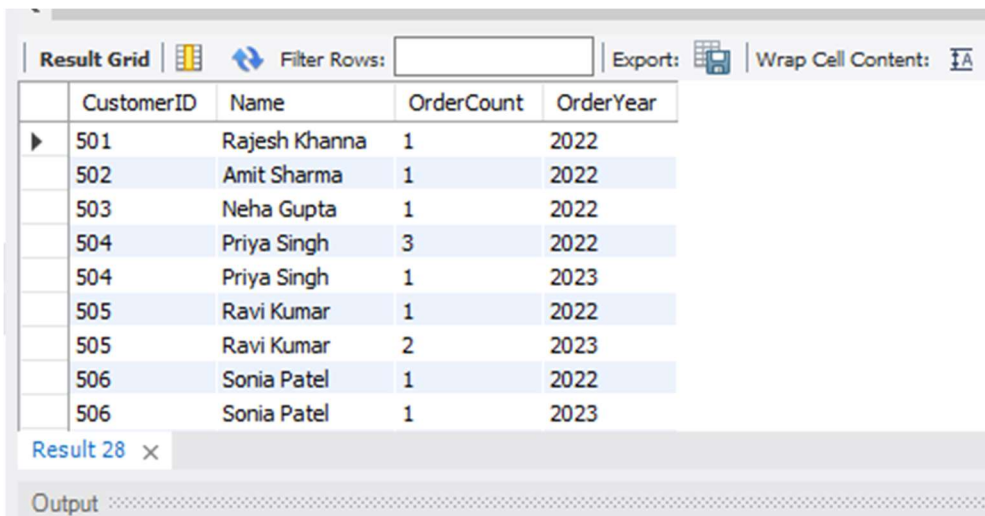
```
SELECT Customers.CustomerID, Customers.Name, COUNT(Orders.OrderID) AS OrderCount, YEAR(Orders.OrderDate) AS OrderYear
```

```
FROM Customers
```

```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
GROUP BY Customers.CustomerID, OrderYear
```

```
ORDER BY Customers.CustomerID;
```



The screenshot shows a SQL query result grid with the following data:

	CustomerID	Name	OrderCount	OrderYear
▶	501	Rajesh Khanna	1	2022
	502	Amit Sharma	1	2022
	503	Neha Gupta	1	2022
	504	Priya Singh	3	2022
	504	Priya Singh	1	2023
	505	Ravi Kumar	1	2022
	505	Ravi Kumar	2	2023
	506	Sonia Patel	1	2022
	506	Sonia Patel	1	2023

Below the table, there is a tab labeled "Result 28" and an "Output" section.

**Q39) Find the category with the lowest average product price.**

```
SELECT Categories.CategoryID, Categories.CategoryName, AVG(Products.Price) AS Avg_Price
```

```
FROM Categories
```

```
JOIN Products ON Products.CategoryID = Categories.CategoryID
```

```
GROUP BY Categories.CategoryID
```

```
ORDER BY Avg_Price ASC
```

