Introduction

This project analyzes Amazon sales data using SQL to uncover trends in product performance, customer behavior, and operational efficiency. The goal is to drive better decision-making in inventory, marketing, and customer management.

Problem Statement

E-commerce platforms struggle with identifying key revenue drivers, managing stock, and improving customer satisfaction. This analysis aims to tackle those issues through data-driven insights on sales, returns, shipping, and seller performance.

Queries & Results

1. **Query the top 10 products by total sales value, including product name, total quantity sold, and total sales value.**

```
ALTER TABLE order_items
ADD total_sale FLOAT;

UPDATE order_items
SET total_sale = quantity * price_per_unit;

SELECT TOP 10
   oi.product_id,
   p.product_name,
   SUM(oi.total_sale) AS total_sale,
   COUNT(o.order_id) AS total_orders
FROM orders AS o
JOIN order_items AS oi
   ON oi.order_id = o.order_id
JOIN products AS p
   ON p.product_id = oi.product_id
GROUP BY oi.product_id, p.product_name
ORDER BY total_sale DESC;
```

	product_id	product_name	total_sale	total_orders
1	8	Apple iMac Pro	629998.739999999	120
2	7	Apple iMac 27-Inch Retina	232198.71	115
3	90	Canon EOS R5 Mirrorless Camera	222299.43	41
4	6	Apple iMac 24-Inch	189798.54	133
5	25	Apple MacBook Pro 16-inch	187499.25	65
6	40	Dell Alienware Aurora R13	177499.29	63
7	26	Apple MacBook Pro 16-inch (20...	162499.35	54
8	43	Dell XPS 17 Laptop	157499.25	68
9	216	Sony A7R IV Mirrorless Camera	155099.53	40
10	193	Canon EOS R6 Mirrorless Camera	144999.42	44

These products are likely candidates for focused promotions and restocking priority.

2. **Calculate total revenue generated by each product category and include the percentage contribution of each category to total revenue.**

```
SELECT p.category_id, c.category_name,
    ROUND(SUM(oi.total_sale), 2) AS total_sale,
    ROUND((SUM(oi.total_sale) / (SELECT SUM(total_sale) FROM order_items)) * 100, 2) AS
percentage_revenue
FROM order_items AS oi
JOIN products AS p ON p.product_id = oi.product_id
LEFT JOIN category AS c ON p.category_id = c.category_id
GROUP BY p.category_id, c.category_name
ORDER BY total_sale DESC;
```

	category_id	category_name	total_sale	percentage_revenue
1	1	electronics	11346709.55	89.73
2	6	Sports & Outdoors	457462.79	3.62
3	5	Toys & Games	354165.59	2.8
4	4	Pet Supplies	262478.77	2.08
5	2	clothing	133775.88	1.06
6	3	home & kitchen	90277.84	0.71

Invest more in categories with high percentage contributions; reevaluate low-performing ones.

3. **Average Order Value (AOV): Compute the average order value for each customer, including only customers with more than 5 orders.**

```
SELECT
```

```
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS full_name,
    ROUND(SUM(oi.total_sale) / COUNT(o.order_id), 2) AS AOV,
    COUNT(o.order_id) AS total_orders
FROM orders AS o
JOIN customers AS c
    ON c.customer_id = o.customer_id
JOIN order_items AS oi
    ON oi.order_id = o.order_id
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(o.order_id) > 5
ORDER BY total_orders DESC;
```

	customer_id	full_name	AOV	total_orders
1	625	Wendy Reed	596.37	127
2	694	Ella Reed	535.84	117
3	647	Gina Reed	506.26	115
4	697	Leo Adams	615.48	114
5	701	Olivia Barnes	655.2	114
6	731	Henry Davis	536.91	114
7	587	Alicia Green	565.12	113
8	693	Henry Harris	576.88	112
9	689	Daniel Green	454.69	112
10	700	Chloe Smith	642.16	110
11	716	Zackary Smith	537.31	109
12	699	Felix Scott	684.68	109
13	742	Patrick Rogers	532.98	109
14	670	William Smith	685.88	108
15	610	Olivia Scott	472.42	108
16	636	Olivia Green	412.83	108

Consider loyalty programs or exclusive offers for customers with high AOV and order counts.

