



# WALMART

## SALES ANALYSIS WITH SQL

# INTRODUCTION

## SQL

- SQL, or Structured Query Language, is a standardized programming language used for managing and manipulating relational databases.
- It allows users to perform various operations, such as querying data, updating records, inserting new data, and deleting existing records.

## WALMART

- Walmart is one of the largest retail corporations in the world, founded by Sam Walton in 1962 in Rogers, Arkansas.
- It operates a chain of hypermarkets, discount department stores, and grocery stores.

# OVERVIEW & SCOPE

This project focuses on the September month sales analysis on the year 2024, designing a comprehensive relational database aimed at efficiently managing Walmart's critical data assets.

This project includes four tables:

## CATEGORY TABLE

Organizing products into categories for better navigation and reporting

## CUSTOMER TABLE

Storing detailed information about each customer, including contact information and payment methods.

## PRODUCT TABLE

Cataloguing product details such as name, category, price, and stock quantity.

## ORDERS TABLE

Tracking customer orders, including order dates, quantities, and total amounts, facilitating efficient order processing and analysis

# OBJECTIVE OF THE PROJECT

## DATA MANAGEMENT

Creates a structured database that allows for easy access and management of customer and product information, leading to improved data integrity and consistency.

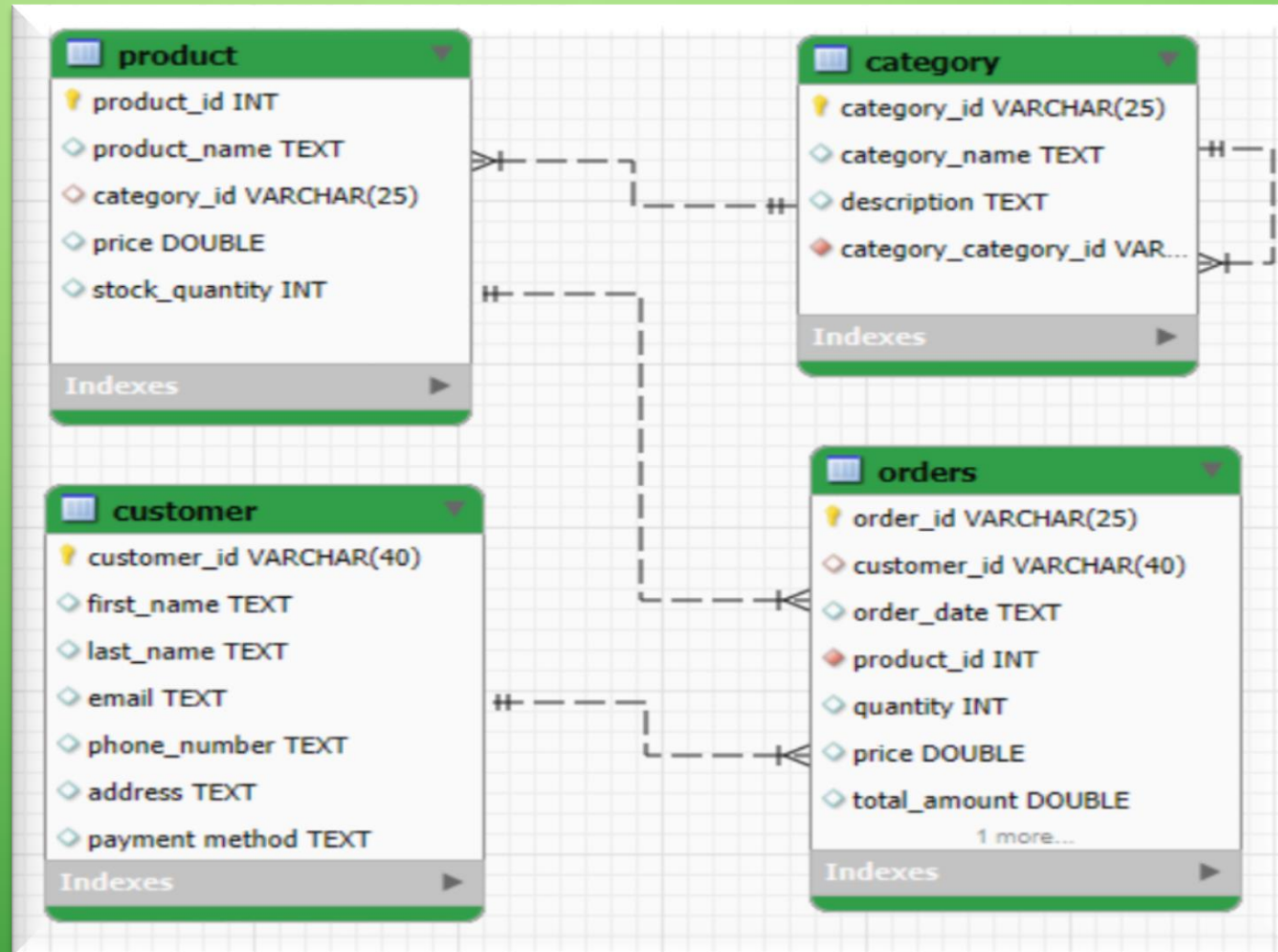
## ORDER PROCESSING

Streamlines the order management process, enabling quicker transaction handling and better tracking of customer purchases

## INSIGHTS & REPORTING

Develop capabilities for data analysis to derive actionable insights, such as sales trends, inventory levels, and customer preferences, which can inform marketing strategies and inventory management.