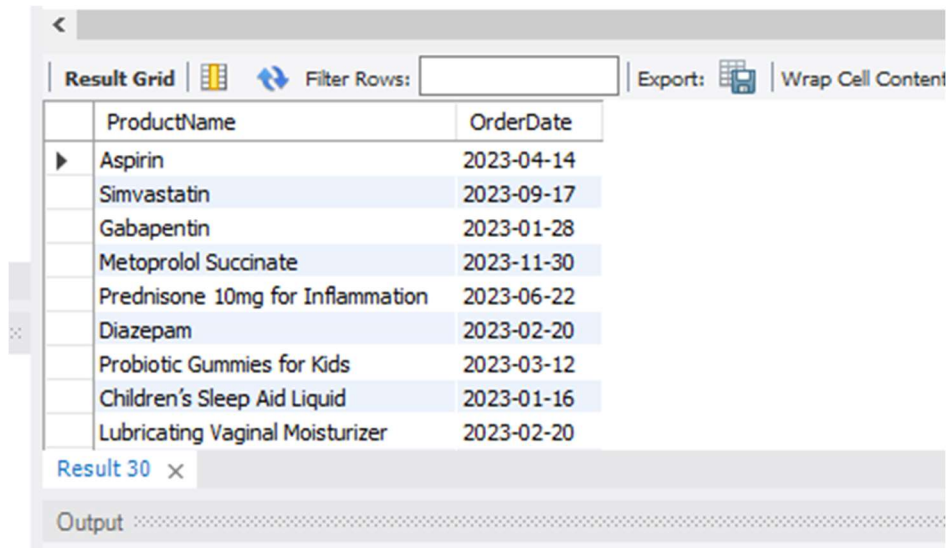
```
LIMIT 1;
```

Result Grid			
		Filter Rows:	
		Export:	
	CategoryID	CategoryName	Avg_Price
▶	103	Pediatrics	20.393333

Result 101 ×

**Q40) List all products that have not been ordered in the year 2022.**

```
SELECT Products.Name AS ProductName, Orders.OrderDate  
  
FROM Products  
  
LEFT JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID  
  
LEFT JOIN Orders ON Orders.OrderID = OrderDetails.OrderID  
  
WHERE Orders.OrderDate IS NULL  
  
OR YEAR(Orders.OrderDate) <> 2022;
```



The screenshot shows a SQL query result grid with the following data:

	ProductName	OrderDate
▶	Aspirin	2023-04-14
	Simvastatin	2023-09-17
	Gabapentin	2023-01-28
	Metoprolol Succinate	2023-11-30
	Prednisone 10mg for Inflammation	2023-06-22
	Diazepam	2023-02-20
	Probiotic Gummies for Kids	2023-03-12
	Children's Sleep Aid Liquid	2023-01-16
	Lubricating Vaginal Moisturizer	2023-02-20

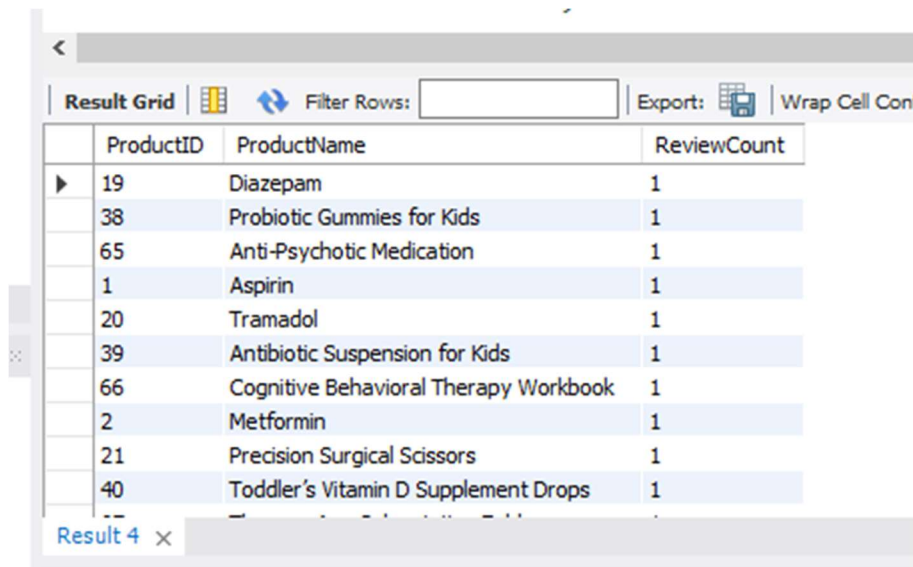
Result 30 x

Output



**Q41) Identify the products that have the most reviews.**

```
SELECT Products.ProductID, Products.Name AS ProductName, COUNT(Reviews.ReviewID) AS ReviewCount
FROM Products
JOIN Reviews ON Reviews.ProductID = Products.ProductID
GROUP BY Products.ProductID
ORDER BY ReviewCount DESC;
```



