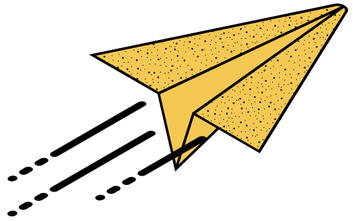
Presented by Shiksha Lahre

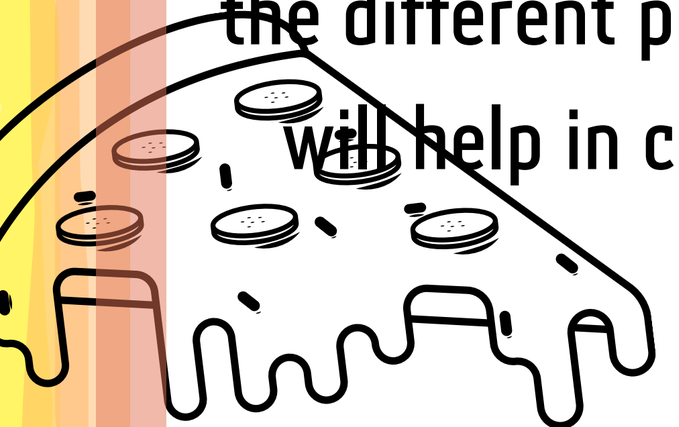


SQLIZZA ANALYSIS

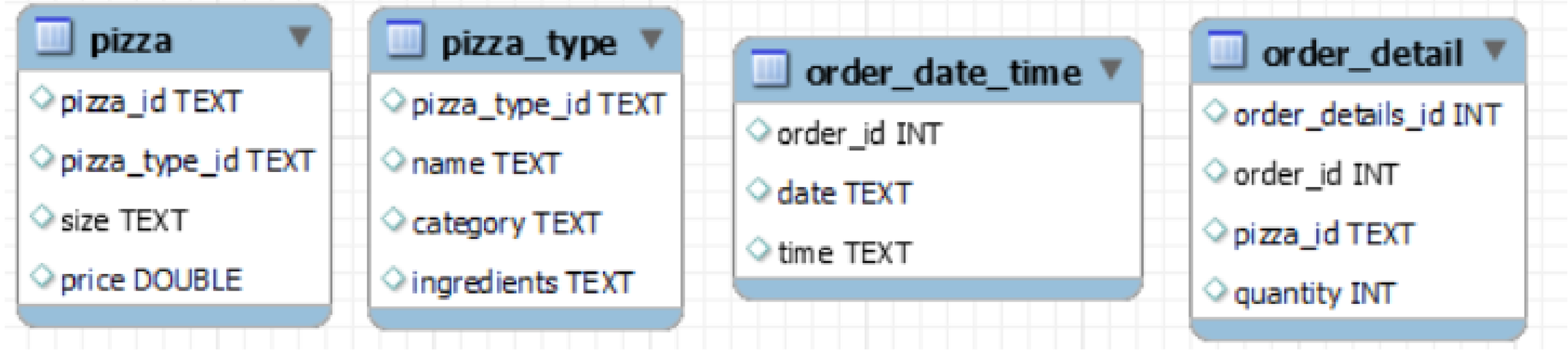
Analyzing company pizza types and sales by SQL

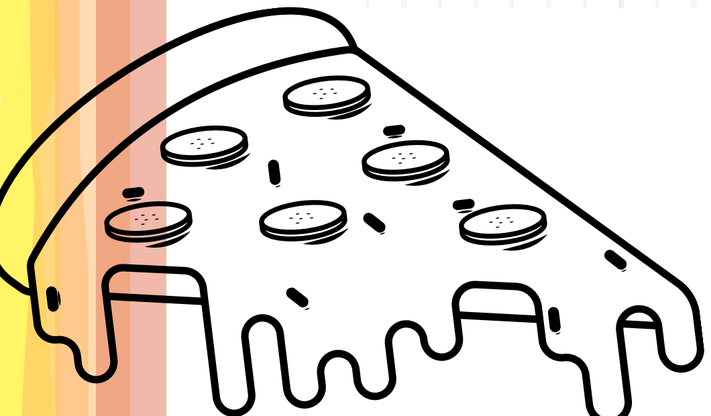
PROJECT PREVIEW

Mr. Shah, the owner of Flame and Crust shop, is currently facing difficulty in analyzing their pizza sales due to a medical emergency with their analyst employee. However, they need to submit their report to the regional owner, who will soon be visiting the shop. As a data analyst, I will assist Mr. Shah in arranging some of the important questions about the different pizza types and the revenue generated by their sales. This will help in creating a comprehensive report for the regional owner.



ENHANCED ENTITY-RELATIONSHIP

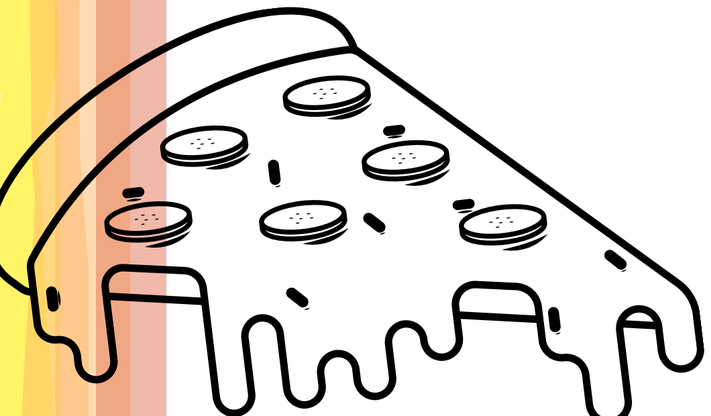




DATASET

pizza

1) pizza_id: Unique identifier of the pizza
2) pizza_type_id: Identifier linking to pizza_type table
3) size: size of pizza (i.e. small, medium, large)
4) price: price of the pizza



