

PIZZAHUT SALES ANALYSIS (USING MY SQL)



INTRODUCTION :

This project presents a comprehensive sales analysis of a pizza business using SQL querying techniques on transactional and product data. The analysis involves joining multiple related tables such as pizza types, individual pizzas, and order details to calculate key performance metrics like revenue by pizza category and size. Advanced SQL functions including aggregation and window functions are utilized to rank pizza categories based on sales performance.

Q)1 - RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED

Schemas

Filter objects

ccdb

pizzahut

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
 - Columns
 - pizza_id
 - pizza_type_id
 - size
 - price
 - Indexes
 - Foreign Keys
 - Triggers
- Views
- Stored Procedures
- Functions

world

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SG SQL Addit...
1 -- Retrieve the total number of orders placed.
2
3 • SELECT
4 COUNT(order_id) AS total_orders
5 FROM
6 orders;
7

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

total_orders
21350

Administration Schemas Information

Table: pizza_types

Columns:

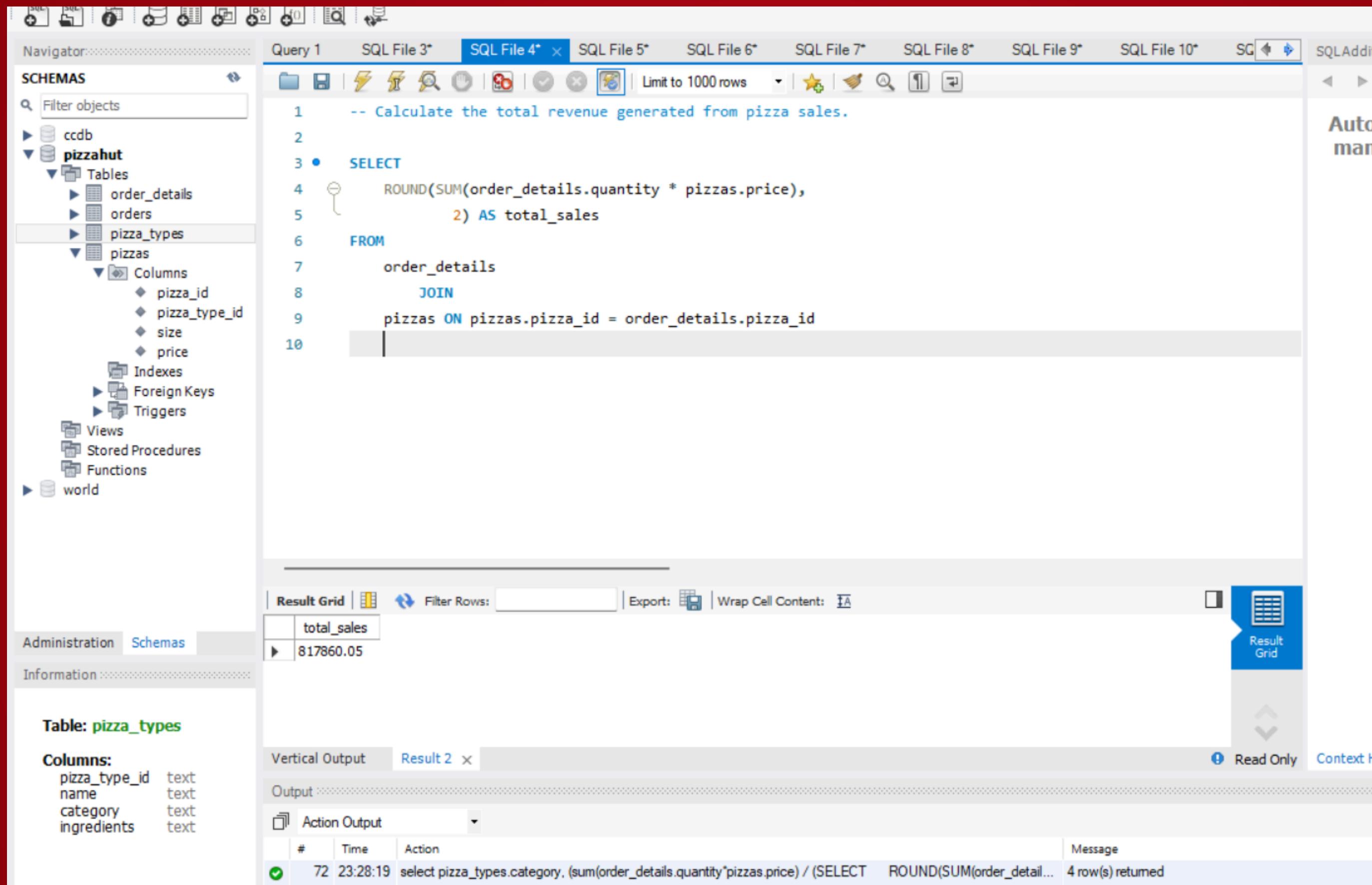
pizza_type_id	text
name	text
category	text
ingredients	text

Result 2 x Read Only Context H...

