

**1.What is the average 'Total\_Spent' for transactions paid with 'Credit Card'?**

**SELECT**

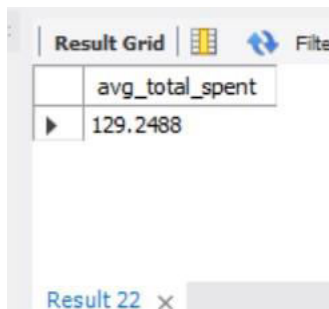
**AVG**(Total\_Spent) **AS** avg\_total\_spent

**FROM**

store\_sales

**WHERE**

Payment\_method = 'Credit Card';



The screenshot shows a 'Result Grid' window with a single column header 'avg\_total\_spent' and one data row containing the value '129.2488'. The window title is 'Result 22'.

avg_total_spent
129.2488

**2.How many transactions were made in 'Online' locations?**

**SELECT**

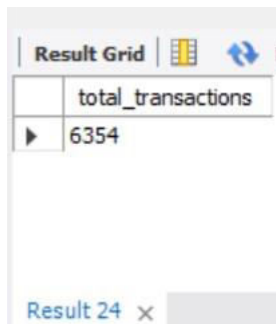
**COUNT**(Transaction\_ID) **AS** total\_transactions

**FROM**

store\_sales

**WHERE**

Location = 'Online';



The screenshot shows a 'Result Grid' window with a single column header 'total\_transactions' and one data row containing the value '6354'. The window title is 'Result 24'.

total_transactions
6354

**#3.Which 'Customer\_ID' has the highest total 'Total\_Spent'?**

**SELECT**

Customer\_ID,

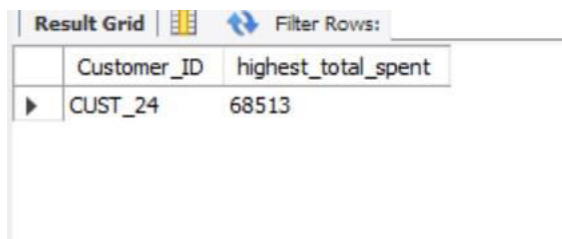
**MAX**(Total\_Spent) **AS** highest\_total\_spent

**FROM**

store\_sales

**GROUP BY**

Customer\_ID;



The screenshot shows a 'Result Grid' with two columns: 'Customer\_ID' and 'highest\_total\_spent'. The first row contains the values 'CUST\_24' and '68513'.

Customer_ID	highest_total_spent
CUST_24	68513

**#4.What is the total 'Quantity' of 'Item 16 BEV' sold?**

**SELECT**

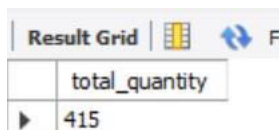
**SUM**(Quantity) **AS** total\_quantity

**FROM**

store\_sales

**WHERE**

Item = 'Item\_16\_BEV';



The screenshot shows a 'Result Grid' with one column: 'total\_quantity'. The first row contains the value '415'.

total_quantity
415

**#5. Calculate the average 'Price\_Per\_Unit' for items in the 'Butchers' category.**

**SELECT**

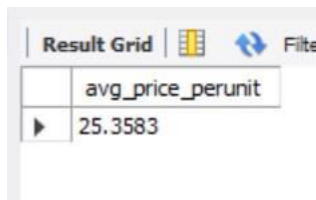
**AVG**(Price\_Per\_Unit) **AS** avg\_price\_perunit

**FROM**

store\_sales

**WHERE**

Category = 'Butchers';



A screenshot of a SQL query result grid. The grid has two columns: 'avg\_price\_perunit' and a value '25.3583'. The grid is titled 'Result Grid' and has a 'Filter' button.

	avg_price_perunit
▶	25.3583

**#6. How many transactions have a 'Discount\_Applied' value of '1'?**

**SELECT**

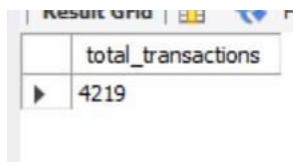
**COUNT**(Transaction\_ID) **AS** total\_transactions

**FROM**

store\_sales

**WHERE**

Discount\_Applied = '1';



A screenshot of a SQL query result grid. The grid has two columns: 'total\_transactions' and a value '4219'. The grid is titled 'Result Grid' and has a 'Filter' button.

	total_transactions
▶	4219

**#7. What is the 'Total\_Spent' for 'CUST\_09' in 'Patisserie' category?**

**SELECT**

Customer\_ID,

Category,

**SUM**(Total\_Spent) **AS** Total\_spent

**FROM**

store\_sales

**WHERE**

Customer\_ID = 'CUST\_09' AND

Category = 'Patisserie';



The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. The grid contains one row of data with the following columns: Customer\_ID, Category, and Total\_spent.

