CUSTOMER AND ORDER MANAGEMENT ANALYSIS

BY
INDIRA PRIYA DHARSHINI JYOTHI

1. List all customers.

```
SELECT * FROM customers;
```

Result Table

Result Grid | Filter Rows: | Export:

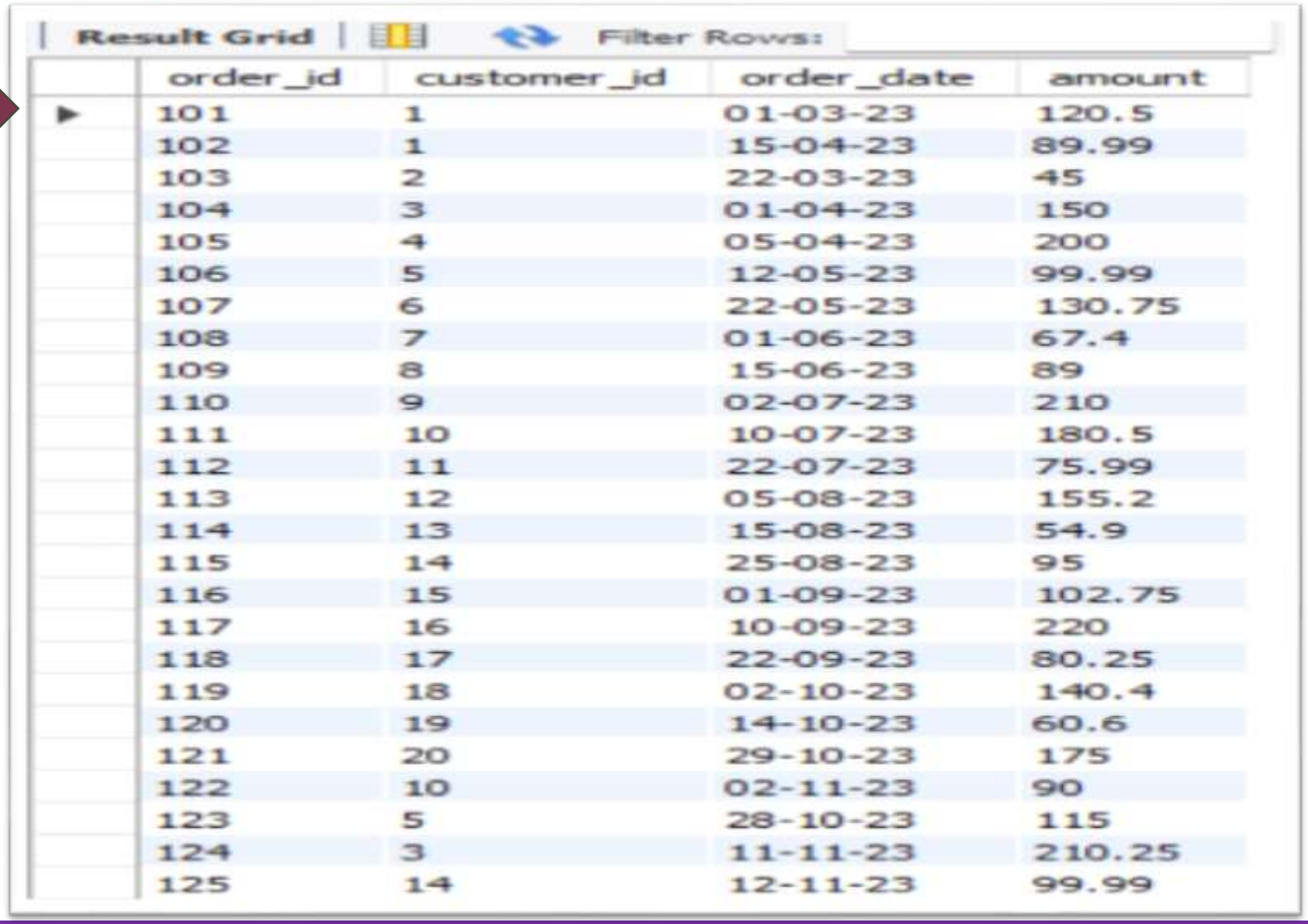
customer_id	customer_name	city	signup_date
1	Alice Brown	New York	12-01-23
2	David Smith	Chicago	05-02-23
3	Maria Lopez	Houston	20-02-23
4	John Carter	Miami	15-03-23
5	Sarah Miller	New York	01-04-23
6	Michael Green	Seattle	10-04-23
7	Emma Wilson	Boston	01-05-23
8	Oliver Jones	Chicago	12-05-23
9	Sophia Davis	Miami	02-06-23
10	James Taylor	Houston	15-06-23
11	Isabella Martin	Seattle	01-07-23
12	Lucas Lee	New York	10-07-23
13	Mia Harris	Boston	01-08-23
14	Ethan Clark	Houston	11-08-23
15	Ava Lewis	Chicago	20-08-23
16	Noah Walker	Miami	01-09-23
17	Liam Hall	Seattle	12-09-23
18	Charlotte King	New York	01-10-23
19	Amelia Scott	Boston	12-10-23
20	Benjamin Young	Houston	25-10-23

- The customer table shows a well-distributed signup pattern across months.
- Multiple customers joined within short intervals, showing steady onboarding.
- Customer diversity supports balanced analytics across the dataset.

2. List all orders.

```
SELECT * FROM orders;
```

Result Table



Result Grid | Filter Rows:

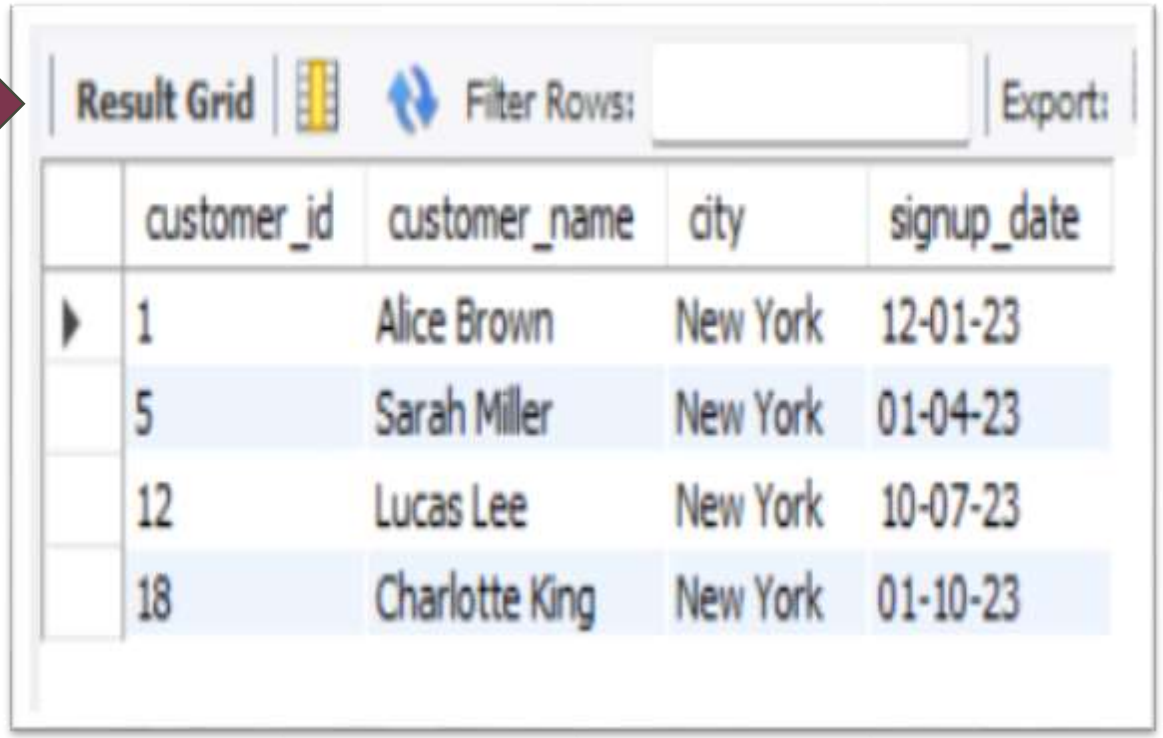
order_id	customer_id	order_date	amount
101	1	01-03-23	120.5
102	1	15-04-23	89.99
103	2	22-03-23	45
104	3	01-04-23	150
105	4	05-04-23	200
106	5	12-05-23	99.99
107	6	22-05-23	130.75
108	7	01-06-23	67.4
109	8	15-06-23	89
110	9	02-07-23	210
111	10	10-07-23	180.5
112	11	22-07-23	75.99
113	12	05-08-23	155.2
114	13	15-08-23	54.9
115	14	25-08-23	95
116	15	01-09-23	102.75
117	16	10-09-23	220
118	17	22-09-23	80.25
119	18	02-10-23	140.4
120	19	14-10-23	60.6
121	20	29-10-23	175
122	10	02-11-23	90
123	5	28-10-23	115
124	3	11-11-23	210.25
125	14	12-11-23	99.99

- Orders vary significantly in amount, indicating mixed spending behaviors.
- Order dates span across different months, showing repeated business activity.
- Some customers contribute multiple orders, while others contribute only one.

