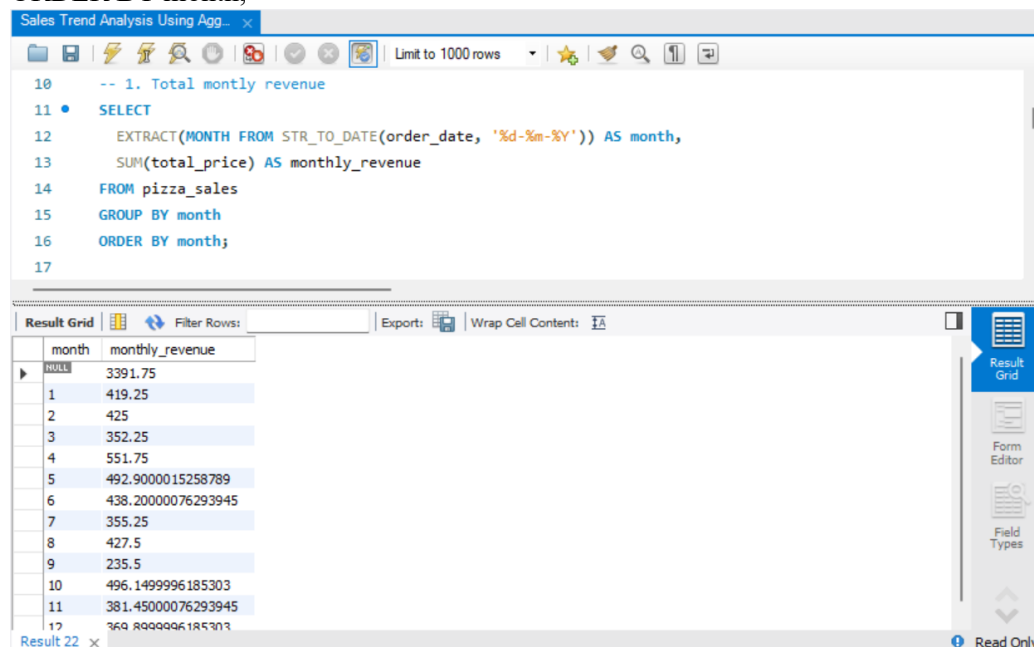


# Sales Trend Analysis Using Aggregations

-- 1. Total montly revenue

```
SELECT
  EXTRACT(MONTH FROM STR_TO_DATE(order_date, '%d-%m-%Y')) AS month,
  SUM(total_price) AS monthly_revenue
FROM pizza_sales
GROUP BY month
ORDER BY month;
```

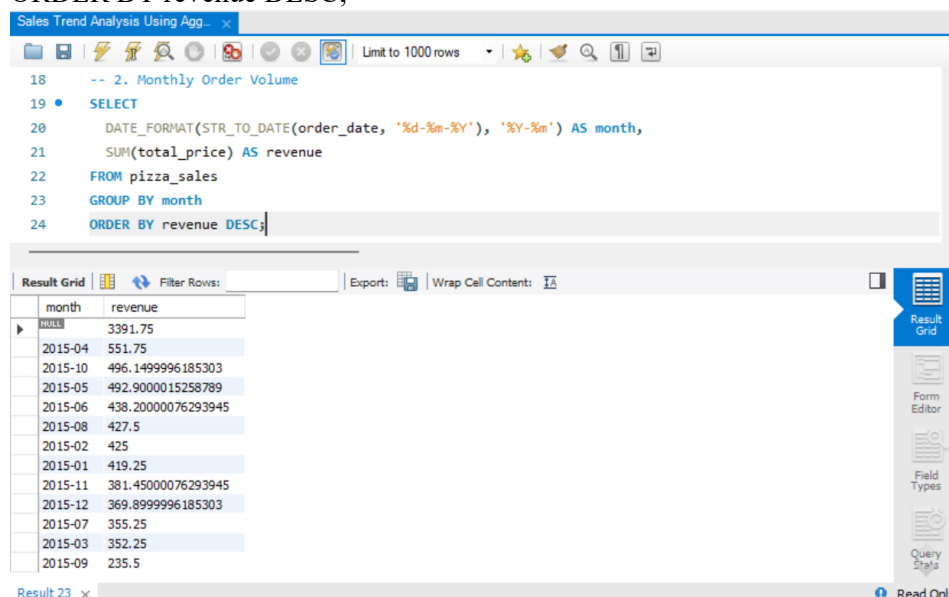


The screenshot shows a SQL IDE window titled "Sales Trend Analysis Using Agg...". The SQL editor contains the query for total monthly revenue. Below the editor, the "Result Grid" shows the output. The first row is NULL with a revenue of 3391.75. Subsequent rows show months 1 through 11 with their respective revenues. The interface includes a toolbar with icons for file operations, a "Limit to 1000 rows" dropdown, and a right sidebar with "Result Grid", "Form Editor", and "Field Types" tabs.