Customer_ID	Category	Total_spent
CUST_09	Patisserie	5666

**#8. List all the unique 'Payment\_Method' used in 'Online' transactions.**

**SELECT**

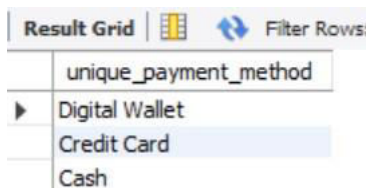
**DISTINCT**(Payment\_method) **AS** unique\_payment\_method

**FROM**

store\_sales

**WHERE**

Location = 'Online';



The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. The grid contains three rows of data under the column 'unique\_payment\_method'.

unique_payment_method
Digital Wallet
Credit Card
Cash

Result 38 x

**#9. What is the earliest 'Transaction\_Date' in the dataset?**

**SELECT**

**MIN**(Transaction\_Date)

**FROM**

store\_sales;

Result Grid		Filter Rows
	MIN(Transaction_Date)	
▶	2022-01-01	

**#10. Calculate the total revenue generated from 'Food' category items.**

**SELECT**

**SUM**(Total\_Spent) **AS** total\_revenue

**FROM**

store\_sales

**WHERE**

Category = 'Food';

Result Grid		Filter Rows
	total_revenue	
▶	194995	

**#11. Identify the top 3 customers who spent the most money on a category 'Milk Products' in year 2022.**

**SELECT**

Customer\_ID,

**SUM**(Total\_Spent) **AS** Total\_Spent

**FROM**

Store\_sales

**WHERE**

Category = 'Milk Products' **AND YEAR**(Transaction\_Date) = 2022

**GROUP BY**

Customer\_ID

**ORDER BY**

Total\_Spent **DESC**

**LIMIT 3;**

Result Grid			Filter Rows:
	Customer_ID	Total_Spent	
▶	CUST_14	4358	
	CUST_22	3380	
	CUST_20	3118	

Result 58 x

**#12. Generate a report showing the percentage change in total sales for each month compared to the previous month.**

**WITH MonthlySales AS (**

**SELECT**

**MONTH(Transaction\_Date) AS month,**

**SUM(Total\_Spent) AS total\_sales**

**FROM**

store\_sales

**GROUP BY**

**MONTH(Transaction\_Date)**

**)**

**SELECT**

month,

total\_sales,

**LAG(total\_sales,1,0)OVER(ORDER BY month) AS Previous\_Month\_Sales,**

**CASE**

**WHEN LAG(total\_sales,1,0) OVER(ORDER BY month) = 0 THEN NULL**

**ELSE**

**(total\_sales - LAG(total\_sales,1,0)OVER(ORDER BY month)) \* 100.0 /**

**LAG(total\_sales,1,0)OVER(ORDER BY month)**

**END AS Percentage\_change**

**FROM**

MonthlySales

**ORDER BY**

month;

Result Grid				
		Filter Rows:		
		Export:		
		Wrap		
	month	Total_Sales	Previous_Month_Sales	Percentage_Change
▶	1	174574	0	NULL
	2	119796	174574	-31.37810
	3	122512	119796	2.26719
	4	125736	122512	2.63158
	5	124705	125736	-0.81997
	6	129902	124705	4.16744
	7	131636	129902	1.33485
	8	123416	131636	-6.24449
	9	129458	123416	4.89564
	10	119522	129458	-7.67508
	11	122455	119522	2.45394
	12	129784	122455	5.98506