4. **Monthly Sales Trend: Query monthly total sales over the past year, displaying current month sale and last month sale.**

```
SELECT year, month, total_sale as current_sale, LAG(total_sale, 1) OVER(ORDER BY month) as last_month_sale
FROM (
    SELECT MONTH(order_date) AS month,
```

```
        YEAR(order_date) AS year,
        ROUND(SUM(oi.total_sale), 2) AS total_sale
    FROM orders AS o
    JOIN order_items AS oi
    ON oi.order_id = o.order_id
    WHERE order_date >= DATEADD(YEAR, -1, GETDATE())
    GROUP BY MONTH(order_date), YEAR(order_date)
) AS t1
ORDER BY month, year;
```

	year	month	current_sale	last_month_sale
1	2025	2	2799.86	NULL
2	2024	3	4235.12	2799.86
3	2024	4	13839.49	4235.12
4	2024	5	25378.36	13839.49
5	2024	6	9563.75	25378.36
6	2024	7	25998.37	9563.75

Use this data for seasonal forecasting and inventory planning. Investigate months with significant drops.

5. **Customer Lifetime Value (CLTV): Calculate the total value of orders placed by each customer over their lifetime, and rank customers based on their CLTV.**

```
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS full_name,
    SUM(oi.total_sale) AS CLTV,
    DENSE_RANK() OVER(ORDER BY SUM(oi.total_sale) DESC) AS cx_ranking
FROM orders AS o
JOIN customers AS c
    ON c.customer_id = o.customer_id
JOIN order_items AS oi
    ON oi.order_id = o.order_id
GROUP BY c.customer_id, c.first_name, c.last_name;
```

	customer_id	full_name	CLTV	cx_ranking
1	554	Yvonne Reed	89029.0900000001	1
2	616	Mia Reed	82350.1800000001	2
3	711	Fred Davis	82179.1700000001	3
4	591	Quinn Davis	79205.23	4
5	748	Nathan Lee	77136.9800000001	5
6	718	Henry Reed	75825.21	6
7	625	Wendy Reed	75738.7300000001	7
8	712	Jack Johnson	75017.1500000001	8
9	669	Zackary Da...	74862.0100000001	9
10	701	Olivia Barnes	74692.81	10
11	680	Yara Davis	74691.55	11

Focus retention strategies on high CLTV customers. Tailored outreach or VIP treatment could boost loyalty.

6. **Inventory Stock Alerts: Query products with stock levels below a certain threshold (e.g., less than 10 units), including last restock date and warehouse information.**

```
SELECT
   i.inventory_id,
   p.product_name,
   i.stock AS current_stock,
   i.last_stock_date,
   i.warehouse_id
FROM inventory AS i
JOIN products AS p
   ON p.product_id = i.product_id
WHERE i.stock < 10;
```

	inventory_id	product_name	current_stock	last_stock_date	warehouse_id
1	607	Pet Water Fountain	1	2022-08-01 00:00:00.000	1
2	609	Pet Blanket	7	2022-10-30 00:00:00.000	1
3	611	Cat Food	4	2023-07-25 00:00:00.000	1
4	612	Dog Training Collar	8	2022-05-04 00:00:00.000	1
5	614	Remote Control Helicopter	5	2023-07-30 00:00:00.000	1
6	615	Magic Markers Set	2	2023-02-21 00:00:00.000	1
7	617	Giant Jenga	6	2023-08-24 00:00:00.000	1
8	618	Play Kitchen Set	8	2022-03-21 00:00:00.000	1
9	622	Hot Wheels Cars	7	2023-12-12 00:00:00.000	1
10	624	Sports Water Bottle	3	2022-09-30 00:00:00.000	1
11	627	Hiking Poles	4	2022-01-15 00:00:00.000	1

Automate reorder processes for items below threshold. Cross-reference with top-selling items.