3. Find customers from New York.

```
SELECT * FROM customers WHERE city='New York';
```

Result Table



Result Grid | Filter Rows: | Export:

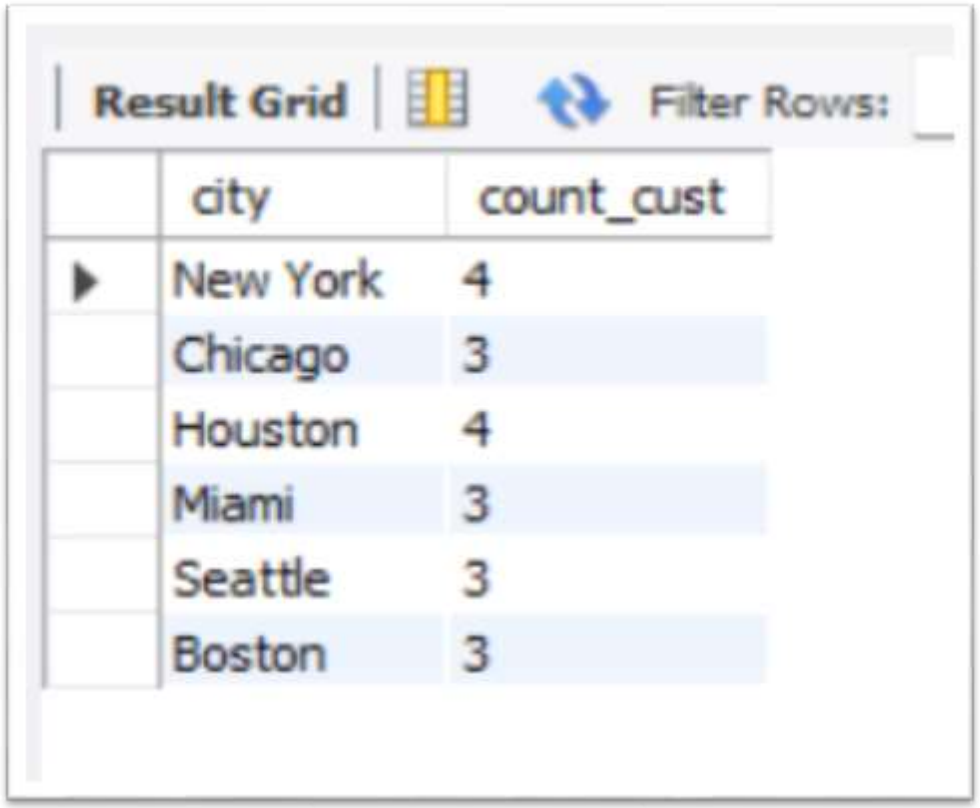
customer_id	customer_name	city	signup_date
1	Alice Brown	New York	12-01-23
5	Sarah Miller	New York	01-04-23
12	Lucas Lee	New York	10-07-23
18	Charlotte King	New York	01-10-23

- It shows whether those customers actively made purchases or not.
- Comparing with the orders table reveals their spending contribution.

4. Count customers per city.

```
SELECT city,count(customer_id) AS count_cust FROM customers GROUP BY city;
```

Result Table



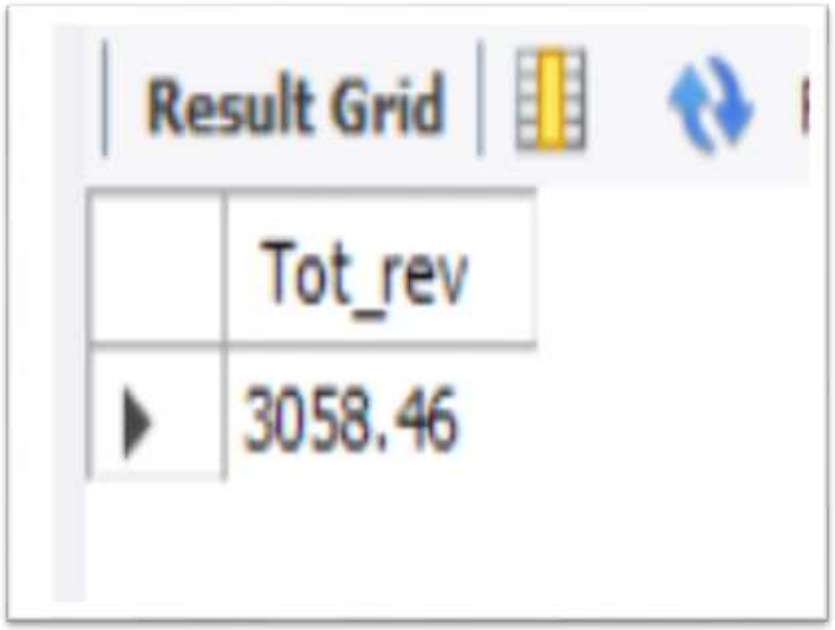
city	count_cust
New York	4
Chicago	3
Houston	4
Miami	3
Seattle	3
Boston	3

- Higher customer count often correlates with higher total orders.
- This distribution helps understand how customer base affects revenue patterns.

5. Total revenue generated.

```
SELECT sum(amount) AS Tot_rev FROM orders;
```

Result Table

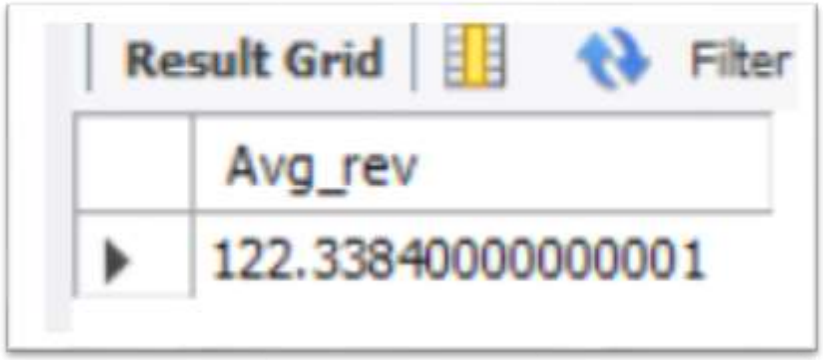


- High-value orders significantly boost overall revenue.
- The mix of small and large orders creates a balanced revenue curve.

6. Average order value.

```
SELECT AVG(amount) AS Avg_rev FROM orders;
```

Result Table



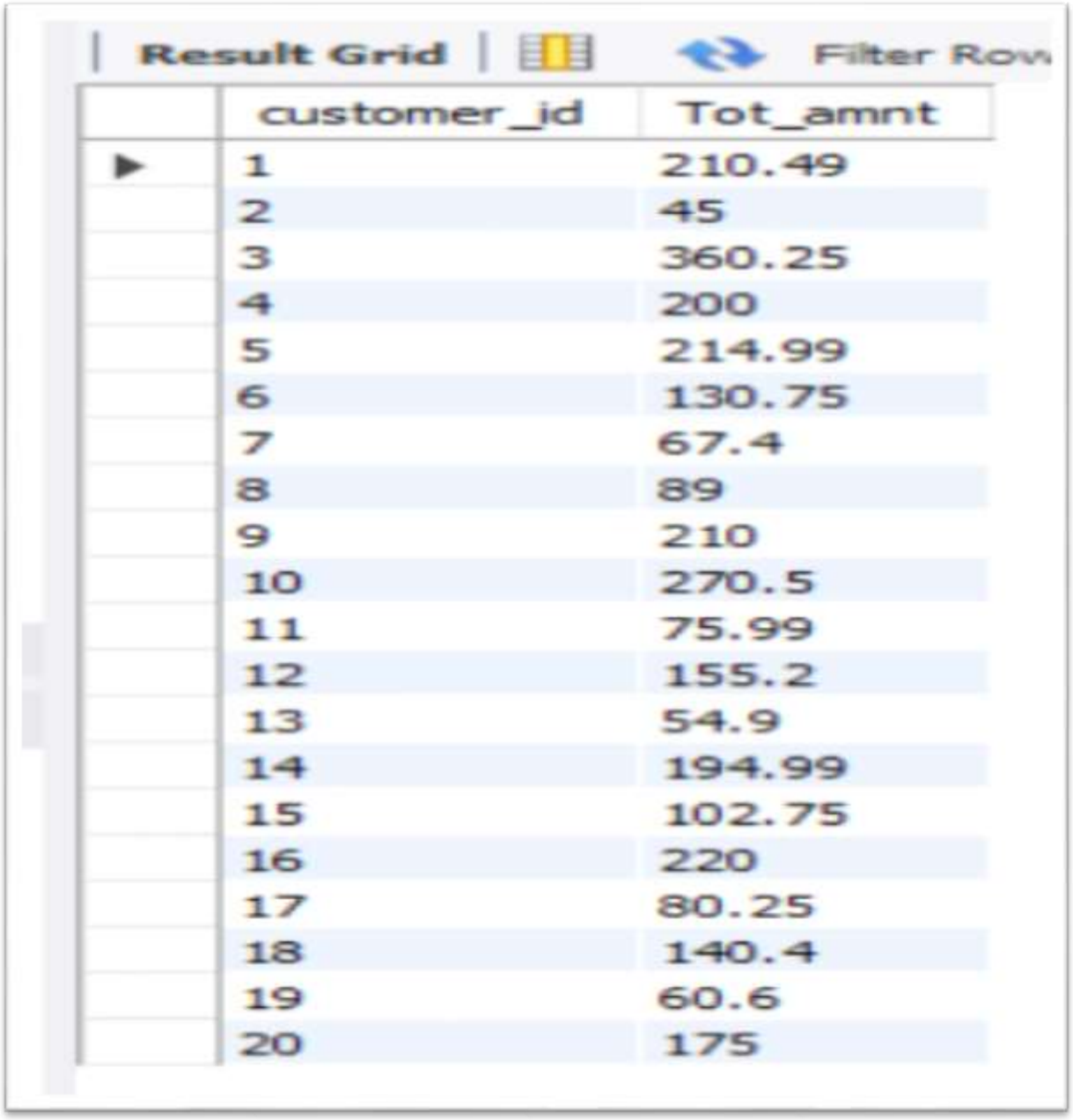
- Average order value reflects both frequent small purchases and occasional large ones.
- Helps compare customer behavior between light spenders and premium spenders.

