PIZZA SALES ANALYSIS

The Queries Are Divided Into Three Parts:

1. Beginner- Page (3-12)
2. Intermediate- Page (13-22)
3. Advance- Page (23-31)

Findings & Recommendations:
Page (32-33)

These are some of the Questions Answered Using MYSQL

1. Total Orders
2. Total Quantity Sold
3. Average Quantity Sold Per Day
4. Top 5 Most Ordered Pizza
5. Rank Pizza Size by Orders
6. Total Orders by Day
7. Best Seller Pizza for Each Month
8. Top 3 Most Positive and Negative by WoW Change
9. Month On Month Change
10. Daily Running Total

BEGINNER QUERIES

Retrieve the total number of orders placed

```
SELECT
    COUNT(DISTINCT order_id) AS 'Total_Orders'
FROM
    pizza_order_details;
```

Total_Orders
21350

Calculate the Revenue generated from pizza sales

```
SELECT
    CAST(SUM(o.quantity * p.price) AS DECIMAL (10 , 2 )) AS Revenue
FROM
    pizza_order_details AS o
        INNER JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id;
```

	Revenue
▶	817860.05

Pizza Prices
(Average, Minimum, Maximum)

```sql
SELECT
    CAST(AVG(price) AS DECIMAL (4 , 2 )) AS 'AVG Price',
    MIN(price) AS 'MIN Price',
    MAX(price) AS 'MAX Price'
FROM
    pizza_order_details AS d
        JOIN
    pizzas AS p ON d.pizza_id = p.pizza_id;
```

AVG Price	MIN Price	MAX Price
16.49	9.75	35.95

Identify the highest-priced pizza

```sql
SELECT
    o.pizza_id AS pizza_name, p.price
FROM
    pizza_order_details AS o
        INNER JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id
ORDER BY price DESC
LIMIT 1;
```

	pizza_name	price
▶	the_greek_xxl	35.95

Alternative (using window function)

```sql
with cte as (
select pizza_types.name as 'Pizza Name', cast(pizzas.price as decimal(10,2)) as 'Price',
rank() over (order by price desc) as rnk
from pizzas
join pizza_types on pizza_types.pizza_type_id = pizzas.pizza_type_id
)
select 'Pizza Name', Price from cte where rnk =1;
```

Pizza sizes by Orders

```sql
SELECT
    pizzas.size,
    COUNT(DISTINCT order_id) AS 'No of Orders',
    SUM(quantity) AS 'Total Quantity Ordered'
FROM
    pizza_order_details
        JOIN
    pizzas ON pizzas.pizza_id = pizza_order_details.pizza_id
GROUP BY pizzas.size
ORDER BY COUNT(DISTINCT order_id) DESC;
```

size	No of Orders	Total Quantity Ordered
L	12736	18956
M	11159	15635
S	10490	14403
XL	544	552
XXL	28	28

Pizza sizes by Revenue

size	Quantity	Revenue
L	18526	375319
M	15385	249382
S	14137	178076
XL	544	14076
XXL	28	1007

```sql
SELECT DISTINCT
    p.size,
    COUNT(o.quantity) AS 'Quantity',
    ROUND(SUM(o.quantity * p.price)) AS 'Revenue'
FROM
    pizza_order_details AS o
        JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY 3 DESC;
```

Top 5 Most Ordered Pizzas

```sql
SELECT
    t.name AS 'Name',
    p.size AS 'Size',
    SUM(quantity) AS 'Total Quantity'
FROM
    pizza_order_details AS d
        JOIN
    pizzas AS p ON d.pizza_id = p.pizza_id
        JOIN
    pizza_types AS t ON t.pizza_type_id = p.pizza_type_id
GROUP BY 1 , 2
ORDER BY 3 DESC
```

Name	Size	Total Quantity
The Big Meat Pizza	S	1914
The Thai Chicken Pizza	L	1410
The Five Cheese Pizza	L	1409
The Four Cheese Pizza	L	1316
The Classic Deluxe Pizza	M	1181

Total No of Pizzas by Sizes

size	No of Pizzas
S	32
M	31
L	31
XL	1
XXL	1

```sql
SELECT
    size, COUNT(size) AS 'No of Pizzas'
FROM
    pizzas
GROUP BY 1;
```

