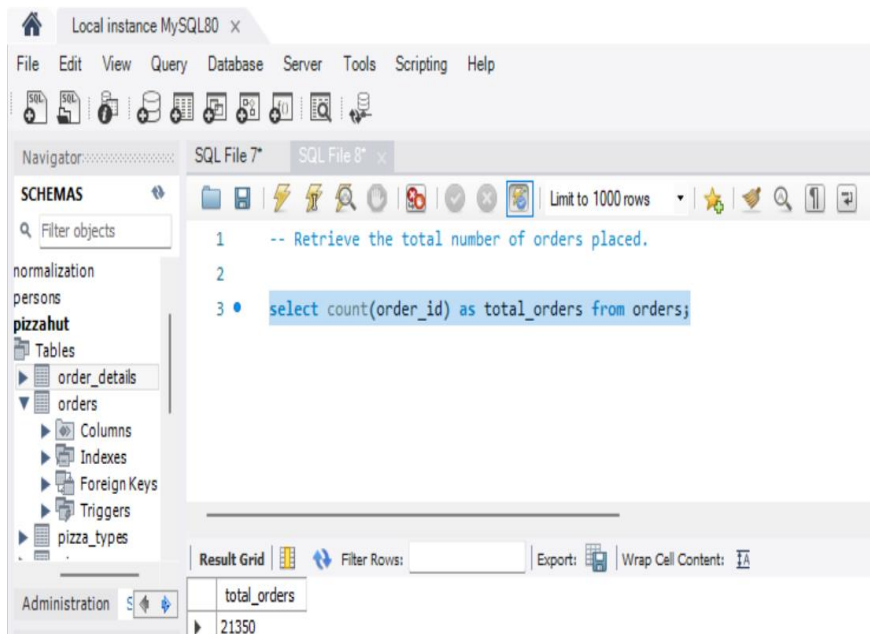


Q1. Retrieve the total number of orders placed.



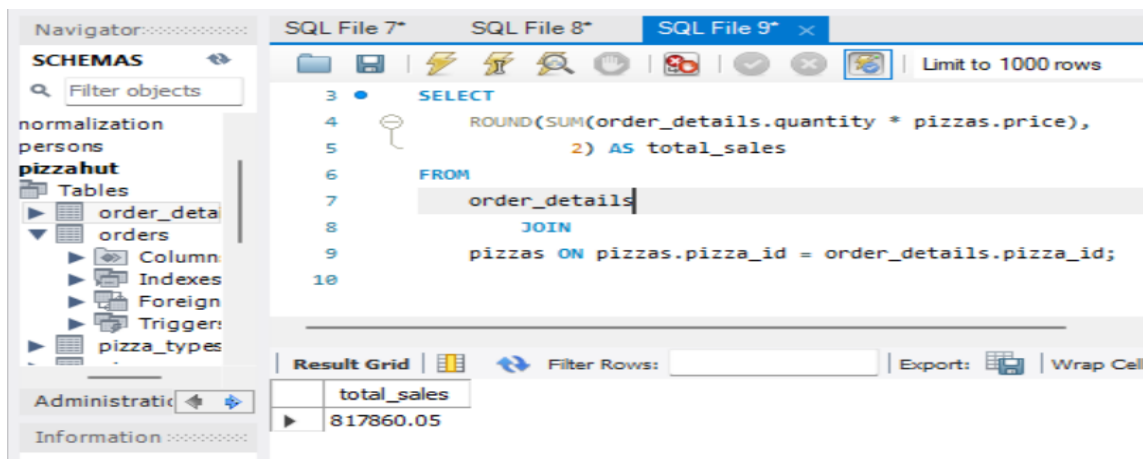
The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the 'pizzahut' database schema, including tables like 'order_details', 'orders', and 'pizza_types'. The main editor window shows a SQL query in 'SQL File 7*':

```
1 -- Retrieve the total number of orders placed.  
2  
3 select count(order_id) as total_orders from orders;
```

The 'Result Grid' at the bottom shows the query result:

total_orders
21350

Q2. Calculate the total revenue generated from pizza sales.