# ENTITY RELATIONSHIP DIAGRAM



# RELATIONSHIPS BETWEEN THE TABLES

## One-to-Many

### Customer to Orders

Each customer can place multiple orders. This relationship allows us to track all purchases made by a customer.

## One-to-Many

### Category to product

Each category can contain multiple products. This helps in organizing products for easier navigation and search.

## Many-to-One

### Orders to Customer

Each order is associated with only one customer. This ensures that all orders can be traced back to the individual who made the purchase.

## Many-to-One

### Orders to Products

Each order contains one specific product. This is essential for inventory management and sales tracking.

# USER CREATION & ACCESS GRANTING

QUERY:


```
Create user Walmart@localhost identified by 'walmart';
```

```
Grant all privileges on walmart.* to 'walmart'@'localhost';
```

OUTPUT:

## MySQL Connections

Local instance MySQL80

 root

 localhost:3306

walmart

 walmart

 127.0.0.1:3306



# CATEGORY TABLE

Select \* from Category;

category_id	category_name	description
C001	Dairy Products	Milk, cheese, yogurt, and other dairy items.
C002	Health & Wellness	Vitamins, supplements, and personal care products.
C003	Electronics	Smartphones, TVs, computers, and accessories.
C004	Office Supplies	Printers, paper, office furniture, and related items.
C005	Furniture	Home and office furniture, including chairs and desks.
C006	School Supplies	Stationery, writing tools, and educational materials.
C007	Apparel	Clothing items for men, women, and children.
C008	Groceries	Fresh fruits&vegetables, Spices and other cooking neccesities

Desc Category;

	Field	Type	Null	Key	Default	Extra
▶	category_name	text	YES		NULL	
	description	text	YES		NULL	
	category_id	varchar(25)	NO	PRI	NULL	



# CUSTOMER TABLE

Select \* from Customer;

