INTRODUCTION-

Welcome to the Amazon Sales Analysis project! Here, we delve into Amazon's sales data to extract insights and trends vital for optimizing sales strategies, understanding customer behaviour, and improving business operations. Through advanced SQL techniques, we'll analyze data to uncover correlations, identify emerging trends, and provide predictive insights. Our focus areas encompass data analysis, sales strategies, customer behaviour, SQL techniques, optimization, insights, trends, and business operations, aiming to empower stakeholders with actionable intelligence for informed decision-making and sustained growth.

DATASET OVERVIEW-

The dataset used in this project consists of Approximately 10,000 rows of data, representing Amazon sales transactions. Along with the sales data, the dataset includes information about customers, products, orders, returns and sellers. Before analysis, the dataset underwent preprocessing to handle missing values and ensure data quality, a crucial step in data analysis workflows. This preprocessing stage ensures the integrity and reliability of our findings, enabling us to draw accurate insights and make informed decisions based on the data.

AMAZON KPIs

1. Total revenue =

```sql
--1. Total_Revenue

SELECT
        ROUND(SUM(sale):: numeric, 2)
        AS total_revenue
FROM orders;
```

Data Output

	total_revenue numeric
1	2296140.50

2. Average order value =

```sql
--02. Average Order Value
SELECT
        ROUND((SUM(sale)/ SUM(quantity)) :: numeric, 2)
        AS avg_order_value
FROM orders;
```

Data Output

	avg_order_value numeric
1	60.64

3. Total orders =

```sql
--03. Total Orders

SELECT
        SUM(quantity) AS total_orders
FROM orders;
```

Data Output

	total_orders bigint
1	37862

4. Average qty per order =

```sql
--04. Average Qty Per Order

SELECT
        SUM(quantity)/ COUNT(DISTINCT(order_id))
        AS avg_qty_per_order
FROM orders;
```

Data Output

	avg_qty_per_order bigint
1	3

5. Average Price Per Unit =

```sql
--05. Average Price Per Unit

SELECT
        ROUND(AVG(price_per_unit) :: numeric , 2)
        AS avg_price_per_unit
FROM orders;
```

Data Output

	avg_price_per_unit numeric
1	60.92

ANALYSIS QUESTIONS RESOLVED-

During the analysis, the following key questions were addressed using SQL queries and data analysis techniques:

1. Identify the Most Active and Most Idle Month.

```sql
--The Most Active Month-

SELECT
       TO_CHAR(order_date, 'Month') AS "Month",
       SUM(quantity) AS total_orders
FROM orders
GROUP BY "Month"
ORDER BY total_orders DESC
LIMIT 1;
```

Data Output

	Month text	total_orders bigint
1	January	3736

```sql
--The Most Idle Month-

SELECT
       TO_CHAR(order_date, 'Month') AS "Month",
       SUM(quantity) AS total_orders
FROM orders
GROUP BY "Month"
ORDER BY total_orders ASC
LIMIT 1;
```

Data Output

	Month text	total_orders bigint
1	June	3003

2. Find out the month wise orders.

```sql
-- Orders By Month

SELECT
       TO_CHAR(order_date, 'Month') AS "Month",
       SUM(quantity) AS total_orders
FROM orders
GROUP BY "Month"
ORDER BY total_orders DESC;
```

Data Output

	Month text	total_orders bigint
1	January	3736
2	July	3194
3	December	3175
4	August	3167
5	October	3137
6	February	3121
7	May	3102
8	April	3095
9	November	3076
10	September	3042
11	March	3014
12	June	3003

3. Find out the orders by states.

```sql
--Orders By State

SELECT
       state,
       SUM(quantity) AS total_orders
       FROM orders
WHERE state IS NOT NULL
GROUP BY state
ORDER BY total_orders DESC;
```

Data Output

	state character varying (30)	total_orders bigint
1	Andhra Pradesh	1552
2	Tamil Nadu	1548
3	Maharashtra	1533
4	Himachal Pradesh	1488
5	Manipur	1468
6	West Bengal	1467
7	Tripura	1426
8	Kerala	1424
9	Jharkhand	1417
10	Rajasthan	1411
11	Arunachal Pradesh	1352
12	Gujarat	1344
13	Meghalaya	1339
14	Sikkim	1332
15	Madhya Pradesh	1322
16	Mizoram	1317
17	Nagaland	1309
18	Uttarakhand	1298
19	Assam	1284
20	Bihar	1283
21	Chhattisgarh	1279
22	Haryana	1265
23	Odisha	1261
24	Telangana	1260
25	Uttar Pradesh	1247
26	Karnataka	1228
27	Punjab	1222
28	Goa	1185