Find the total quantity of each pizza category ordered
(this will help us to understand the category which customers prefer the most)

```
SELECT
    t.category, SUM(o.quantity) AS 'Total Quantity'
FROM
    pizza_order_details AS o
        JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id
        JOIN
    pizza_types AS t ON p.pizza_type_id = t.pizza_type_id
GROUP BY t.category
ORDER BY SUM(o.quantity) DESC;
```

category	Total Quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Total Pizzas Sold

```
    SELECT
    sum(quantity) as 'Total Quantity Sold'
FROM
    pizza_order_details;
```

Total Quantity Sold
49574

Determine the distribution of orders by hour of the day

(At which time the orders are maximum. For inventory management and resource allocation)

```sql
SELECT
    HOUR(time) AS 'Hour of the day',
    COUNT(order_details_id) AS 'No of Orders'
FROM
    pizza_order_details AS d
        JOIN
    pizza_orders AS p ON d.order_id = p.order_id
GROUP BY HOUR(time)
ORDER BY COUNT(order_details_id) DESC;
```

Hour of the day	No of Orders
12	6545
13	6214
18	5359
17	5143
19	4350
16	4185
14	3521
20	3487
15	3170
11	2672
21	2528
22	1370
23	68
10	17
9	4

Total No of Pizzas in each Category

(excluding sizes)

```
SELECT
    category, COUNT(DISTINCT pizza_type_id) AS 'No of pizzas'
FROM
    pizza_types
GROUP BY category
ORDER BY COUNT(DISTINCT pizza_type_id);
```

category	No of pizzas
Chicken	6
Classic	8
Supreme	9
Veggie	9

Total Pizzas by Category

(including Sizes)

category	No of Pizzas
Chicken	18
Classic	26
Supreme	25
Veggie	27

```
SELECT
    category, COUNT(category) AS 'No of Pizzas'
FROM
    pizzas AS p
        JOIN
    pizza_types AS t ON p.pizza_type_id = t.pizza_type_id
GROUP BY category;
```

Top 10 Pizza Types by Revenue

```
SELECT DISTINCT
    o.pizza_id as 'Pizza Type',
    COUNT(o.quantity) AS 'Quantity',
    ROUND(SUM(o.quantity * p.price)) AS 'Revenue'
FROM
    pizza_order_details AS o
        JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id
GROUP BY o.pizza_id
ORDER BY 3 DESC;
```

Pizza Type	Quantity	Revenue
thai_ckn_l	1365	29258
five_cheese_l	1359	26066
four_cheese_l	1273	23622
spicy_ital_l	1088	23012
big_meat_s	1811	22968
southw_ckn_l	993	21082
bbq_ckn_l	967	20584
cali_ckn_l	895	19235
classic_dlx_m	1159	18896
mexicana_l	844	17557

Total No of Pizza Types

```
SELECT
    COUNT(pizza_id) AS 'Pizza Type'
FROM
    pizzas;
```

Pizza Type
96

INTERMEDIATE QUERIES

Peak Sales Periods
(Time of Day)

```sql
SELECT
    COUNT(order_details_id) AS 'No of Orders',
    CASE
        WHEN HOUR(time) BETWEEN 9 AND 11 THEN 'Morning'
        WHEN HOUR(time) BETWEEN 12 AND 16 THEN 'Afternoon'
        WHEN HOUR(time) BETWEEN 17 AND 20 THEN 'Evening'
        ELSE 'Night'
    END AS 'Period of Day'
FROM
    pizza_order_details AS d
        JOIN
    pizza_orders AS p ON d.order_id = p.order_id
GROUP BY 2
ORDER BY 1 DESC;
```

No of Orders	Period of Day
23635	Afternoon
18339	Evening
3966	Night
2693	Morning

Top 5 Pizzas by Revenue

```sql
SELECT DISTINCT
    t.name,
    COUNT(o.quantity) AS 'Quantity',
    SUM(o.quantity * p.price) AS 'Revenue'
FROM
    pizza_order_details AS o
        JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id
        JOIN
    pizza_types AS t ON p.pizza_type_id = t.pizza_type_id
GROUP BY t.name
ORDER BY 3 DESC
LIMIT 5;
```

name	Quantity	Revenue
The Thai Chicken Pizza	2315	43434.25
The Barbecue Chicken Pizza	2372	42768
The California Chicken Pizza	2302	41409.5
The Classic Deluxe Pizza	2416	38180.5
The Spicy Italian Pizza	1887	34831.25