customer_id	first_name	last_name	email	phone_number	address	payment method
C001	John	Doe	john.doe@example.com	(202) 555-0171	123 Maple St, Springfield	Ewallet
C002	Jane	Smith	jane.smith@example.com	(303) 555-0192	456 Oak St, Springfield	Cash
C003	Michael	Johnson	michael.j@example.com	(415) 555-0234	789 Pine St, Shelbyville	Credit card
C004	Emily	Davis	emily.davis@example.com	(646) 555-0125	101 Elm St, Capital City	Ewallet
C005	Robert	Brown	robert.brown@example.com	(702) 555-0183	202 Cedar St, Ogdenville	Ewallet
C006	Linda	Wilson	linda.wilson@example.com	(818) 555-0279	303 Birch St, North Haverbrook	Ewallet
C007	David	Miller	david.miller@example.com	(512) 555-0345	404 Walnut St, Springfield	Ewallet
C008	Sarah	Taylor	sarah.taylor@example.com	(617) 555-0467	505 Cherry St, Shelbyville	Ewallet
C009	James	Anderson	james.anderson@example.com	(213) 555-0119	606 Poplar St, Capital City	Credit card
C010	Patricia	Thomas	patricia.thomas@example.com	(305) 555-0248	707 Ash St, Ogdenville	Credit card
C011	Emma	Smith	Emma.Smith@example.com	(407) 555-0382	808 Maple St, Springfield	Ewallet
C012	Liam	Johnson	Liam.Johnson@example.com	(512) 555-0156	909 Oak St, Springfield	Cash
C013	Olivia	Williams	Olivia.Williams@example.com	(720) 555-0473	123 Pine St, Shelbyville	Ewallet
C014	Noah	Brown	Noah.Brown@example.com	(818) 555-0264	234 Elm St, Capital City	Ewallet
C015	Ava	Jones	Ava.Jones@example.com	(503) 555-0145	345 Cedar St, Ogdenville	Cash
C016	Elijah	Garcia	Elijah.Garcia@example.com	(312) 555-0361	456 Birch St, North Haverbrook	Cash
C017	Isabella	Miller	Isabella.Miller@example.com	(408) 555-0220	567 Walnut St, Springfield	Credit card
C018	Lucas	Davis	Lucas.Davis@example.com	(602) 555-0314	678 Cherry St, Shelbyville	Credit card
C019	Sophia	Rodriguez	Sophia.Rodriguez@example.com	(706) 555-0396	789 Poplar St, Capital City	Credit card
C020	Mason	Martinez	Mason.Martinez@example.com	(831) 555-0450	890 Ash St, Ogdenville	Ewallet
C021	Mia	Hernandez	Mia.Hernandez@example.com	(917) 555-0167	101 Maple St, Shelbyville	Ewallet
C022	Logan	Lopez	Logan.Lopez@example.com	(757) 555-0231	202 Oak St, Capital City	Ewallet
C023	Amelia	Gonzalez	Amelia.Gonzalez@example.com	(801) 555-0354	303 Pine St, Ogdenville	Credit card
C024	Ethan	Wilson	Ethan.Wilson@example.com	(619) 555-0198	404 Elm St, Springfield	Ewallet
C025	Harper	Anderson	Harper.Anderson@example.com	(718) 555-0303	505 Cedar St, North Haverbrook	Ewallet
C026	James	Thomas	James.Thomas@example.com	(925) 555-0487	606 Birch St, Shelbyville	Credit card
C027	Evelyn	Taylor	Evelyn.Taylor@example.com	(414) 555-0335	707 Walnut St, Capital City	Cash
C028	Aiden	Moore	Aiden.Moore@example.com	(405) 555-0128	808 Cherry St, Springfield	Credit card
C029	Abigail	Jackson	Abigail.Jackson@example.com	(516) 555-0273	909 Poplar St, Ogdenville	Cash
C030	Jackson	Martin	Jackson.Martin@example.com	(901) 555-0179	111 Ash St, North Haverbrook	Cash
C031	Ella	Lee	Ella.Lee@example.com	(505) 555-0132	222 Maple St, Capital City	Credit card

Desc Customer;

Field	Type	Null	Key	Default	Extra
customer_id	varchar(40)	NO	PRI	NULL	
first_name	text	YES		NULL	
last_name	text	YES		NULL	
email	text	YES		NULL	
phone_number	text	YES		NULL	
address	text	YES		NULL	
payment_method	text	YES		NULL	

# PRODUCT TABLE

Select \* from Product;

product_id	product_name	category_id	price	stock_quantity
1	Whole Milk	C001	2.99	100
2	Laptop	C003	999.99	50
3	Sofa	C005	699.99	20
4	Multivitamin Tablets	C002	12.99	150
5	Inkjet Printer	C004	119.99	30
6	Backpack	C006	39.99	200
7	Cheddar Cheese	C001	5.49	80
8	Smartphone	C003	599.99	75
9	Dining Table	C005	499.99	15
10	Greek Yogurt	C001	1.99	120
11	Spiral Notebook	C006	2.49	300
12	Bluetooth Headphones	C003	49.99	100
13	Highlighters	C006	7.99	50
14	Fish Oil Capsules	C002	14.99	60
15	4K TV	C003	799.99	25
16	Protein Powder	C002	29.99	40
17	Coffee Table	C005	149.99	18
18	Whiteboard Markers	C006	7.49	200
19	Desk Organizer	C004	24.99	50
20	Cream Cheese	C001	2.79	90
21	Recliner Chair	C005	399.99	10
22	Ream of Printer Paper	C004	5.99	120
23	Ballpoint Pens	C006	6.99	200
24	External Hard Drive	C003	1.99	120
25	Herbal Tea	C002	3.99	130
26	Filing Cabinet	C004	4.99	80
27	Ice Cream	C008	89.99	150
28	Wireless Mouse	C004	4.49	100
29	TV Stand	C003	24.99	30