7. Total spending per customer.

```sql
SELECT
        customer_id,
        sum(amount) AS Tot_amnt
        FROM orders
        GROUP BY customer_id;
```

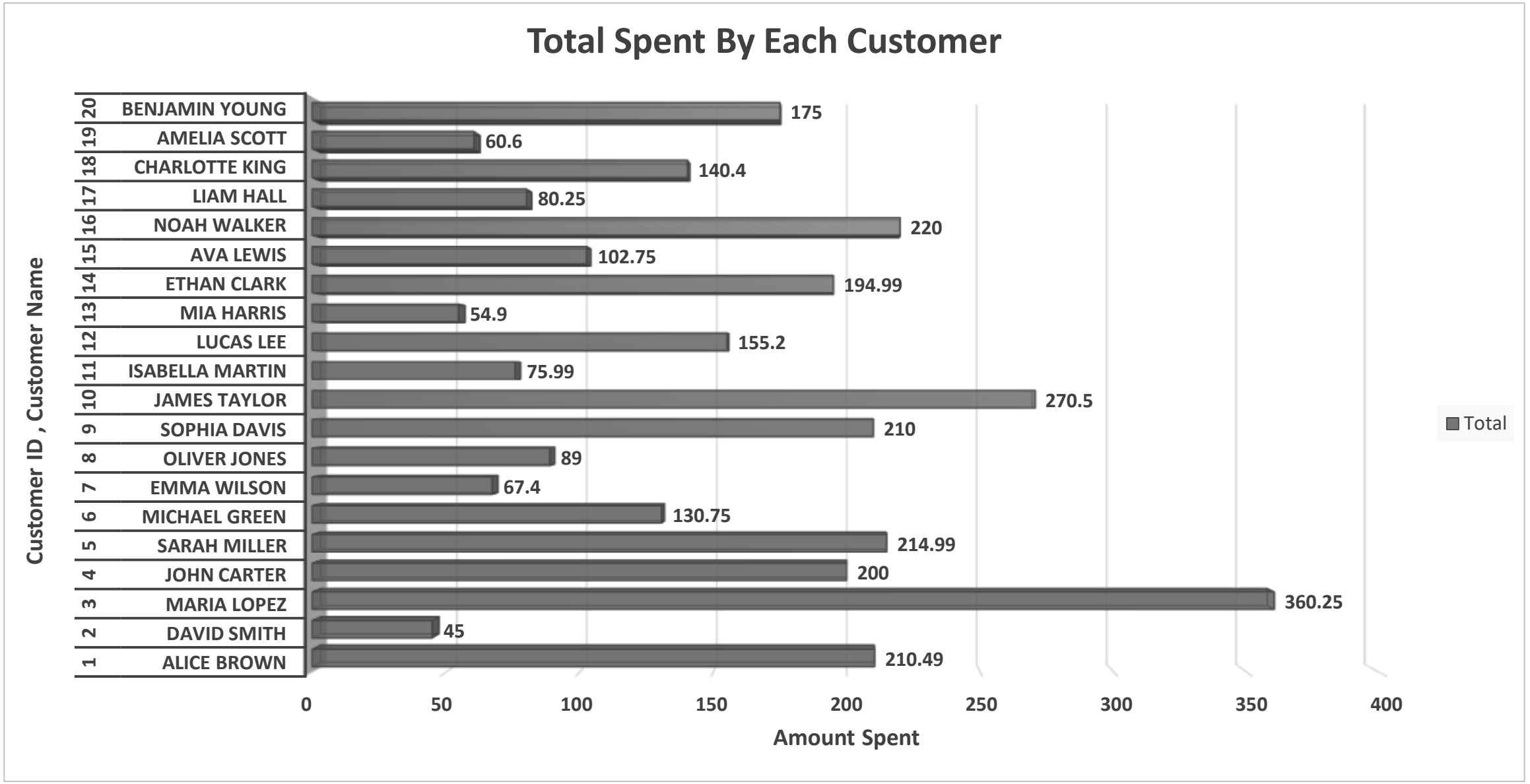
- This comparison clearly separates high-value and low-value customer groups with repeated orders and high contribution ,less order and least contribution to the revenue.

Result Table



customer_id	Tot_amnt
1	210.49
2	45
3	360.25
4	200
5	214.99
6	130.75
7	67.4
8	89
9	210
10	270.5
11	75.99
12	155.2
13	54.9
14	194.99
15	102.75
16	220
17	80.25
18	140.4
19	60.6
20	175

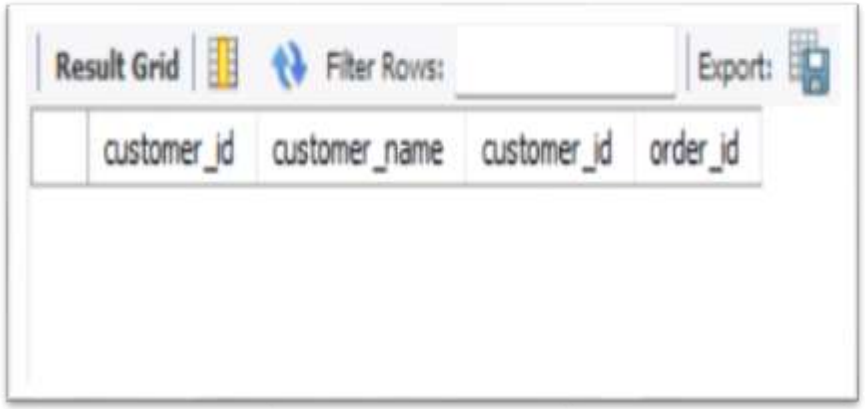
Graphical Representation



8. Customers with no orders.

```
SELECT
      c.customer_id,
      c.customer_name,
      o.customer_id,
      o.order_id FROM customers c
      LEFT JOIN orders o
      ON c.customer_id=o.customer_id
      WHERE o.customer_id IS NULL;
```

Result Table

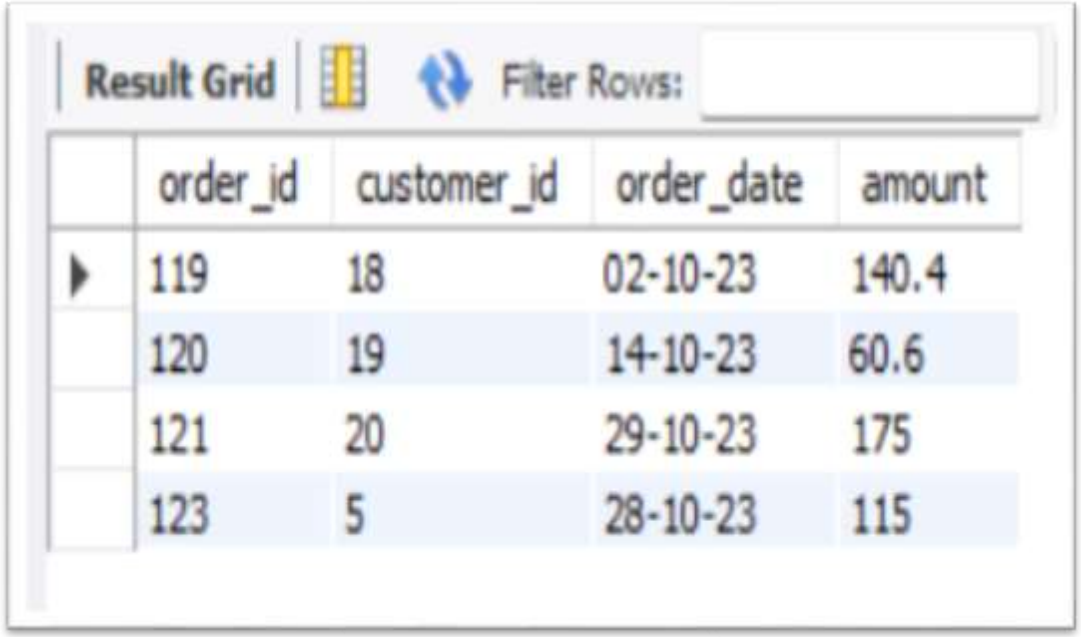


- These customers signed up but never made a purchase.
- Their inactivity highlights missed revenue opportunities.

9. Orders made in October.

```
SELECT *
        FROM orders
        WHERE MONTH(order_date)='10';
```

Result Table

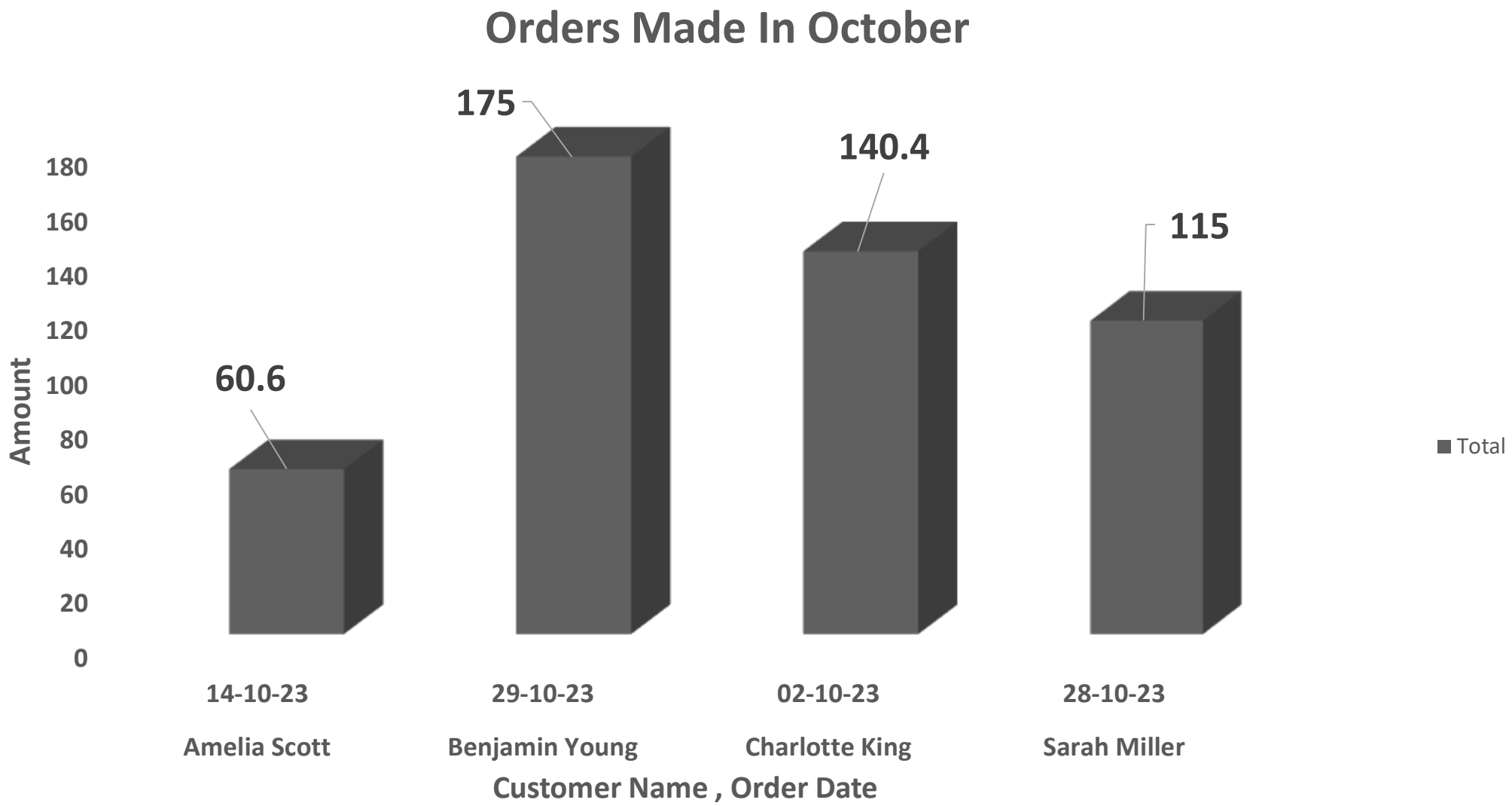


Result Grid | Filter Rows:

order_id	customer_id	order_date	amount
119	18	02-10-23	140.4
120	19	14-10-23	60.6
121	20	29-10-23	175
123	5	28-10-23	115

- Monthly filtering highlights customer activity during that period.
- It shows whether the month had high or low business.

