FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

# Pizza sales data analysis

Group 4:
Fernanda C. Gangas
Vicente Coronado
Matías Aguirre

May 31, 2023

# Milestone 1

**Data set description:** The data set is for a fictional pizza shop and includes information about orders and different types of pizza. The data can be found here (link)

**Description and motivation of the problem:** The motivation behind this problem lies in understanding and analyzing customer consumption patterns over time at the pizza shop. By studying sales data and the most popular pizza types, it's possible to identify trends, preferences, and customer behaviors. This information can be used to optimize ingredient selection, create more effective promotional offers, adjust the menu, and enhance the overall customer experience at the pizza shop. Analyzing consumption patterns can also help identify opportunities to introduce new pizza types or adapt the existing offerings to meet changing customer demands. In summary, the goal is to use the dataset to gain valuable insights that drive the growth and success of the pizza shop business.

**Chosen option:** Data analysis report.

**Initial data exploration:**

- Table 1: orders (order_id, time, date)
- Table 2: order_details (order_details_id, order_id, pizza_id, quantity)
- Table 3: pizzas (pizza_id, pizza_type_id, size, price)
- Table 4: pizza_types (pizza_type_id, name, category, ingredients, price)

**Number of rows per table:**

- *Table 1:* 21,4k rows
- *Table 2:* 48,6k rows
- *Table 3:* 96 rows
- *Table 4:* 32 rows

**Type of data:**

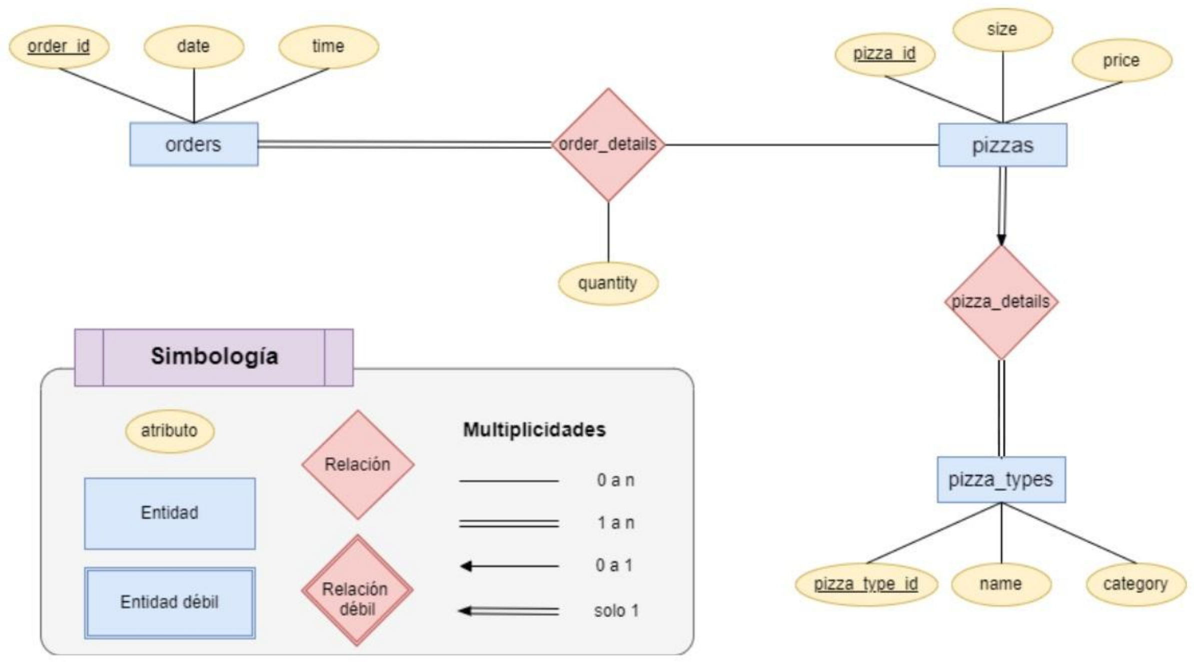
- ***order_id:*** Int
- ***time:*** Date
- ***date:*** Date
- ***order_details_id:*** Int
- ***order_id:*** Int
- ***pizza_id:*** String
- ***quantity:*** Int
- ***pizza_id:*** String
- ***pizza_type_id:*** String
- ***size:*** String
- ***price:*** Float
- ***pizza_type_id:*** String
- ***name:*** String
- ***category:*** String
- ***ingredients:*** String (is a list of ingredients used in different types of pizza. Ingredients are separated by commas and are listed on the menu. All types of pizza include mozzarella cheese, even if not specified, and all include tomato sauce, unless another sauce is specified.)
- ***price:*** String

**Questions:**

1. What are the 3 best-selling types of pizza each month?
2. What are the pizzas with the greatest benefits?
3. For how many months was each type of pizza the least sold?
4. In which month did each type of pizza reach its peak sales?
5. What are the historical monthly sales figures for the pizza shop?
6. What is the average number of pizzas sold for each day of the week?

