

Query 1 x SQL File 3* SQL File 4* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL



```
1  -- 1 Retrieve the total number of orders placed.  
2  
3  • SELECT  
4      COUNT(order_id) AS total_orders  
5  FROM  
6      orders;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

	total_orders
▶	21350



```
1  -- Calculate the total revenue generated from pizza sales.
2
3  •  SELECT
4      ROUND(SUM(order_details.quantity * pizzas.price),
5             2) AS total_Revenue
6  FROM
7      order_details
8      JOIN
9      pizzas ON pizzas.pizza_id = order_details.pizza_id;
10
11
```

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

	total_Revenue
--	---------------

▶	817860.05
---	-----------

Query 1 SQL File 3 SQL File 4 x SQL File 7 SQL File 8 SQL File 9 SQL File 10



```
1  -- Identify the highest-priced pizza.
2
3  •  SELECT
4      pizza_types.name, pizzas.price
5  FROM
6      pizza_types
7      JOIN
8      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9  ORDER BY pizzas.price DESC
10 LIMIT 1;
```

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

	name	price
▶	The Greek Pizza	35.95

Query 1 SQL File 3* SQL File 4* **SQL File 7* x** SQL File 8* SQL File 9* SQL File 10*



Limit to 1000 rows

```
1  -- 4 Identify the most common pizza size ordered.
2
3  • SELECT
4      pizzas.size,
5      COUNT(order_details.order_details_id) AS order_count
6  FROM
7      pizzas
8      JOIN
9      order_details ON pizzas.pizza_id = order_details.pizza_id
10 GROUP BY pizzas.size
11 ORDER BY order_count DESC;
```

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

	size	order_count
▶	L	18526
	M	15385
	S	14137
	XL	544
	XXL	28

Query 1 SQL File 3* SQL File 4* SQL File 7* **SQL File 8* x** SQL File 9* SQL File 10* SQL

         | Limit to 1000 rows     





```
1  -- 5 List the top 5 most ordered pizza types along with their quantities.
2
3  •  SELECT
4      pt.name AS pizza_name,
5      SUM(od.quantity) AS total_quantity_ordered
6  FROM
7      pizza_types AS pt
8      JOIN
9      pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
10     JOIN
11     order_details AS od ON p.pizza_id = od.pizza_id
12 GROUP BY pt.name
```

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









	pizza_name	total_quantity_ordered
▶	The Classic Deluxe Pizza	2453
	The Barbecue Chicken Pizza	2432
	The Hawaiian Pizza	2422
	The Pepperoni Pizza	2418
	The Thai Chicken Pizza	2371








```
1  -- 6 | Join the necessary tables to find the total quantity of each pizza category ordered.
2
3  • SELECT
4      pizza_types.category,
5      SUM(order_details.quantity) AS total_quantity
6  FROM
7      pizza_types
8      JOIN
9      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10     JOIN
11     order_details ON order_details.pizza_id = pizzas.pizza_id
12 GROUP BY pizza_types.category
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	category	total_quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050



Limit to 1000 rows



1

-- 7 Determine the distribution of orders by hour of the day.

2

3 • SELECT

4 HOUR(time) AS hour, COUNT(order_id) AS order_count

5 FROM


6 orders


7 GROUP BY HOUR(time);


8

9

Result Grid

 Filter Rows:

Export: 

Wrap Cell Content: 

	hour	order_count
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642
	21	1198
	22	663
	23	28
	10	8
	9	1



```
1  -- 9 Group the orders by date and calculate the average number of pizzas ordered per day.
2
3  • SELECT
4      ROUND(AVG(quantity), 0)
5  FROM
6      (SELECT
7          orders.date, SUM(order_details.quantity) AS quantity
8      FROM
9          orders
10     JOIN order_details ON orders.order_id = order_details.order_id
11     GROUP BY orders.date) AS order_quantity;
```

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

	ROUND(AVG(quantity), 0)
--	-------------------------

▶	138
---	-----



```
1  -- 10 Determine the top 3 most ordered pizza types based on revenue.  
2  Execute the selected portion of the script or everything, if there is no selection  
3  • SELECT  
4      pizza_types.name,  
5      SUM(order_details.quantity * pizzas.price) AS revenue  
6  FROM  
7      pizza_types  
8      JOIN  
9      pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id  
10     JOIN  
11     order_details ON order_details.pizza_id = pizzas.pizza_id  
12 GROUP BY pizza_types.name  
13 ORDER BY revenue DESC  
14 LIMIT 3;
```

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

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5



```
1  -- 11 Calculate the percentage contribution of each pizza type to total revenue.
```

Stop the query being executed (the connection to the DB server will not be restarted and any open transactions will remain open)

```
3  • select pizza_types.category,  
4     round(sum(order_details.quantity * pizzas.price) / (select  
5     round(sum(order_details.quantity * pizzas.price),2) as total_sales  
6     from order_details join pizzas  
7     on pizzas.pizza_id = order_details.pizza_id) *100,2) as revenue  
8     from pizza_types join pizzas  
9     on pizza_types.pizza_type_id = pizzas.pizza_type_id  
10    join order_details  
11    on order_details.pizza_id = pizzas.pizza_id  
12    group by pizza_types.category  
13    order by revenue desc;
```

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

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

Limit to 1000 rows

```
1  -- 12 Analyze the cumulative revenue generated over time.
2
3  • select date,
4     sum(revenue) over (order by date ) as cum_revenue from
5  (select orders.date,
6     sum(order_details.quantity * pizzas.price) as revenue
7     from order_details join pizzas
8     on order_details.pizza_id = pizzas.pizza_id
9     join orders
10    on orders.order_id = order_details.order_id
11    group by orders.date) as sales;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

	date	cum_revenue
▶	2015-01-01 00:00:00	2713.8500000000004
	2015-01-02 00:00:00	5445.75
	2015-01-03 00:00:00	8108.15
	2015-01-04 00:00:00	9863.6
	2015-01-05 00:00:00	11929.55
	2015-01-06 00:00:00	14358.5
	2015-01-07 00:00:00	16560.7
	2015-01-08 00:00:00	19399.05
	2015-01-09 00:00:00	21526.4
	2015-01-10 00:00:00	23990.350000000002
	2015-01-11 00:00:00	25862.65
	2015-01-12 00:00:00	27781.7
	2015-01-13 00:00:00	29831.300000000003
	2015-01-14 00:00:00	32358.700000000004
	2015-01-15 00:00:00	34343.500000000001
	2015-01-16 00:00:00	36937.650000000001
	2015-01-17 00:00:00	39001.750000000001

```

1  -- 13 Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2
3
4  • select name, revenue from
5  (select category, name, revenue,
6   rank() over(partition by category order by revenue desc) as rn
7   from
8   (select pizza_types.category, pizza_types.name,
9    sum((order_details.quantity) * pizzas.price) as revenue
10   from pizza_types join pizzas
11    on pizza_types.pizza_type_id = pizzas.pizza_type_id
12   join order_details
13    on order_details.pizza_id = pizzas.pizza_id
14   group by pizza_types.category, pizza_types.name ) as a) as b
15  where rn <= 3;

```

 Result Grid Filter Rows: Export: Wrap Cell Content:

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5
	The Classic Deluxe Pizza	38180.5
	The Hawaiian Pizza	32273.25
	The Pepperoni Pizza	30161.75
	The Spicy Italian Pizza	34831.25
	The Italian Supreme Pizza	33476.75
	The Sicilian Pizza	30940.5
	The Four Cheese Pizza	32265.70000000065
	The Mexicana Pizza	26780.75
	The Five Cheese Pizza	26066.5