Monthly Sales

```sql
SELECT
    MONTHNAME(date2) AS 'Month',
    SUM(p.quantity) AS 'Quantity',
    ROUND(SUM(p.quantity * price), 2) AS 'Sales'
FROM
    pizza_orders AS o
        JOIN
    pizza_order_details AS p ON o.order_id = p.order_id
        JOIN
    pizzas ON p.pizza_id = pizzas.pizza_id
GROUP BY 1;
```

Month	Quantity	Sales
January	4232	69793.3
February	3961	65164.3
March	4261	70397.1
April	4151	68736.8
May	4328	71402.75
June	4107	68230.2
July	4403	72733.85
August	4168	68278.25
September	3890	64180.05
October	3883	64027.6
November	4266	70395.35
December	3938	64734.3

Calculate the Average No of Pizzas Ordered per Day

```sql
with cte as(
SELECT
    o.date, SUM(p.quantity) AS 'Total'
FROM
    pizza_order_details AS p join orders as o on p.order_id=o.order_id group by o.date)
SELECT
    ROUND(AVG(`Total`)) AS 'Daily AVG Quantity'
FROM
    cte;
```

Daily AVG Quantity
139

Alternative (using Sub Query)

```sql
SELECT
    AVG(`Total Pizza Ordered`) AS 'Daily AVG Quantity'
FROM
    (SELECT
        o.date, SUM(p.quantity) AS 'Total Pizza Ordered'
    FROM
        pizza_order_details AS p
    JOIN orders AS o ON p.order_id = o.order_id
    GROUP BY o.date) AS pizzas_ordered;
```

Weekly Sales

```sql
SELECT
    WEEKOFYEAR(date2) AS 'Week',
    SUM(quantity) AS 'Quantity',
    ROUND(SUM(quantity * price), 2) AS 'Sales'
FROM
    pizza_orders AS o
        JOIN
    pizza_order_details AS p ON p.order_id = o.order_id
        JOIN
    pizzas ON p.pizza_id = pizzas.pizza_id
GROUP BY 1
ORDER BY 1;
```

Week	Quantity	Sales
43	783	12921.55
44	952	15742.2
45	1021	16885.95
46	941	15430.75
47	1052	17305.15
48	1030	17046.75
49	1021	16717.9
50	944	15514.9
51	948	15661.2
52	788	13020.15

Week	Quantity	Sales
22	1024	16914.15
23	977	16319.25
24	907	15122
25	959	15867.95
26	954	15738.95
27	1092	18083.75
28	944	15685.3
29	991	16490.95
30	969	15852.15
31	901	14712.6
32	992	16214.8
33	999	16376.1
34	904	14812
35	899	14729.15
36	1002	16519.2
37	992	16330.95
38	809	13301.95
39	886	14759.1
40	788	12902.15
41	941	15531.55
42	846	13988.75

Week	Quantity	Sales
1	1012	16685.2
2	901	14944.45
3	977	15957.4
4	897	14804.25
5	1049	17275.15
6	958	15653.25
7	965	15978.7
8	945	15547.8
9	991	16458
10	966	16028.4
11	1017	16619.05
12	878	14526
13	1021	16887.2
14	995	16413.45
15	989	16561.7
16	956	15782
17	926	15255.65
18	939	15345.55
19	1053	17353.1
20	976	16127.85
21	921	15370.45

Calculate the Percentage Contribution of Top 19 Pizzas by Pizza name to Total Revenue