The screenshot shows a database query result grid. The grid has four columns: ProductID, ProductName, and ReviewCount. The results are ordered by ReviewCount in descending order. The first row shows ProductID 19, ProductName Diazepam, and ReviewCount 1. The second row shows ProductID 38, ProductName Probiotic Gummies for Kids, and ReviewCount 1. The third row shows ProductID 65, ProductName Anti-Psychotic Medication, and ReviewCount 1. The fourth row shows ProductID 1, ProductName Aspirin, and ReviewCount 1. The fifth row shows ProductID 20, ProductName Tramadol, and ReviewCount 1. The sixth row shows ProductID 39, ProductName Antibiotic Suspension for Kids, and ReviewCount 1. The seventh row shows ProductID 66, ProductName Cognitive Behavioral Therapy Workbook, and ReviewCount 1. The eighth row shows ProductID 2, ProductName Metformin, and ReviewCount 1. The ninth row shows ProductID 21, ProductName Precision Surgical Scissors, and ReviewCount 1. The tenth row shows ProductID 40, ProductName Toddler's Vitamin D Supplement Drops, and ReviewCount 1. The grid is titled 'Result 4' and has a 'Filter Rows' button and an 'Export' button.

ProductID	ProductName	ReviewCount
19	Diazepam	1
38	Probiotic Gummies for Kids	1
65	Anti-Psychotic Medication	1
1	Aspirin	1
20	Tramadol	1
39	Antibiotic Suspension for Kids	1
66	Cognitive Behavioral Therapy Workbook	1
2	Metformin	1
21	Precision Surgical Scissors	1
40	Toddler's Vitamin D Supplement Drops	1

**Q42) List customers along with the number of products they have reviewed.**

```
SELECT Customers.CustomerID,  
  
       Customers.Name,  
  
       (  
  
       SELECT COUNT(*)  
  
       FROM Reviews  
  
       WHERE Reviews.CustomerID = Customers.CustomerID  
  
       ) AS ReviewCount  
  
FROM  
  
Customers;
```

943 • **SELECT Customers.CustomerID,**

<

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

	CustomerID	Name	ReviewCount
▶	501	Rajesh Khanna	1
	502	Amit Sharma	2
	503	Neha Gupta	1
	504	Priya Singh	2
	505	Ravi Kumar	1
	506	Sonia Patel	1
	507	Vikram Mehta	2
	508	Kavita Desai	1
	509	Anil Verma	1
	510	Pooja Rao	1

Result 1 ×

Output

**Q43) Get the details of the most expensive product in each category.**

```
SELECT p.ProductID, p.Name AS ProductName, p.Price, p.StockQuantity, p.CategoryID
FROM Products p
WHERE p.Price = (
    SELECT MAX(p2.Price)
    FROM Products p2
    WHERE p2.CategoryID = p.CategoryID
);
```

Result Grid					
Filter Rows:		Edit:		Export/Import:	
	ProductID	ProductName	Price	StockQuantity	CategoryID
▶	6	Ibuprofen	65.00	451	101
	30	LED Surgical Light	1250.00	14	102
	47	Pediatric Probiotic Powder	66.00	130	103
	49	Bioidentical Hormone Replacement Therapy	48.00	55	104
	67	Therapy App Subscription Folder	40.00	70	105
	81	Defibrillator Unit	1500.00	20	106
*	NULL	NULL	NULL	NULL	NULL

Products 2 ×

**Q44) List the top 5 customers who have placed the most orders.**

```
SELECT Customers.Name, COUNT(Orders.OrderID) AS OrderCount
```


```
FROM Customers
```


```
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
```

```
GROUP BY Customers.Name
```

```
ORDER BY OrderCount DESC
```