4. Find percentage of sales according to category.

```sql
--%age by category

SELECT
        category,
        ROUND((SUM(sale) * 100/ (SELECT SUM(sale) FROM orders)) :: numeric, 2)
        AS revenue
        FROM orders
WHERE category IS NOT NULL
GROUP BY category
ORDER BY revenue DESC;
```

Data Output

	category character varying (55)	revenue numeric
1	Technology	36.41
2	Furniture	32.32
3	Office Supplies	31.24

5. Find percentage of sales according to sub category.

```sql
--%age by sub_category

SELECT
        sub_category,
        ROUND((SUM(sale) * 100/ (SELECT SUM(sale) FROM orders)) :: numeric, 2) AS revenue
        FROM orders
WHERE sub_category IS NOT NULL
GROUP BY sub_category
ORDER BY revenue DESC;
```

Data Output

	sub_category character varying (155)	revenue numeric
1	Phones	14.37
2	Chairs	14.32
3	Storage	9.75
4	Tables	9.01
5	Binders	8.86
6	Machines	8.24
7	Accessories	7.29
8	Copiers	6.51
9	Bookcases	4.99
10	Appliances	4.61
11	Furnishings	3.99
12	Paper	3.42
13	Supplies	2.03
14	Art	1.18
15	Envelopes	0.72
16	Labels	0.54
17	Fasteners	0.13

6. Find Out Best and Worst Seller Product.

```sql
--Best selling product

SELECT
        p.product_id,
        p.product_name,
        ROUND(SUM(sale) :: numeric, 2) AS revenue
FROM orders AS o
JOIN products AS p
        ON p.product_id = o.product_id
GROUP BY
        p.product_id,
        p.product_name
ORDER BY revenue DESC
LIMIT 5;
```

Data Output

	product_id character varying (10)	product_name character varying (255)	revenue numeric
1	P1696	Canon imageCLASS 2200 ...	61599.83
2	P1031	Fellowes PB500 Electric P	27453.38
3	P191	Cisco TelePresence System	22638.48
4	P600	HON 5400 Series Task Chai	21870.57
5	P718	GBC DocuBind TL300 Electr	19823.48

```sql
--Worst Selling product

SELECT
        p.product_id,
        p.product_name,
        ROUND(SUM(sale) :: numeric, 2) AS revenue
FROM orders AS o
JOIN products AS p
        ON p.product_id = o.product_id
GROUP BY
        p.product_id,
        p.product_name
ORDER BY revenue ASC
LIMIT 5;
```

Data Output

	product_id character varying (10)	product_name character varying (255)	revenue numeric
1	P1817	Eureka Disposable Bags...	1.62
2	P1809	Avery 5	5.76
3	P1586	Xerox 20	6.48
4	P1689	Grip Seal Envelopes	7.07
5	P1717	Avery Hi-Liter Pen Style	7.70

7. Find out the top 5 customers who made the highest profits.

```sql
--Find Out the top 5 customers who made the highest profits

SELECT
    c.customer_id,
    c.customer_name,
    profit.total_profit
FROM customers AS c
    JOIN
        (SELECT
        o.customer_id,
        ROUND((SUM(sale - (p.cogs * o.quantity))):: numeric, 2)
        AS total_profit
FROM orders
        AS o
JOIN products
        AS p
ON p.product_id = o.product_id
GROUP BY o.customer_id)
        AS profit

ON c.customer_id = profit.customer_id
ORDER BY
        profit.total_profit DESC
LIMIT 5;
```

Data Output

	customer_id character varying (10)	customer_name character varying (55)	total_profit numeric
1	CS103	Bhavya	10474.71
2	CS188	Shalini	8309.22
3	CS218	Vaibhav	7745.36
4	CS106	Deepti	7090.34
5	CS146	Aditi	6540.18

8. Find out the average quantity ordered per category.

```sql
--Find out the average quantity order.

SELECT
        category,
        ROUND(AVG(quantity)) AS avg_qty
FROM orders
WHERE category IS NOT NULL
GROUP BY category;
```