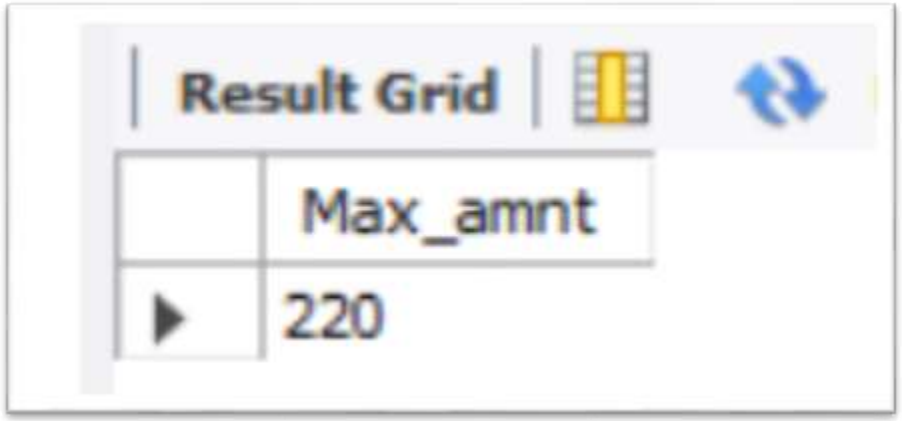
Graphical Representation



10. Highest order amount.

```
SELECT
    MAX(amount) AS Max_amnt
    FROM orders ;
```

Result Table



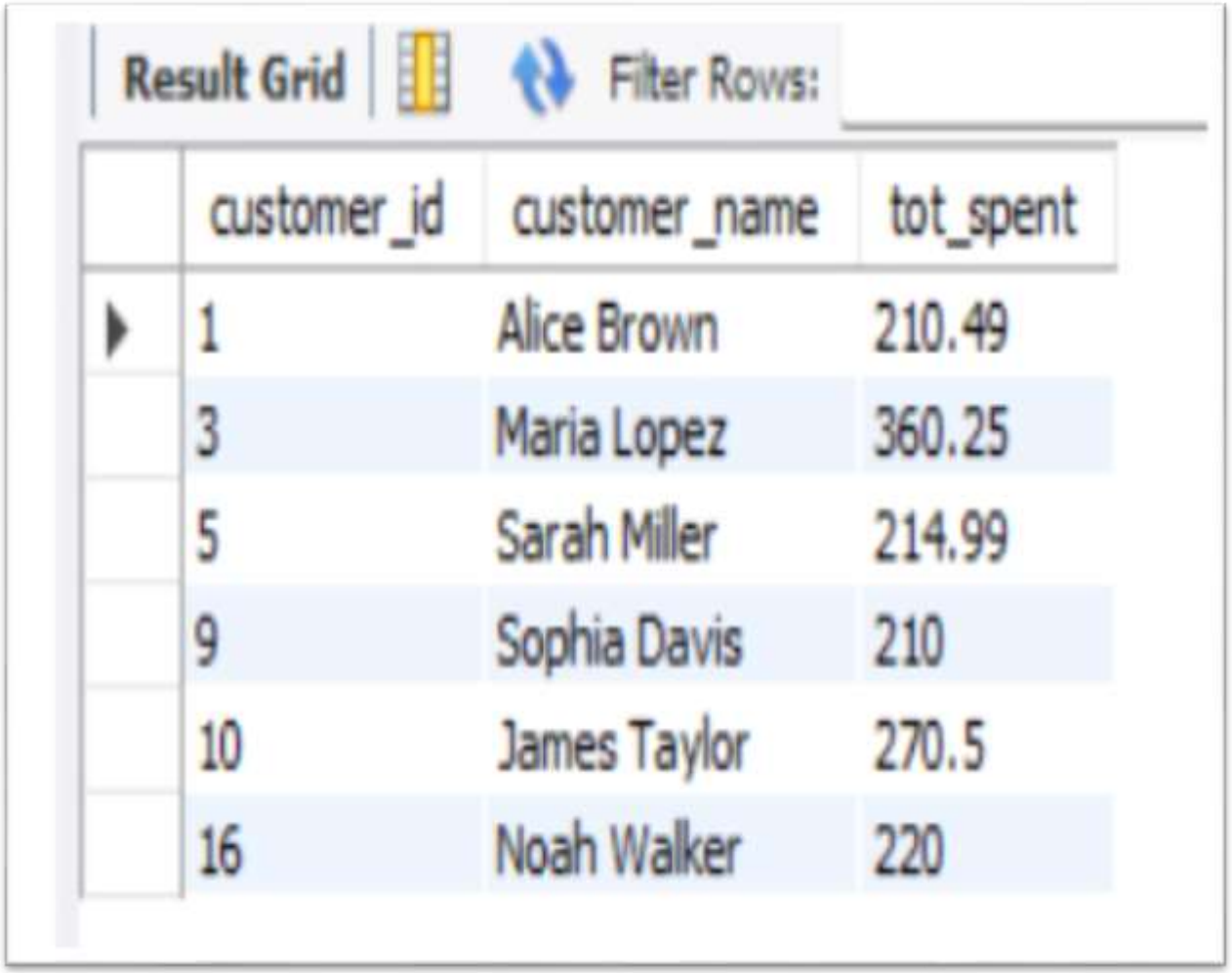
- The largest order stands out clearly compared to the rest.
- It significantly influences total revenue and average order value

11. Customers who spent more than 200 total.

```sql
SELECT
    c.customer_id,
    c.customer_name,
    SUM(o.amount) AS tot_spent
    FROM customers c
LEFT JOIN orders o
    ON c.customer_id=o.customer_id
    GROUP BY c.customer_id,c.customer_name
    HAVING SUM(o.amount)>200;
```

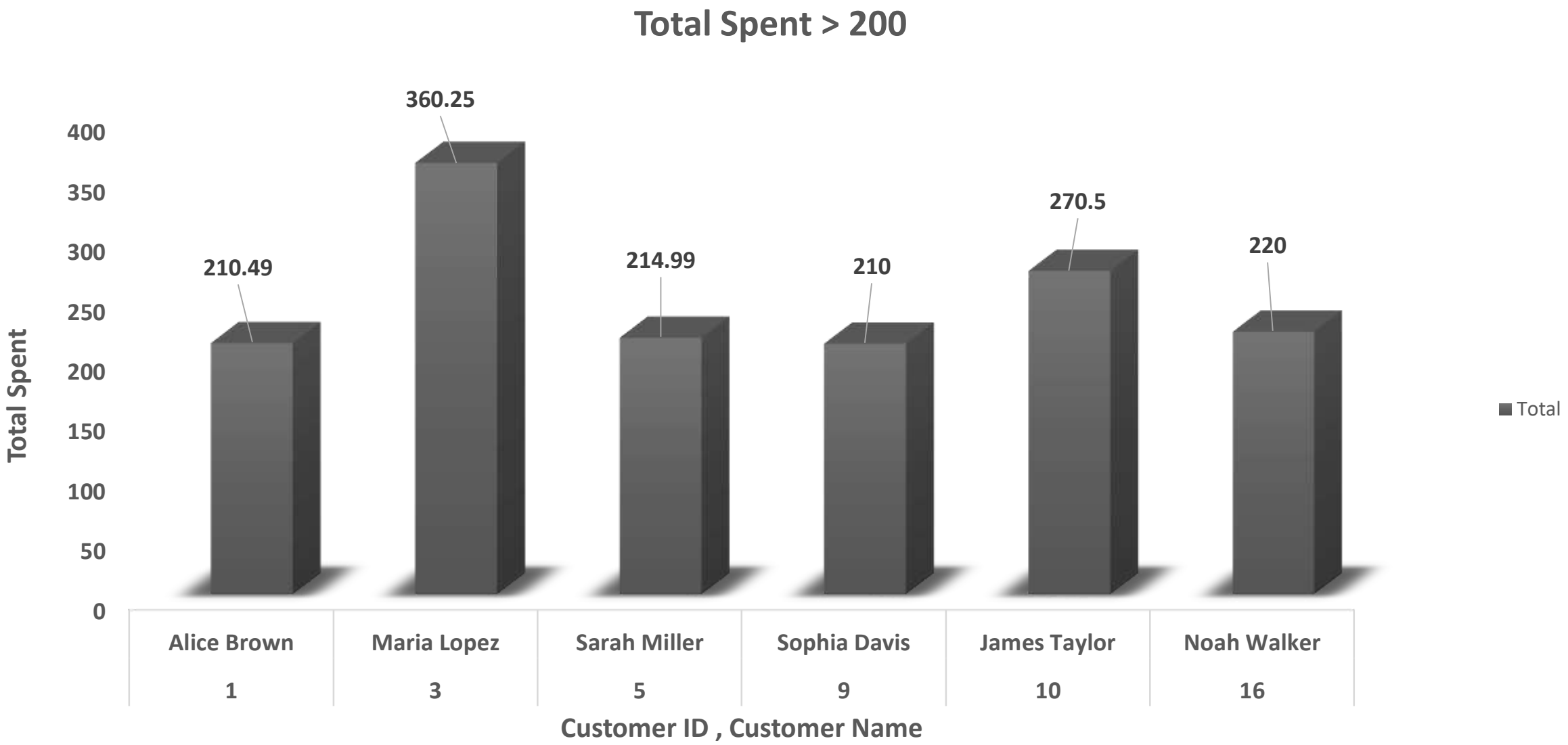
- These customers form the premium spending group.
- Their contributions make up a major percentage of total revenue

Result Table



customer_id	customer_name	tot_spent
1	Alice Brown	210.49
3	Maria Lopez	360.25
5	Sarah Miller	214.99
9	Sophia Davis	210
10	James Taylor	270.5
16	Noah Walker	220

Graphical Representation



12. Number of orders per customer.

```sql
SELECT
      c.customer_id,
      c.customer_name,
      count(o.order_id) AS orders_count
      FROM customers c
LEFT JOIN orders o
      ON c.customer_id=o.customer_id
      GROUP BY c.customer_id,c.customer_name;
```

- This analysis helps categorize customers into frequent and infrequent buyers.
- Some customers place multiple orders, Others place only one order.