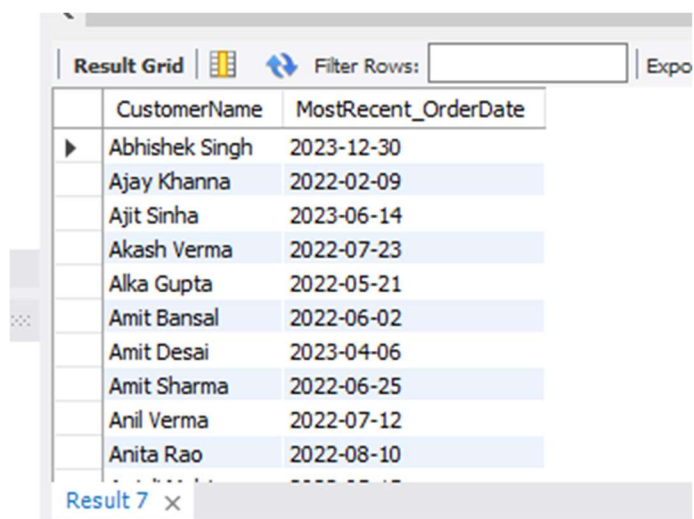
```
LIMIT 5;
```

Result Grid 			Filter Rows: <input type="text"/>	
	Name	OrderCount		
▶	Priya Singh	4		
	Ravi Kumar	3		
	Vikram Mehta	3		
	Kavita Desai	3		
	Sonia Patel	2		

Result 3 

**Q45) Find the most recent order date for each customer.**

```
SELECT Customers.Name AS CustomerName, MAX(Orders.OrderDate) AS MostRecent_OrderDate  
  
FROM Customers  
  
JOIN Orders ON Orders.CustomerID = Customers.CustomerID  
  
GROUP BY CustomerName;
```



The screenshot shows a SQL query result grid with two columns: CustomerName and MostRecent\_OrderDate. The results are sorted by the order date in descending order. The first row is Abhishek Singh with a date of 2023-12-30. The last row is Anita Rao with a date of 2022-08-10. The interface includes a 'Filter Rows' search bar and an 'Export' button.

CustomerName	MostRecent_OrderDate
Abhishek Singh	2023-12-30
Ajay Khanna	2022-02-09
Ajit Sinha	2023-06-14
Akash Verma	2022-07-23
Alka Gupta	2022-05-21
Amit Bansal	2022-06-02
Amit Desai	2023-04-06
Amit Sharma	2022-06-25
Anil Verma	2022-07-12
Anita Rao	2022-08-10

**Q46) List products that have been ordered more than 10 times and have a price greater than Rs 50.**

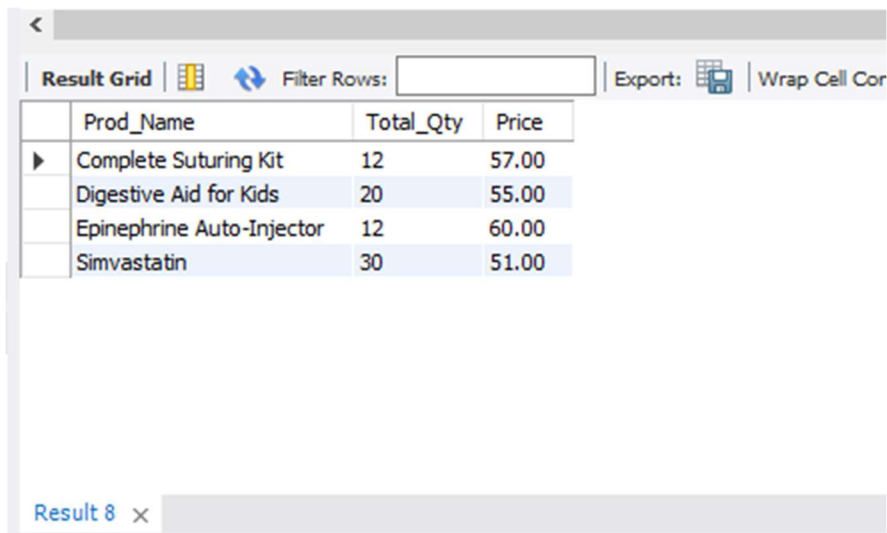
```
SELECT Products.Name AS Prod_Name, SUM(OrderDetails.Quantity) AS Total_Qty, Products.Price
```

```
FROM Products
```

```
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
```

```
GROUP BY Products.Name, Products.Price
```

```
HAVING Total_Qty > 10 AND Price > 50;
```



The screenshot shows a database query result grid with the following columns: Prod\_Name, Total\_Qty, and Price. The results are as follows:

Prod_Name	Total_Qty	Price
Complete Suturing Kit	12	57.00
Digestive Aid for Kids	20	55.00
Epinephrine Auto-Injector	12	60.00
Simvastatin	30	51.00

The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Cor' option. The result is labeled 'Result 8'.

