DAY 4 – ADVANCED SQL ANALYSIS (CASE & SUBQUERIES)

Dataset: Superstore Sales Dataset
Tool Used: MySQL Workbench
Analysis Type: Advanced SQL & Conditional Logic

1.Objective of Day 4

The objective of Day 4 was to apply advanced SQL concepts such as **CASE statements**, **subqueries**, and **ranking logic** to analyze profitability and identify top- and bottom-performing products.

2.Why Advanced SQL Is Important

Advanced SQL is important because it allows analysts to:

- Apply business rules directly inside queries
- Categorize data based on conditions
- Compare performance against benchmarks
- Identify high-impact and low-impact entities

3.SQL Analysis Performed

3.1 Order Profit Classification Using CASE

```
SELECT
  order_id,
  sales,
  profit,
  CASE
    WHEN profit > 0 THEN 'Profitable'
    WHEN profit = 0 THEN 'Break-even'
    ELSE 'Loss'
  END AS order_status
FROM orders;
```

order_id	sales	profit	order_status
CA-2016-152156	261.96	41.91	Profitable
CA-2016-152156	731.94	219.58	Profitable
CA-2016-138688	14.62	6.87	Profitable
US-2015-108966	957.58	-383.03	Loss
US-2015-108966	22.37	2.52	Profitable
CA-2014-115812	48.86	14.17	Profitable
CA-2014-115812	7.28	1.97	Profitable
CA-2014-115812	907.15	90.72	Profitable
CA-2014-115812	18.50	5.78	Profitable
CA-2014-115812	114.90	34.47	Profitable
CA-2014-115812	1706.18	85.31	Profitable
CA-2014-115812	911.42	68.36	Profitable
CA-2017-114412	15.55	5.44	Profitable
CA-2016-161389	407.98	132.59	Profitable
US-2015-118983	68.81	-123.86	Loss
US-2015-118983	2.54	-3.82	Loss
CA-2014-105893	665.88	13.32	Profitable
CA-2014-167164	55.50	9.99	Profitable
CA-2014-143336	8.56	2.48	Profitable
CA-2014-143336	213.48	16.01	Profitable
CA-2014-143336	22.72	7.38	Profitable
CA-2016-137330	19.46	5.06	Profitable

3.2 Count of Profitable vs Loss Orders

```
SELECT
  CASE
    WHEN profit > 0 THEN 'Profitable'
    ELSE 'Loss'
  END AS order_type,
  COUNT(*) AS total_orders
FROM orders
GROUP BY order_type;
```

order_type	total_orders
Profitable	8058
Loss	1936

3.3 Products Performing Above Average Profit (Subquery)

```
SELECT
  sub_category,
  ROUND(SUM(profit),2) AS total_profit
FROM orders
GROUP BY sub_category
HAVING SUM(profit) >
(
  SELECT AVG(profit)
  FROM orders
)
ORDER BY total_profit DESC;
```

sub_category	total_profit
Chairs	26590.15
Labels	5546.18
Storage	21279.05
Furnishings	13059.25
Art	6527.96
Phones	44516.25
Binders	30221.64
Appliances	18138.07
Paper	34053.34
Accessories	41936.78
Envelopes	6964.10
Fasteners	949.53
Machines	3384.73
Copiers	55617.90

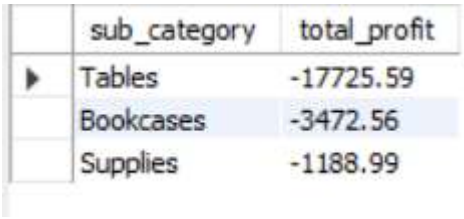
3.4 Top 3 Products by Profit

```
SELECT
  sub_category,
  ROUND(SUM(profit),2) AS total_profit
FROM orders
GROUP BY sub_category
ORDER BY total_profit DESC
LIMIT 3;
```

sub_category	total_profit
Copiers	55617.90
Phones	44516.25
Accessories	41936.78

3.5 Bottom 3 Products by Profit

```
SELECT
  sub_category,
  ROUND(SUM(profit),2) AS total_profit
FROM orders
GROUP BY sub_category
ORDER BY total_profit ASC
LIMIT 3;
```



	sub_category	total_profit
▶	Tables	-17725.59
	Bookcases	-3472.56
	Supplies	-1188.99

Key Insights from Day 4

- CASE statements enable powerful conditional analysis.
- Subqueries help compare performance against overall benchmarks.
- A small set of products drives most of the profit.
- Loss-making products can be clearly identified using SQL.

Business Recommendations

Based on Day 4 analysis:

- Focus on high-profit products for growth.
- Review discount and cost structures for loss-making products.
- Use CASE-based classification for profitability reporting.
- Optimize or discontinue consistently unprofitable products.

Key Learnings from Day 4

- Advanced SQL increases analytical depth.
- CASE and subqueries are essential for real-world data analysis.
- SQL can directly support business decision-making.
- Ranking and filtering improve insight quality.