Desc Product;

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	
product_name	text	YES		NULL	
category_id	varchar(25)	YES	MUL	NULL	
price	double	YES		NULL	
stock_quantity	int	YES		NULL	

# ORDERS TABLE

Select \* from Orders;

order_id	customer_id	order_date	product_id	quantity	price	total_amount	status
OD01	C003	28-09-2024	45	7	39.99	279.93	Shipped
OD02	C007	29-09-2024	12	1	49.99	49.99	Delivered
OD03	C012	14-09-2024	67	1	13.99	13.99	Processing
OD04	C005	15-09-2024	34	4	3.29	13.16	Delivered
OD05	C008	06-09-2024	2	2	999.99	1999.98	Shipped
OD06	C019	26-09-2024	58	3	3.99	11.97	Delivered
OD07	C006	08-09-2024	23	2	6.99	13.98	Shipped
OD08	C014	23-09-2024	77	1	8.99	8.99	Delivered
OD09	C002	10-09-2024	11	1	2.49	2.49	Processing
OD10	C010	09-09-2024	5	4	5.49	21.96	Shipped
OD11	C015	07-09-2024	39	2	5.99	11.98	Shipped
OD12	C004	22-09-2024	54	1	3.99	3.99	Delivered
OD13	C009	21-09-2024	1	3	2.99	8.97	Processing
OD14	C001	25-09-2024	30	5	199.99	999.95	Delivered
OD15	C011	13-09-2024	76	1	4	4	Shipped
OD16	C020	17-09-2024	8	2	599.99	1199.98	Delivered
OD17	C017	18-09-2024	66	1	3.99	3.99	Shipped
OD18	C013	27-09-2024	24	10	1.99	19.9	Delivered
OD19	C018	02-09-2024	52	1	9.99	9.99	Processing
OD20	C016	05-09-2024	19	4	24.99	99.96	Shipped
OD21	C023	12-09-2024	73	1	7.99	7.99	Shipped
OD22	C025	20-09-2024	15	3	799.99	2399.97	Delivered
OD23	C029	30-09-2024	40	20	6.49	129.8	Processing
OD24	C030	03-09-2024	27	2	89.99	179.98	Cancelled
OD25	C024	19-09-2024	65	1	6.99	6.99	Shipped
OD26	C021	01-09-2024	35	4	3.49	13.96	Delivered
OD27	C022	16-09-2024	9	1	2.79	2.79	Shipped
OD28	C028	24-09-2024	17	5	149.99	749.95	Delivered
OD29	C026	11-09-2024	71	5	11.99	59.95	Processing

Desc Orders;

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	
product_name	text	YES		NULL	
category_id	varchar(25)	YES	MUL	NULL	
price	double	YES		NULL	
stock_quantity	int	YES		NULL	

# ANALYSIS

Find the date with the highest total sales in the month

QUERY:

```
Select order_date,  
sum(o.total_amount) as  
total_sales  
from orders  
group by order_date  
order by total_sales desc  
Limit 1;
```

OUTPUT:

Result Grid			Filter Rows:	Export:
	order_date	total_sales		
▶	20-09-2024	2489.9599999999996		





# ANALYSIS

Retrieve the customer details whose delivery status is processing or cancelled

QUERY:

```
Select concat(cu.first_name," ",cu.last_name)
as customer_name, cu.email, cu.address, cu.phone_number,
cu.payment_method, o.`status`
From customer cu
Join orders o on cu.customer_id=o.customer_id
Join product p on o.product_id=p.product_id
Join category ca on p.category_id=ca.category_id
Where o.`status`='cancelled' or o.`status`='processing';
```

