



LARANA PIZZA

# LARANA PIZZA





LARANA PIZZA

# WELCOME TO LARANA PIZZA

Hi, my name is Rajdip Ghosh and this is my first SQL Pizza Sales Project that analyzes sales data to uncover insights into customer preferences, revenue trends, and operational performance. By addressing questions of varying complexity, the project explores key metrics like total orders, revenue, top-selling pizza types, and order patterns by time and category.

Basic analyses identify popular pizza sizes, top 5 pizzas by quantity, and highest-priced items. Intermediate queries reveal category-wise distributions, hourly order trends, and revenue from top pizza types. Advanced analyses include revenue contribution percentages, cumulative revenue trends, and top-performing pizzas by category.

This project highlights the use of SQL to transform data into actionable insights, enabling better menu optimization, operational planning, and customer satisfaction strategies.





LARANA PIZZA

## RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
332      -- Retrieve the total number of orders placed.  
333 •  SELECT  
334      COUNT(order_id) AS total_orders  
335  FROM  
336      orders;  
337  
338  
339  
340  
341  
342
```

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

total_orders
21350





# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

The screenshot shows a MySQL query editor interface with the following SQL code:

```
334
335  -- Calculate the total revenue generated from pizza sales.
336 •  SELECT
337      ROUND(SUM(orders_details.quantity * pizzas.price),
338            2) AS total_sales
339  FROM
340      orders_details
341      JOIN
342          pizzas ON orders_details.pizza_id = pizzas.pizza_id;
343
344
```

The result grid displays one row with the column 'total\_sales' containing the value '817860.05'.

total_sales
817860.05



LARANA PIZZA

# IDENTIFY THE HIGHEST-PRICED PIZZA.

```
348      -- Identify the highest-priced pizza.
349 •  SELECT
350      pizza_types.name, pizzas.price
351  FROM
352      pizza_types
353      JOIN
354      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
355  ORDER BY pizzas.price DESC
356  LIMIT 1;
357

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


|   | name            | price |
|---|-----------------|-------|
| ▶ | The Greek Pizza | 35.95 |


```



LARANA PIZZA

# IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

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



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



THE CLASSIC DELUXE PIZZA

SCHEMAS

Filter objects

- amazon
- college
- pizza\_hut**
- Tables
  - orders
  - orders\_details
  - pizza\_types
  - pizzas
- Views
- Stored Procedures
- Functions
- practiceq3
- sakila
- sys
- world

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

373 • SELECT

374 pizza\_types.name, SUM(orders\_details.quantity) AS quantity

375 FROM

376 pizza\_types

377 JOIN

378 pizzas ON pizza\_types.pizza\_type\_id = pizzas.pizza\_type\_id

379 JOIN

380 orders\_details ON orders\_details.pizza\_id = pizzas.pizza\_id

381 GROUP BY pizza\_types.name

382 ORDER BY quantity DESC

383 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



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



```
386  --- Join the necessary tables to find the total quantity of each pizza category ordered.
387 •   SELECT
388     pizza_types.category,
389     SUM(orders_details.quantity) AS total_quantity
390   FROM
391     pizza_types
392       JOIN
393     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
394       JOIN
395     orders_details ON orders_details.pizza_id = pizzas.pizza_id
396   GROUP BY pizza_types.category
397   ORDER BY total_quantity DESC;
```

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

category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050





LARANA PIZZA

## DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```
398
399 -- Determine the distribution of orders by hour of the day
400 • SELECT
401     HOUR(order_time) AS hours, COUNT(order_id) AS order_count
402 FROM
403     orders
404 GROUP BY HOUR(order_time);
405
406
407
```

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

hours	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1070

Result 23 ×





LARANA PIZZA

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

```
407  -- Join relevant tables to find the category-wise distribution of pizzas.  
408 • SELECT  
409     category, COUNT(pizza_type_id)  
410   FROM  
411     pizza_types  
412   GROUP BY category;  
413  
414  
415  
416
```

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

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

Result 26 ×



LARANA PIZZA

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

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

- Toolbar:** Includes icons for file operations, search, and connection management.
- Query Editor:** Displays the following SQL code:

```
415
416    -- Group the orders by date and calculate the average number of pizzas ordered per day
417 •  SELECT
418        ROUND(AVG(quantity), 0) as avg_pizzas_ordered_per_day
419    FROM
420        (SELECT
421            orders.order_date, SUM(orders_details.quantity) AS quantity
422        FROM
423            orders
424        JOIN orders_details ON orders.order_id = orders_details.order_id
425        GROUP BY orders.order_date) AS order_quantity;
426
```
- Result Grid:** Shows the output of the query:

avg_pizzas_ordered_per_day
138



LARANA PIZZA

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

The screenshot shows a MySQL Workbench interface with a query editor and a result grid. The query is:

```
428      -- Determine the top 3 most ordered pizza types based on revenue.
429 •  SELECT
430      pizza_types.name,
431      SUM(orders_details.quantity * pizzas.price) AS revenue
432  FROM
433      pizza_types
434      JOIN
435      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
436      JOIN
437      orders_details ON orders_details.pizza_id = pizzas.pizza_id
438  GROUP BY pizza_types.name
439  ORDER BY revenue DESC
440  LIMIT 3;
```

The result grid displays the following data:

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



LARANA PIZZA

## CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```
444  -- Calculate the percentage contribution of each pizza type to total revenue.
445
446 • SELECT
447     pizza_types.category,
448     ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
449         ROUND(SUM(orders_details.quantity * pizzas.price),
450             2)
451     FROM
452         orders_details
453     JOIN
454         pizzas ON pizzas.pizza_id = orders_details.pizza_id) * 100,
455         2) AS revenue
456     FROM
457     pizza_types
458     JOIN
459         pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
460     JOIN
461         orders_details ON orders_details.pizza_id = pizzas.pizza_id
462     GROUP BY pizza_types.category
463     ORDER BY revenue DESC;
```

Result Grid		Filter Rows:
	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68



LARANA PIZZA

# ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
465
466  -- Analyze the cumulative revenue generated over time.
467 • select order_date,
468    sum(revenue) over(order by order_date) as cumm_revenue
469  from
470  (select orders.order_date, sum(orders_details.quantity * pizzas.price) as revenue
471   from orders_details
472   join pizzas
473   on orders_details.pizza_id = pizzas.pizza_id
474   join orders
475   on orders.order_id = orders_details.order_id
476   group by orders.order_date) as sales;
```

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

order_date	cumm_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 46 ×





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

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

	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 ×

Output



LARANA PIZZA

# THANK YOU!