Result 61 x

**#13. List all items where the total revenue generated exceeds the average revenue for all items.**

**SELECT**

Item,

**SUM**(Total\_Spent) **AS** Item\_revenue

**FROM**

store\_sales

**GROUP BY**

Item

**HAVING**

**SUM**(Total\_Spent) > (**SELECT AVG**(Total\_Spent) **FROM** store\_sales);

Result Grid   Filter Rows:		
	Item	Item_revenue
▶	Item_10_PAT	2562
	Item_17_MILK	13456
	Item_12_BUT	10687
	Item_16_BEV	11427
	Item_6_FOOD	4841
	NULL	79145
	Item_1_FOOD	675
	Item_16_FUR	7543
	Item_22_BUT	19738
	Item_3_BUT	3272
	Item_2_FOOD	753
	Item_24_PAT	12260
	Item_16_MILK	17274
	Item_17_PAT	14500
	Item_13_EHE	11109
	Item_7_BEV	5544
	Item_4_EHE	2962

Result 67 × 

**#14. Identify the month with the highest revenue for each payment method.**

**WITH** MonthlyPaymentMethod **AS** (

**SELECT**

Payment\_Method,

**MONTH**(Transaction\_Date) **AS** month,

**SUM**(Total\_Spent) **AS** total\_revenue,

**ROW\_NUMBER**()**OVER**(**PARTITION BY** Payment\_Method **ORDER BY**  
**SUM**(Total\_Spent) **DESC**) **AS** rn

**FROM**

store\_sales

**GROUP BY**

Payment\_Method,

**MONTH**(Transaction\_Date)

)

**SELECT**



Payment\_Method,  
month,  
total\_revenue

**FROM**

MonthlyPaymentMethod

**WHERE**

rn = 1;

Result Grid			
Filter Rows:			
	Payment_Method	month	total_revenue
▶	Cash	1	59815
	Credit Card	1	56822
	Digital Wallet	1	57937

Result 73 x

**#15. Find the percentage of transactions that were made with a discount, compared to those without a discount.**

**SELECT**

Discount\_Applied,

**COUNT**(Transaction\_ID) **AS** Transaction\_Count,

**COUNT**(Transaction\_ID)\***100.0** / (**SELECT COUNT**(\*) **FROM** store\_sales) **AS** Percentage

**FROM**

store\_sales

**GROUP BY**

Discount\_Applied;

Result Grid			
Filter Rows:			
	Discount_Applied	Transaction_Count	Percentage
▶	1	4219	33.55070
	0	4157	33.05765
	NULL	4199	33.39165

**#16.Create a new column called 'Revenue\_Per\_Unit' by dividing 'Total\_Spent' by 'Quantity'. Calculate the average 'Revenue\_Per\_Unit' for each category.**

**SELECT**

Category,

**AVG**(Revenue\_Per\_Unit) **AS** Avg\_Revenue\_Per\_Unit

**FROM**

(**SELECT**

Category,

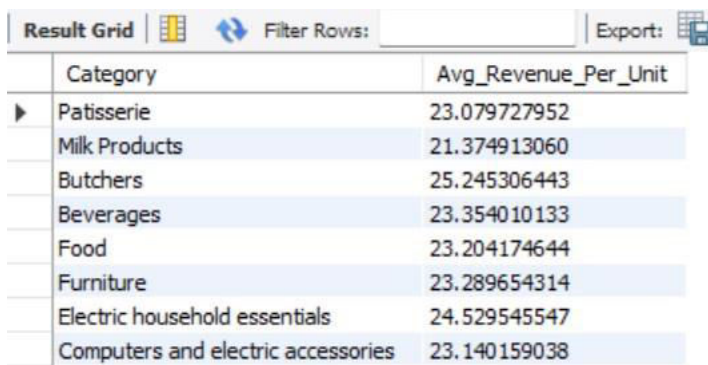
Total\_Spent \* **100.0** / Quantity **AS** Revenue\_Per\_Unit

**FROM**

store\_sales) **AS** revenue

**GROUP BY**

Category;



The screenshot shows a database query result grid with two columns: 'Category' and 'Avg\_Revenue\_Per\_Unit'. The grid contains 9 rows of data. The categories listed are Patisserie, Milk Products, Butchers, Beverages, Food, Furniture, Electric household essentials, and Computers and electric accessories. The average revenue per unit values are displayed to the right of each category name.

