

Pizza_Sales

SQL PROJECT

Basic Questions:

1: Retrieve the total number of orders placed.

The screenshot shows the MySQL Workbench interface with a query editor window. The title bar says "Local instance 3306". The left sidebar shows "Administration" and "Schemas" sections with "Schemas" expanded to show "bank", "ccdb", "Entity", "new_schema", and "pizzahut". The main area has tabs "Query 1" and "SQL File 3". The "Query 1" tab contains the following SQL code:

```
1 1: Retrieve the total number of orders placed.  
2 Ans: 21350 is the total number of orders placed.  
3  
4 select count(order_id) as total_orders from orders;  
5  
6  
7  
8
```

The "Result Grid" tab shows a single row with "total_orders" and "21350". The status bar at the bottom indicates "1 error found". A context help message on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The bottom status bar also shows the query was run at 11:49:53.

2: Calculate the total revenue generated from pizza sales.

The screenshot shows the MySQL Workbench interface with a query editor and results grid.

Query Editor:

```
1 -- 2: Calculate the total revenue generated from pizza sales.
2 -- Ans: 817860.05 is the total revenue generated.
3
4 • SELECT
5   ROUND(SUM(order_details.quantity * pizzas.price),
6         2) AS total_sales
7
8   FROM
9     order_details
10    JOIN
11      pizzas ON pizzas.pizza_id = order_details.pizza_id
```

Results Grid:

total_sales
817860.05

Status Bar:

Result 1 | Read Only

Action Output

Time	Action	Response	Duration / Fetch Time
11:59:34	select round(sum(order_details.quantity * pizzas.price),2) as total_sales from or...	1 row(s) returned	0.052 sec / 0.000035...

Formatted 1 statements.

3: Identify the highest-priced pizza.

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes 'File', 'Edit', 'Tools', 'Help', and tabs for 'Local instance 3306', 'Administration', 'Schemas', 'Query 1', 'SQL File 3*', and 'SQL File 4*'. The 'Schemas' panel on the left shows the 'pizzanut' schema with tables: 'order_details', 'orders', 'pizzas', and 'pizza_types'. The central workspace contains a query editor with the following SQL code:

```
1 -- 3: Identify the highest-priced pizza.
2 -- Ans: The Greek Pizza is the highest priced pizza.
3
4 • select pizza_types.name, pizzas.price
5   from pizza_types join pizzas
6     on pizza_types.pizza_type_id = pizzas.pizza_type_id
7   order by pizzas.price desc limit 1;
```

The right side of the interface has a 'Context Help' window with the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." A vertical toolbar on the right lists 'Result Grid' (selected), 'Form Editor', 'Field Types', 'Query Stats', and 'Execution Plan'. The bottom status bar indicates 'Query Completed'.

Result Grid

name	price
The Greek Pizza	35.95

Action Output 0

	Time	Action	Response	Duration / Fetch Time
3	12:08:00	select pizza_types.name, pizzas.price from pizza_types join pizzas on pizza_typ...	1 row(s) returned	0.0018 sec / 0.00006...

4: Identify the most common pizza size ordered.

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:`-- 4: Identify the most common pizza size ordered.
-- Ans: Large pizza size is most common pizza size ordered.
--
4 • select pizzas.size, count(order_details.order_details_id) as order_count
from pizzas join order_details
on pizzas.pizza_id = order_details.pizza_id
group by pizzas.size order by order_count desc;`

The result grid shows the following data:

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

The status bar at the bottom indicates "Query Completed".

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

The screenshot shows the Oracle SQL Developer interface with the following details:

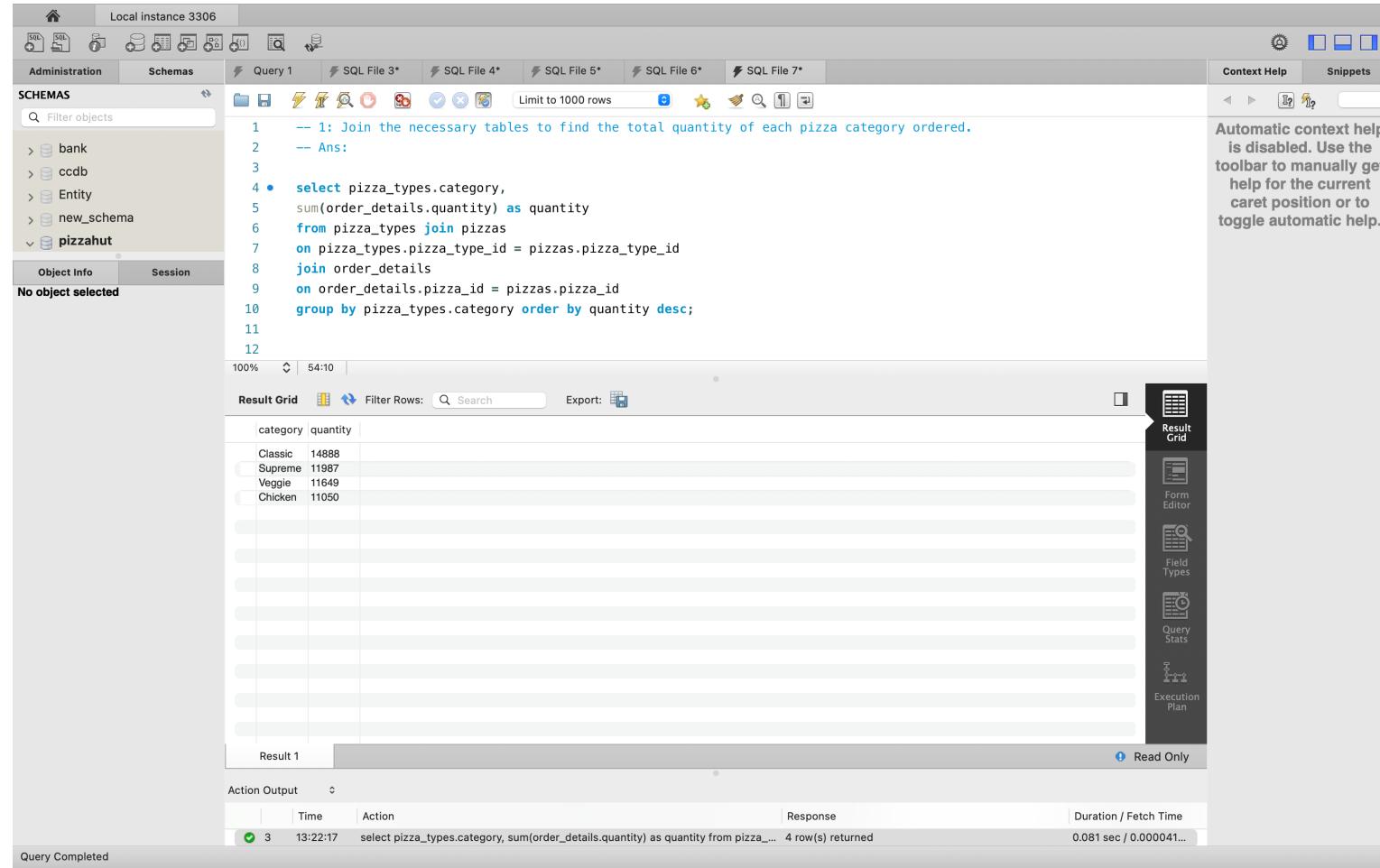
- Toolbar:** Local instance 3306, Administration, Schemas, Query 1, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, Context Help, Snippets.
- Schemas:** bank, ccdb, Entity, new_schema, pizzahut. The pizzahut schema is selected.
- Query Editor:** Contains the following SQL code:

```
1  -- 5: List the top 5 most ordered pizza types along with their quantities.
2  -- Ans: The classic deluxe pizza, barbecue chicken pizza, hawaiian pizza, pepperoni pizza and thai chicken pizza are the
3  -- top 5 most ordered pizza types.
4
5  • select pizza_types.name,
6    sum(order_details.quantity) as quantity
7  from pizza_types join pizzas
8  on pizza_types.pizza_type_id = pizzas.pizza_type_id
9  join order_details
10 on order_details.pizza_id = pizzas.pizza_id
11 group by pizza_types.name order by quantity desc limit 5;
```
- Result Grid:** Shows the results of the query in a table format.

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
- Right Panel:** Displays a message about context help being disabled and a sidebar with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.
- Bottom Status:** Action Output: 2 rows, Time: 13:03:20, Action: select pizza_types.name, sum(order_details.quantity) as quantity from pizza_typ..., Response: 5 row(s) returned, Duration / Fetch Time: 0.085 sec / 0.000032...
- Message Bar:** Query Completed

Intermediate Questions:

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



The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query editor contains the following SQL code:

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

The results grid displays the following data:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

The status bar at the bottom indicates "Query Completed".

2: Determine the distribution of orders by hour of the day.