(To understand % of contribution of pizzas in the total revenue)

```sql
SELECT DISTINCT
    t.name,
    COUNT(o.quantity) AS 'Quantity',
    CONCAT(ROUND((SUM(o.quantity * p.price) / 817860.05 * 100),
                 1),
           '%') AS 'contribution to Revenue'
FROM
    pizza_order_details AS o
        JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id
        JOIN
    pizza_types AS t ON p.pizza_type_id = t.pizza_type_id
GROUP BY t.name
ORDER BY 3 DESC;
```

name	Quantity	contribution to Revenue
The Thai Chicken Pizza	2315	5.3%
The Barbecue Chicken Pizza	2372	5.2%
The California Chicken Pizza	2302	5.1%
The Classic Deluxe Pizza	2416	4.7%
The Spicy Italian Pizza	1887	4.3%
The Southwest Chicken Pizza	1885	4.2%
The Italian Supreme Pizza	1849	4.1%
The Prosciutto and Arugula Pizza	1428	3%
The Vegetables + Vegetables Pizza	1510	3%
The Hawaiian Pizza	2370	3.9%
The Four Cheese Pizza	1850	3.9%
The Sicilian Pizza	1887	3.8%
The Pepperoni Pizza	2369	3.7%
The Greek Pizza	1406	3.5%
The Mexicana Pizza	1456	3.3%
The Five Cheese Pizza	1359	3.2%
The Italian Capocollo Pizza	1414	3.1%
The Pepper Salami Pizza	1422	3.1%
The Chicken Pesto Pizza	961	2%

Analyse the cumulative revenue generated over day

```sql
with cte as (
SELECT
    date AS 'Date',
    CAST(SUM(quantity * price) AS DECIMAL (10 , 2 )) AS Revenue
FROM
    pizza_order_details
        JOIN
    orders ON pizza_order_details.order_id = orders.order_id
        JOIN
    pizzas ON pizzas.pizza_id = pizza_order_details.pizza_id
GROUP BY date
)
select Date, Revenue, sum(Revenue) over (order by date) as 'Cumulative Sum'
from cte
group by date, Revenue;
```

Date	Revenue	Cumulative Sum
01-01-2023	2713.85	2713.85
01-02-2023	3189.20	5903.05
01-03-2023	1598.55	7501.60
01-04-2023	2176.85	9678.45
01-05-2023	2571.95	12250.40
01-06-2023	3067.75	15318.15
01-07-2023	2231.50	17549.65
01-08-2023	2440.55	19990.20
01-09-2023	2352.85	22343.05
01-10-2023	3202.15	25545.20
01-11-2023	1986.65	27531.85
01-12-2023	2076.70	29608.55
02-01-2023	2731.90	32340.45
02-02-2023	2328.60	34669.05
02-03-2023	2379.05	37048.10
02-04-2023	2547.15	39595.25
02-05-2023	2400.20	41995.45
02-06-2023	2449.95	44445.40
02-07-2023	2294.80	46740.20
02-08-2023	1910.15	48650.35
02-09-2023	1865.55	50515.90

The data was till 31-12-2023 but could fit till 02-09-2023

Pizzas Price Segmentation

```sql
SELECT
    price 'Prices',
    SUM(quantity) 'Total Quantity',
    CASE
        WHEN price >= 9.75 AND price <= 16.3 THEN 'Low Price'
        WHEN price >= 16.4 AND price <= 22.85 THEN 'medium Price'
        WHEN price >= 22.86 AND price <= 29.4 THEN 'High Price'
        ELSE 'Very High'
    END AS 'Price Range'
FROM
    pizza_order_details AS d
        JOIN
    pizzas AS p ON d.pizza_id = p.pizza_id
GROUP BY 1
ORDER BY 2 DESC;
```

Prices	Total Quantity	Price Range
20.75	8891	medium Price
12	5744	Low Price
16	4522	Low Price
16.75	4380	medium Price
16.5	4111	medium Price
12.5	3380	Low Price
20.25	3093	medium Price
12.75	2529	Low Price
20.5	2026	medium Price
18.5	1409	medium Price
17.95	1316	medium Price
16.25	1136	Low Price
10.5	1020	Low Price
12.25	850	Low Price
9.75	751	Low Price
15.25	728	Low Price
14.75	586	Low Price
11	578	Low Price
25.5	552	High Price
23.65	490	High Price
13.25	483	Low Price
14.5	397	Low Price
17.5	384	medium Price
21	190	medium Price
35.95	28	Very High

