

All
About
Pizza!



PIZZA SALES SQL PROJECT

Analyzing pizza sales data to extract insights on
revenue, orders, and popular pizzas.

Basic:

Retrieve the total number of orders placed.

Calculate the total revenue generated from pizza sales.

Identify the highest-priced pizza.

Identify the most common pizza size ordered.

List the top 5 most ordered pizza types along with their quantities.

Intermediate:

Join the necessary tables to find the total quantity of each pizza category ordered.

Determine the distribution of orders by hour of the day.

Join relevant tables to find the category-wise distribution of pizzas.

Group the orders by date and calculate the average number of pizzas ordered per day.

Determine the top 3 most ordered pizza types based on revenue.

Advanced:

Calculate the percentage contribution of each pizza type to total revenue.

Analyze the cumulative revenue generated over time.

Determine the top 3 most ordered pizza types based on revenue for each pizza category.




```
Query 1 x SQL File 1* SQL File 2* SQL File 3* SQL File 4*
Limit to 1000 rows

1 • create database pizzahut;
2 • use pizzahut;
3 • create table orders (
4   order_id int not null,
5   order_date date not null,
6   order_time time not null,
7   primary key(order_id));
8
9 • create table order_details (
10  order_details_id int not null,
11  order_id int not null,
12  pizza_id text not null,
13  quantity int not null,
14  primary key(order_details_id));
15
16
```



Liceria & Co.

[Home](#)

About Us

Content

Contact Us

```
-- Calculate the total revenue generated from pizza sales.
SELECT
    round(sum(order_details.quantity * pizzas.price),-- selecting quant and price for cal
          2) AS total_sales
FROM
    order_details
    JOIN
    pizzas
ON pizzas.pizza_id = order_details.pizza_id -- common between 2 tables
```

The screenshot shows the SQL Server Enterprise Manager interface. The top menu bar includes "File", "Server", "Tools", "Scripting", and "Help". Below the menu is a toolbar with icons for file operations and database management. The main area displays several tabs: "Query 1", "SQL File 1*", "SQL File 2*", "SQL File 3*", "SQL File 4*", and "SQL File 5*". The active tab is "SQL File 1*", which contains the following SQL code:

```
-- Retrieve the total number of orders placed.
select count(order_id) as total_orders from orders;
```

Below the query editor, there is a section for "Result Grid" and "Filter Rows:". The "Result Grid" shows a single row with the column name "total_orders" and the value "21350".

	total_orders
▶	21350

A watermark "Hannah Morales" is visible across the bottom right portion of the image.

Liceria & Co.

[Home](#)

About Us

Content

Contact Us

```
-- Identify the highest-priced pizza.
select pizza_types.name,pizzas.price -- selection of two tables name and price
from pizza_types join pizzas
on pizza_types.pizza_type_id=pizzas.pizza_type_id -- common part in both table is pizza id
order by pizzas.price desc limit 1;-- limit 1 → After sorting, LIMIT 1 tells SQL to return only 1 row
```

The screenshot shows a SQL IDE interface. The top menu bar includes 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main editor area displays a SQL query in a file named 'SQL File 4*'. The query is as follows:

```

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

```

Below the query editor, the 'Result Grid' tab is active, showing the results of the query. The grid has two columns: 'size' and 'order_count'. The results are as follows:

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

Liceria & Co.

```
1  -- List the top 5 most ordered pizza types along with their quantities.
2  • select pizza_types.name,
3     sum(order_details.quantity) as quantity
4  from pizza_types join pizzas
5  on pizza_types.pizza_type_id=pizzas.pizza_type_id
6  join order_details
7  on order_details.pizza_id=pizzas.pizza_id
8  group by pizza_types.name order by quantity desc limit 5;
```

