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';
```

Result Grid | Filte

avg_total_spent
129.2488

Result 22 ×

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

```
SELECT
    COUNT(Transaction_ID) AS total_transactions
FROM
    store_sales
WHERE
    Location = 'Online';
```

Result Grid

total_transactions
6354

Result 24 ×

#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;
```

Result Grid | Filter Rows:

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';
```

Result Grid | F

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';
```

Result Grid | Filte

	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';
```

Result Grid

	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';
```

Result Grid | Filter Rows:

	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';
```

Result Grid | Filter Rows:

	unique_payment_method
▶	Digital Wallet
	Credit Card
	Cash

Result 38 ×

#9.What is the earliest 'Transaction_Date' in the dataset?

```
SELECT
        MIN(Transaction_Date)
FROM
        store_sales;
```

Result Grid | Filter Rov

	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 | Fi

	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

#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

#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

#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: | Expor

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;
```

Result Grid | Filter Rows: | Export:

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';

Result Grid

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

Result 87

#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

Result 95

#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