Result Table

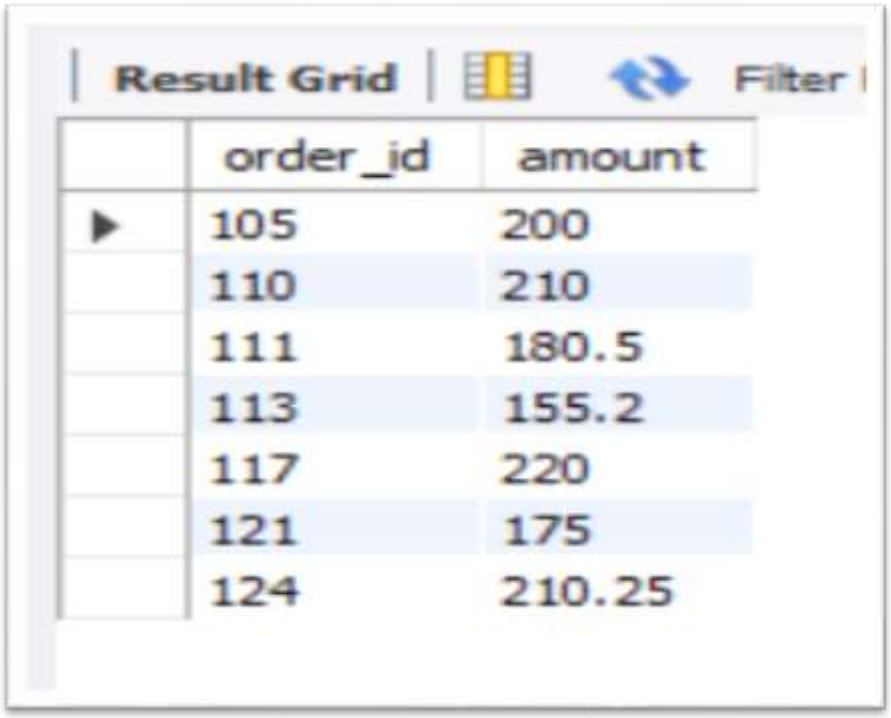
Result Grid | Filter Rows:

customer_id	customer_name	orders_count
1	Alice Brown	2
2	David Smith	1
3	Maria Lopez	2
4	John Carter	1
5	Sarah Miller	2
6	Michael Green	1
7	Emma Wilson	1
8	Oliver Jones	1
9	Sophia Davis	1
10	James Taylor	2
11	Isabella Martin	1
12	Lucas Lee	1
13	Mia Harris	1
14	Ethan Clark	2
15	Ava Lewis	1
16	Noah Walker	1
17	Liam Hall	1
18	Charlotte King	1
19	Amelia Scott	1
20	Benjamin Young	1

13. Orders over $150.

```
SELECT order_id ,amount FROM orders WHERE amount > 150;
```

Result Table



Result Grid | Filter

order_id	amount
105	200
110	210
111	180.5
113	155.2
117	220
121	175
124	210.25

- High-value orders contribute disproportionately to total revenue.
- Comparing both tables shows which customers are high-value buyers.

Graphical Representation

14. First 5 customers by signup date.

```
SELECT
        customer_id,
        customer_name,
        signup_date
        FROM customers
        ORDER BY signup_date LIMIT 5;
```

Result Table

Result Grid | Filter Rows:

customer_id	customer_name	signup_date
5	Sarah Miller	01-04-23
7	Emma Wilson	01-05-23
11	Isabella Martin	01-07-23
13	Mia Harris	01-08-23
16	Noah Walker	01-09-23

- Early customers often have stronger long-term engagement.
- Their purchasing pattern reveals how early adoption affects spending

15. Last order of each customer.

```sql
WITH OrderRank AS (
    SELECT customer_id,order_id,amount,
    ROW_NUMBER()
    OVER (PARTITION BY customer_id ORDER BY order_id) AS rn
    FROM Orders
    )
SELECT * FROM OrderRank WHERE rn = 2;
```

Result Table

Result Grid | Filter Rows:

customer_id	order_id	amount	rn
1	102	89.99	2
3	124	210.25	2
5	123	115	2
10	122	90	2
14	125	99.99	2

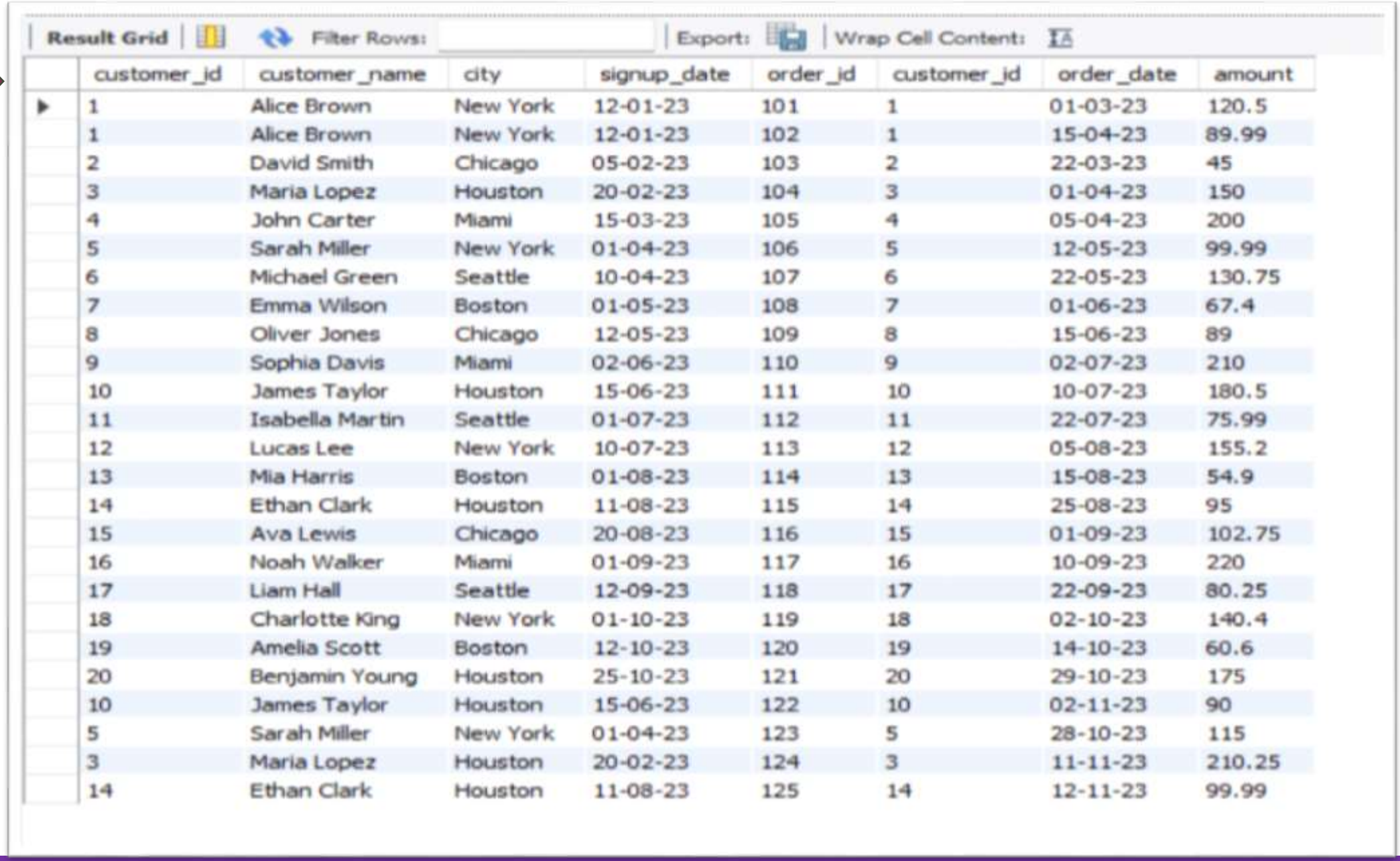
- Recent orders indicate currently active customers.
- Older last-order dates suggest declining engagement

16. Join customer + order details.

```sql
SELECT
    c.customer_id,
    c.customer_name,
    c.city,
    c.signup_date,
    o.order_id,
    o.customer_id,
    o.order_date,
    o.amount
FROM customers c
INNER JOIN orders o
    ON c.customer_id = o.customer_id;
```

- Joining both tables shows complete customer activity in one view.
- It reveals which customers are contributing the most to revenue.