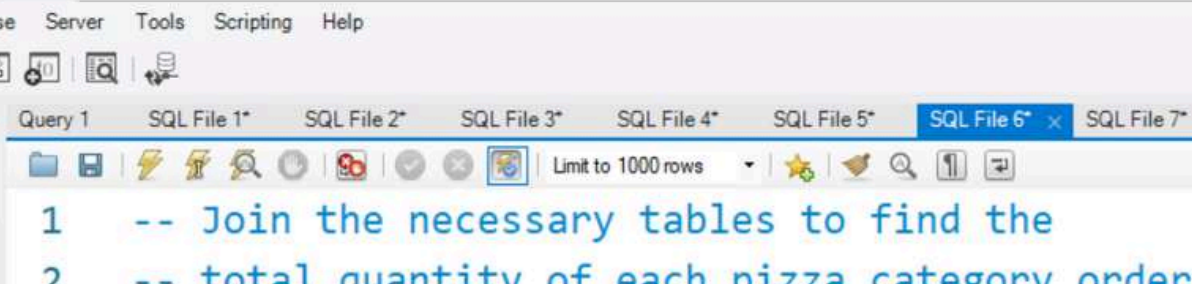
Result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	name	quantity		
▶	The Classic Deluxe Pizza	2453		
	The Barbecue Chicken Pizza	2432		
	The Hawaiian Pizza	2422		
	The Pepperoni Pizza	2418		
	The Thai Chicken Pizza	2371		

Home

About Us

Content

Contact Us



The screenshot shows a MySQL IDE window with a query editor. The query is as follows:

```

1  -- Join the necessary tables to find the
2  -- total quantity of each pizza category ordered.
3 • select pizza_types.category,
4     sum(order_details.quantity) as quantity
5  from pizza_types join pizzas
6  on pizza_types.pizza_type_id=pizzas.pizza_type_id
7  join order_details
8  on order_details.pizza_id=pizzas.pizza_id
9  group by pizza_types.category order by quantity desc

```

Result Grid		Filter Rows:	Export:	Wrap Cell Contents:
	category	quantity		
▶	Classic	14888		
	Supreme	11987		
	Veggie	11649		
	Chicken	11050		

Query 1SQL File 1*SQL File 2*SQL File 3*SQL File 4*SQL File 5*SQL File 6*SQL File 7* xSQL File 8*SQL File 9*SQL File 10*SQL File 11*

Limit to 1000 rows

1 -- Determine the distribution of orders by hour of the day.
2 • SELECT
3 **hour**(order_time) **AS** hours, **count**(order_id) **AS** order_count
4 **FROM**
5 orders
6 **GROUP BY** **hour**(order_time)
7

Result Grid

	hours	order_count
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642

Result 2 x

Query 1SQL File 1*SQL File 2*SQL File 3*SQL File 4*SQL File 5*SQL File 6*SQL File 7*SQL File 8* xSQL File 9*