## OUTPUT:

<div> Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content:  </div>						
	customer_name	email	address	phone_number	payment_method	status
▶	Liam Johnson	Liam.Johnson@example.com	909 Oak St, Springfield	(512) 555-0156	Cash	Processing
	Jane Smith	jane.smith@example.com	456 Oak St, Springfield	(303) 555-0192	Cash	Processing
	James Anderson	james.anderson@example.com	606 Poplar St, Capital City	(213) 555-0119	Credit card	Processing
	Lucas Davis	Lucas.Davis@example.com	678 Cherry St, Shelbyville	(602) 555-0314	Credit card	Processing
	Abigail Jackson	Abigail.Jackson@example.com	909 Poplar St, Ogdenville	(516) 555-0273	Cash	Processing
	Jackson Martin	Jackson.Martin@example.com	111 Ash St, North Haverbrook	(901) 555-0179	Cash	Cancelled
	James Thomas	James.Thomas@example.com	606 Birch St, Shelbyville	(925) 555-0487	Credit card	Processing
	Chloe Harris	Chloe.Harris@example.com	666 Cedar St, North Haverbrook	(323) 555-0412	Ewallet	Processing
	Michael Robinson	Michael.Robinson@example.com	223 Ash St, Shelbyville	(954) 555-0317	Cash	Processing
	Samuel Scott	Samuel.Scott@example.com	889 Birch St, Capital City	(630) 555-0210	Cash	Processing
	Hazel Hall	Hazel.Hall@example.com	890 Cedar St, Capital City	(512) 555-0392	Ewallet	Processing
	John Mitchell	John.Mitchell@example.com	323 Cherry St, Springfield	(407) 555-0341	Ewallet	Processing
	David Nguyen	David.Nguyen@example.com	123 Cherry St, Ogdenville	(406) 555-0285	Ewallet	Processing
	Noah Brown	Noah.Brown@example.com	234 Elm St, Capital City	(818) 555-0264	Ewallet	Processing
	Ellie Allen	Ellie.Allen@example.com	556 Pine St, Ogdenville	(718) 555-0462	Cash	Processing

(Note: limited results are shown due to lack of space)



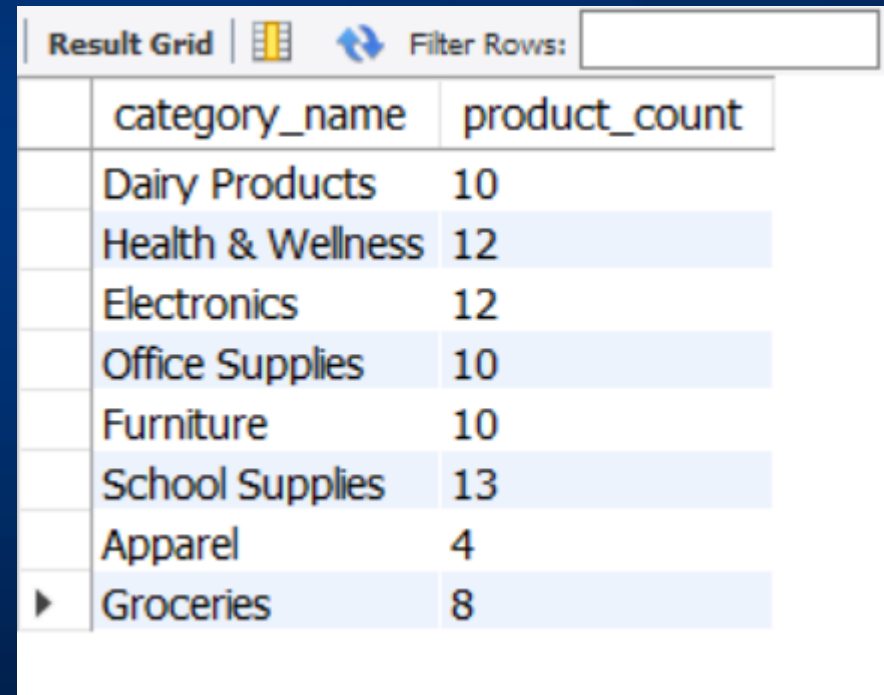
# ANALYSIS

List all categories and the number of products in each category.

QUERY:

```
Select
category.category_name,
count(product.product_id)
as product_count
From category
Left join product
On category.category_id =
product.category_id
Group by
category.category_id;
```

OUTPUT:



The screenshot shows a 'Result Grid' window with a toolbar containing a 'Filter Rows' button and a search input field. The table below displays the results of the SQL query, with columns 'category\_name' and 'product\_count'.

	category_name	product_count
	Dairy Products	10
	Health & Wellness	12
	Electronics	12
	Office Supplies	10
	Furniture	10
	School Supplies	13
	Apparel	4
▶	Groceries	8





# ANALYSIS

Retrieve the total stock for each category and the number of products in that category.