Result Table



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	customer_name	city	signup_date	order_id	customer_id	order_date	amount
1	Alice Brown	New York	12-01-23	101	1	01-03-23	120.5
1	Alice Brown	New York	12-01-23	102	1	15-04-23	89.99
2	David Smith	Chicago	05-02-23	103	2	22-03-23	45
3	Maria Lopez	Houston	20-02-23	104	3	01-04-23	150
4	John Carter	Miami	15-03-23	105	4	05-04-23	200
5	Sarah Miller	New York	01-04-23	106	5	12-05-23	99.99
6	Michael Green	Seattle	10-04-23	107	6	22-05-23	130.75
7	Emma Wilson	Boston	01-05-23	108	7	01-06-23	67.4
8	Oliver Jones	Chicago	12-05-23	109	8	15-06-23	89
9	Sophia Davis	Miami	02-06-23	110	9	02-07-23	210
10	James Taylor	Houston	15-06-23	111	10	10-07-23	180.5
11	Isabella Martin	Seattle	01-07-23	112	11	22-07-23	75.99
12	Lucas Lee	New York	10-07-23	113	12	05-08-23	155.2
13	Mia Harris	Boston	01-08-23	114	13	15-08-23	54.9
14	Ethan Clark	Houston	11-08-23	115	14	25-08-23	95
15	Ava Lewis	Chicago	20-08-23	116	15	01-09-23	102.75
16	Noah Walker	Miami	01-09-23	117	16	10-09-23	220
17	Liam Hall	Seattle	12-09-23	118	17	22-09-23	80.25
18	Charlotte King	New York	01-10-23	119	18	02-10-23	140.4
19	Amelia Scott	Boston	12-10-23	120	19	14-10-23	60.6
20	Benjamin Young	Houston	25-10-23	121	20	29-10-23	175
10	James Taylor	Houston	15-06-23	122	10	02-11-23	90
5	Sarah Miller	New York	01-04-23	123	5	28-10-23	115
3	Maria Lopez	Houston	20-02-23	124	3	11-11-23	210.25
14	Ethan Clark	Houston	11-08-23	125	14	12-11-23	99.99

17. Total revenue per city.

```
SELECT
     c.city,SUM(o.amount) AS total_revenue FROM Customers c
INNER JOIN Orders o
     ON c.customer_id = o.customer_id GROUP BY c.city;
```

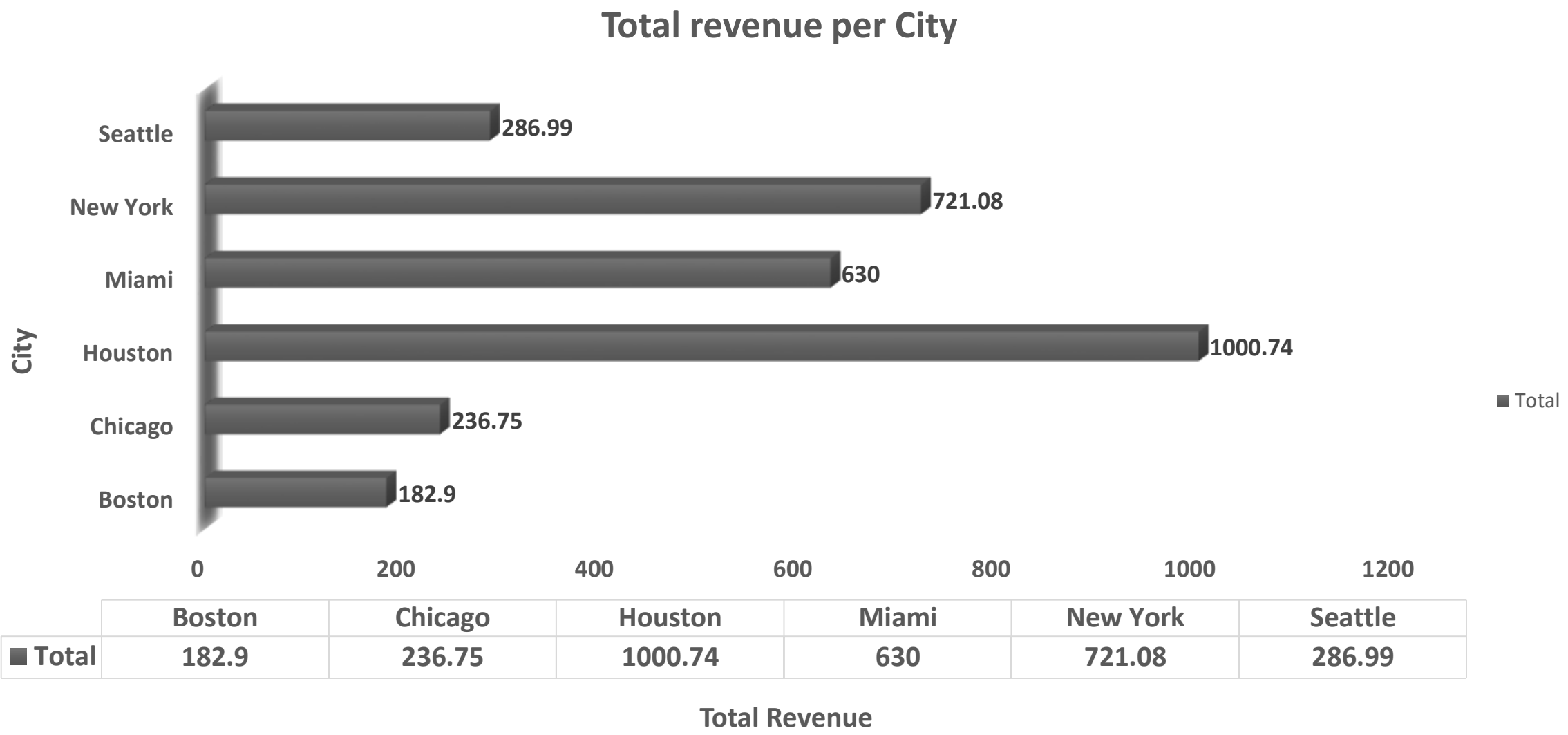
Result Table

Result Grid | Filter Rows:

city	total_revenue
New York	721.08
Chicago	236.75
Houston	1000.74
Miami	630
Seattle	286.99
Boston	182.9

- Revenue differs based on where more active customers are.
- Some locations have fewer customers but higher spending.

Graphical Representation



	Boston	Chicago	Houston	Miami	New York	Seattle
Total	182.9	236.75	1000.74	630	721.08	286.99

18. Customers who joined after July.

```sql
SELECT
    customer_id,
    customer_name,
    city,
    signup_date
    FROM customers
    WHERE MONTH(signup_date)>7;
```

- These newer customers show how quickly new users convert into buyers.
- Comparing their orders reveals early engagement levels.

Result Table

Result Grid | Filter Rows: | Export:

customer_id	customer_name	city	signup_date
13	Mia Harris	Boston	01-08-23
14	Ethan Clark	Houston	11-08-23
15	Ava Lewis	Chicago	20-08-23
16	Noah Walker	Miami	01-09-23
17	Liam Hall	Seattle	12-09-23
18	Charlotte King	New York	01-10-23
19	Amelia Scott	Boston	12-10-23
20	Benjamin Young	Houston	25-10-23

19. Monthly order revenue.

```
SELECT
    MONTH(order_date) AS month_num,
    SUM(amount)
    FROM orders GROUP BY month_num ;
```

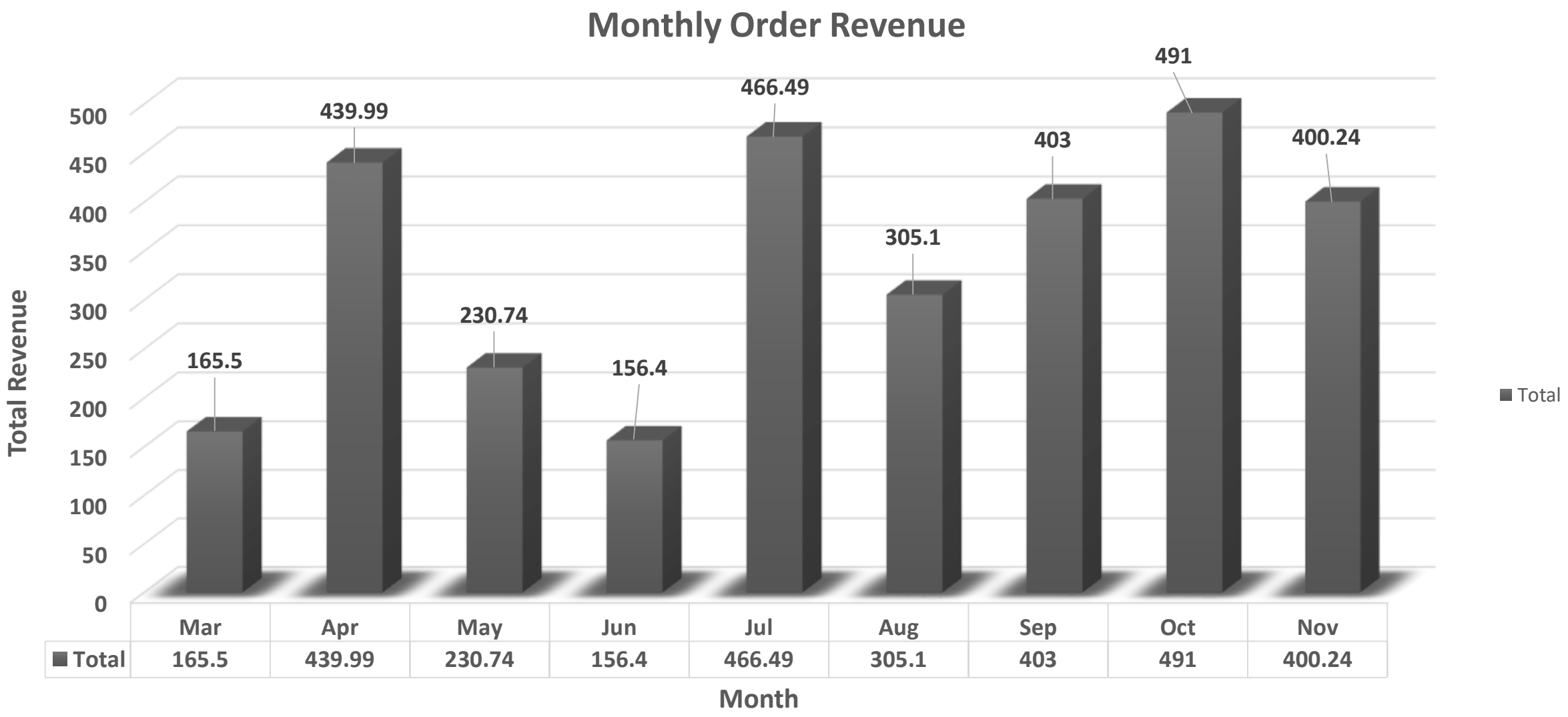
Result Table

Result Grid | Filter Rows:

month_num	SUM(amount)
3	165.5
4	439.99
5	230.74
6	156.4
7	466.49
8	305.1
9	403
10	491
11	400.24

- Revenue varies month to month based on customer activity.
- Some months show clear spikes, driven by multiple high-value orders.
- Other months show fewer orders.