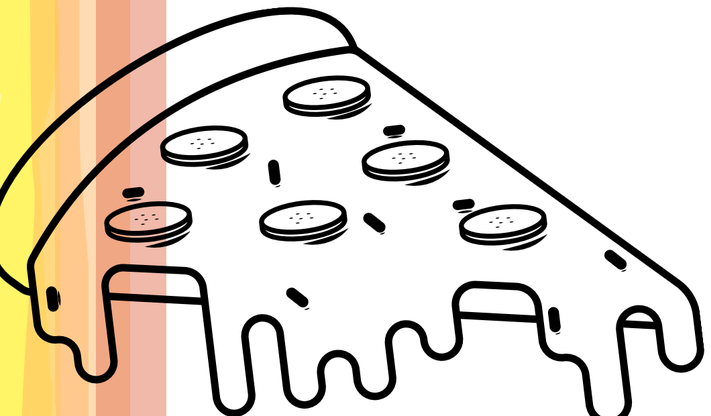
pizza_type

1) pizza_type_id: Unique identifer for the pizza type
2) name: Regular name of the pizza
3) category: Category of pizza (vegeterian, meat, etc.)
4) ingredients: Ingredients in a particular pizza

DATASET

Order_details

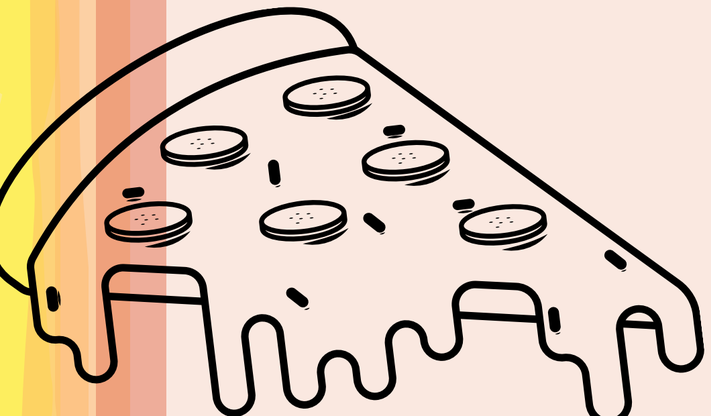
1) order_detail_id: Unique identifier of each order
2) order_id: identifier linking to Order_date_time table
3) pizza_id: Identifier linking to pizza table
4) quantity: Number of pizza ordered



Order_date_time

1) order_id: Unique identifier of the order
2) date: date of the order made
3) time: time of the order

QUERIES
&
OUTPUT

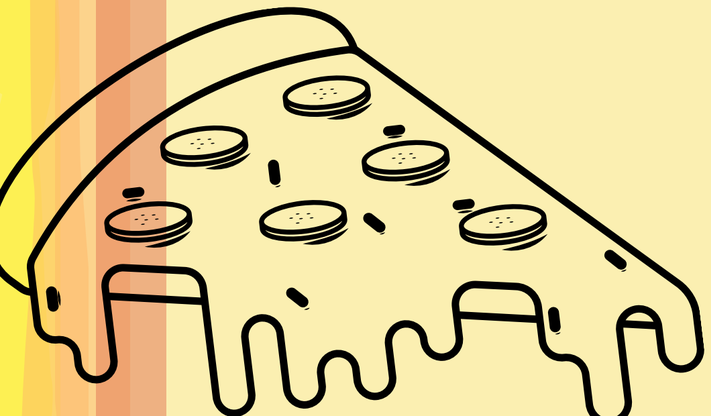


Query:

```
# Total number of pizzas placed
SELECT COUNT(*) AS Total_pizzas_placed
FROM order_date_time;
```

Output:

	Total_pizzas_placed
▶	21350

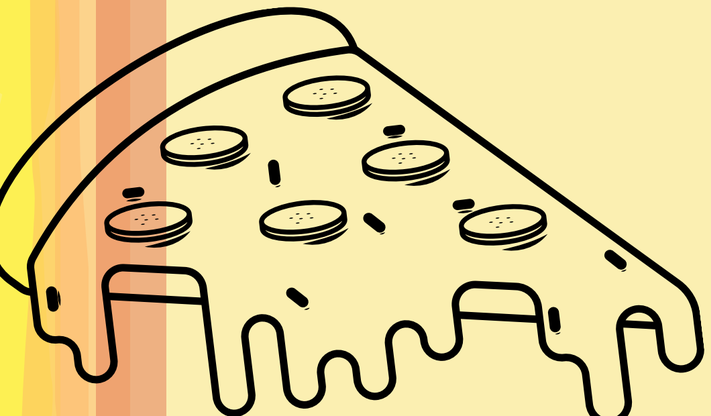


Query:

```
#Finding the first and last order date of the pizza in the record
SELECT MIN(Date) AS `First order on`,
        MAX(Date) AS `Last order on`
FROM Order_date_time;
```

Output:

	First order on	Last order on
▶	01-01-2015	31-12-2015

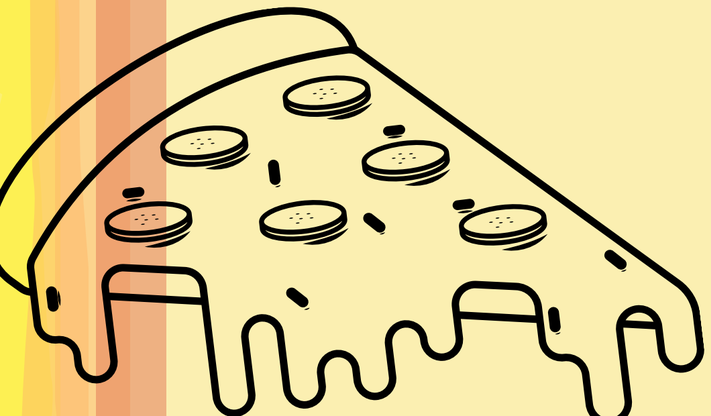


Query:

```
# List all the available pizza types
SELECT DISTINCT pizza_type_id
FROM pizza;
```