month	monthly_revenue
NULL	3391.75
1	419.25
2	425
3	352.25
4	551.75
5	492.9000015258789
6	438.20000076293945
7	355.25
8	427.5
9	235.5
10	496.1499996185303
11	381.45000076293945
12	369.8999996185303

-- 2. Monthly Order Volume

```
SELECT
  DATE_FORMAT(STR_TO_DATE(order_date, '%d-%m-%Y'), '%Y-%m') AS month,
  SUM(total_price) AS revenue
FROM pizza_sales
GROUP BY month
ORDER BY revenue DESC;
```



The screenshot shows the same SQL IDE window with the second query. The SQL editor contains the query for monthly order volume ordered by revenue in descending order. The "Result Grid" shows the output with months formatted as 'YYYY-MM'. The first row is NULL with a revenue of 3391.75. Subsequent rows show months from 2015-04 to 2015-09 with their respective revenues. The interface is consistent with the first screenshot.

month	revenue
NULL	3391.75
2015-04	551.75
2015-10	496.1499996185303
2015-05	492.9000015258789
2015-06	438.20000076293945
2015-08	427.5
2015-02	425
2015-01	419.25
2015-11	381.45000076293945
2015-12	369.8999996185303
2015-07	355.25
2015-03	352.25
2015-09	235.5

-- 3. Top 5 Revenue-Generating Months

```
SELECT EXTRACT(YEAR_MONTH FROM STR_TO_DATE(order_date, '%d-%m-%Y')) AS month,
  SUM(total_price) AS revenue
FROM pizza_sales
GROUP BY month
```

ORDER BY revenue DESC

LIMIT 5;

The screenshot shows a SQL IDE window titled "Sales Trend Analysis Using Agg...". The query editor contains the following SQL code:

```
-- 3. Top 5 Revenue-Generating Months
SELECT EXTRACT(YEAR_MONTH FROM STR_TO_DATE(order_date, '%d-%m-%Y')) AS month,
SUM(total_price) AS revenue
FROM pizza_sales
GROUP BY month
ORDER BY revenue DESC
LIMIT 5;
```

The result grid below the query shows the following data:

month	revenue
201504	551.75
201510	496.1499996185303
201505	492.9000015258789
201506	438.20000076293945

-- 4. Monthly Revenue for a Specific Year

```
SELECT EXTRACT(MONTH FROM STR_TO_DATE(order_date, '%d-%m-%Y')) AS month,
SUM(total_price) AS revenue
FROM pizza_sales
WHERE EXTRACT(YEAR FROM STR_TO_DATE(order_date, '%d-%m-%Y')) = 2015
GROUP BY month
ORDER BY month;
```

The screenshot shows a SQL IDE window titled "Sales Trend Analysis Using Agg...". The query editor contains the following SQL code:

```
-- 4. Monthly Revenue for a Specific Year
SELECT EXTRACT(MONTH FROM STR_TO_DATE(order_date, '%d-%m-%Y')) AS month,
SUM(total_price) AS revenue
FROM pizza_sales
WHERE EXTRACT(YEAR FROM STR_TO_DATE(order_date, '%d-%m-%Y')) = 2015
GROUP BY month
ORDER BY month;
```

The result grid below the query shows the following data:

month	revenue
1	419.25
2	425
3	352.25
4	551.75
5	492.9000015258789
6	438.20000076293945
7	355.25
8	427.5
9	235.5
10	496.1499996185303
11	381.45000076293945
12	369.8999996185303

-- 5. Most Popular Pizza Sizes by Volume

```
SELECT pizza_size, COUNT(*) AS total_orders
FROM pizza_sales
GROUP BY pizza_size
ORDER BY total_orders DESC;
```

Sales Trend Analysis Using Agg. x

Limit to 1000 rows

```

41
42 -- 5. Most Popular Pizza Sizes by Volume
43 • SELECT pizza_size, COUNT(*) AS total_orders
44 FROM pizza_sales
45 GROUP BY pizza_size
46 ORDER BY total_orders DESC;
47

```

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

	pizza_size	total_orders
▶	L	182
	M	171
	S	142
	XL	5

Result 26 x Read Only

Result Grid, Form Editor, Field Types, Query Stats

-- 6. Revenue by Pizza Category

```

SELECT pizza_category, SUM(total_price) AS revenue
FROM pizza_sales
GROUP BY pizza_category
ORDER BY revenue DESC;

```

Sales Trend Analysis Using Agg. x

Limit to 1000 rows

```

47
48 -- 6. Revenue by Pizza Category
49 • SELECT pizza_category, SUM(total_price) AS revenue
50 FROM pizza_sales
51 GROUP BY pizza_category
52 ORDER BY revenue DESC;
53