The screenshot shows the Oracle SQL Developer interface with the following details:

- Toolbar:** Local instance 3306, Administration, Schemas, Query 1 through Query 8, Context Help, Snippets.
- Schemas:** bank, ccdb, Entity, new_schema, pizzahut. The pizzahut schema is selected.
- Object Info:** No object selected.
- Code Editor:** Contains the following SQL query:

```
1  -- 2: Determine the distribution of orders by hour of the day.
2  -- Ans:
3
4  •  SELECT
5      HOUR(order_time) AS hour, COUNT(order_id) AS order_count
6  FROM
7      orders
8  GROUP BY HOUR(order_time);
```
- Result Grid:** Shows the distribution of orders by hour of the day. The data is as follows:

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

- Action Output:** Shows the execution details:

Action	Time	Response	Duration / Fetch Time
SELECT HOUR(order_time) AS hour, COUNT(order_id) AS order_count FROM...	17:42:22	15 row(s) returned	0.015 sec / 0.000017...
- Status Bar:** Query Completed.

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

The screenshot shows the Oracle SQL Developer interface. The top navigation bar displays "Local instance 3306". The left sidebar shows the "SCHEMAS" tree with schemas: bank, ccdb, Entity, new_schema, and pizzahut. The main workspace contains a query editor with the following SQL code:

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

A tooltip on the right side of the screen states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The results grid shows the following data:

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

The bottom status bar indicates the query was completed at 17:47:54 with a duration of 0.00094 sec / 0.000... and 4 row(s) returned.

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

The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query editor contains the following SQL code:

```
-- 4: Group the orders by date and calculate the average number of pizzas ordered per day.  
-- Ans: 138 is the average number of pizzas ordered per day.  
  
4 • select round(avg(quantity),0) as avg_pizza_ordered_per_day from  
(select orders.order_date, sum(order_details.quantity) as quantity  
from orders join order_details  
on orders.order_id = order_details.order_id  
group by orders.order_date) as order_quantity ;
```

The results grid displays a single row with the value 138 under the column header "avg_pizza_ordered_per_...". A tooltip on the right side of the interface states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

At the bottom of the interface, the status bar shows the following information:

- Action Output
- Time: 18:12:06
- Action: select round(avg(quantity).0) as avg_pizza_ordered_per_day from (select orders...
- Response: 1 row(s) returned
- Duration / Fetch Time: 0.054 sec / 0.00003...

Query Completed

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

The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Includes icons for Home, Local instance 3306, SQL, Schemas, and various file operations.
- Schemas Panel:** Shows available schemas: bank, ccdb, Entity, new_schema, and pizzahut. The pizzahut schema is currently selected.
- Query Editor:** Displays the following SQL query:

```
1 -- 5: Determine the top 3 most ordered pizza types based on revenue.
2 -- Ans: The thai chicken pizza, barbecue chicken pizza and the california chicken pizza are the 3 most ordered pizza
3 -- types based on revenue.
4
5 • select pizza_types.name,
6   sum(order_details.quantity * pizzas.price) as revenue
7   from pizza_types join pizzas
8   on pizzas.pizza_type_id = pizza_types.pizza_type_id
9   join order_details
10  on order_details.pizza_id = pizzas.pizza_id
11  group by pizza_types.name order by revenue desc limit 3;
```
- Result Grid:** Shows the results of the query in a grid format:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
- Right Sidebar:** Contains a message about context help being disabled and a toolbar for manual help.
- Status Bar:** Shows "Query Completed".

Advanced Questions:

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

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

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

The results grid shows the following data:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Query stats at the bottom:

Action	Time	Action	Response	Duration / Fetch Time
1	18:48:51	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;	4 row(s) returned	0.100 sec / 0.000011...

2: Analyze the cumulative revenue generated over time.

The screenshot shows the Oracle SQL Developer interface with a dark theme. The top navigation bar includes tabs for Administration, Schemas, and Snippets. The Schemas panel on the left lists several schemas: bank, ccdb, Entity, new_schema, and pizzahut. The central workspace displays a SQL query:

```
1 -- 2: Analyze the cumulative revenue generated over time.
2 -- Ans:
3
4 • select order_date,
5   sum(revenue) over(order by order_date) as cum_revenue
6   from
7   (select orders.order_date,
8     sum(order_details.quantity * pizzas.price) as revenue
9     from order_details join pizzas
10    on order_details.pizza_id = pizzas.pizza_id
11    join orders
12    on orders.order_id = order_details.order_id
13   group by orders.order_date) as sales;
14
```

A tooltip message on the right side of the interface states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." Below the query, the Result Grid shows the output:

order_date	cum_revenue
2015-01-01	2713.850000000004
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	21526.4
2015-01-10	23990.35000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.30000000003
2015-01-14	32358.70000000004
2015-01-15	34343.50000000001
▶ 2015-01-16	36937.65000000001

The bottom status bar indicates "Query Completed".

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

The screenshot shows the Oracle SQL Developer interface with a query editor window. The query is:

```
1 -- 3: Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2 -- Ans:
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 b
15   where rn <= 3;
```

The result grid shows the following data:

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	32265.7000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

Context Help message: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Query completed at 21:34:57.