



# Dominos Business Analytics

## Key Performance Indicators : (using SQL)

- 1) Select the total number of orders placed

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

- Entities:** order\_details, orders, pizza\_types, pizzas.
- Query:**

```
1 --Select the total number of orders placed
2 Select
3 count(order_id) as total_orders
4 from
5 orders
```

- Result:** A single row with the value 21350.
- Database:** dominos.db
- Time:** 6ms

- 2) Calculate total revenue generated from Pizza sales

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

- Entities:** order\_details, orders, pizza\_types, pizzas.
- Query:**

```
1 --Calculate total revenue generated from Pizza sales
2 Select
3 Round(SUM((order_details.quantity * pizzas.price)), 1) as total_sales
4 from
5 order_details
6 join pizzas on pizzas.pizza_id = order_details.pizza_id
```

- Result:** A single row with the value 817860.1.
- Database:** dominos.db
- Time:** 27ms

### 3) Identify the highest price pizza

The screenshot shows a database interface with a code editor and a results table. The code is:

```
1 --Identify the highest price pizza
2 Select
3   pizzas_types.name,
4   pizzas.price
5   from
6   pizzas_types
7   join pizzas on pizzas_types.pizza_type_id = pizzas.pizza_type_id
8   order by
9   pizzas.price desc
10 limit
11 1
```

The results table shows one row:

	name	price
1	The Greek Pizza	35.95

At the bottom, it says "dominos.db" and "sqlite".

### 4) Identify the most common size pizza ordered

The screenshot shows a database interface with a code editor and a results table. The code is:

```
1 --Identity the most common size pizza ordered
2 Select
3   pizzas.size,
4   count(order_details.order_details_id) as order_count
5   from
6   pizzas
7   join order_details on pizzas.pizza_id = order_details.pizza_id
8   group by
9   pizzas.size
10 order by
11   order_count desc
```

The results table shows five rows:

	size	order_count
1	L	18526
2	M	15385
3	S	14137
4	XL	544
5	XXL	28

At the bottom, it says "dominos.db" and "sqlite".

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

The screenshot shows a database interface with a code editor and a results table. The code is a SQL query to find the top 5 most ordered pizza types:

```

1 --List the top 5 most ordered pizza types along with their quantities.
2 Select
3   pizza_types.name,
4   sum(order_details.quantity) as Quantity
5   from
6   pizza_types
7   join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
8   join order_details on order_details.pizza_id = pizzas.pizza_id
9   group by
10  pizza_types.name
11  order by
12  Quantity desc
13  limit
14  5

```

The results table displays the top 5 pizza types and their quantities:

	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

At the bottom, it shows the database is 'dominos.db' and the query took 62ms.

- 6) Join the necessary tables to find the total quantity of each pizza category ordered.

The screenshot shows a database interface with a code editor and a results table. The code is a SQL query to find the total quantity of each pizza category ordered:

```

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

```

The results table displays the total quantity for each pizza category:

	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

At the bottom, it shows the database is 'dominos.db' and the query took 60ms.

## 7) Determine the distribution of orders by hour of the day

The screenshot shows a database interface with a query editor and a results table. The query is:

```
1 --Determine the distribution of orders by hour of the day
2 Select
3 strftime('%H', time) as hour_time,
4 count(order_id) as order_count
5 from
6 orders
7 group by
8 hour_time
```

The results table has two columns: `hour_time` and `order_count`. The data is:

hour_time	order_count
09	1
10	8
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

At the bottom, it says `dominos.db`, `sqlite`, `15`, `0 affected`, and `14ms`.

## 8) Join the relevant tables to find the category wise distribution of pizzas

The screenshot shows a database interface with a query editor and a results table. The query is:

```
1 --Join the relevant tables to find the category wise distribution of pizzas
2 Select
3 category,
4 count(name)
5 from
6 pizza_types
7 group by
8 category
```

The results table has two columns: `category` and `count(name)`. The data is:

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

At the bottom, it says `dominos.db`, `sqlite`, `4`, `0 affected`, and `2ms`.