```

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

	pizza_category	revenue
▶	Chicken	2249.75
	Classic	2239.5
	Supreme	2038.849998474121
	Veggie	1808.7500038146973

Result 27 x Read Only

Result Grid, Form Editor, Field Types, Query Stats

-- 7. Top 5 Pizzas by Revenue

```

SELECT pizza_name, SUM(total_price) AS revenue
FROM pizza_sales
GROUP BY pizza_name
ORDER BY revenue DESC
LIMIT 5;

```

Sales Trend Analysis Using Agg...

```

53
54 -- 7. Top 5 Pizzas by Revenue
55 • SELECT pizza_name, SUM(total_price) AS revenue
56 FROM pizza_sales
57 GROUP BY pizza_name
58 ORDER BY revenue DESC
59 LIMIT 5;

```

Result Grid

	pizza_name	revenue
▶	The California Chicken Pizza	540
	The Thai Chicken Pizza	503.5
	The Barbecue Chicken Pizza	472.25
	The Classic Deluxe Pizza	373.5
	The Pepperoni Pizza	362.25

Result 28 x Read Only

-- 8. Top 5 Pizzas by Quantity Sold

```

SELECT pizza_name, SUM(quantity) AS total_quantity
FROM pizza_sales
GROUP BY pizza_name
ORDER BY total_quantity DESC
LIMIT 5;

```

Sales Trend Analysis Using Agg...

```

61 -- 8. Top 5 Pizzas by Quantity Sold
62 • SELECT pizza_name, SUM(quantity) AS total_quantity
63 FROM pizza_sales
64 GROUP BY pizza_name
65 ORDER BY total_quantity DESC
66 LIMIT 5;
67

```

Result Grid

	pizza_name	total_quantity
▶	The California Chicken Pizza	32
	The Pepperoni Pizza	27
	The Barbecue Chicken Pizza	27
	The Hawaiian Pizza	26
	The Thai Chicken Pizza	26

Result 29 x Read Only

-- 9. Order Volume and Revenue by Day of Week

```

SELECT DAYNAME(STR_TO_DATE(order_date, '%d-%m-%Y')) AS day_of_week,
COUNT(DISTINCT order_id) AS order_volume,
SUM(total_price) AS revenue
FROM pizza_sales
GROUP BY day_of_week
ORDER BY revenue DESC;

```

Sales Trend Analysis Using Agg...

```

68 -- 9. Order Volume and Revenue by Day of Week
69 • SELECT DAYNAME(STR_TO_DATE(order_date, '%d-%m-%Y')) AS day_of_week,
70       COUNT(DISTINCT order_id) AS order_volume,
71       SUM(total_price) AS revenue
72 FROM pizza_sales
73 GROUP BY day_of_week
74 ORDER BY revenue DESC;

```

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

day_of_week	order_volume	revenue
NULL	199	3391.75
Thursday	53	953.6499996185303
Friday	51	849.3999996185303
Monday	44	766.25
Saturday	43	687.75
Tuesday	41	679.7000007629395
Wednesday	35	566.4500007629395
Sunday	28	441.9000015258789

Result 30 x | Read Only

-- 10. Monthly Revenue by Pizza Category  
 SELECT pizza\_category, EXTRACT(YEAR FROM STR\_TO\_DATE(order\_date, '%d-%m-%Y')) AS year,  
 EXTRACT(MONTH FROM STR\_TO\_DATE(order\_date, '%d-%m-%Y')) AS month,  
 SUM(total\_price) AS revenue  
 FROM pizza\_sales  
 GROUP BY pizza\_category, year, month  
 ORDER BY pizza\_category, year, month;

Sales Trend Analysis Using Agg...

```

76 -- 10. Monthly Revenue by Pizza Category
77 • SELECT pizza_category, EXTRACT(YEAR FROM STR_TO_DATE(order_date, '%d-%m-%Y')) AS year,
78       EXTRACT(MONTH FROM STR_TO_DATE(order_date, '%d-%m-%Y')) AS month,
79       SUM(total_price) AS revenue
80 FROM pizza_sales
81 GROUP BY pizza_category, year, month
82 ORDER BY pizza_category, year, month;

```

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

pizza_category	year	month	revenue
Chicken	2015	6	166.75
Chicken	2015	7	95.75
Chicken	2015	8	121.25
Chicken	2015	9	33.5
Chicken	2015	10	175.5
Chicken	2015	11	20.75
Chicken	2015	12	37.5
Classic	NULL	NULL	894.75
Classic	2015	1	148.75
Classic	2015	2	93.5
Classic	2015	3	39
Classic	2015	4	135.75
Classic	2015	5	120

Result 32 x | Read Only