Data Output

	category character varying (55)	avg_qty numeric
1	Furniture	4
2	Office Supplies	4
3	Technology	4

9. Determine the top 5 products whose revenue has decreased compared to the previous year.

```sql
--Determine the top 5 products whose revenue has decreased compared to the previous year.

WITH revenue_comparison AS (
    SELECT
        o.product_id,
        EXTRACT(YEAR FROM o.order_date) AS order_year,
        ROUND(SUM(o.sale)::numeric, 2) AS total_revenue
    FROM
        orders AS o
    GROUP BY
        o.product_id,
        EXTRACT(YEAR FROM o.order_date)
)
SELECT
    p.product_id,
    p.product_name,
    previous_year.total_revenue AS previous_year_revenue,
    current_year.total_revenue AS current_year_revenue

FROM
    revenue_comparison AS current_year
JOIN
    revenue_comparison AS previous_year
ON
    current_year.product_id = previous_year.product_id
    AND current_year.order_year = previous_year.order_year + 1
JOIN
    products AS p
ON
    p.product_id = current_year.product_id
WHERE
    current_year.total_revenue < previous_year.total_revenue
ORDER BY
    (previous_year.total_revenue - current_year.total_revenue) DESC
LIMIT 5;
```

Data Output

	product_id character varying (10)	product_name character varying (255)	previous_year_revenue numeric	current_year_revenue numeric
1	P1696	Canon imageCLASS 2200 ...	51099.86	10499.97
2	P1392	GBC Ibimaster 500 Manual	12708.37	760.98
3	P782	Lexmark MX611dhe Monoc...	11219.93	3059.98
4	P1031	Fellowes PB500 Electric P	11693.10	4575.57
5	P1253	Canon PC1060 Personal Las	8819.87	2799.96

10. Identify the highest profitable sub-category.

```sql
--Identify the highest profitable sub-category.

SELECT
        sub_category,
        ROUND(SUM(sale) :: numeric, 2) AS total_revenue
FROM orders
WHERE sub_category IS NOT NULL
GROUP BY sub_category
ORDER BY total_revenue DESC
LIMIT 5;
```

Data Output

	sub_category character varying (155)	total_revenue numeric
1	Phones	329916.17
2	Chairs	328842.70
3	Storage	223843.59
4	Tables	206965.68
5	Binders	203325.37

11. Calculate the profit margin percentage for each sale (Profit divided by Sales).

```sql
--Calculate the profit margin percentage for each sale (Profit divided by Sales).

SELECT
    p.product_id,
    p.price,
    p.cogs,
    o.sale,
    ROUND((o.sale-(p.cogs * o.quantity)) * 100/ o.sale) AS profit
FROM
    products AS p
JOIN
    orders AS o
ON p.product_id = o.product_id
ORDER BY profit DESC;
```

Data Output

	product_id character varying (10)	price double precision	cogs double precision	sale double precision	profit double precision
1	P22	5.68	3.01	247.84	90
2	P22	5.68	3.01	61.96	90
3	P22	5.68	3.01	74.35	88
4	P22	5.68	3.01	49.57	88
5	P79	2.47	1.31	31.56	83

12. Calculate the percentage contribution of each sub-category.

```sql
--Calculate the percentage contribution of each sub_category.

SELECT
    sub_category,
    ROUND(SUM(sale) :: numeric, 2) AS revenue,
    ROUND((SUM(sale) *100 / (SELECT SUM(sale) FROM orders)) :: numeric, 2)
    AS contribution
FROM
    orders
WHERE
    sub_category IS NOT NULL
GROUP BY
    sub_category
ORDER BY
    contribution DESC;
```

Data Output

	sub_category character varying (155)	revenue numeric	contribution numeric
1	Phones	329916.17	14.37
2	Chairs	328842.70	14.32
3	Storage	223843.59	9.75
4	Tables	206965.68	9.01
5	Binders	203325.37	8.86
6	Machines	189238.68	8.24
7	Accessories	167380.31	7.29
8	Copiers	149528.01	6.51
9	Bookcases	114556.91	4.99
10	Appliances	105943.66	4.61
11	Furnishings	91636.66	3.99
12	Paper	78479.24	3.42
13	Supplies	46673.52	2.03
14	Art	27118.80	1.18
15	Envelopes	16476.38	0.72
16	Labels	12486.30	0.54
17	Fasteners	3021.23	0.13