QUERY:

```
Select ca.category_id, ca.category_name,  
Count(p.product_id) as product_count,  
Sum(p.stock_quantity) as total_stock  
From category ca  
Join product p on ca.category_id = p.category_id  
Group by ca.category_id, ca.category_name  
Order by total_stock desc;
```

OUTPUT:

Result Grid   Filter Rows: <input type="text"/>					Export:  Wrap Cell C
	category_id	category_name	product_count	total_stock	
▶	C006	School Supplies	13	1915	
	C002	Health & Wellness	12	1155	
	C001	Dairy Products	10	900	
	C008	Groceries	8	810	
	C003	Electronics	12	720	
	C004	Office Supplies	10	515	
	C007	Apparel	4	325	
	C005	Furniture	10	261	

# ANALYSIS

List customers along with their frequency of order.

QUERY:

```
Select cu.customer_id,  
Concat(cu.first_name," ",cu.last_name) as customer_name,  
Count(o.order_id) as total_orders,  
If (count(o.order_id)>=2, 'frequent buyer','Regular Buyer')  
as frequency_of_purchase  
From customer cu  
Left join orders o on cu.customer_id = o.customer_id  
Group by cu.customer_id;
```

OUTPUT:

Result Grid				
		Filter Rows:		
		Export:		
		Wrap Cell Content:		
	customer_id	customer_name	total_orders	frequency_of_purchase
▶	C001	John Doe	2	frequent buyer
	C002	Jane Smith	2	frequent buyer
	C003	Michael Johnson	1	Regular Buyer
	C004	Emily Davis	2	frequent buyer
	C005	Robert Brown	2	frequent buyer
	C006	Linda Wilson	1	Regular Buyer
	C007	David Miller	2	frequent buyer
	C008	Sarah Taylor	2	frequent buyer
	C009	James Anderson	1	Regular Buyer
	C010	Patricia Thomas	1	Regular Buyer
	C011	Emma Smith	2	frequent buyer
	C012	Liam Johnson	2	frequent buyer
	C013	Olivia Williams	1	Regular Buyer
	C014	Noah Brown	2	frequent buyer
	C015	Ava Jones	2	frequent buyer
	C016	Elijah Garcia	1	Regular Buyer

(Note: limited results are shown due to lack of space)

# ANALYSIS

Retrieve the total number of products ordered and the total sales for each category

QUERY:

```
Select c.category_name,  
Count(o.product_id) as total_products_ordered,  
Round(sum(o.total_amount),0) as total_sales_amount  
From category c  
Join product p on c.category_id = p.category_id  
Join orders o on p.product_id = o.product_id  
Group by c.category_id;
```

OUTPUT:

Result Grid				Filter Rows:	Export:	Wrap Cell Content
	category_name	total_products_ordered	total_sales_amount			
▶	Dairy Products	15	557			
	Health & Wellness	15	772			
	Electronics	14	7700			
	Office Supplies	14	2257			
	Furniture	10	3948			
	School Supplies	14	1690			
	Apparel	6	166			
	Groceries	12	2319			

# ANALYSIS

Classify customers into spending tiers based on their total spending.

QUERY:

```
Select cu.customer_id, cu.first_name, cu.last_name,  
Round(sum(o.total_amount),0) as total_spending,  
Case  
    When sum(o.total_amount) >= 1000 then 'Platinum'  
    When sum(o.total_amount) between 500 and 999 then 'Gold'  
    When sum(o.total_amount) between 100 and 499 then 'Silver'  
    Else 'Bronze'  
End as spending_tier  
From customer cu  
Left join orders o on cu.customer_id = o.customer_id  
Group by cu.customer_id  
Order by total_spending desc;
```



OUTPUT:

Result Grid					
		Filter Rows:		Export:	Wrap Cell Content:
	customer_id	first_name	last_name	total_spending	spending_tier
▶	C025	Harper	Anderson	2400	Platinum
	C008	Sarah	Taylor	2364	Platinum
	C038	Henry	Ramirez	1454	Platinum
	C001	John	Doe	1400	Platinum
	C020	Mason	Martinez	1242	Platinum
	C002	Jane	Smith	902	Gold
	C005	Robert	Brown	813	Gold
	C027	Evelyn	Taylor	782	Gold
	C046	Samuel	Scott	750	Gold
	C028	Aiden	Moore	750	Gold
	C043	Ellie	Allen	708	Gold
	C011	Emma	Smith	704	Gold
	C045	Zoey	Wright	550	Gold
	C049	Nora	Hill	525	Gold
	C030	Jackson	Martin	430	Silver
	C034	Benjamin	White	410	Silver

(Note: limited results are shown due to lack of space)

# ANALYSIS

Show products and indicate if they are "Best Seller" based on quantity sold.