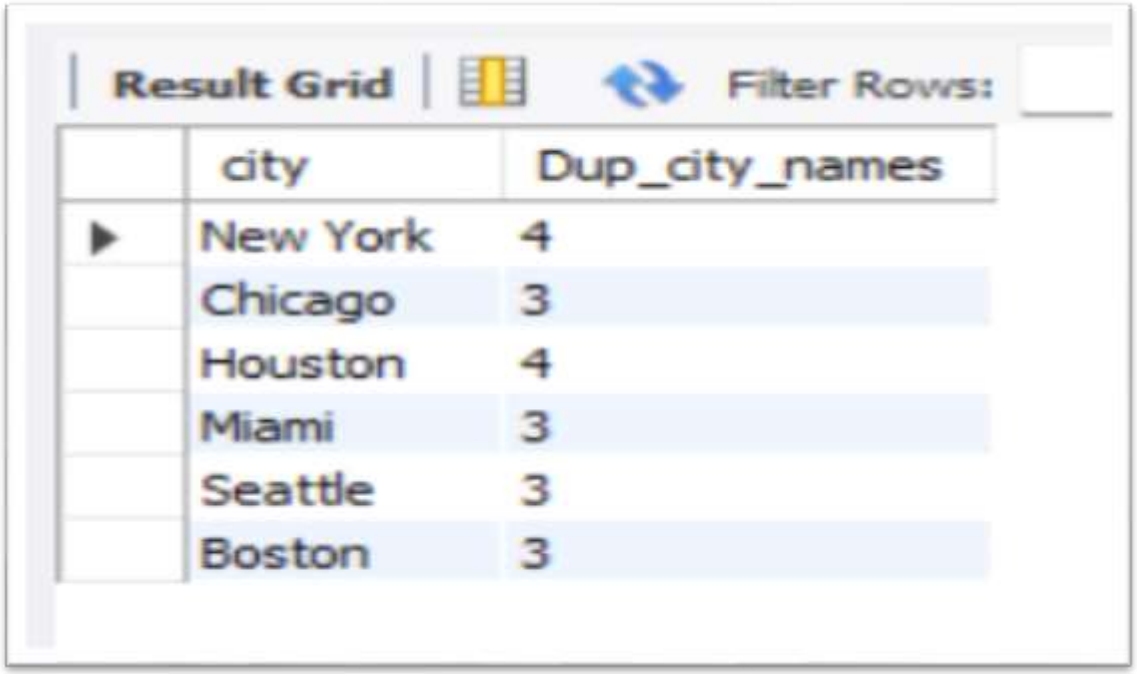
Graphical Representation



20. Find duplicate city names.

```sql
SELECT
    city,
    count(*) AS Dup_city_names
    FROM customers
    GROUP BY city
    HAVING count(*)>1;
```

Result Table



Result Grid | Filter Rows:

city	Dup_city_names
New York	4
Chicago	3
Houston	4
Miami	3
Seattle	3
Boston	3

- The duplication highlights key areas of customer presence.
- This cluster often results in higher total order volume.

21. Top 5 highest spending customers.

```sql
SELECT
    c.customer_id,
    c.customer_name,
    SUM(o.amount) AS total_spent
    FROM customers c
    INNER JOIN orders o
    ON c.customer_id=o.customer_id
    GROUP BY c.customer_id,c.customer_name
    ORDER BY total_spent DESC LIMIT 5;
```

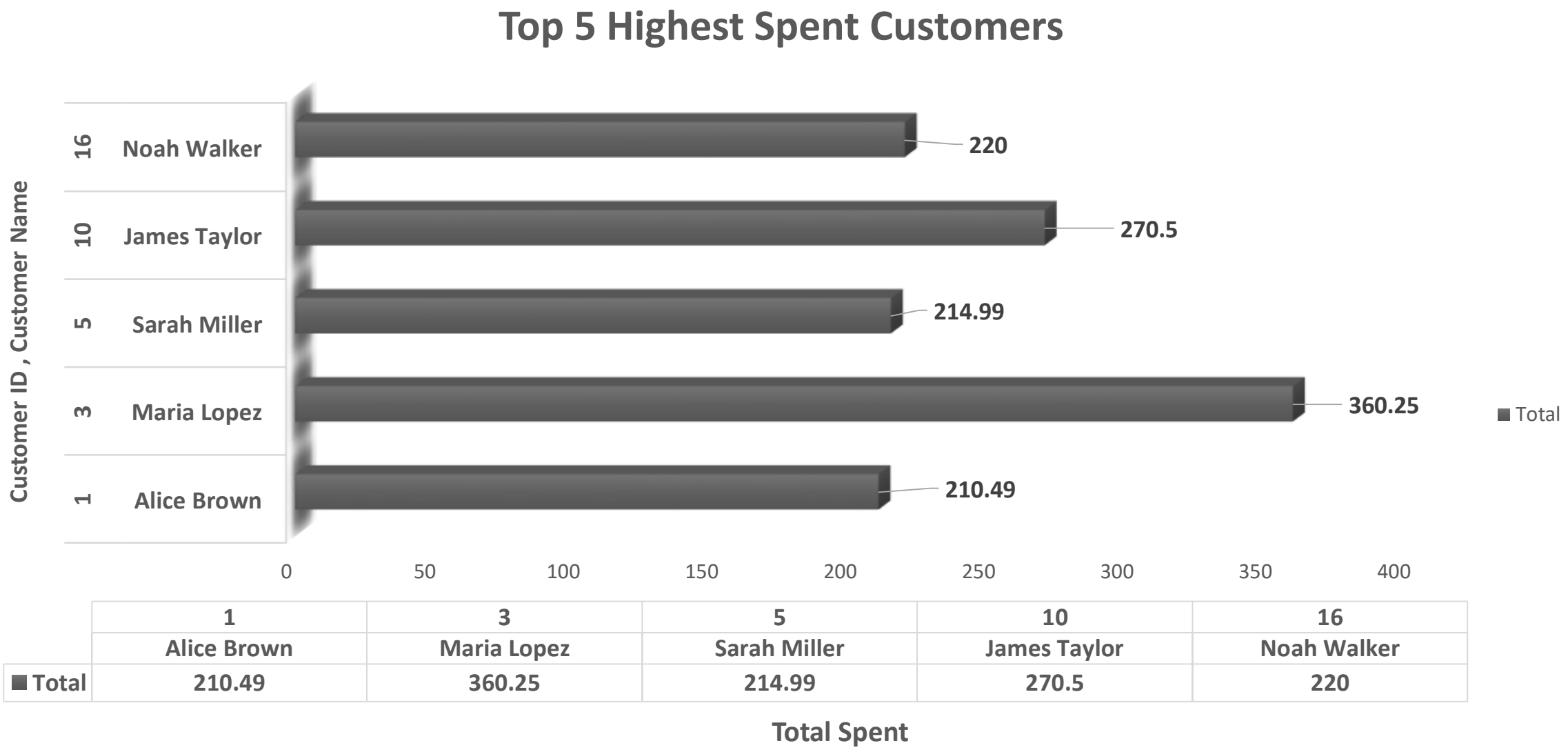
- These customers contribute the majority of the total revenue.
- The ranking highlights clear differences in spending behavior.

Result Table

Result Grid | Filter Rows:

customer_id	customer_name	total_spent
3	Maria Lopez	360.25
10	James Taylor	270.5
16	Noah Walker	220
5	Sarah Miller	214.99
1	Alice Brown	210.49

Graphical Representation



	1	3	5	10	16
	Alice Brown	Maria Lopez	Sarah Miller	James Taylor	Noah Walker
Total	210.49	360.25	214.99	270.5	220

22. Orders sorted by amount (descending).

```sql
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.amount
    FROM customers c
    INNER JOIN orders o
    ON c.customer_id=o.customer_id
    ORDER BY o.amount DESC;
```

- Sorting highlights the gap between low and high-value orders.
- Largest orders appear at the top, showing premium transactions.

Result Table

Result Grid | Filter Rows: | Export:

customer_id	customer_name	order_id	amount
16	Noah Walker	117	220
3	Maria Lopez	124	210.25
9	Sophia Davis	110	210
4	John Carter	105	200
10	James Taylor	111	180.5
20	Benjamin Young	121	175
12	Lucas Lee	113	155.2
3	Maria Lopez	104	150
18	Charlotte King	119	140.4
6	Michael Green	107	130.75
1	Alice Brown	101	120.5
5	Sarah Miller	123	115
15	Ava Lewis	116	102.75
5	Sarah Miller	106	99.99
14	Ethan Clark	125	99.99
14	Ethan Clark	115	95
10	James Taylor	122	90
1	Alice Brown	102	89.99
8	Oliver Jones	109	89
17	Liam Hall	118	80.25
11	Isabella Martin	112	75.99
7	Emma Wilson	108	67.4
19	Amelia Scott	120	60.6
13	Mia Harris	114	54.9
2	David Smith	103	45

23. Customers and their total number of orders (0 included).

```sql
SELECT
    c.customer_id,
    c.customer_name,
    COUNT(o.order_id) AS tot_orders
    FROM customers c
    LEFT JOIN orders o
    ON c.customer_id=o.customer_id
    GROUP BY c.customer_id,c.customer_name;
```