**Q47) List products that have a discount applied and are priced above Rs 30.**

SELECT Products.Name, Products.Price, Discounts.DiscountAmount

FROM Products

JOIN Discounts ON Discounts.ProductID = Products.ProductID

WHERE Products.Price > 30;

Result Grid			
Filter Rows:		Export:	
Wrap Cell Content:			
	Name	Price	DiscountAmount
▶	First Aid Kit	45.00	40
	Prenatal Vitamins with DHA	32.00	20
	Tramadol	32.89	35
	Levothyroxine	32.00	25
	Therapy App Subscription Folder	40.00	10
	Kids' Cough Suppressant	32.00	35
	Precision Surgical Scissors	38.00	30
	LED Surgical Light	1250.00	40
	Hydrochlorothiazide	61.00	15
	Bioidentical Hormone Replacement Therapy	48.00	25

**Q48) Find customers who have placed orders across multiple years.**

```
SELECT Orders.CustomerID, Customers.Name
```

```
FROM Orders
```

```
JOIN Customers ON Customers.CustomerID = Orders.CustomerID
```

```
GROUP BY Orders.CustomerID
```

```
HAVING COUNT(DISTINCT YEAR(OrderDate)) > 1;
```

Result Grid			Filter Rows:
	CustomerID	Name	
▶	504	Priya Singh	
	505	Ravi Kumar	
	506	Sonia Patel	
	507	Vikram Mehta	
	508	Kavita Desai	

Result 7 ×





**Q49) Find orders that were shipped on the same day they were placed.**

```
SELECT Orders.OrderID, Orders.OrderDate, Shipping.ShipDate
```

```
FROM Orders
```

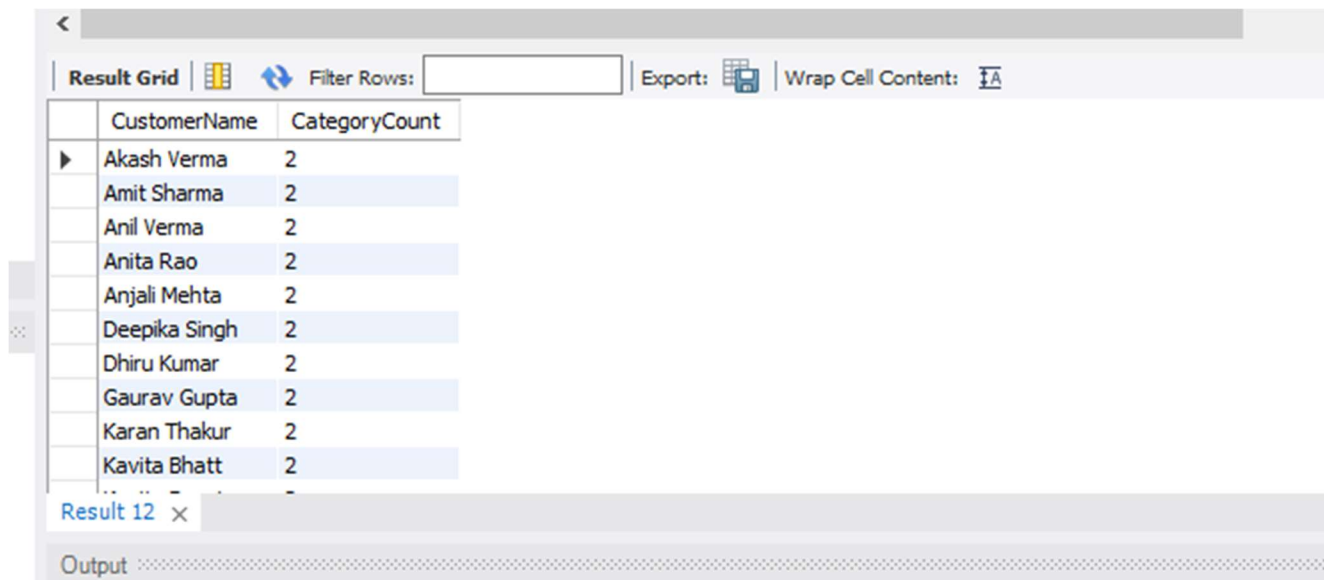
```
JOIN Shipping ON Shipping.OrderID = Orders.OrderID
```

```
WHERE Shipping.ShipDate = Orders.OrderDate;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	OrderID	OrderDate	ShipDate
▶	201	2022-05-05	2022-05-05
	202	2022-09-05	2022-09-05
	203	2022-04-25	2022-04-25
	204	2022-01-25	2022-01-25
	205	2022-08-30	2022-08-30
	206	2022-10-10	2022-10-10
	207	2022-06-25	2022-06-25
	208	2022-07-12	2022-07-12
	209	2022-07-15	2022-07-15
	210	2022-08-21	2022-08-21
Result 8 x			

**Q50) List customers who have made purchases in more than one category.**