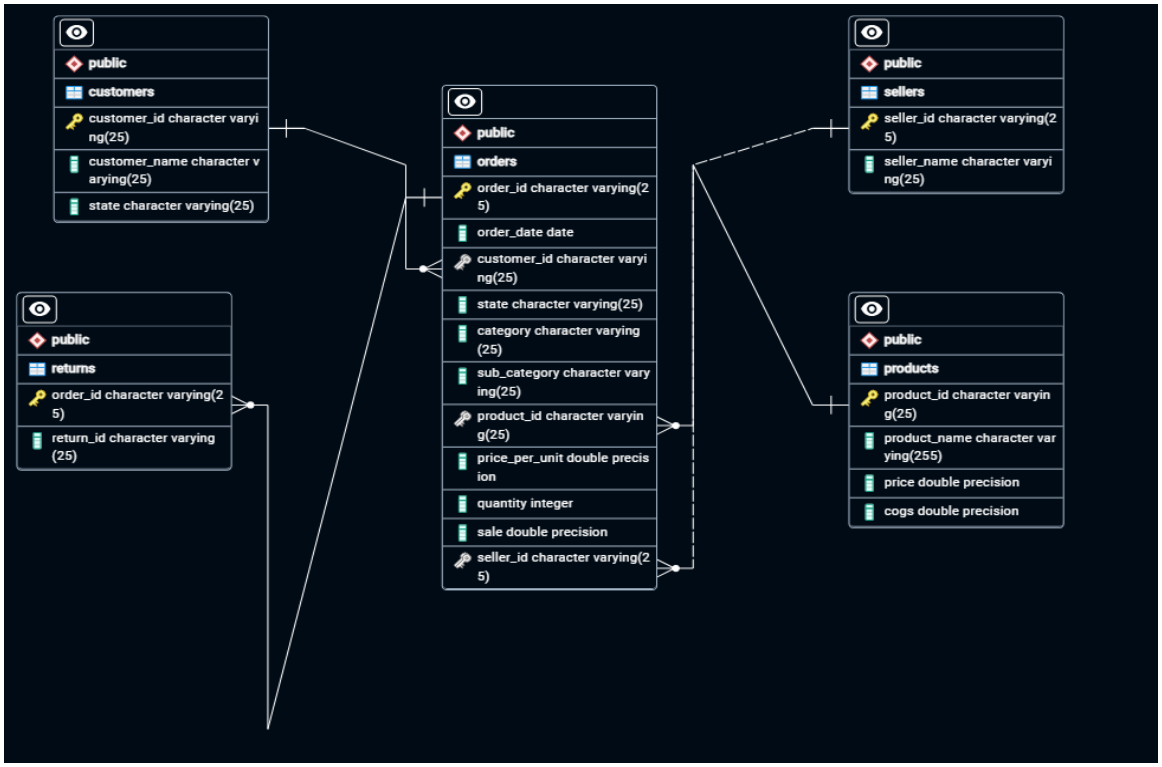
13. Identify the top 2 categories that have received maximum returns and their return percentage.

```sql
SELECT
    o.category,
    COUNT(r.return_id) AS return_count,
    ROUND(((COUNT(r.return_id) / CAST((SELECT COUNT(return_id) FROM return) AS FLOAT)) * 100) ::numeric,2)
    AS return_percentage
FROM
    orders AS o
JOIN
    return AS r ON o.order_id = r.order_id
GROUP BY
    o.category
ORDER BY
    return_count DESC
LIMIT 2;
```

Data Output

	category character varying (75)	return_count bigint	return_percentage numeric
1	Office Supplies	177	59.60
2	Furniture	64	21.55

Entity Relationship Diagram (ERD)



An Entity-Relationship Diagram (ERD) has been created to visualize the relationships between the tables in the dataset. This diagram provides a clear understanding of the data structure and helps in identifying key entities and their attributes.

CONCLUSION –

In conclusion, our project endeavours to offer meaningful insights into Amazon's sales patterns, customer inclinations, and various factors shaping e-commerce dynamics. By meticulously scrutinizing the dataset and delving into pivotal inquiries, our aim is to empower stakeholders with actionable intelligence, enabling them to make informed decisions and refine their sales strategies for optimal outcomes.