Output:

pizza_type_id
bbq_ckn
cali_ckn
ckn_alfredo
ckn_pesto
southw_ckn
thai_ckn
big_meat
classic_dlx
hawaiian
ital_cpclo
napolitana
pep_msh_pep
pepperoni
the_greek
calabrese
ital_supr
peppr_salami
prsc_argla
sicilian
soppressata
spicy_ital
spinach_supr
five_cheese
four_cheese
green_garden
ital_veggie
mediterraneo
mexicana
spin_pesto
spinach_fet
veggie_veg

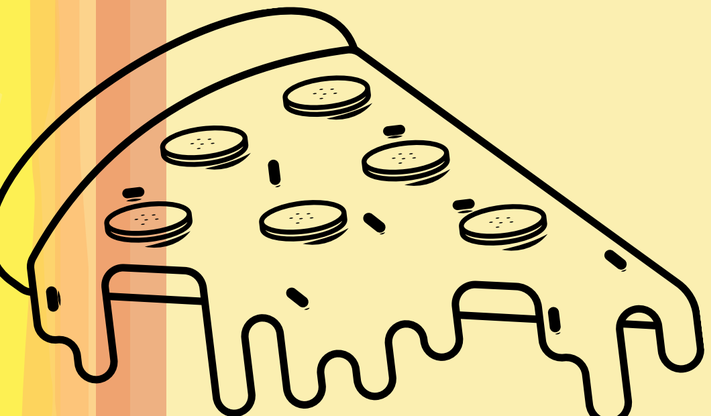


Query:

```
# List all the pizza sizes available
SELECT DISTINCT size
FROM pizza;
```

Output:

	size
▶	S
	M
	L
	XL
	XXL

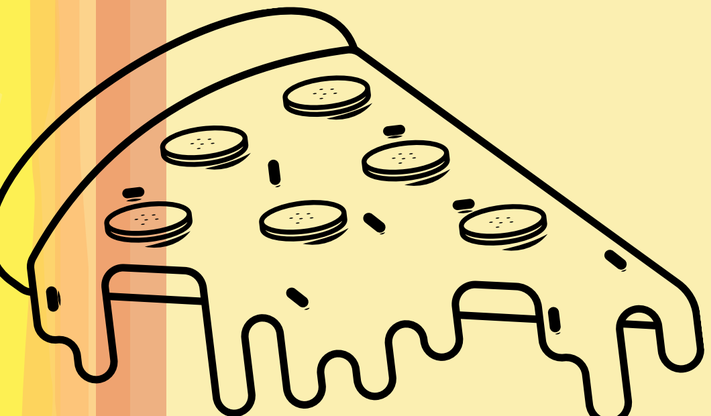


Query:

```
#List the maximum and minimum price of the pizza in Dollars
SELECT CONCAT('$'," ",MAX(price)) AS Maximum_price,
       CONCAT('$'," ",MIN(price)) AS Minimum_price
FROM pizza;
```

Output:

	Maximum_price	Minimum_price
▶	$ 35.95	$ 9.75

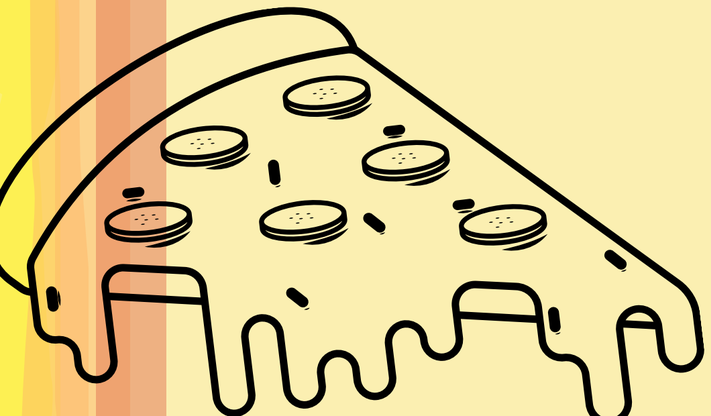


Query:

```
#Minimum and Maximum price pizza of each size
SELECT size,
        CONCAT("$ ",MAX(price)) AS Maximum_price,
        CONCAT("$ ",MIN(price)) as Minimum_price
FROM pizza
GROUP BY size
ORDER BY FIELD(size, 'S', 'M', 'L', 'XL', 'XXL');
```

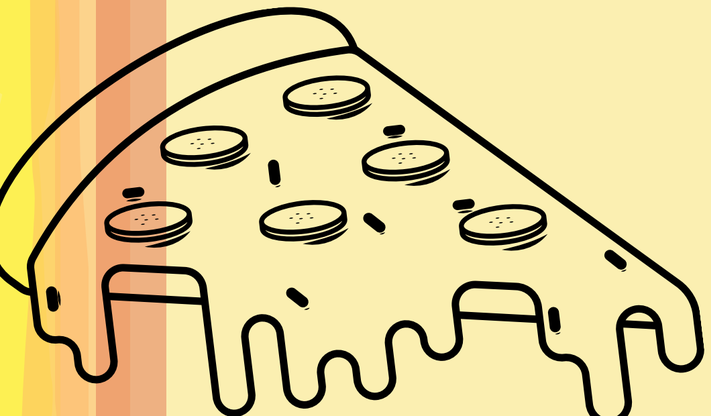
Output:

size	Maximum_price	Minimum_price
S	$ 23.65	$ 9.75
M	$ 16.75	$ 12.5
L	$ 21	$ 15.25
XL	$ 25.5	$ 25.5
XXL	$ 35.95	$ 35.95



Query:

```
#list prices of pizza available for each pizza type
SELECT pizza_type_id,
        GROUP_CONCAT(DISTINCT CONCAT(" $"," ",price)
                     ORDER BY price ASC SEPARATOR ',')
                     AS `Range of Prices`
FROM pizza
GROUP BY pizza_type_id
ORDER BY pizza_type_id;
```



Output:

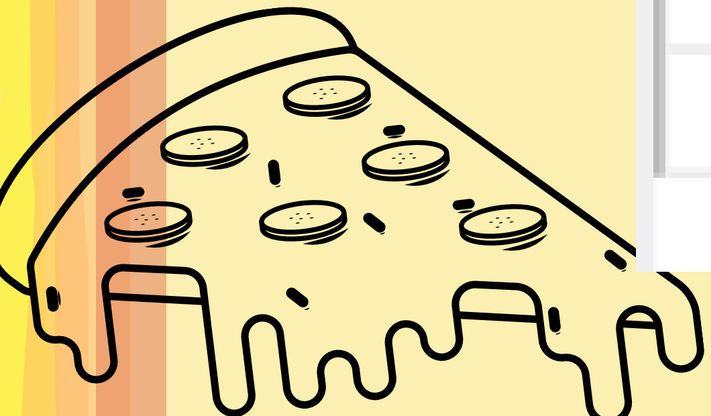
pizza_type_id	Range of Prices
bbq_ckn	$ 12.75, $ 16.75, $ 20.75
big_meat	$ 12, $ 16, $ 20.5
brie_carre	$ 23.65
calabrese	$ 12.25, $ 16.25, $ 20.25
cali_ckn	$ 12.75, $ 16.75, $ 20.75
ckn_alfredo	$ 12.75, $ 16.75, $ 20.75
ckn_pesto	$ 12.75, $ 16.75, $ 20.75
classic_dlx	$ 12, $ 16, $ 20.5
five_cheese	$ 12.5, $ 15.5, $ 18.5
four_cheese	$ 11.75, $ 14.75, $ 17.95
green_garden	$ 12, $ 16, $ 20.25
hawaiian	$ 10.5, $ 13.25, $ 16.5
ital_cpcllo	$ 12, $ 16, $ 20.5
ital_supr	$ 12.5, $ 16.5, $ 20.75
ital_veggie	$ 12.75, $ 16.75, $ 21
mediterraneo	$ 12, $ 16, $ 20.25
mexicana	$ 12, $ 16, $ 20.25
napolitana	$ 12, $ 16, $ 20.5
pep_msh_pep	$ 11, $ 14.5, $ 17.5
pepperoni	$ 9.75, $ 12.5, $ 15.25
peppr_salami	$ 12.5, $ 16.5, $ 20.75
prsc_argla	$ 12.5, $ 16.5, $ 20.75
sicilian	$ 12.25, $ 16.25, $ 20.25
soppressata	$ 12.5, $ 16.5, $ 20.75
southw_ckn	$ 12.75, $ 16.75, $ 20.75

Query:

```sql
# Find pizza types with prices significantly higher (30%) than the average price of pizzas of the same size.
SELECT pizza_type_id, price
FROM pizza
WHERE price > (SELECT AVG(price) FROM pizza) * 1.3;
```

Output:

pizza_type_id	price
the_greek	25.5
the_greek	35.95
brie_carre	23.65

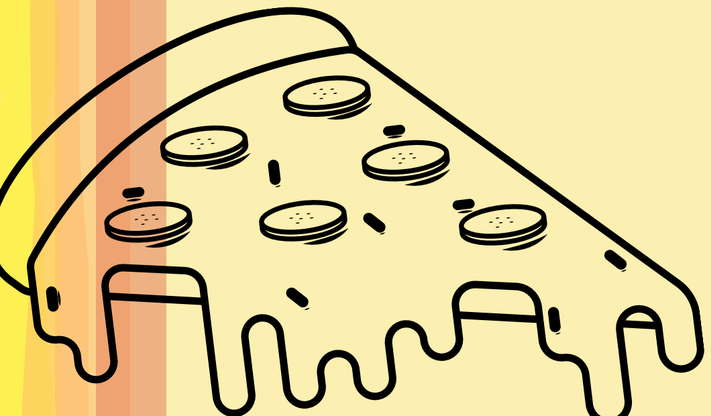


Query:

```
#Find all unique categories of pizzas
SELECT DISTINCT category
FROM pizza_type;
```

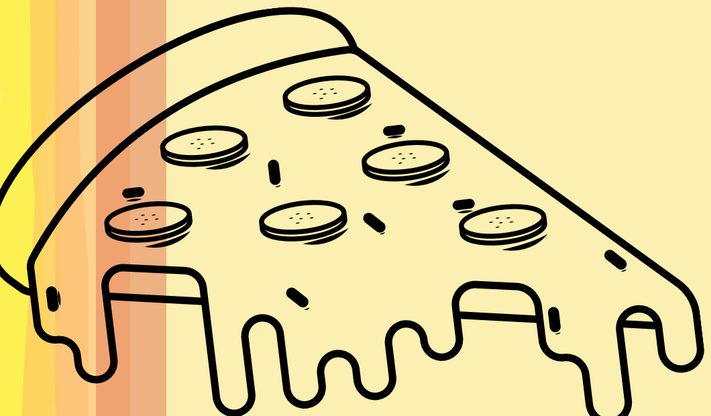
Output:

category
Chicken
Classic
Supreme
Veggie



Query:

```sql
#Finding if there is any common ingredient present in all pizza
SELECT TRIM(ingredient) AS common_ingredient
FROM (
    SELECT
        t.pizza_type_id,
        TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(t.ingredients, ',', numbers.n), ',', -1)) AS ingredient
    FROM
        pizza_type t
    JOIN (
        SELECT 1 AS n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5
        UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9 UNION ALL SELECT 10
    ) numbers ON CHAR_LENGTH(t.ingredients) - CHAR_LENGTH(REPLACE(t.ingredients, ',', '')) >= numbers.n - 1
) AS ingredients_list
GROUP BY ingredient
HAVING COUNT(DISTINCT pizza_type_id) = (SELECT COUNT(*) FROM pizza_type);
```



Output:

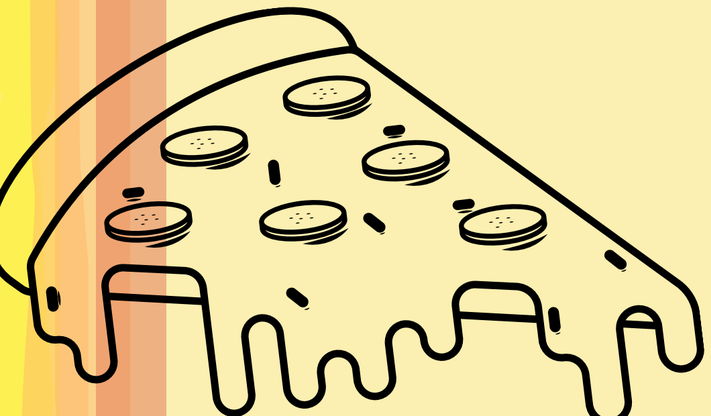
	common_ingredient

Query:

```
#Find the date with maximum number of orders;
SELECT date,
        COUNT(date) AS `Number of Orders`
FROM Order_date_time
GROUP BY date ORDER BY `Number of Orders` DESC
LIMIT 1;
# it is the day after thanksgiving celebrated as black friday. Day for spreading random act of kindness; here donating pizza
```

Output:

	date	Number of Orders
▶	27-11-2015	115

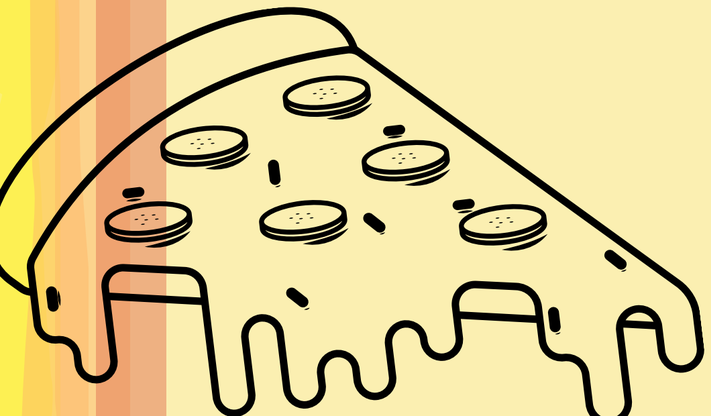


Query:

```sql
#Calculate total quantity of each type of pizza
SELECT p.pizza_id,
    pt.name AS pizza_type,
    SUM(od.quantity) AS total_quantity
FROM order_detail od
JOIN pizza p
ON od.pizza_id = p.pizza_id
JOIN pizza_type pt
ON p.pizza_type_id = pt.pizza_type_id
GROUP BY p.pizza_id, pt.name
ORDER BY total_quantity DESC;
```

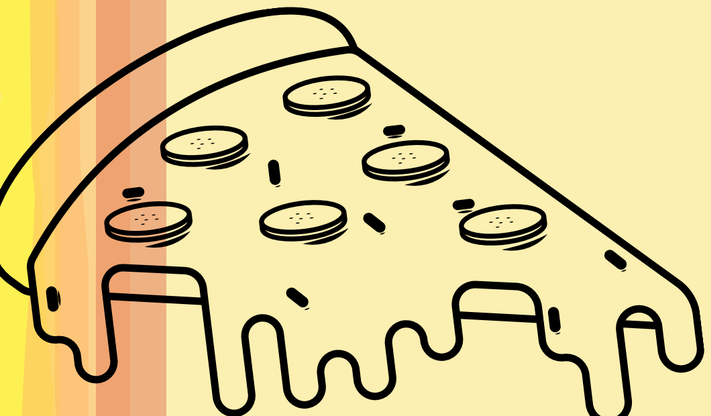
Output:

pizza_id	pizza_type	total_quantity
big_meat_s	The Big Meat Pizza	1914
thai_ckn_l	The Thai Chicken Pizza	1410
five_cheese_l	The Five Cheese Pizza	1409
four_cheese_l	The Four Cheese Pizza	1316
classic_dlx_m	The Classic Deluxe Pizza	1181
spicy_ital_l	The Spicy Italian Pizza	1109
hawaiian_s	The Hawaiian Pizza	1020
southw_ckn_l	The Southwest Chicken Pizza	1016
bbq_ckn_l	The Barbecue Chicken Pizza	992
bbq_ckn_m	The Barbecue Chicken Pizza	956
cali_ckn_m	The California Chicken Pizza	944
ital_supr_m	The Italian Supreme Pizza	941
pepperoni_m	The Pepperoni Pizza	939
cali_ckn_l	The California Chicken Pizza	927
hawaiian_l	The Hawaiian Pizza	919
mexicana_l	The Mexicana Pizza	867
classic_dlx_s	The Classic Deluxe Pizza	799
sicilian_s	The Sicilian Pizza	751
pepperoni_s	The Pepperoni Pizza	751
ital_supr_l	The Italian Supreme Pizza	747
ital_cpclo_l	The Italian Capocollo Pizza	732
pepperoni_l	The Pepperoni Pizza	728
ckn_alfredo_m	The Chicken Alfredo Pizza	703
peppr_salami_l	The Pepper Salami Pizza	696



Query:

```sql
#List Pizzas with the Most Popular Size
SELECT p.size,
    COUNT(*) AS popularity
FROM pizza p
JOIN order_detail od
ON p.pizza_id = od.pizza_id
GROUP BY p.size
ORDER BY popularity DESC;
```



Output:

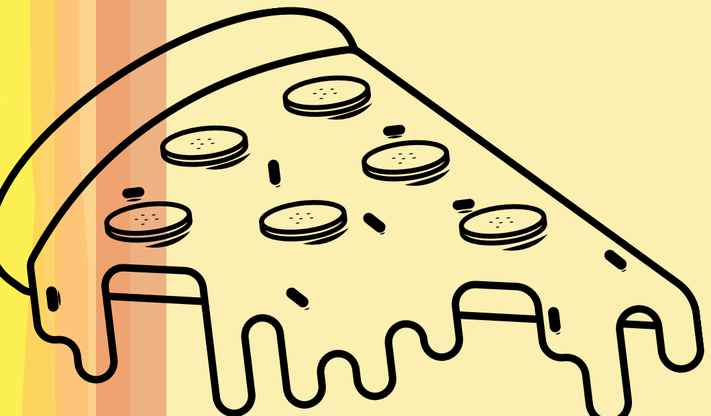
size	popularity
L	18526
M	15385
S	14137
XL	544
XXL	28