Action Output

#	Time	Action	Message
72	22-28-19	select pizza_types.category, (sum(order_details.quantity*pizzas.price)) / (SELECT ROUND(SUM(order_detail...)	4 row(s) returned

Q2) CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.



The screenshot shows a SQL development environment with the following details:

- Schemas:** The current schema is **pizzahut**, which contains tables like **order_details**, **orders**, **pizza_types**, and **pizzas**.
- Query Editor:** The query being run is:

```
-- Calculate the total revenue generated from pizza sales.  
--  
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
```
- Result Grid:** The result of the query is displayed in a grid:

total_sales
817860.05
- Message Bar:** A message at the bottom indicates "4 row(s) returned".

Q3) IDENTIFY THE HIGHEST-PRICED PIZZA.

The screenshot shows a SQL database interface with the following details:

- Navigator:** Shows the schema structure. The **pizzahut** schema is expanded, showing tables like **order_details**, **orders**, **pizza_types**, and **pizzas**. The **pizzas** table is selected.
- SQL Editor:** The current tab is **SQL File 5***. The query is:

```
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;
11
```

- Result Grid:** Shows the result of the query:

	name	price
▶	The Greek Pizza	35.95

- Table Information:** Details for the **pizza_types** table:
 - Columns:**

pizza_type_id	text
name	text
category	text
ingredients	text
- Action Output:** Shows the execution log:

#	Time	Action	Message
72	23:28:19	select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail...)	4 row(s) returned
73	23:44:42	select orders.order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_da...	Error Code: 1066. Not unique table/alias: 'orders'

Q4) IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

Schemas

Filter objects

ccdb

pizzahut

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
 - Columns
 - pizza_id
 - pizza_type_id
 - size
 - price
 - Indexes
 - Foreign Keys
 - Triggers
- Views
- Stored Procedures
- Functions

world

SCHEMAS

Navigator

Query 1 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 Addition

-- Identify the most common pizza size ordered.

1

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;

Administration Schemas

Information

Table: pizza_types

Columns:

pizza_type_id	text
name	text
category	text
ingredients	text

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Read Only Context Help

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

Vertical Output Result 1 × Output Action Output # Time Action Message

72 23:28:19 select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail... 4 row(s) returned

Q5) LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

SCHEMAS

Filter objects

ccdb

pizzahut

Tables

order_details

orders

pizza_types

pizzas

Columns

- ◆ pizza_id
- ◆ pizza_type_id
- ◆ size
- ◆ price

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

world

-- List the top 5 most ordered pizza types
-- along with their quantities.
1
2
3
4 • SELECT
5 pizza_types.name, SUM(order_details.quantity) AS 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.name
13 ORDER BY quantity DESC
14 LIMIT 5;
15
16

Administration Schemas

Information

Table: pizza_types

Columns:

	pizza_type_id	name	category	ingredients
1	text	text	text	text
2	text	text	text	text
3	text	text	text	text
4	text	text	text	text

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content: | Fetch rows: | Result Grid | Read Only | Context

	name	quantity
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

Result 1 x Output Action Output

Q6) JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

Schemas

```
1 -- Join the necessary tables to find the total quantity of each pizza category ordered.
2
3
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;
```

Administration Schemas

Information

Table: pizza_types

Columns:

pizza_type_id	text
name	text
category	text
ingredients	text

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Read Only Context Help Snippets

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Action Output

#	Time	Action	Message
72	23:28:19	select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail...)	4 row(s) returned

Q7) DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

Schemas

Filter objects

ccdb

pizzahut

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
 - Columns
 - ◆ pizza_id
 - ◆ pizza_type_id
 - ◆ size
 - ◆ price
 - Indexes
 - Foreign Keys
 - Triggers
- Views
- Stored Procedures
- Functions

world

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SG

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

1
2
3 • SELECT
4 HOUR(order_time), COUNT(order_id) AS order_count
5 FROM
6 orders
7 ✘ GROUP BY HOUR(order_time);

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Read Only Context Help Snippets

hour(order_time)	order_count
11	1231
12	2520
13	2455
14	1472
15	1468

Output

Action Output

#	Time	Action	Message
72	23:28:19	select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail...))	4 row(s) returned
73	23:44:42	select orders.order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_da...	Error Code: 1066. Not unique table/alias: 'orders'

Q8) JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

Schemas

- ccdb
- pizzahut
 - Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
 - Columns
 - pizza_id
 - pizza_type_id
 - size
 - price
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - world

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* x SG SQLAdditions

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

Automatic context help is available. You can manually get help by hovering over the icons or using the F1 key.