# Milestone 2

## E/R Model:



## Translation to the relational model:

- **Entities (**those that are in italics and colored are foreign keys**)**
  - orders (

    order_id: int,

    time: time,

    date: date

    )
  - pizzas (

    pizza_id: string,

    size: string,

    price: float

    )
  - pizza_types (

    pizza_type_id: string,

name: string,

category: string

)

- **Relations**
    - order_details (

        *order_id*: int,

        *pizza_id*: string,

        quantity: int

        )

    - pizza_details (

        *pizza_id*: string,

        *pizza_type_id*: string

        )

The entity "order_details" was transformed into a strong relationship, and the attribute "order_details_id" was removed because we found it unnecessary. This was because the foreign keys "order_id" and "pizza_id" together formed a candidate key for this relationship, allowing us to find a quantity. This change was necessary because, for instance, it is not possible to order two large pepperoni pizzas in a single order.

**Normalization of the generated schema:**

In pursuit of normalization, we have decided as a group to remove the column from the scheme **ingredients**, since for each tuple there is a list of ingredients and we decided as a group to remove the column from the schema **order_details_id** since the foreign keys of that entity served as the candidate key.

It is in Boyce-Codd Normal Form (BCNF) since each entity has only one primary key, and this key functionally determines each of the attributes within that entity. As for the relationships, no foreign key functionally determines the other, so it can only be the case that the super key, consisting of the foreign keys, functionally determines them or that the super key functionally determines the "quantity" attribute. Therefore, it is stated to be in BCNF.

## Milestone 3

**SQL scripts used:**

```
/*
Creating the PizzaShop schema to later create tables
*/
CREATE SCHEMA PizzaShop;
ALTER USER cc3201 SET search_path TO PizzaShop, public;
```

```
--Creating tables
--Entities
/*
Create a table named "Orders" in the schema "PizzaShop". The table
has three columns: "order_id" as a primary key of integer type,
"date" as a non-null date, and "time" as a non-null time.
*/
CREATE TABLE PizzaShop.Orders (
      order_id INTEGER PRIMARY KEY,
      date DATE NOT NULL,
      time TIME NOT NULL
);

/*
Create a table named "Pizzas" in the schema "PizzaShop". The table
has three columns: "pizza_id" as a primary key of string type with
a maximum length of 30 characters, "size" as a non-null string
with a maximum length of 3 characters, and "price" as a decimal
greater than or equal to zero, assuming that there can be
promotions with a free product, with a maximum of two digits to
the right of the decimal point and a maximum of ten digits to the
left of the decimal point, considering that a pizza price will not
exceed ten digits.
*/
CREATE TABLE PizzaShop.Pizzas (
      pizza_id VARCHAR(30) PRIMARY KEY,
      size VARCHAR(3) NOT NULL,
      price DECIMAL(10, 2) CHECK (price >= 0)
);

/*
```

```
Create a table named "Pizza_Types" in the schema "PizzaShop". The
table has three columns: "pizza_type_id" as a primary key of
string type with a maximum length of 30 characters, "name" as a
non-null string with a maximum length of 100 characters, and
"category" as a non-null string with a maximum length of 30
characters.
*/
CREATE TABLE PizzaShop.Pizza_Types (
      pizza_type_id VARCHAR(30) PRIMARY KEY,
      name VARCHAR(100) NOT NULL,
      category VARCHAR(30) NOT NULL
);

--Relationships
/*
Create a table named "Order_Details" in the schema "PizzaShop".
The table has three columns: "order_id" as a foreign key
referencing the "Orders" table, "pizza_id" as a foreign key
referencing the "Pizzas" table, and "quantity" as an integer
greater than or equal to 1.
*/
CREATE TABLE PizzaShop.Order_Details (
      order_id INTEGER REFERENCES Orders(order_id),
      pizza_id VARCHAR(30) REFERENCES Pizzas(pizza_id),
      quantity INTEGER CHECK (quantity >= 1)
);

/*
Create a table named "Pizza_Details" in the schema "PizzaShop".
The table has two columns: "pizza_id" as a foreign key referencing
the "Pizzas" table, and "pizza_type_id" as a foreign key
referencing the "Pizza_Types" table.
*/
CREATE TABLE PizzaShop.Pizza_Details (
      pizza_id VARCHAR(30) REFERENCES Pizzas(pizza_id),
      pizza_type_id VARCHAR(30) REFERENCES
Pizza_Types(pizza_type_id)
);
```

```
--Insert data
```