Pizza Price Segmentation

```sql
SELECT
    price 'Prices',
    SUM(quantity) 'Total Quantity',
    CAST(SUM(quantity * price) AS DECIMAL (10 , 2 )) AS 'Revenue',
    CASE
        WHEN price >= 9.75 AND price <= 16.3 THEN 'Low Price'
        WHEN price >= 16.4 AND price <= 22.85 THEN 'medium Price'
        WHEN price >= 22.86 AND price <= 29.4 THEN 'High Price'
        ELSE 'Very High'
    END AS 'Price Range'
FROM
    pizza_order_details AS d
        JOIN
    pizzas AS p ON d.pizza_id = p.pizza_id
GROUP BY 1
ORDER BY 3 DESC;
```

Prices	Total Quantity	Revenue	Price Range
20.75	8891	184488.25	medium Price
16.75	4380	73365.00	medium Price
16	4522	72352.00	Low Price
12	5744	68928.00	Low Price
16.5	4111	67831.50	medium Price
20.25	3093	62633.25	medium Price
12.5	3380	42250.00	Low Price
20.5	2026	41533.00	medium Price
12.75	2529	32244.75	Low Price
18.5	1409	26066.50	medium Price
17.95	1316	23622.20	medium Price
16.25	1136	18460.00	Low Price
25.5	552	14076.00	High Price
23.65	490	11588.50	High Price
15.25	728	11102.00	Low Price
10.5	1020	10710.00	Low Price
12.25	850	10412.50	Low Price
14.75	586	8643.50	Low Price
9.75	751	7322.25	Low Price
17.5	384	6720.00	medium Price
13.25	483	6399.75	Low Price
11	578	6358.00	Low Price
14.5	397	5756.50	Low Price
21	190	3990.00	medium Price
35.95	28	1006.60	Very High

ADVANCE QUERIES

Top 3 Pizzas from each Category by Revenue

```sql
with cte as (
SELECT
    t.name,
    t.category,
    ROUND(SUM(o.quantity * p.price), 2) AS 'Revenue'
FROM
    pizza_order_details AS o
        JOIN
    pizzas AS p ON o.pizza_id = p.pizza_id
        JOIN
    pizza_types AS t ON p.pizza_type_id = t.pizza_type_id
GROUP BY t.name , t.category)
,cte1 as(
select category, name, `Revenue`, rank()
over(partition by category order by `Revenue` desc)
as rnk from cte)
SELECT
    category, name, `Revenue`
FROM
    cte1
WHERE
    rnk IN (1 , 2, 3)
ORDER BY category , `Revenue` DESC;
```

category	name	Revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75
Supreme	The Sicilian Pizza	30940.5
Veggie	The Four Cheese Pizza	32265.7
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

Four Categories
Chicken, Classic, Supreme, Veggie

```sql
WITH cte AS (
    SELECT
        MONTHNAME(date2) AS `Month`,
        SUM(p.quantity) AS `Quantity`,
        ROUND(SUM(p.quantity * price), 2) AS `Sales`,
        LEAD(ROUND(SUM(p.quantity * price), 2), 1) OVER () AS `nxSales`
    FROM
        pizza_orders AS o
    JOIN
        pizza_order_details AS p
        ON o.order_id = p.order_id
    JOIN
        pizzas
        ON p.pizza_id = pizzas.pizza_id
    GROUP BY
        MONTHNAME(date2))
SELECT
    `Month`,
    `Quantity`,
    `Sales`,
    COALESCE(
        LAG(
            CAST(
                ((nxSales - Sales) / Sales * 100) AS DECIMAL(5, 2)
            )) OVER(),
            0
        ) AS `Monthly Change %`
FROM cte;
```

Month on Month % Change

Month	Quantity	Sales	Monthly Change %
January	4232	69793.3	0.00
February	3961	65164.3	-6.63
March	4261	70397.1	8.03
April	4151	68736.8	-2.36
May	4328	71402.75	3.88
June	4107	68230.2	-4.44
July	4403	72733.85	6.60
August	4168	68278.25	-6.13
September	3890	64180.05	-6.00
October	3883	64027.6	-0.24
November	4266	70395.35	9.95
December	3938	64734.3	-8.04