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

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

Result 1 x Read Only Context Help Snippets

Action Output

#	Time	Action	Message
72	23:28:19	select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail...)	4 row(s) returned
73	23:44:42	select orders.order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_da...	Error Code: 1066. Not unique table/alias: 'orders'

QS) GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

SQL Navigator: 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

SCHEMAS

- ccdb
- pizzahut
 - Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
 - Columns
 - pizza_id
 - pizza_type_id
 - size
 - price
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- world

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

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

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

round(avg(quantity),0)
138

Administration Schemas Information

Table: pizza_types

Columns:

pizza_type_id	text
name	text
category	text
ingredients	text

Result 2 x Read Only Context Help Snippets

Output

Action Output

Message

Q10) DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

Schemas

Filter objects

ccdb

pizzahut

- Tables
 - order_details
 - orders
 - pizza_types**
 - pizzas
- Columns
 - pizza_id
 - pizza_type_id
 - size
 - price
- Indexes
- Foreign Keys
- Triggers
- Views
- Stored Procedures
- Functions

world

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

SELECT pizza_types.name,
SUM(order_details.quantity * pizzas.price) AS revenue
FROM pizza_types
JOIN pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;

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

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

Administration Schemas

Information

Table: pizza_types

Columns:

pizza_type_id	text
name	text
category	text
ingredients	text

Vertical Output Result 1 X Read Only Context Help Snippets

Action Output

#	Time	Action	Message
72	23:28:19	select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail...)	4 row(s) returned
73	23:44:42	select orders.order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_da...	Error Code: 1066. Not unique table/alias: 'orders'
74	23:47:59	select order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_date, sum...	358 row(s) returned

Q 11) CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

SQL File 13*

Navigator: Filter objects

SCHEMAS

- ccdb
- pizzahut
 - Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
 - Columns
 - pizza_id
 - pizza_type_id
 - size
 - price
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - world

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

```
1   -- Calculate the percentage contribution of each pizza type to total revenue.
2
3   ● SELECT
4       pizza_types.category,
5           (SUM(order_details.quantity * pizzas.price) / (SELECT
6               ROUND(SUM(order_details.quantity * pizzas.price),
7                   2) AS total_sales
8
9               FROM
10                  order_details
11                  JOIN
12                      pizzas ON pizzas.pizza_id = order_details.pizza_id)) * 100 AS revenue
13
14   FROM
15       pizza_types
16       JOIN
17           pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
18       JOIN
19           order_details ON order_details.pizza_id = pizzas.pizza_id
20   GROUP BY pizza_types.category
21   ORDER BY revenue DESC;
```

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

	category	revenue
▶	Classic	26.90596025566967
	Supreme	25.45631126009862
	Chicken	23.955137556847287
	Veggie	23.682590927384577

Administration Schemas

Information

Tables: pizza_types

Result Grid



Q12) ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- ccdb
- pizzahut
 - Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
 - Columns
 - pizza_id
 - pizza_type_id
 - size
 - price
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- world

File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQL File 14* SQL File 15*

-- Analyze the cumulative revenue generated over time.

```
1  -- Analyze the cumulative revenue generated over time.
2
3
4 • select order_date,
5   sum(revenue) over(order by order_date) as cum_revenue
6
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;
```

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

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

order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55

Result 1 x Read Only Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
72	23:28:19	select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail...)	4 row(s) returned	0.203 sec / 0.000 sec
73	23:44:42	select orders.order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_da...	Error Code: 1066. Not unique table/alias: 'orders'	0.000 sec
74	23:47:59	select order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_date, sum...	358 row(s) returned	0.141 sec / 0.000 sec

Q13) DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

File Edit View Query Database Server Tools Scripting Help

SOL SQL Navigator File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQL File 14* SQL File 15* SQLAdditions

SCHEMAS
Filter objects
ccdb
pizzahut
Tables
order_details
orders
pizza_types
pizzas
Columns
pizza_id
pizza_type_id
size
price
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
world

```
1 -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
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;
15
```

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

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

Result 2 x Read Only Context Help Snippets

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
72	23:28:19	select pizza_types.category, (sum(order_details.quantity*pizzas.price) / (SELECT ROUND(SUM(order_detail...)	4 row(s) returned	0.203 sec / 0.000 sec
73	23:44:42	select orders.order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_da...	Error Code: 1066. Not unique table/alias: 'orders'	0.000 sec
74	23:47:59	select order_date, sum(revenue) over(order by order_date) as cum_revenue from (select orders.order_date, sum...	358 row(s) returned	0.141 sec / 0.000 sec



PIZZAHUT

THANK YOU!