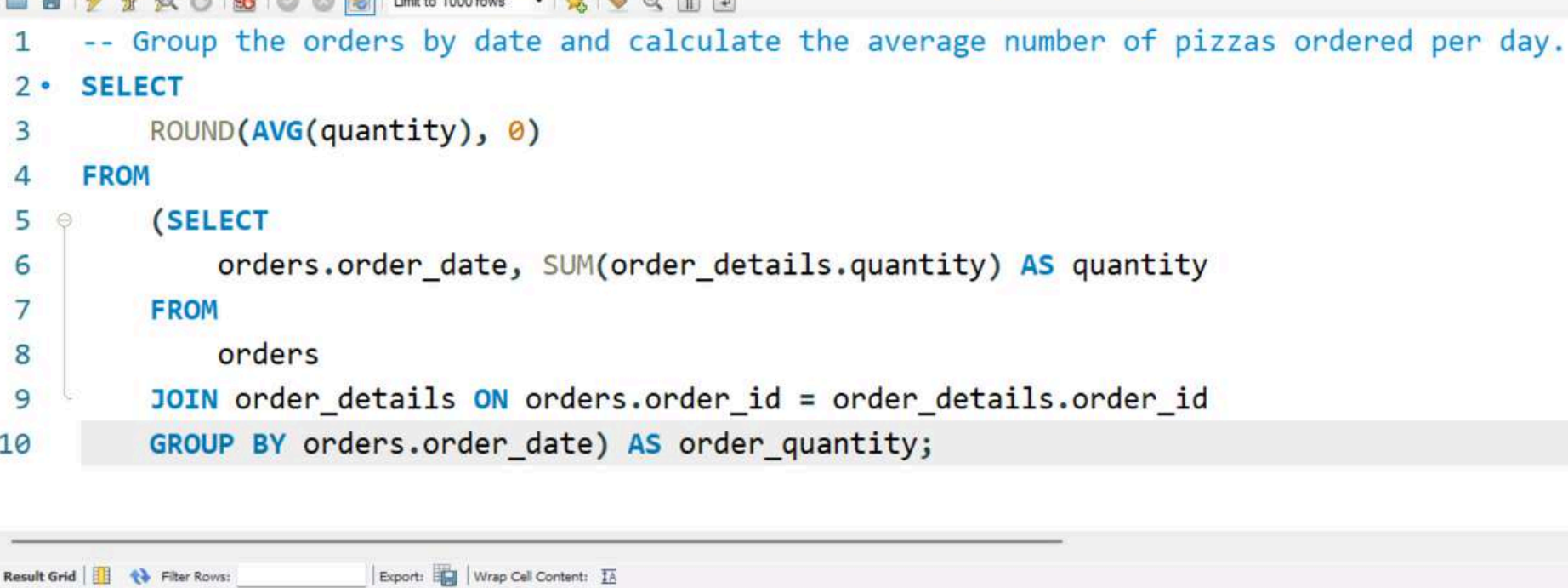
Limit to 1000 rows

1 -- Join relevant tables to find
2 -- the category-wise distribution of pizzas.
3
4 • SELECT
5 category, **count**(name)
6 **FROM**
7 pizza_types
8 **GROUP BY** category;
9

Result Grid

	category	count(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

Liceria & Co.