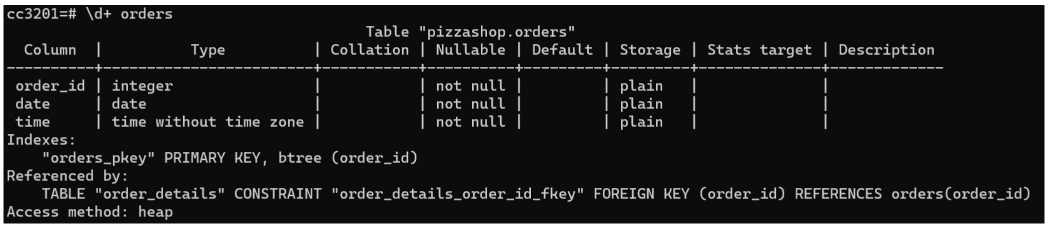
```
--Entities
--Orders
/*
Load the data from the "orders.csv" file into the "Orders" table
of the database.
*/
\copy orders (order_id, date, time) FROM 'orders.csv' DELIMITER
',' CSV HEADER;

--Pizzas
/*
First, we create a temporary table with the same data types as the
"pizzas.csv" file.
*/
CREATE TABLE tabla_temporal (pizza_id VARCHAR(30), pizza_type_id
VARCHAR(30), size VARCHAR(3), price numeric(10,2));

/*
Then, we load the data from the "pizzas.csv" file into the
temporary table.
*/
\copy tabla_temporal (pizza_id, pizza_type_id, size, price) FROM
'pizzas.csv' DELIMITER ',' CSV HEADER;

/*
Then, we copy the data that we want to keep from the temporary
table to the "Pizzas" table.
*/
INSERT INTO pizzas (pizza_id, size, price)
SELECT pizza_id, size, price
FROM tabla_temporal;

/*
And finally, we delete the temporary table.
*/
DROP TABLE tabla_temporal;

--Pizza_types
/*
Load the data from the "pizza_types.csv" file into the
"Pizza_Types" table of the database.
*/
```

```
\copy pizza_types (pizza_type_id, name, category) FROM
'pizza_types.csv' DELIMITER ',' CSV HEADER;

--Relationships
--Order_details
/*
First, we create a temporary table with the same data types as the
"order_details.csv" file.
*/
CREATE TABLE tabla_temporal (order_details_id INTEGER, order_id
INTEGER, pizza_id VARCHAR(30), quantity INTEGER);

/*
Then, we load the data from the "order_details.csv" file into the
temporary table.
*/
\copy tabla_temporal (order_details_id, order_id, pizza_id,
quantity) FROM 'order_details.csv' DELIMITER ',' CSV HEADER;

/*
Then, we copy the data that we want to keep from the temporary
table to the "Order_Details" table.
*/
INSERT INTO order_details (order_id, pizza_id, quantity)
SELECT order_id, pizza_id, quantity
FROM tabla_temporal;

/*
And finally, we delete the temporary table.
*/
DROP TABLE tabla_temporal;

--Pizza_details
/*
First, we create a temporary table with the same data types as the
"pizzas.csv" file.
*/
CREATE TABLE tabla_temporal (pizza_id VARCHAR(30), pizza_type_id
VARCHAR(30), size VARCHAR(3), price numeric(10,2));

/*
Then, we load the data from the "pizzas.csv" file into the
```