Category	Avg_Revenue_Per_Unit
Patisserie	23.079727952
Milk Products	21.374913060
Butchers	25.245306443
Beverages	23.354010133
Food	23.204174644
Furniture	23.289654314
Electric household essentials	24.529545547
Computers and electric accessories	23.140159038

**#17.Identify customers who have made purchases in both 'Online' and 'In-store' locations.**

**SELECT**

**DISTINCT**(s.Customer\_ID)

**FROM**

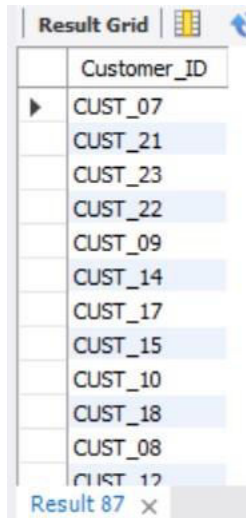
store\_sales s

**INNER JOIN**

store\_sales s1 **ON** s.Customer\_ID = s1.Customer\_ID

## WHERE

s.Location = 'Online' AND s1.Location = 'In-store';



The screenshot shows a 'Result Grid' window with a list of customer IDs. The first row is 'CUST\_07', followed by 'CUST\_21', 'CUST\_23', 'CUST\_22', 'CUST\_09', 'CUST\_14', 'CUST\_17', 'CUST\_15', 'CUST\_10', 'CUST\_18', 'CUST\_08', and 'CUST\_12'. The grid has a header row with 'Customer\_ID' and a status bar at the bottom indicating 'Result 87'.

Customer_ID
CUST_07
CUST_21
CUST_23
CUST_22
CUST_09
CUST_14
CUST_17
CUST_15
CUST_10
CUST_18
CUST_08
CUST_12

**#18. Create a pivot table showing the total 'Total\_Spent' for each 'Category' and 'Payment\_Method' combination. (Requires pivot table creation)**

**SELECT DISTINCT** Payment\_Method

**FROM** store\_sales;

## SELECT

Category,

**SUM(CASE WHEN Payment\_Method = 'Digital Wallet' THEN Total\_Spent ELSE 0 END) AS**  
'Digital Wallet',

**SUM(CASE WHEN Payment\_Method = 'Credit Card' THEN Total\_Spent ELSE 0 END) AS**  
'Credit Card',


**SUM(CASE WHEN Payment\_Method = 'Cash' THEN Total\_Spent ELSE 0 END) AS 'Cash'**

## FROM

store\_sales

## GROUP BY

Category;

Result Grid				
Filter Rows:		Export:  Wrap Cell Content:		
	Category	Digital Wallet	Credit Card	Cash
▶	Patisserie	60696	59998	61655
	Milk Products	59459	60878	59932
	Butchers	62718	72895	72693
	Beverages	65551	61382	70328
	Food	61620	63602	69773
	Furniture	63025	64839	67616
	Electric household essentials	75349	60566	68067
	Computers and electric accessories	59333	63400	68121

Result 91 ×

**#19. Identify 'Customer\_ID' who have made more than 2 transactions and have an average 'Total\_Spent' above 100.**

**SELECT**

Customer\_ID,

**COUNT**(Transaction\_ID) **AS** transactions,

**AVG**(Total\_Spent) **AS** avg\_total\_spent

**FROM**

store\_sales

**GROUP BY**

Customer\_ID

**HAVING**

transactions > 2 AND avg\_total\_spent >100;

Result Grid			
Filter Rows:			
	Customer_ID	transactions	avg_total_spent
▶	CUST_09	519	123.4719
	CUST_22	501	130.3650
	CUST_02	488	132.9829
	CUST_06	481	127.5761
	CUST_05	544	129.9089
	CUST_07	491	131.7809
	CUST_21	498	132.3298
	CUST_23	513	134.2328
	CUST_25	476	127.1467
	CUST_14	484	130.0064
	CUST_15	519	126.1058
	CUST_17	487	127.0395

**#20. Create a new column called 'Day\_of\_Week' based on the 'Transaction\_Date'. Analyze which day of the week has the highest 'Total\_Spent'.**

**SELECT**

**DAYNAME(Transaction\_Date) AS Day\_of\_Week,**

**SUM(Total\_Spent) AS Total\_Spent**

**FROM**

store\_sales

**GROUP BY**

DAYNAME(Transaction\_Date)

**ORDER BY**

Total\_Spent **DESC**

**LIMIT 1;**

Result Grid		
Filter Rows:		
	Day_of_Week	Total_Spent
▶	Friday	232980