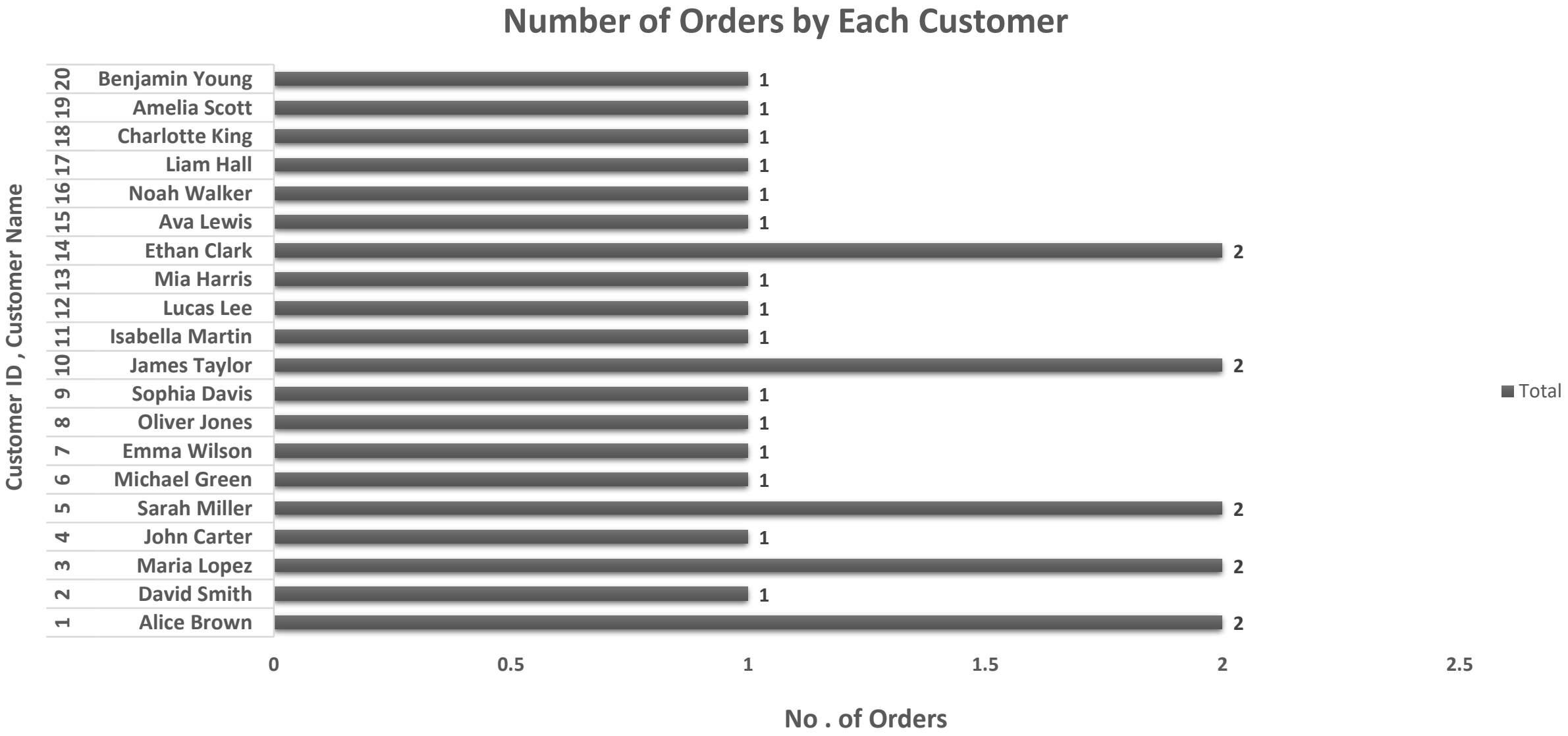
- Combines data to show both active and inactive customers.
- Highlights customers who signed up but never purchased.

Result Table

Result Grid | Filter Rows: | Export:

customer_id	customer_name	tot_orders
1	Alice Brown	2
2	David Smith	1
3	Maria Lopez	2
4	John Carter	1
5	Sarah Miller	2
6	Michael Green	1
7	Emma Wilson	1
8	Oliver Jones	1
9	Sophia Davis	1
10	James Taylor	2
11	Isabella Martin	1
12	Lucas Lee	1
13	Mia Harris	1
14	Ethan Clark	2
15	Ava Lewis	1
16	Noah Walker	1
17	Liam Hall	1
18	Charlotte King	1
19	Amelia Scott	1
20	Benjamin Young	1

Graphical Representation



24. Rank customers by spending (window function).

```sql
SELECT
    c.customer_id,
    c.customer_name,
    SUM(o.amount) AS total_spent,
    DENSE_RANK() OVER (
        ORDER BY SUM(o.amount) DESC
    ) AS spend_rank
FROM customers c
JOIN orders o
    ON c.customer_id = o.customer_id
GROUP BY
    c.customer_id,
    c.customer_name
ORDER BY total_spent DESC;
```

- Ranking showcases which customers contribute most to revenue.
- It highlights spending differences not visible in raw data.

Result Table

Result Grid | Filter Rows: | Export: | Wrap

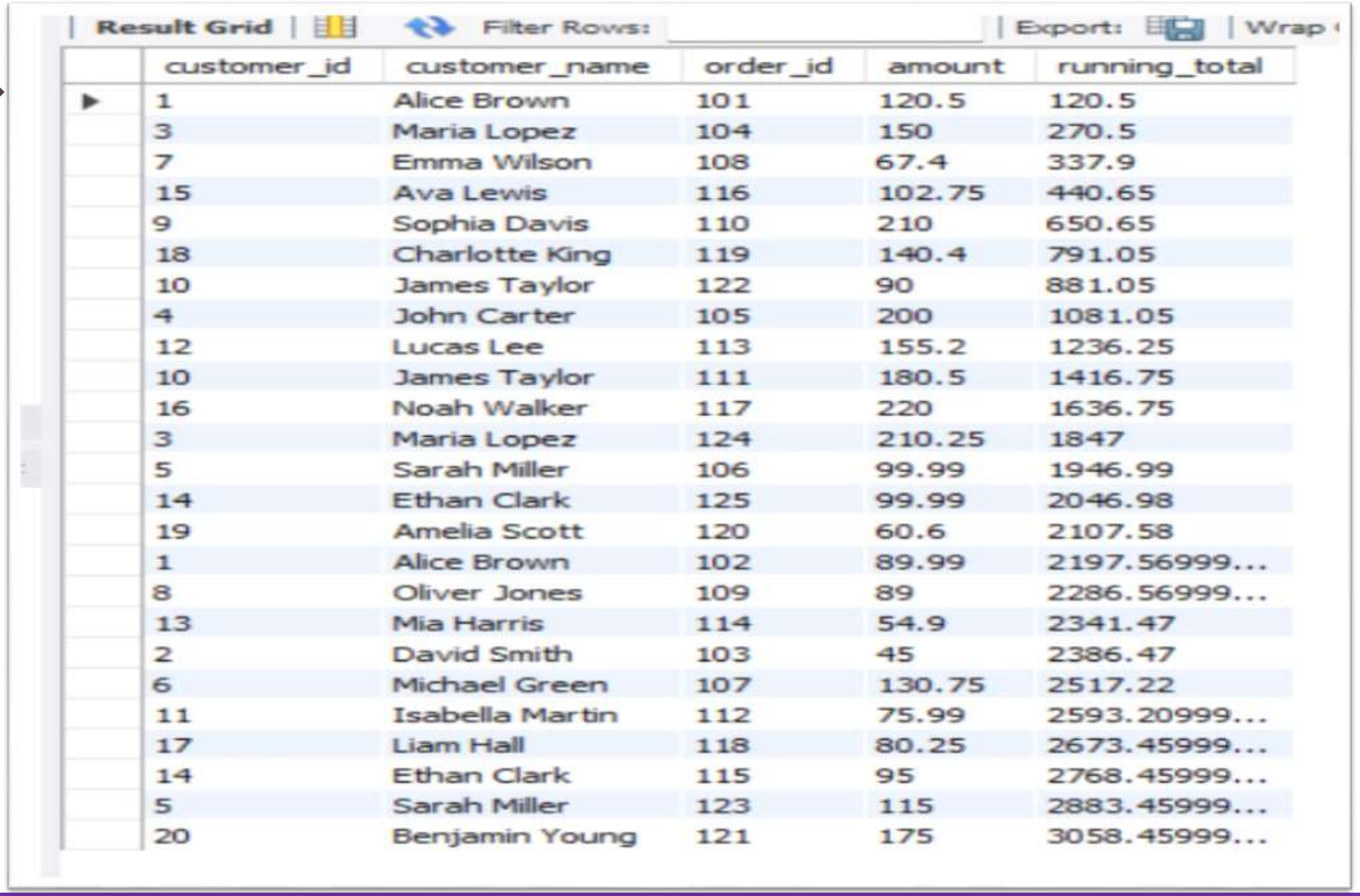
customer_id	customer_name	total_spent	spend_rank
3	Maria Lopez	360.25	1
10	James Taylor	270.5	2
16	Noah Walker	220	3
5	Sarah Miller	214.99	4
1	Alice Brown	210.49	5
9	Sophia Davis	210	6
4	John Carter	200	7
14	Ethan Clark	194.99	8
20	Benjamin Young	175	9
12	Lucas Lee	155.2	10
18	Charlotte King	140.4	11
6	Michael Green	130.75	12
15	Ava Lewis	102.75	13
8	Oliver Jones	89	14
17	Liam Hall	80.25	15
11	Isabella Martin	75.99	16
7	Emma Wilson	67.4	17
19	Amelia Scott	60.6	18
13	Mia Harris	54.9	19
2	David Smith	45	20

25. Running total of order amounts (window function).

```sql
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.amount,
    SUM(o.amount) OVER(
        ORDER BY o.order_date
    ) AS running_total
FROM customers c
JOIN orders o
    ON c.customer_id = o.customer_id;
```

- Cumulative revenue shows how earnings build over time.
- This trend helps identify the most impactful transactions.

Result Table



Result Grid | Filter Rows: | Export: | Wrap

customer_id	customer_name	order_id	amount	running_total
1	Alice Brown	101	120.5	120.5
3	Maria Lopez	104	150	270.5
7	Emma Wilson	108	67.4	337.9
15	Ava Lewis	116	102.75	440.65
9	Sophia Davis	110	210	650.65
18	Charlotte King	119	140.4	791.05
10	James Taylor	122	90	881.05
4	John Carter	105	200	1081.05
12	Lucas Lee	113	155.2	1236.25
10	James Taylor	111	180.5	1416.75
16	Noah Walker	117	220	1636.75
3	Maria Lopez	124	210.25	1847
5	Sarah Miller	106	99.99	1946.99
14	Ethan Clark	125	99.99	2046.98
19	Amelia Scott	120	60.6	2107.58
1	Alice Brown	102	89.99	2197.56999...
8	Oliver Jones	109	89	2286.56999...
13	Mia Harris	114	54.9	2341.47
2	David Smith	103	45	2386.47
6	Michael Green	107	130.75	2517.22
11	Isabella Martin	112	75.99	2593.20999...
17	Liam Hall	118	80.25	2673.45999...
14	Ethan Clark	115	95	2768.45999...
5	Sarah Miller	123	115	2883.45999...
20	Benjamin Young	121	175	3058.45999...


Thank
you