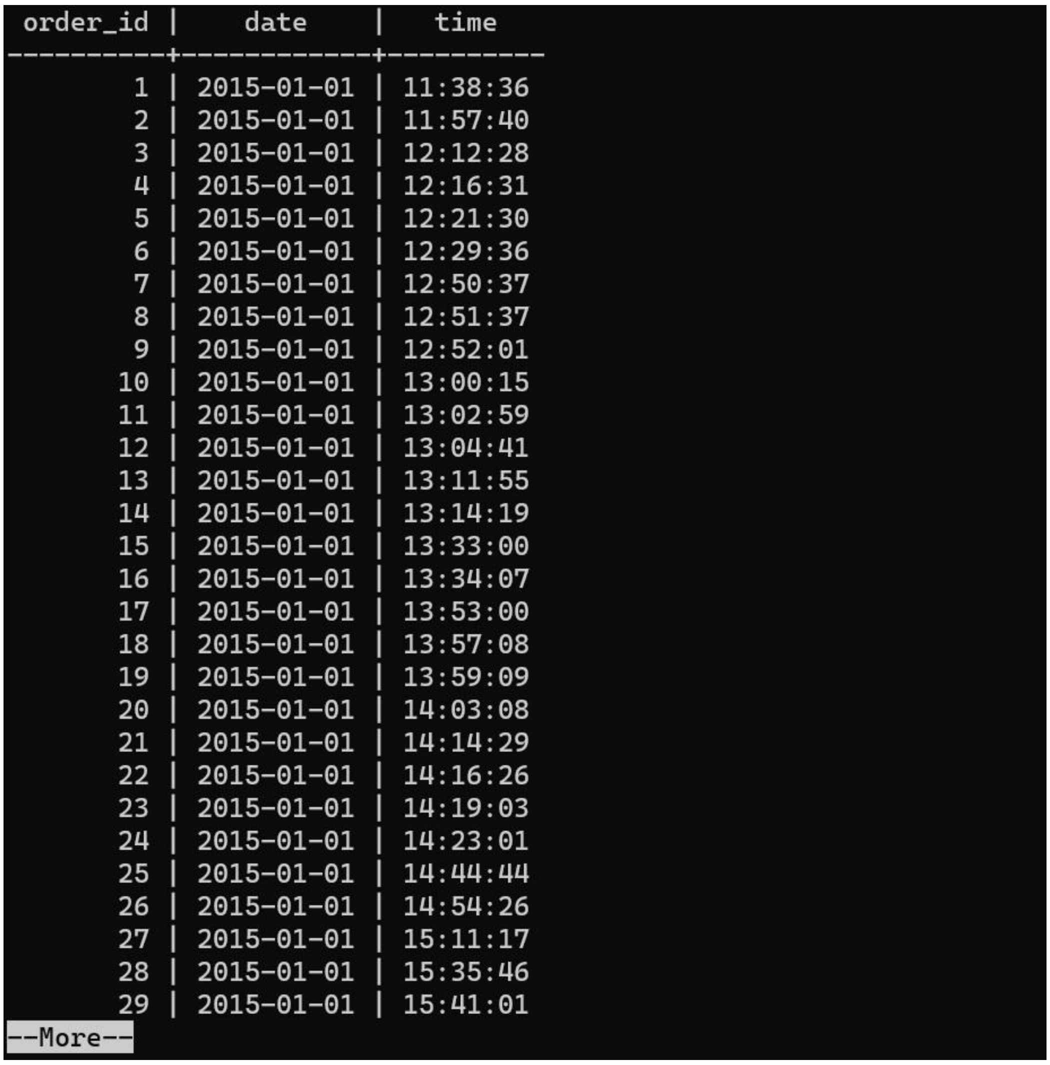
```
temporary table.
*/
\copy tabla_temporal (pizza_id, pizza_type_id, size, price) FROM
'pizzas.csv' DELIMITER ',' CSV HEADER;

/*
Then, we copy the data that we want to keep from the temporary
table to the "Pizza_Details" table.
*/
INSERT INTO pizza_details (pizza_id, pizza_type_id)
SELECT pizza_id, pizza_type_id
FROM tabla_temporal;

/*
And finally, we delete the temporary table.
*/
DROP TABLE tabla_temporal;
```

**End State Description:**

- **orders:** This table has a total of 21.350 rows, no tuples were removed from the original table, nor its data, so it only required copying and pasting the data into the database table.

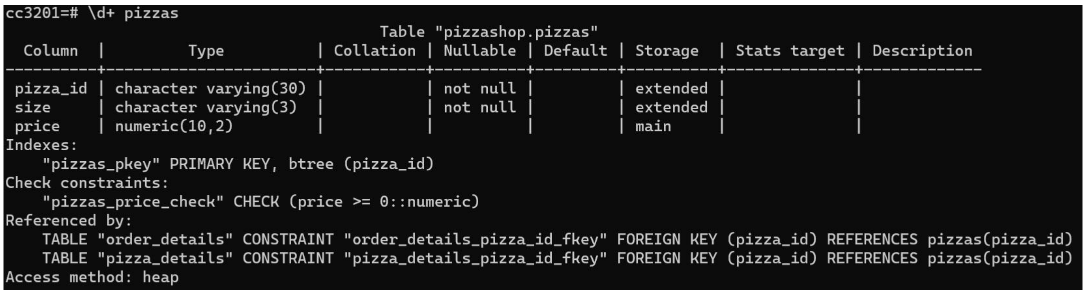


```
cc3201=# \d+ orders
                                        Table "pizzashop.orders"
  Column  |          Type          | Collation | Nullable | Default | Storage | Stats target | Description
----------+------------------------+-----------+----------+---------+---------+--------------+-------------
 order_id | integer                |           | not null |         | plain   |              |
 date     | date                   |           | not null |         | plain   |              |
 time     | time without time zone |           | not null |         | plain   |              |
Indexes:
    "orders_pkey" PRIMARY KEY, btree (order_id)
Referenced by:
    TABLE "order_details" CONSTRAINT "order_details_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
Access method: heap
```