Week on Week Change %

```
with cte as(
select weekofyear(date2) as 'Weeks',
sum(quantity) as 'Quantity',
round(sum(quantity*price),2) as 'Sales',
lead(round(sum(quantity*price),2))
over(order by weekofyear(date2))
as 'nxweek' from pizza_orders as o
join pizza_order_details as p
on p.order_id=o.order_id join pizzas
on p.pizza_id=pizzas.pizza_id group by 1)
select Weeks, Quantity, Sales,
lag(cast(((nxweek-Sales)/Sales*100) as
decimal(4,2))) over() as 'Weekly Change %'
from cte group by Weeks;
```

Weeks	Quantity	Sales	Weekly Change %
43	783	12921.55	-7.63
44	952	15742.2	21.83
45	1021	16885.95	7.27
46	941	15430.75	-8.62
47	1052	17305.15	12.15
48	1030	17046.75	-1.49
49	1021	16717.9	-1.93
50	944	15514.9	-7.20
51	948	15661.2	0.94
52	788	13020.15	-16.86

Weeks	Quantity	Sales	Weekly Change %	Weeks	Quantity	Sales	Weekly Change %
22	1024	16914.15	10.04	1	1012	16685.2	NULL
23	977	16319.25	-3.52	2	901	14944.45	-10.43
24	907	15122	-7.34	3	977	15957.4	6.78
25	959	15867.95	4.93	4	897	14804.25	-7.23
26	954	15738.95	-0.81	5	1049	17275.15	16.69
27	1092	18083.75	14.90	6	958	15653.25	-9.39
28	944	15685.3	-13.26	7	965	15978.7	2.08
29	991	16490.95	5.14	8	945	15547.8	-2.70
30	969	15852.15	-3.87	9	991	16458	5.85
31	901	14712.6	-7.19	10	966	16028.4	-2.61
32	992	16214.8	10.21	11	1017	16619.05	3.69
33	999	16376.1	0.99	12	878	14526	-12.59
34	904	14812	-9.55	13	1021	16887.2	16.25
35	899	14729.15	-0.56	14	995	16413.45	-2.81
36	1002	16519.2	12.15	15	989	16561.7	0.90
37	992	16330.95	-1.14	16	956	15782	-4.71
38	809	13301.95	-18.55	17	926	15255.65	-3.34
39	886	14759.1	10.95	18	939	15345.55	0.59
40	788	12902.15	-12.58	19	1053	17353.1	13.08
41	941	15531.55	20.38	20	976	16127.85	-7.06
42	846	13988.75	-9.93	21	921	15370.45	-4.70

Top 3 Positive and Negative Weeks

```sql
with cte as(
select weekofyear(date2) as 'Weeks', sum(quantity) as 'Quantity',
round(sum(quantity*price),2) as 'Sales',
lead(round(sum(quantity*price),2))
over(order by weekofyear(date2)) as 'nxweek'
from pizza_orders as o join pizza_order_details as p
on p.order_id=o.order_id join pizzas on p.pizza_id=pizzas.pizza_id
group by 1),
cte1 as(
select lead(Weeks) over() as 'Weeks', lead(Quantity)
over() as 'Quantity', lead(Sales) over() as 'Sales',
((nxweek-Sales)/Sales*100) as 'Weekly Change %',
rank() over(order by ((nxweek-Sales)/Sales*100) desc) as rnk
from cte)
(select Weeks, Quantity, Sales, `Weekly Change %`
from cte1 where rnk in (49,50,51) or rnk in (1,2,3)
order by 4 desc);
```