```
SELECT Customers.Name AS CustomerName, COUNT(DISTINCT Products.CategoryID) AS CategoryCount
FROM Customers
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID
JOIN Products ON Products.ProductID = OrderDetails.ProductID
GROUP BY CustomerName
HAVING CategoryCount > 1;
```

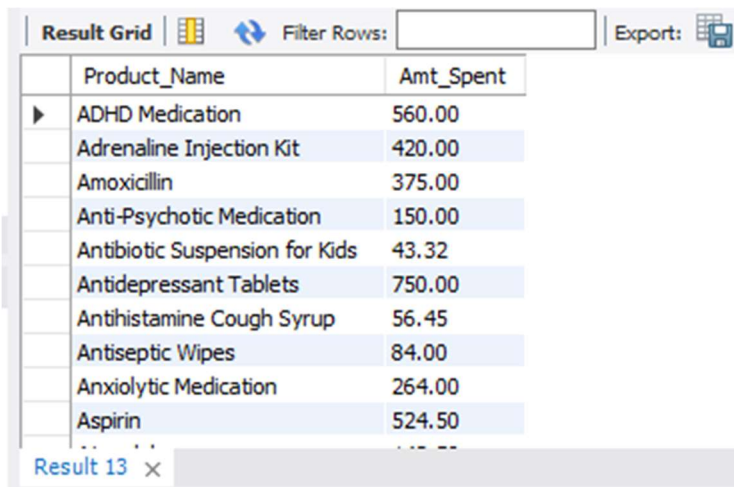


The screenshot shows a SQL query result grid with the following columns: CustomerName and CategoryCount. The grid displays 12 rows, all with a CategoryCount of 2. The interface includes a 'Result Grid' tab, a 'Filter Rows' search bar, and an 'Export' button. Below the grid, there is a 'Result 12' tab and an 'Output' section.

CustomerName	CategoryCount
Akash Verma	2
Amit Sharma	2
Anil Verma	2
Anita Rao	2
Anjali Mehta	2
Deepika Singh	2
Dhiru Kumar	2
Gaurav Gupta	2
Karan Thakur	2
Kavita Bhatt	2

**Q51) List all products along with the total amount spent on them.**