The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the 'pizzahut' database schema. The main editor window shows a SQL query in 'SQL File 9*':

```
3 SELECT  
4     ROUND(SUM(order_details.quantity * pizzas.price),  
5           2) AS total_sales  
6 FROM  
7     order_details  
8 JOIN  
9     pizzas ON pizzas.pizza_id = order_details.pizza_id;  
10
```

The 'Result Grid' at the bottom shows the query result:

total_sales
817860.05

Q3. Identify the highest-priced pizza.

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

```
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;
```

The results grid shows the following data:

name	price
The Greek Pizza	35.95

Q4. Identify the most common pizza size ordered.

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

```
1 -- Identify the most common pizza size ordered.
2 • select quantity, count(order_details_id)
3 from order_details group by quantity;
4
5 • SELECT
6     pizzas.size,
7     COUNT(order_details.order_details_id) AS order_count
8 FROM
9     pizzas
10    JOIN
11    order_details ON pizzas.pizza_id = order_details.pizza_id
12 GROUP BY pizzas.size
13 ORDER BY order_count DESC;
14
```

The results grid shows the following data:

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

Q5. List the top 5 most ordered pizza types along with their quantities.

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

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

The results grid shows the following data:

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

Q6. Join the necessary tables to find the total quantity of each pizza category Category ordered

The screenshot shows a SQL IDE with a query editor and a result grid. The query is as follows:

```

1  -- Join the necessary tables to find the
2  -- total quantity of each pizza category Category ordered
3  use pizzahut;
4  SELECT
5      pizza_types.category,
6      SUM(order_details.quantity) AS quantity
7  FROM
8      pizza_types
9      JOIN
10     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11     JOIN
12     order_details ON order_details.pizza_id = pizzas.pizza_id
13 GROUP BY pizza_types.category
14 ORDER BY quantity DESC;

```

The result grid shows the following data:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Q7. Determine the distribution of orders by hour of the day.

The screenshot shows a SQL IDE with a query editor and a result grid. The query is as follows:

```

1  -- Determine the distribution of orders by hour of the day.
2
3  SELECT
4      HOUR(order_time) AS Hours, COUNT(order_id) AS order_count
5  FROM
6      orders
7  GROUP BY HOUR(order_time);

```

The result grid shows the following data:

Hours	order_count
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

Q8. Join relevant tables to find the category-wise distribution of pizzas.

The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 7*', which contains the following SQL query:

```
1  -- Join relevant tables to find the
2  -- category-wise distribution of pizzas.
3
4  • select category, count(name) from pizza_types
5     group by category;
6
```

Below the query editor is the 'Result Grid' showing the results of the query:

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

Q9. Group the orders by date and calculate the average number of pizzas ordered per day.

The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 8*', which contains the following SQL query:

```
1  -- Group the orders by date and calculate the average number of pizzas ordered per day.
2
3  • SELECT
4     ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day
5  FROM
6     (SELECT
7        orders.order_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.order_date) AS order_quantity;
12
13
14
```

Below the query editor is the 'Result Grid' showing the results of the query:

avg_pizza_ordered_per_day
138

Q10. Determine the top 3 most ordered pizza types based on revenue.

The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 9*', which contains the following SQL query:

```
1  -- Determine the top 3 most ordered pizza types based on revenue.
2
3  • select pizza_types.name,
4     sum(order_details.quantity * pizzas.price) as revenue
5  from pizza_types join pizzas
6  on pizzas.pizza_type_id = pizza_types.pizza_type_id
7  join order_details
8  on order_details.pizza_id = pizzas.pizza_id
9  group by pizza_types.name order by revenue desc limit 3;
10
11
12
```

Below the query editor is the 'Result Grid' showing the results of the query:

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

Q11. Calculate the percentage contribution of each pizza type to total revenue.