Weeks	Quantity	Sales	Weekly Change %
44	952	15742.2	21.829037538066267
41	941	15531.55	20.379549144909955
5	1049	17275.15	16.690477396693527
28	944	15685.3	-13.263012372986802
52	788	13020.15	-16.863650294996557
38	809	13301.95	-18.547604395335238

```sql
with cte as(
SELECT
    d.pizza_id,
    t.name AS 'Name',
    p.size AS 'Size',
    SUM(quantity) AS 'Total Quantity',
    MONTHNAME(date2) AS 'Month'
FROM
    pizza_order_details AS d
        JOIN
    pizza_orders AS o ON d.order_id = o.order_id
        JOIN
    pizzas AS p ON d.pizza_id = p.pizza_id
        JOIN
    pizza_types AS t ON p.pizza_type_id = t.pizza_type_id
GROUP BY 1 , 2 , 3 , 5
ORDER BY 5 , 4 DESC),
cte1 as(
select *, row_number() over(partition by month order by 'total quantity')
as rnk from cte)
(SELECT
    `Name`, `Size`, `Total Quantity`, `Month`, rnk
FROM
    cte1
WHERE
    rnk IN (1 , 2, 3, 4, 5)
ORDER BY 5 , 3 DESC);
```

Top 5 Best Seller Pizzas Every Month

Name	Size	Total Quantity	Month	rnk
The Big Meat Pi...	S	190	May	1
The Big Meat Pi...	S	185	July	1
The Big Meat Pi...	S	176	March	1
The Big Meat Pi...	S	174	November	1
The Big Meat Pi...	S	160	August	1
The Big Meat Pi...	S	157	December	1
The Big Meat Pi...	S	151	October	1
The Big Meat Pi...	S	151	February	1
The Big Meat Pi...	S	150	January	1
The Big Meat Pi...	S	142	September	1
The Big Meat Pi...	S	139	April	1
The Big Meat Pi...	S	139	June	1
The Five Chees...	L	139	July	2
The Five Chees...	L	138	January	2
The Thai Chicke...	L	136	December	2
The Five Chees...	L	125	March	2
The Five Chees...	L	124	May	2
The Five Chees...	L	124	June	2
The Thai Chicke...	L	123	September	2

There were 60 rows, I was able to fit only this much

```sql
with cte as(
SELECT COUNT(d.order_details_id) 'Total Orders', WEEKDAY(date2) AS 'week',
case
    when weekday(date2) = 0 then 'Monday'
    when weekday(date2) = 1 then 'Tuesday'
    when weekday(date2) = 2 then 'Wednesday'
    when weekday(date2) = 3 then 'Thursday'
    when weekday(date2) = 4 then 'Friday'
    when weekday(date2) = 5 then 'Saturday'
    when weekday(date2) = 6 then 'Sunday'
end as "weekday"
from
    pizza_orders as o
        join
    pizza_order_details as d on o.order_id=d.order_id
group by 2,3
order by 1 desc)
SELECT
    `Total Orders`, `Weekday`
FROM
    cte;
```

Day of the Week Analysis

(Which day gets most orders)

Total Orders	Weekday
8106	Monday
7355	Tuesday
7323	Sunday
6800	Saturday
6753	Friday
6379	Thursday
5917	Wednesday

Top 15 Days

using Temp Table

```sql
create temporary table daily_rev as
SELECT
    date AS 'Date',
    CAST(SUM(quantity * price) AS DECIMAL (10 , 2 )) AS Revenue
FROM
    pizza_order_details
        JOIN
    orders ON pizza_order_details.order_id = orders.order_id
        JOIN
    pizzas ON pizzas.pizza_id = pizza_order_details.pizza_id
GROUP BY date;

SELECT
    *
FROM
    daily_rev
WHERE
    Revenue > 3000
ORDER BY revenue desc;

drop table daily_rev;
```

Date	Revenue
27-11-2023	4422.45
26-11-2023	4405.95
15-10-2023	4320.20
04-07-2023	3864.20
03-07-2023	3443.00
15-05-2023	3386.15
24-07-2023	3204.40
01-10-2023	3202.15
01-02-2023	3189.20
06-11-2023	3157.50
17-07-2023	3131.65
01-06-2023	3067.75
08-05-2023	3052.30
14-08-2023	3016.60
29-05-2023	3001.20