```
SELECT Products.Name AS Product_Name, SUM(OrderDetails.Quantity*Products.Price)
FROM Products
JOIN OrderDetails ON OrderDetails.ProductID = Products.ProductID
GROUP BY Products.Name;
```



The screenshot shows a SQL query result grid with the following data:

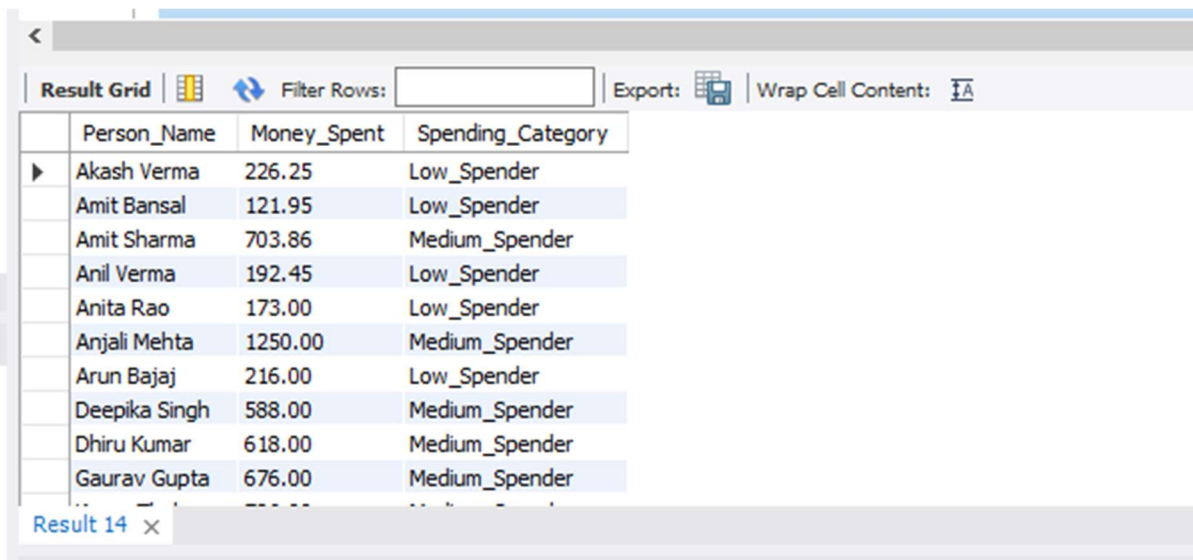
	Product_Name	Amt_Spent
▶	ADHD Medication	560.00
	Adrenaline Injection Kit	420.00
	Amoxicillin	375.00
	Anti-Psychotic Medication	150.00
	Antibiotic Suspension for Kids	43.32
	Antidepressant Tablets	750.00
	Antihistamine Cough Syrup	56.45
	Antiseptic Wipes	84.00
	Anxiolytic Medication	264.00
	Aspirin	524.50

Result 13

**Q52) Classify customers based on the total amount they have spent into three categories:**

**'Low Spender', 'Medium Spender', and 'High Spender'.**

```
SELECT Customers.Name AS Person_Name, SUM(OrderDetails.Quantity*Products.Price) AS Money_Spent,  
  
CASE  
  
    WHEN SUM(OrderDetails.Quantity*Products.Price) > 2000 THEN "High_Spender"  
  
    WHEN SUM(OrderDetails.Quantity*Products.Price) BETWEEN 500 AND 2000 THEN "Medium_Spender"  
  
    ELSE "Low_Spender"  
  
END AS Spending_Category  
  
FROM Customers  
  
JOIN Orders ON Orders.CustomerID = Customers.CustomerID  
  
JOIN OrderDetails ON OrderDetails.OrderID = Orders.OrderID  
  
JOIN Products ON Products.ProductID = OrderDetails.ProductID  
  
GROUP BY Customers.Name;
```



The screenshot shows a SQL query result grid with the following columns: Person\_Name, Money\_Spent, and Spending\_Category. The data is as follows:

Person_Name	Money_Spent	Spending_Category
Akash Verma	226.25	Low_Spender
Amit Bansal	121.95	Low_Spender
Amit Sharma	703.86	Medium_Spender
Anil Verma	192.45	Low_Spender
Anita Rao	173.00	Low_Spender
Anjali Mehta	1250.00	Medium_Spender
Arun Bajaj	216.00	Low_Spender
Deepika Singh	588.00	Medium_Spender
Dhiru Kumar	618.00	Medium_Spender
Gaurav Gupta	676.00	Medium_Spender