7. **Top Performing Sellers: Find the top 5 sellers based on total sales value, including both successful and failed orders, and display their percentage of successful orders.**

```
WITH top_sellers AS (
   SELECT TOP 5 s.seller_id, s.seller_name, SUM(oi.total_sale) AS total_sale
   FROM orders AS o
   JOIN sellers AS s ON o.seller_id = s.seller_id
   JOIN order_items AS oi ON oi.order_id = o.order_id
   GROUP BY s.seller_id, s.seller_name
   ORDER BY SUM(oi.total_sale) DESC
),
sellers_report AS (
   SELECT o.seller_id, ts.seller_name, o.order_status, COUNT(*) AS order_count
   FROM orders AS o
   JOIN top_sellers AS ts ON ts.seller_id = o.seller_id
   WHERE o.order_status NOT IN ('Inprogress', 'Returned')
   GROUP BY o.seller_id, ts.seller_name, o.order_status
)
SELECT seller_id,
     seller_name,
     SUM(CASE WHEN order_status = 'Completed' THEN order_count ELSE 0 END) AS
Completed_orders,
     SUM(CASE WHEN order_status = 'Cancelled' THEN order_count ELSE 0 END) AS
Cancelled_orders,
     SUM(order_count) AS total_orders,
     CAST(SUM(CASE WHEN order_status = 'Completed' THEN order_count ELSE 0 END) AS
FLOAT) /
     CAST(SUM(order_count) AS FLOAT) * 100 AS successful_orders_percentage
FROM sellers_report
GROUP BY seller_id, seller_name
```

ORDER BY seller_id, seller_name;

	seller_id	seller_name	Completed_orders	Cancelled_orders	total_orders	successful_orders_percentage
1	1	AmazonBasics	1713	42	1755	97.6068376068376
2	2	AnkerDirect	1854	67	1921	96.5122332118688
3	3	Tech Armor	1751	36	1787	97.9854504756575
4	4	iSaddle	1804	48	1852	97.4082073434125
5	5	Ailun	1759	50	1809	97.2360420121614

Consider expanding partnerships with top sellers. Support sellers with low success rates to reduce cancellations.

8. **Most Returned Products: Query the top 10 products by the number of returns, displaying the return rate as a percentage of total units sold for each product.**

```
SELECT TOP 10
   p.product_id,
   p.product_name,
   SUM(oi.quantity) AS total_units_sold,
   SUM(CASE WHEN o.order_status = 'Returned' THEN oi.quantity ELSE 0 END) AS
total_units_returned,
   CAST(SUM(CASE WHEN o.order_status = 'Returned' THEN oi.quantity ELSE 0 END) AS FLOAT) /
   CAST(SUM(oi.quantity) AS FLOAT) * 100 AS return_percentage
FROM order_items AS oi
JOIN products AS p ON oi.product_id = p.product_id
JOIN orders AS o ON o.order_id = oi.order_id
GROUP BY p.product_id, p.product_name
ORDER BY return_percentage DESC;
```

	product_id	product_name	total_units_sold	total_units_returned	return_percentage
1	749	Pet Travel Water Bottle	3	3	100
2	743	Dog Toothpaste	8	6	75
3	745	Cat Wand Toy	7	5	71.4285714285714
4	570	Yoga Mat	10	7	70
5	301	Canon EOS 77D Camera Kit	11	6	54.5454545454545
6	355	Women's Denim Jacket	15	8	53.3333333333333
7	742	Pet Exercise Wheel	14	7	50
8	628	Running Watch	13	6	46.1538461538462
9	523	Vegetable Peeler	19	8	42.1052631578947
10	462	Cutting Board Set	12	5	41.6666666666667

Review these products for quality or description issues. Returns erode profit margins and customer trust.

9. **Top 10 Products with Highest Decreasing Revenue Ratio: Compare 2022 and 2023 revenue, return product_id, product_name, category_name, 2022 revenue, and the revenue decrease ratio.**

```
WITH last_year_sale AS (
    SELECT
        p.product_id,
        p.product_name,
        SUM(oi.total_sale) AS revenue_2022
    FROM orders AS o
    JOIN order_items AS oi ON o.order_id = oi.order_id
    JOIN products AS p ON p.product_id = oi.product_id
    WHERE YEAR(o.order_date) = 2022
    GROUP BY p.product_id, p.product_name
),

current_year_sale AS (
    SELECT
        p.product_id,
        p.product_name,
        SUM(oi.total_sale) AS revenue_2023
    FROM orders AS o
    JOIN order_items AS oi ON o.order_id = oi.order_id
    JOIN products AS p ON p.product_id = oi.product_id
    WHERE YEAR(o.order_date) = 2023
    GROUP BY p.product_id, p.product_name
)

SELECT TOP 10
    ls.product_id,
```

```
    ls.product_name,
    ls.revenue_2022 AS last_year_revenue,
    cs.revenue_2023 AS current_year_revenue,
    ROUND(ls.revenue_2022 - cs.revenue_2023, 2) AS rev_diff,
    ROUND(CAST(cs.revenue_2023 - ls.revenue_2022 AS FLOAT) / CAST(ls.revenue_2022 AS
FLOAT) * 100, 2) AS revenue_dec_ratio
FROM last_year_sale AS ls
JOIN current_year_sale AS cs
    ON ls.product_id = cs.product_id
WHERE
    ls.revenue_2022 > cs.revenue_2023
ORDER BY revenue_dec_ratio DESC;
```