Query:

```sql
#Calculate average revenue per order
SELECT ROUND(AVG(order_revenue),3) AS average_revenue_per_order
FROM (SELECT od.order_id,
             SUM(p.price * od.quantity) AS order_revenue
    FROM order_detail od
    JOIN pizza p
    ON od.pizza_id = p.pizza_id
    GROUP BY od.order_id
) AS order_revenues;
```

Output:

average_revenue_per_order
38.307

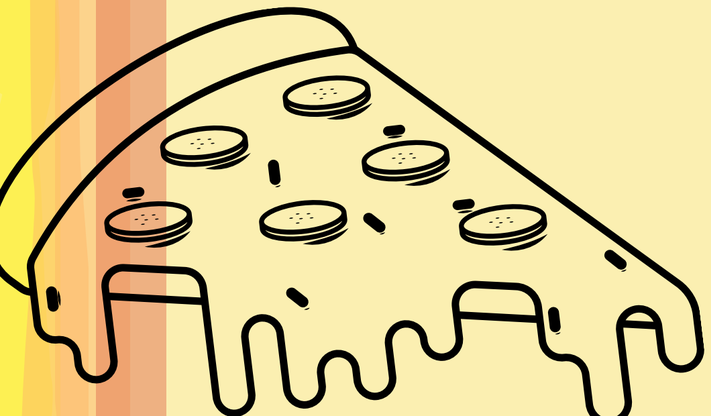


Query:

```
#Total revenue generated from pizza sales
SELECT CONCAT('$ ',ROUND(SUM(o.quantity*p.price))) AS Total_revenue_generated
FROM Order_detail o
JOIN  pizza p
ON o.pizza_id=p.pizza_id;
```

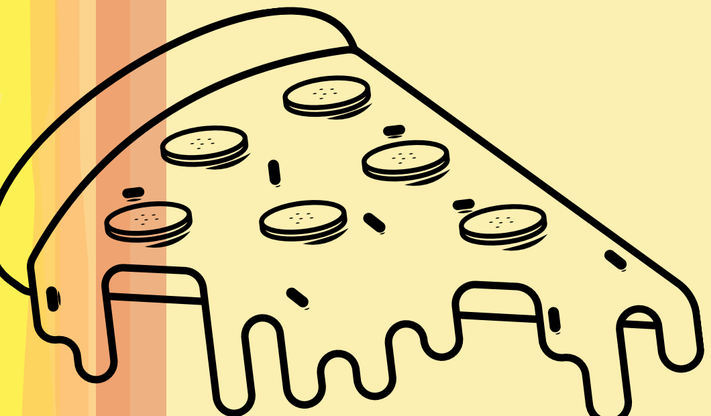
Output:

Total_revenue_generated
$ 817860



Query:

```sql
# The highest price pizza
SELECT pt.name,
        p.price
FROM pizza_type pt
JOIN pizza p
ON pt.pizza_type_id=p.pizza_type_id
WHERE p.price=(SELECT MAX(price)
FROM pizza);
```



Output:

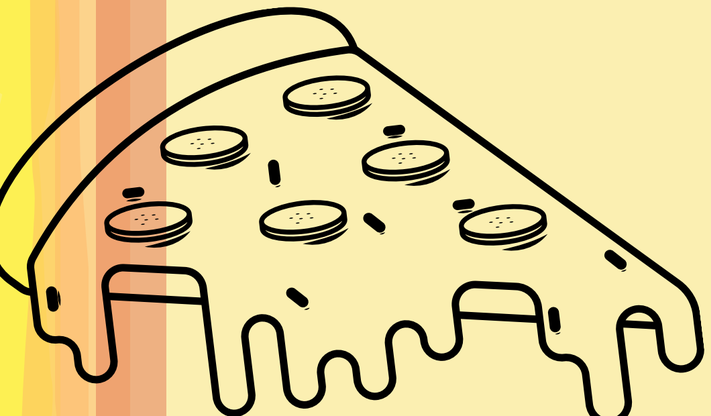
	name	price
▶	The Greek Pizza	35.95

Query:

```sql
#The top 5 most ordered pizza type along with their quantity
SELECT pt.name, SUM(o.quantity) AS Total_quantity
FROM pizza p
JOIN Order_detail o
ON p.pizza_id=o.pizza_id
JOIN pizza_type pt
on pt.pizza_type_id=p.pizza_type_id
GROUP BY pt.name
ORDER BY `Total_quantity`DESC
LIMIT 5;
```

Output:

name	Total_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

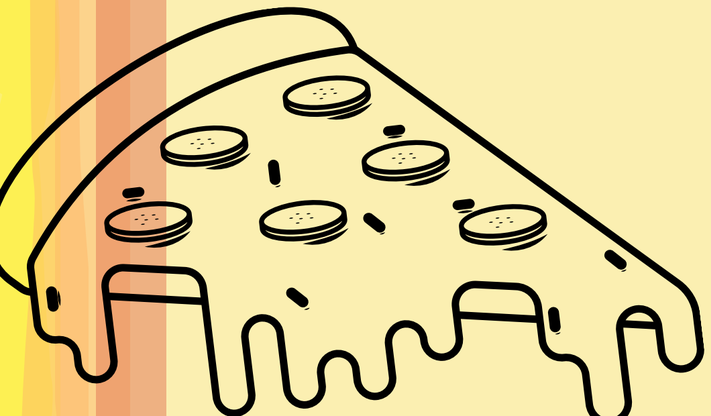


Query:

```sql
#Quantity of each pizza category sold
SELECT pt.category, SUM(o.quantity) AS Total_quantity
FROM Order_detail o
JOIN pizza p
ON o.pizza_id=p.pizza_id
JOIN pizza_type pt
ON pt.pizza_type_id=p.pizza_type_id
GROUP BY pt.category
ORDER BY Total_quantity DESC;
```