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

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

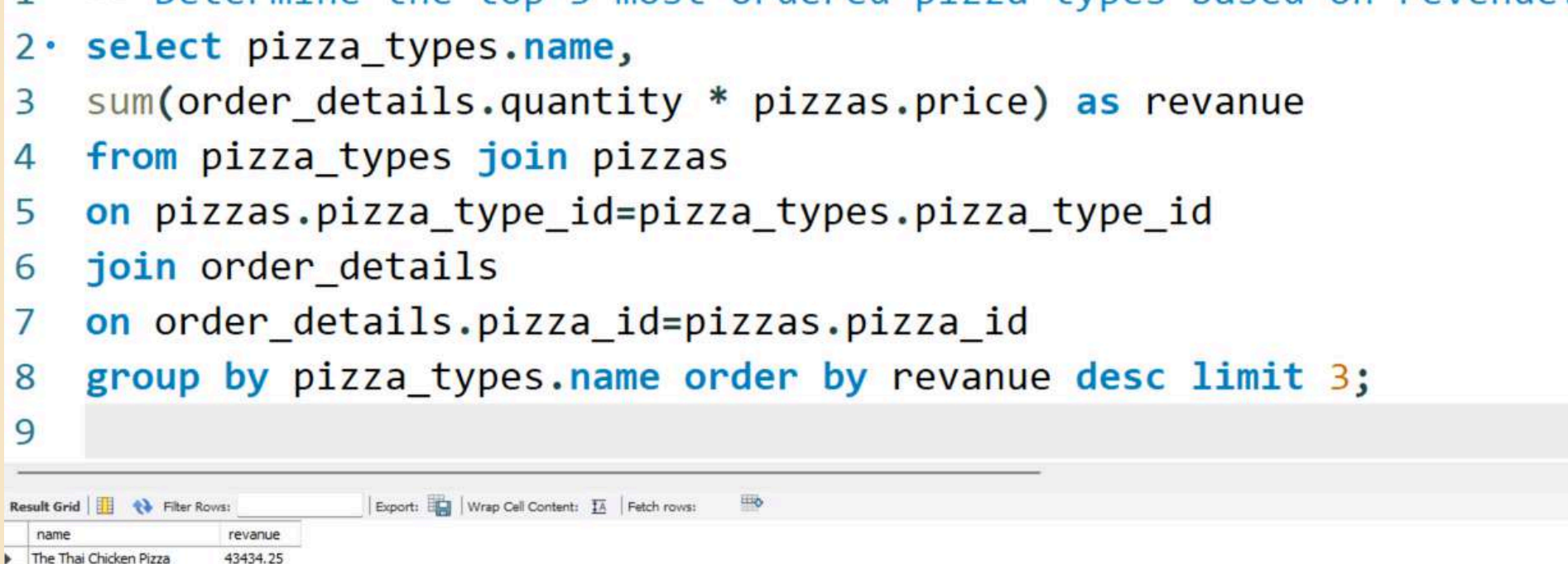
round(avg(quantity),0)
138

Home

About Us

Content

Contact Us



The screenshot shows a SQL IDE interface with multiple tabs at the top labeled 'Query 1', 'SQL File 1*', 'SQL File 2*', 'SQL File 3*', 'SQL File 4*', 'SQL File 5*', 'SQL File 6*', 'SQL File 7*', 'SQL File 8*', 'SQL File 9*', 'SQL File 10*', 'SQL File 11*', 'SQL File 12*', and 'SQL File 13*'. The 'SQL File 10*' tab is active. The main editor area contains a SQL query:

```
1 -- Determine the top 3 most ordered pizza types based on revenue.
2 • select pizza_types.name,
3    sum(order_details.quantity * pizzas.price) as revenue
4 from pizza_types join pizzas
5 on pizzas.pizza_type_id=pizza_types.pizza_type_id
6 join order_details
7 on order_details.pizza_id=pizzas.pizza_id
8 group by pizza_types.name order by revenue desc limit 3;
9
```

Below the editor, the 'Result Grid' tab is active, displaying the query results in a table:

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

Liceria & Co.

Home

About Us

Content

Contact Us

```
Query 1    SQL File 1*    SQL File 2*    SQL File 3*    SQL File 4*    SQL File 5*    SQL File 6*    SQL File 7*    SQL File 8*    SQL File 9*
-- Calculate the percentage contribution of each pizza type to total revenue.
select pizza_types.category,
round(sum(order_details.quantity * pizzas.price) / (SELECT
ROUND (SUM(order_details.quantity * pizzas.price),
2) AS total_sales
FROM
order_details
JOIN
pizzas ON pizzas.pizza_id = order_details.pizza_id) *100,2) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id= pizzas.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

Result Grid



	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68





```
Query 1  SQL File 1*  SQL File 2*  SQL File 3*  SQL File 4*  SQL File 5*  SQL File 6*  SQL File 7*  SQL File
Limit to 1000 rows

1  -- Analyze the cumulative revenue generated over time.
2  -- Show revenue growing day by day
3  •  select order_date,
4      -- Add today's revenue + all previous days = running total
5      sum(revenue) over (order by order_date) as cum_revenue
6  from (
7      -- Find money made each day
8      select orders.order_date,
9          sum(order_details.quantity * pizzas.price) as revenue
10     from order_details
11     join pizzas on order_details.pizza_id = pizzas.pizza_id
12     join orders on orders.order_id = order_details.order_id
13     group by orders.order_date
14 ) as sales;
```

Result Grid

	order_date	cum_revenue
▶	2015-01-01	2713.85000000000004
	2015-01-02	5445.75
	2015-01-03	8108.15
	2015-01-04	9863.6
	2015-01-05	11929.55
	2015-01-06	14358.5
	2015-01-07	16560.7
	2015-01-08	19399.05
	2015-01-09	21576.4

```
Query 1  SQL File 1*  SQL File 2*  SQL File 3*  SQL File 4*  SQL File 5*  SQL File 6*  SQL File 7*  SQL
 Limit to 1000 rows  
1  -- Determine the top 3 most ordered pizza types based on revenue
2
3  •  select name, revenue from
4  (select category, name, revenue,
5   rank() over(partition by category order by revenue desc) as rn --
6   from
7   (select pizza_types.category, pizza_types.name,
8    sum((order_details.quantity) * pizzas.price) as revenue
9    from pizza_types join pizzas
10   on pizza_types.pizza_type_id = pizzas.pizza_type_id
11   join order_details
12   on order_details.pizza_id = pizzas.pizza_id
13   group by pizza_types.category, pizza_types.name) as a) as b
14  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

THANK YOU