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

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

- Entities:** order\_details, orders, pizza\_types, pizzas.
- Query:**

```

1 --Group the orders by date and calculate the average number of pizzas ordered per day
2 Select
3 round(avg(quantity), 0) as average_pizza_ordered_perday
4 from
5 (
6   Select
7     orders.date,
8     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
13    orders.date

```
- Results:** average\_pizza\_ordered\_perday | 1 | 138
- Database:** dominos.db
- Run Status:** 1 rows, 0 affected, 39ms

- 10) Determine top 3 most ordered pizza based on revenue

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

- Entities:** order\_details, orders, pizza\_types, pizzas.
- Query:**

```

1 --Determine top 3 most ordered pizza based on revenue
2 Select
3   pizza_types.name,
4   sum(order_details.quantity * pizzas.price) as revenue
5 from
6   pizza_types
7   join pizzas on pizzas.pizza_type_id = pizza_types.pizza_type_id
8   join order_details on order_details.pizza_id = pizzas.pizza_id
9 group by
10  pizza_types.name
11 order by
12  revenue desc
13 limit
14  3

```
- Results:**

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Piz...	42768
3	The California Chicken Piz...	41409.5
- Database:** dominos.db
- Run Status:** 3 rows, 0 affected, 65ms

## 11) Calculate the percentage contribution of each pizza type to total revenue.

The screenshot shows a database interface with a query editor and a results table. The query calculates the percentage contribution of each pizza type to total revenue. The results table lists four pizza types: Classic, Supreme, Chicken, and Veggie, along with their category and revenue values.

```

1 --Calculate the percentage contribution of each pizza type to total revenue.
2 Select
3   pizza_types.category,
4   round(
5     sum(order_details.quantity * pizzas.price) /
6       Select
7         round(sum(order_details.quantity * pizzas.price), 2) as total_sales
8       from
9         order_details
10        join pizzas on pizzas.pizza_id = order_details.pizza_id
11      ) * 100,
12      2
13    ) as revenue
14  from
15    pizza_types
16    join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
17    join order_details on order_details.pizza_id = pizzas.pizza_id

```

	category	revenue
1	Classic	26.91
2	Supreme	25.46
3	Chicken	23.96
4	Veggie	23.68

dominos.db    sqlite    4    0 affected    86ms

## 12) Analyze the cumulative revenue generate over time

The screenshot shows a database interface with a query editor and a results table. The query analyzes the cumulative revenue generated over time. The results table lists daily cumulative revenue from January 1, 2015, to January 17, 2015.

```

1 --Analyze the cumulative revenue generate over time
2 select
3   date,
4   sum(revenue) over (
5     order by
6     date
7   ) as cum_revenue
8 from
9   (
10    Select
11      orders.date,
12      sum(order_details.quantity * pizzas.price) as revenue
13    from
14      order_details
15      join pizzas on order_details.pizza_id = pizzas.pizza_id
16      join orders on orders.order_id = order_details.order_id
17    group by
18      orders.date
19  ) as sales

```

	date	cum_revenue
1	2015-01-01	2713.85
2	2015-01-02	5445.75
3	2015-01-03	8108.15
4	2015-01-04	9863.6
5	2015-01-05	11929.55
6	2015-01-06	14358.5
7	2015-01-07	16560.7
8	2015-01-08	19399.05
9	2015-01-09	21526.399999999998
10	2015-01-10	23990.35
11	2015-01-11	25862.649999999998
12	2015-01-12	27781.7
13	2015-01-13	29831.3
14	2015-01-14	32358.7
15	2015-01-15	34343.5
16	2015-01-16	36937.65
17	2015-01-17	39001.75

dominos.db    sqlite    358    0 affected    51ms

13) Determine the top 3 most ordered pizza based on revenue for each pizza category.

The screenshot shows a database query editor interface with the following details:

- Entities:** order\_details, orders, pizza\_types, pizzas.
- Query:**

```
--Determine the top 3 most ordered pizza based on revenue for each pizza category.
Select
    name,
    revenue
from
(
    Select
        category,
        name,
        revenue,
        rank() over (
            partition by
                category
            order by
                revenue desc
        ) as rn
    from
    (
        Select
            pizza_types.category,
            pizza_types.name,
            sum(order_details.quantity * pizzas.price) 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,
            pizza_types.name
        ) as a
    ) as b
where
    rn <= 3
```
- Results:** A table showing the top 3 most ordered pizzas by category and their revenue.

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5
4	The Classic Deluxe Pizza	38180.5
5	The Hawaiian Pizza	32273.25
6	The Pepperoni Pizza	30161.75
7	The Spicy Italian Pizza	34831.25
8	The Italian Supreme Pizza	33476.75
9	The Sicilian Pizza	30940.5