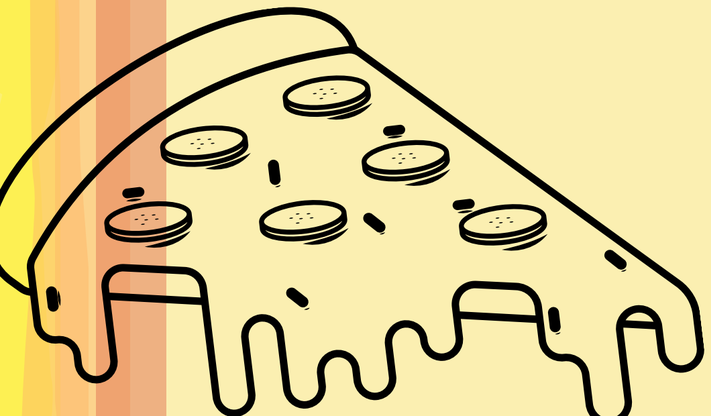
Output:

category	Total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050



Query:

```sql
#Distribution of orders by hours of the day
SELECT HOUR(time) AS `Hour of the day`,
        COUNT(DISTINCT order_id) AS `No. of Orders`
FROM Order_date_time
GROUP BY HOUR(time)
ORDER BY `No. of Orders` DESC;
#It shows mostly order has been done around lunch time.
```



Output:

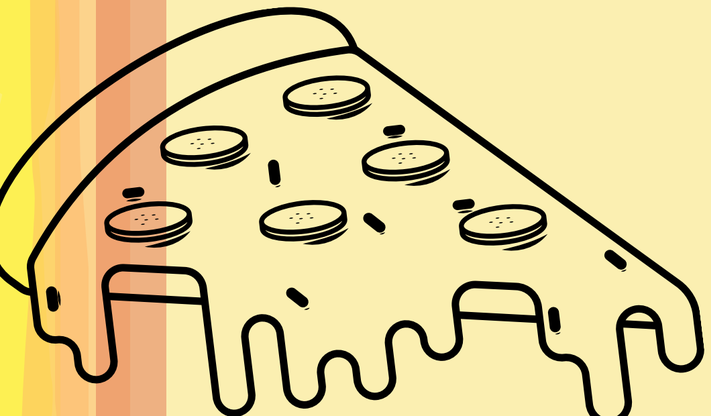
	Hour of the day	No. of Orders
▶	12	2520
	13	2455
	18	2399
	17	2336
	19	2009
	16	1920
	20	1642
	14	1472
	15	1468
	11	1231
	21	1198
	22	663
	23	28
	10	8
	9	1

Query:

```
#Average pizza ordered per day
SELECT ROUND(AVG(total_quantity)) AS average_orders_per_day FROM
(SELECT o.date, SUM(od.quantity) AS total_quantity
FROM Order_detail od
JOIN Order_date_time o
ON o.order_id = od.order_id
GROUP BY o.date) AS Daily_average_orders;
```

Output:

average_orders_per_day
138

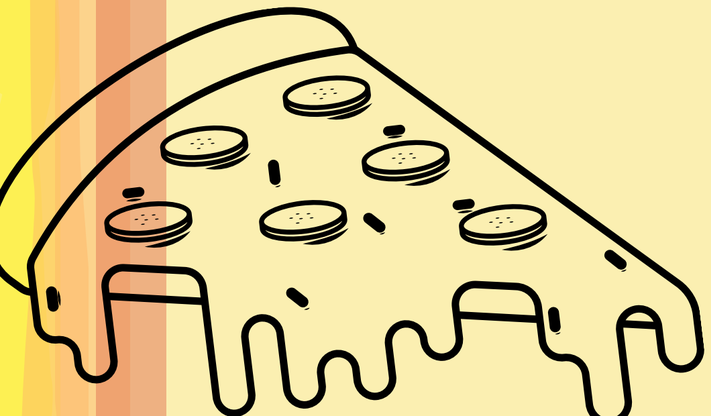


Query:

```sql
#Top 5 ordered pizzas based on revenue
SELECT pt.name,
    CONCAT("$ ",ROUND(SUM(p.price * o.quantity))) AS Total_revenue
FROM Order_detail o
JOIN pizza p
ON o.pizza_id = p.pizza_id
JOIN pizza_type pt
ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY Total_revenue DESC
LIMIT 5;
```

Output:

name	Total_revenue
The Thai Chicken Pizza	$ 43434
The Barbecue Chicken Pizza	$ 42768
The California Chicken Pizza	$ 41410
The Classic Deluxe Pizza	$ 38180
The Spicy Italian Pizza	$ 34831

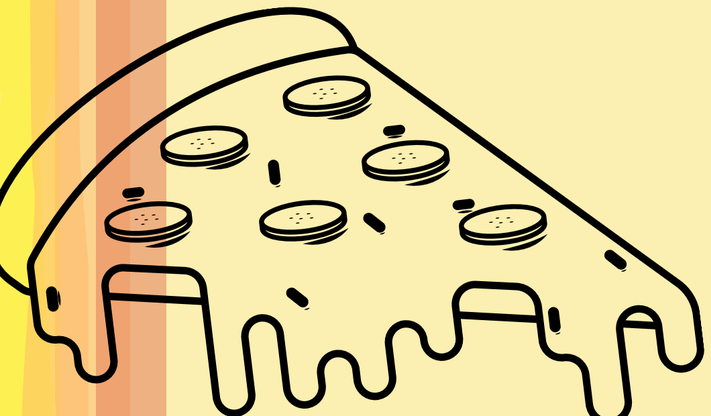


Query:

```sql
#Percentage contribution of each pizza type to revenue
SELECT pt.category,
    CONCAT(ROUND(SUM(o.quantity * p.price) / (SELECT SUM(o.quantity * p.price)
    FROM Order_detail o JOIN pizza p ON o.pizza_id = p.pizza_id) * 100, 2),"%") AS Revenue_Percentage
FROM pizza_type pt
JOIN pizza p
ON pt.pizza_type_id = p.pizza_type_id
JOIN Order_detail o
ON o.pizza_id = p.pizza_id
GROUP BY pt.category
ORDER BY Revenue_Percentage DESC;
```