The screenshot shows a SQL IDE with a query editor and a result grid. The query calculates the percentage contribution of each pizza type to total revenue by joining the `order_details`, `pizzas`, and `pizza_types` tables. It uses `round` and `sum` functions to calculate the revenue for each category and then divides it by the total sales.

```
1 -- Calculate the percentage contribution of each
2 -- pizza type to total revenue.
3 • select pizza_types.category,
4 round((sum(order_details.quantity * pizzas.price)/ (select
5 round(sum(order_details.quantity * pizzas.price),2)
6 as total_sales
7 from order_details
8 join pizzas on pizzas.pizza_id = order_details.pizza_id))*100,2) 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 order by revenue desc;
14
```

The result grid shows the following data:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Q12. Analyze the cumulative revenue generated over time.

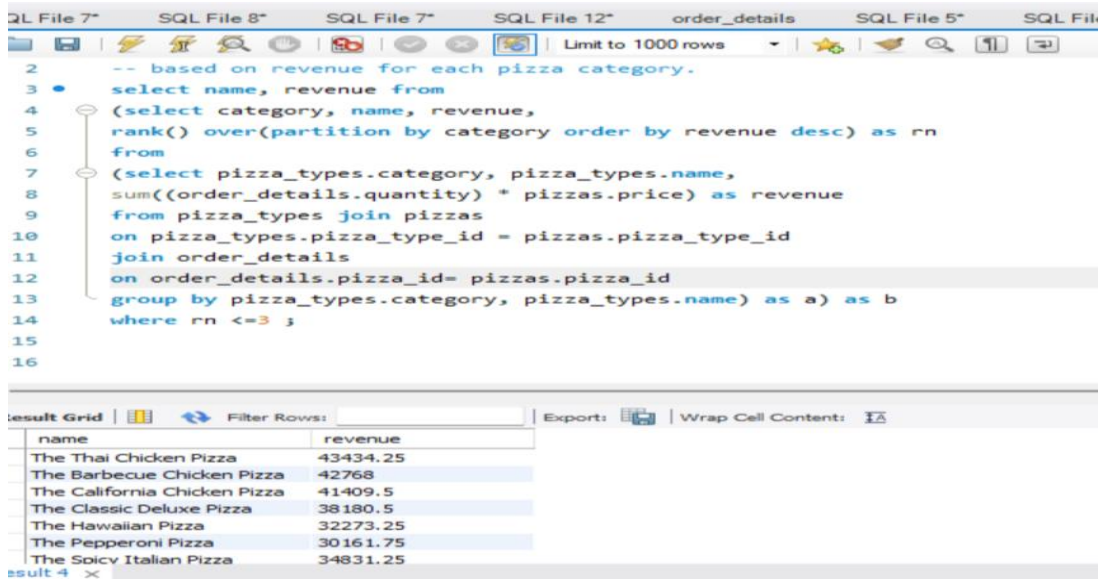
The screenshot shows a SQL IDE with a query editor and a result grid. The query analyzes the cumulative revenue generated over time by joining the `orders`, `order_details`, and `pizzas` tables. It uses `sum` and `over` functions to calculate the cumulative revenue for each order date.

```
1 -- Analyze the cumulative revenue generated over time.
2
3 • select order_date,
4 sum(revenue) over(order by order_date) as cum_revenue
5 from
6 (select orders.order_date,
7 sum(order_details.quantity * pizzas.price) as revenue
8 from order_details join pizzas
9 on order_details.pizza_id = pizzas.pizza_id
10 join orders
11 on orders.order_id = order_details.order_id
12 group by orders.order_date) as sales ;
13
```

The result grid shows the following data:

order_date	cum_revenue
2015-01-01 00:00:00	2713.85000000000004
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

Q13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.



The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 12*' with the file name 'order_details'. The query editor contains the following SQL code:

```
2  -- based on revenue for each pizza category.
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 ;
15
16
```

Below the query editor is the 'Result Grid' showing the results of the query. The grid has two columns: 'name' and 'revenue'. The results are as follows:

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

At the bottom left of the result grid, it says 'result 4 >'. The top of the result grid has a 'Filter Rows:' field and an 'Export:' button. The 'Wrap Cell Content:' option is set to 'Off'.