QUERY:

```
Select p.product_id, p.product_name,  
Sum(o.quantity) as total_quantity_sold,  
If (Sum(o.quantity)>=10,'Best Seller','Regular') as  
sales_category  
From product p  
Left join orders o on p.product_id = o.product_id  
Group by p.product_id  
Order by total_quantity_sold desc;
```

OUTPUT:

Result Grid					Filter Rows:	Export:	Wrap Cell Content:
	product_id	product_name	total_quantity_sold	sales_category			
▶	40	Nightstand	20	Best Seller			
	5	Inkjet Printer	19	Best Seller			
	27	Ice Cream	12	Best Seller			
	18	Whiteboard Markers	10	Best Seller			
	50	Bookshelf	10	Best Seller			
	45	Vitamin C Gummies	10	Best Seller			
	20	Cream Cheese	10	Best Seller			
	24	External Hard Drive	10	Best Seller			
	22	Ream of Printer Paper	9	Regular			
	75	Yogurt Parfait	8	Regular			
	74	Chicken Breast	8	Regular			
	46	Deodorant Stick	8	Regular			
	65	Salt and Pepper Shak...	7	Regular			
	13	Highlighters	7	Regular			
	23	Ballpoint Pens	7	Regular			
	30	Desk Lamp	7	Regular			

(Note: limited results are shown due to lack of space)

# ANALYSIS

## Effectiveness of Customer Payment Method

QUERY:

```
Select cu.payment_method,  
Round(sum(o.total_amount),0) as total_spent,  
Case  
    When Sum(o.total_amount) >= 10000 then 'Very Effective'  
    When Sum(o.total_amount) between 3000 and 6999 then  
'Effective'  
    Else 'Needs Improvement'  
End as effectiveness  
From customer cu  
Join orders o on cu.customer_id = o.customer_id  
Group by cu.payment_method;
```

OUTPUT:

Result Grid				Filter Rows:	Export:	Wrap C
	payment_method	total_spent	effectiveness			
▶	Ewallet	11357	Very Effective			
	Cash	5658	Effective			
	Credit card	2394	Needs Improvement			





# ANALYSIS

Comparison of the highest order total for each customer to the average order total.

QUERY:

```
Select cu.customer_id, cu.first_name, cu.last_name,  
Max(o.total_amount) as max_order,  
(Select avg(total_amount) from orders  
Where customer_id = cu.customer_id) as average_order,  
Case  
    When max(o.total_amount) > (select avg(total_amount) from  
orders where customer_id = cu.customer_id) then 'Above Average'  
    Else 'Below Average'  
End as order_comparison  
From customer cu  
Join orders o on cu.customer_id = o.customer_id  
Group by cu.customer_id, cu.first_name, cu.last_name;
```

## OUTPUT:

Result Grid     Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 						
	customer_id	first_name	last_name	max_order	average_order	order_comparison
▶	C001	John	Doe	999.95	699.97	Above Average
	C002	Jane	Smith	899.9	451.195	Above Average
	C003	Michael	Johnson	279.93	279.93	Below Average
	C004	Emily	Davis	3.99	3.39	Above Average
	C005	Robert	Brown	799.96	406.56	Above Average
	C006	Linda	Wilson	13.98	13.98	Below Average
	C007	David	Miller	49.99	44.995000000000005	Above Average
	C008	Sarah	Taylor	1999.98	1181.97	Above Average
	C009	James	Anderson	8.97	8.97	Below Average
	C010	Patricia	Thomas	21.96	21.96	Below Average
	C011	Emma	Smith	699.99	351.995	Above Average
	C012	Liam	Johnson	119.97	66.98	Above Average
	C013	Olivia	Williams	19.9	19.9	Below Average
	C014	Noah	Brown	16.98	12.985	Above Average
	C015	Ava	Jones	11.98	7.985	Above Average
	C016	Elijah	Garcia	99.96	99.96	Below Average

(Note: limited results are shown due to lack of space)