Create stored procedure to search Pizza details

```sql
CREATE PROCEDURE pizza_details(IN p_type_ids CHAR(100))
BEGIN
    SET @sql_query = CONCAT('
        SELECT
            t.pizza_type_id,
            t.name,
            t.category,
            t.ingredients,
            p.size,
            p.price
        FROM pizza_types AS t
        JOIN pizzas AS p ON t.pizza_type_id = p.pizza_type_id
        WHERE t.pizza_type_id IN (', p_type_ids, ')');

    -- Execute the dynamically constructed query
    PREPARE stmt FROM @sql_query;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END$$
delimiter ;

-- type pizza_type_id in () for result
CALL pizza_details("'bbq_ckn', 'big_meat'");

-- drop stored procedure
drop procedure if exists pizza_details;
```

pizza_type_id	name	category	ingredients	size	price
bbq_ckn	The Barbecue Chicken Pizza	Chicken	Barbecued Chicken, Red Peppers, Green Peppe...	S	12.75
bbq_ckn	The Barbecue Chicken Pizza	Chicken	Barbecued Chicken, Red Peppers, Green Peppe...	M	16.75
bbq_ckn	The Barbecue Chicken Pizza	Chicken	Barbecued Chicken, Red Peppers, Green Peppe...	L	20.75
big_meat	The Big Meat Pizza	Classic	Bacon, Pepperoni, Italian Sausage, Chorizo Sau...	S	12
big_meat	The Big Meat Pizza	Classic	Bacon, Pepperoni, Italian Sausage, Chorizo Sau...	M	16
big_meat	The Big Meat Pizza	Classic	Bacon, Pepperoni, Italian Sausage, Chorizo Sau...	L	20.5

Findings from Pizza Sales Data

- **1. Top-Performing Categories and Pizzas:**
  - Chicken Pizzas: The "Thai Chicken Pizza" and "Southwest Chicken Pizza" are bestsellers, especially in Large (L) size.
  - Classic Pizzas: "The Big Meat Pizza" dominates in Small (S), while "The Classic Deluxe Pizza" performs well across sizes.
  - Supreme Pizzas: "The Spicy Italian Pizza" and "The Sicilian Pizza" excel in Large (L) size.
  - Veggie Pizzas: The "Five Cheese Pizza" generates significant revenue, exclusively in Large (L).
- **2. Size Preferences:**
  - Large (L) and Medium (M) sizes drive most revenue and sales across all categories.
  - Extra-Large (XL) and XXL sizes show low demand or limited availability.
- **3. Underperforming Pizzas:**
  - "The Chicken Alfredo Pizza" and "The Green Garden Pizza" have low sales across all sizes.
- **4. Category Trends:**
  - Chicken and Classic categories lead in revenue, while Veggie pizzas show potential for growth with targeted efforts.
- **5. Time-Based Sales Trends:**
  - Monthly Trends: Revenue and total quantity sold peak in December, indicating strong seasonality. January and February show lower sales, suggesting potential for targeted campaigns during off-peak months.
  - Weekly Patterns: Weekends (Friday-Sunday) generate higher sales, emphasizing the need to focus on weekend promotions.
  - Hourly Sales: Sales are concentrated between 6 PM to 9 PM, aligning with dinner hours. There is minimal activity outside these hours.
- **6. Time-Sensitive Category Performance:**
  - Peak Days for Categories: Chicken and Classic pizzas perform exceptionally well on weekends, while Veggie pizzas see a steady demand throughout the week.

Recommendations

- **1. Focus on Popular Sizes (L & M):**
  - Offer combo deals and upselling incentives for Small (S) to Large (L) upgrades.
- **2. Revive Underperforming Pizzas:**
  - Promote pizzas like "The Chicken Alfredo Pizza" through discounts or bundled offers.
- **3. Boost Larger Size Sales:**
  - Introduce “XXL Family Specials” or party-size promotions targeting group orders.
- **4. Create Category-Specific Campaigns:**
  - Run targeted promotions like “Chicken Lovers Week” or “Classic Pizza Fest.”
- **5. Capitalize on Peak Times:**
  - Offer exclusive "Happy Hour" discounts during 6 PM to 9 PM to maximize peak sales.
  - Introduce weekend-specific combos targeting high-traffic days.
- **6. Target Day-Specific Categories:**
  - Promote Chicken and Classic pizzas during weekends with deals like “Weekend Meat Mania.”
  - Encourage midweek Veggie pizza sales through targeted ads or loyalty discounts.

I look forward to discussing these insights and recommendations further. Your feedback and any additional information to refine these strategies would be greatly appreciated.

THANK YOU