	product_id	product_name	last_year_revenue	current_year_revenue	rev_diff	revenue_dec_ratio
1	712	Kidâ€™s Swing Set	5839.27	5839.27	0	0
2	710	Ride-On Car	6149.59	6149.59	0	0
3	680	Play Tent	1474.41	1449.42	24.99	-1.69
4	3	Apple AirPods Pro	9499.61999999999	9249.63	249.99	-2.63
5	8	Apple iMac Pro	174999.65	169999.66	4999.99	-2.86
6	38	Apple Watch Ultra	22399.72	21599.73	799.99	-3.57
7	713	Skateboard Deck	2679.33	2559.36	119.97	-4.48
8	16	Apple iPhone 12 Mini	13299.81	12599.82	699.99	-5.26
9	88	Sony WH-1000XM4 Wireless Headphones	5949.83	5599.84	349.99	-5.88
10	125	Sony WH-1000XM5 Wireless Headphones	6799.83	6399.84	399.99	-5.88

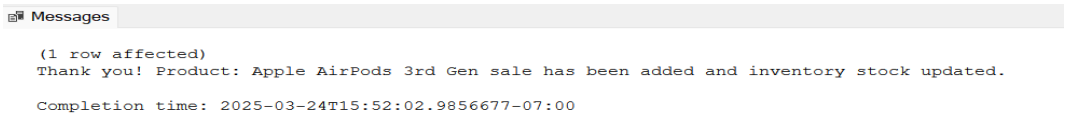
Consider removing or discounting these products. Investigate market trends or customer feedback behind decline.

10. **Store Procedure: Create a procedure to update stock in the inventory table after a product is sold.**

```
SELECT * FROM inventory
WHERE product_id = 1; -- airpod 3rd gen 55 stock
CREATE PROCEDURE add_sales
(
    @p_order_id INT,
    @p_customer_id INT,
    @p_seller_id INT,
    @p_order_item_id INT,
```

```
    @p_product_id INT,
    @p_quantity INT
)
AS
BEGIN
    -- Declare all variables
    DECLARE @v_count INT;
    DECLARE @v_price FLOAT;
    DECLARE @v_product VARCHAR(50);

    -- Fetching product name and price based on product ID
    SELECT
        @v_price = price,
        @v_product = product_name
    FROM products
    WHERE product_id = @p_product_id;

    -- Checking stock and product availability in inventory
    SELECT
        @v_count = COUNT(*)
    FROM inventory
    WHERE
        product_id = @p_product_id
        AND
        stock >= @p_quantity;
    IF @v_count > 0
    BEGIN
        -- Add into orders and order_items tables
        -- Insert into orders
        INSERT INTO orders(order_id, order_date, customer_id, seller_id)
        VALUES
        (@p_order_id, GETDATE(), @p_customer_id, @p_seller_id);
```

```
    -- Add into order_items
    INSERT INTO order_items(order_item_id, order_id, product_id, quantity, price_per_unit,
total_sale)
    VALUES
    (@p_order_item_id, @p_order_id, @p_product_id, @p_quantity, @v_price, @v_price *
@p_quantity);

    -- Update inventory
    UPDATE inventory
    SET stock = stock - @p_quantity
    WHERE product_id = @p_product_id;

    -- Print the message
    PRINT 'Thank you! Product: ' + @v_product + ' sale has been added and inventory stock
updated.';
  END
  ELSE
  BEGIN
    -- Print the message for unavailable product
    PRINT 'Thank you for your info, the product: ' + @v_product + ' is not available.';
  END
END;
GO

-- Example call to the procedure
EXEC add_sales 25006, 2, 5, 25004, 1, 14;
```



Integrate with front-end systems to maintain real-time stock accuracy and avoid overselling.