Output:

category	Revenue_Percentage
Classic	26.91%
Supreme	25.46%
Chicken	23.96%
Veggie	23.68%

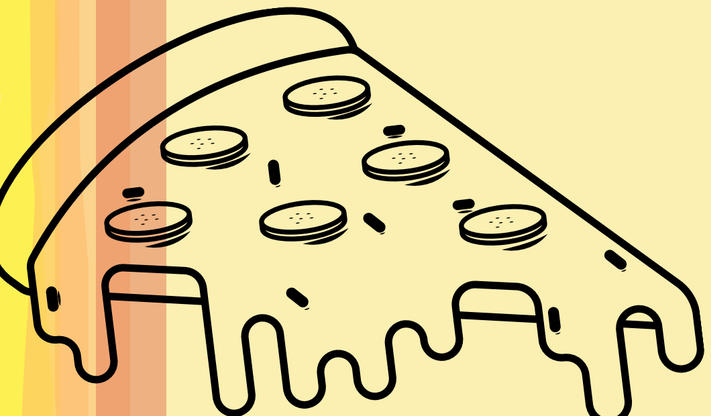


Query:

```sql
#Cumulative Revenue generated over time
SELECT o.date,
    ROUND(SUM(od.quantity * p.price), 2) AS Daily_revenue,
    ROUND(SUM(SUM(od.quantity * p.price)) OVER (ORDER BY o.date), 2) AS Cumulative_revenue
FROM Order_date_time o
JOIN Order_detail od
ON o.order_id = od.order_id
JOIN pizza p
ON p.pizza_id = od.pizza_id
GROUP BY o.date
ORDER BY o.date;
```

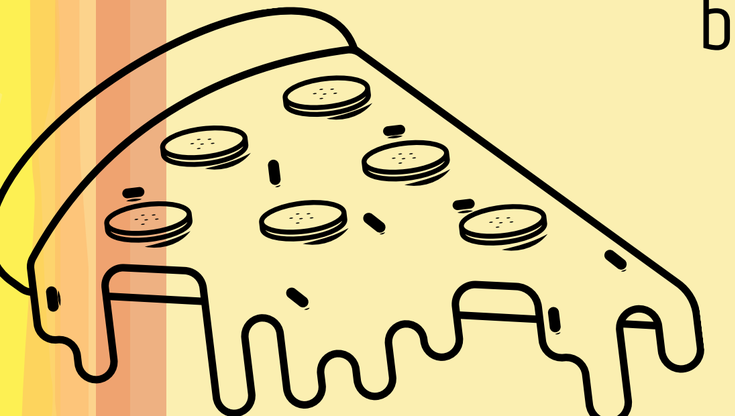
Output:

date	Daily_revenue	Cumulative_revenue
01-01-2015	2713.85	2713.85
01-02-2015	3189.2	5903.05
01-03-2015	1598.55	7501.6
01-04-2015	2176.85	9678.45
01-05-2015	2571.95	12250.4
01-06-2015	3067.75	15318.15
01-07-2015	2231.5	17549.65
01-08-2015	2440.55	19990.2
01-09-2015	2352.85	22343.05
01-10-2015	3202.15	25545.2
01-11-2015	1986.65	27531.85
01-12-2015	2076.7	29608.55
02-01-2015	2731.9	32340.45
02-02-2015	2328.6	34669.05
02-03-2015	2379.05	37048.1
02-04-2015	2547.15	39595.25
02-05-2015	2400.2	41995.45
02-06-2015	2449.95	44445.4
02-07-2015	2294.8	46740.2
02-08-2015	1910.15	48650.35
02-09-2015	1865.55	50515.9
02-10-2015	2074.85	52590.75
02-11-2015	2299.1	54889.85
02-12-2015	2214.1	57103.95
29-07-2015	1923.25	766680.7
29-08-2015	2035	768715.7
29-09-2015	2762.05	771477.75
29-10-2015	2169.75	773647.5
29-11-2015	1899	775546.5
29-12-2015	1353.25	776899.75
30-01-2015	2270.3	779170.05
30-03-2015	2255.45	781425.5
30-04-2015	2667.5	784093
30-05-2015	2486.95	786579.95
30-06-2015	2210.95	788790.9
30-07-2015	2349.15	791140.05
30-08-2015	1494.6	792634.65
30-09-2015	2198.05	794832.7
30-10-2015	2736.6	797569.3
30-11-2015	2223.25	799792.55
30-12-2015	1337.8	801130.35
31-01-2015	2417.85	803548.2
31-03-2015	2756.6	806304.8
31-05-2015	1717.65	808022.45
31-07-2015	2095.4	810117.85
31-08-2015	2081.35	812199.2
31-10-2015	2744.85	814944.05
31-12-2015	2916	817860.05



CONCLUSION

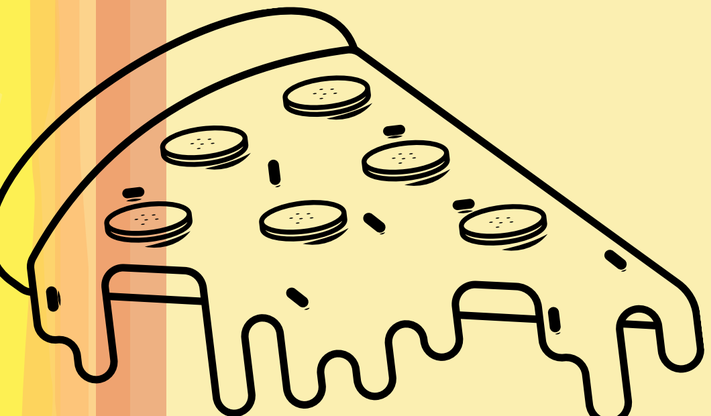
Through our sales analysis, we examined various aspects of pizza orders, including customer preferences, overall sales trends, and both product-specific and cumulative revenue insights. Leveraging these findings, the pizza store can make data-driven decisions to improve the customer experience, boost sales, and achieve sustainable growth.



Fun Pepperoni

Why does a pizza maker breakup with the dough?

Because it was too kneady!!



Thankyou!!