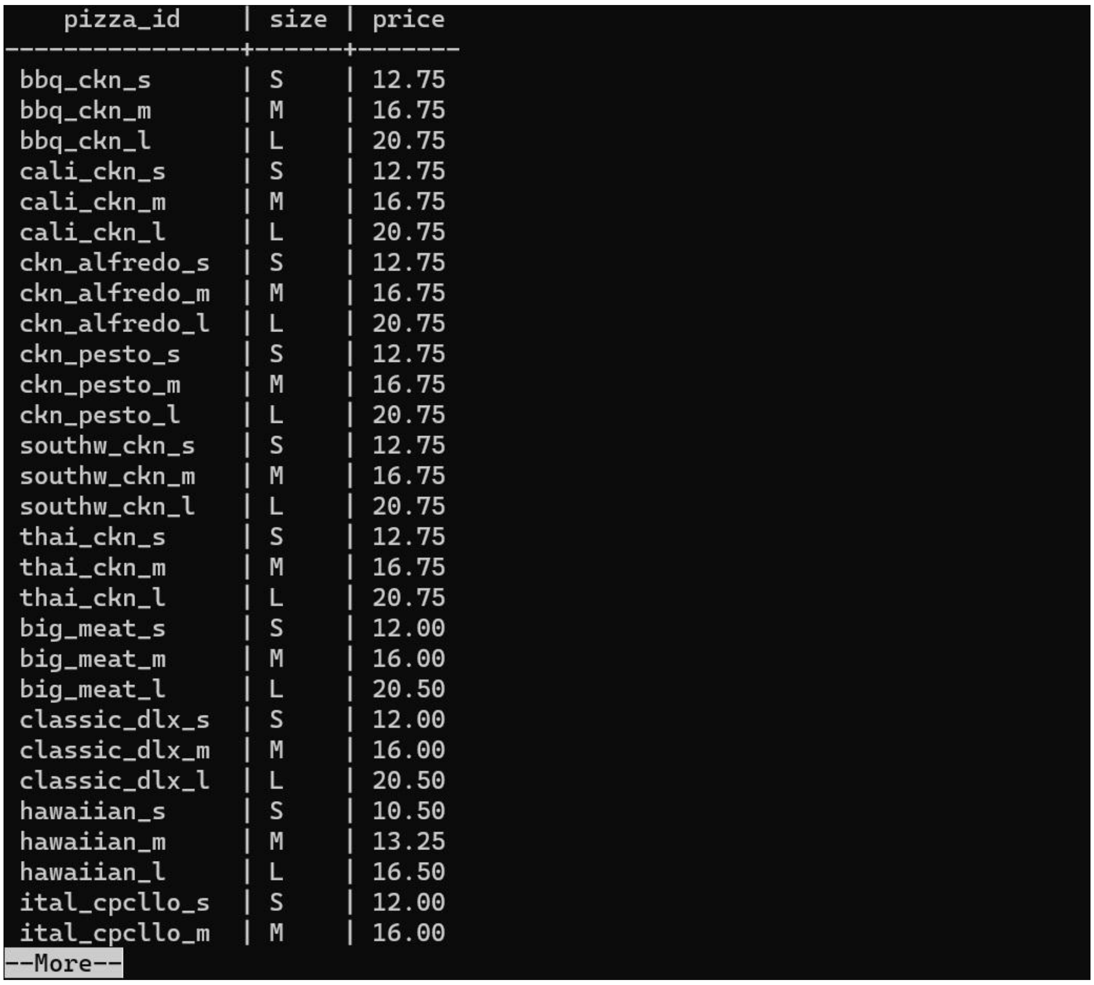


```
 order_id |    date    |   time
----------+------------+----------
        1 | 2015-01-01 | 11:38:36
        2 | 2015-01-01 | 11:57:40
        3 | 2015-01-01 | 12:12:28
        4 | 2015-01-01 | 12:16:31
        5 | 2015-01-01 | 12:21:30
        6 | 2015-01-01 | 12:29:36
        7 | 2015-01-01 | 12:50:37
        8 | 2015-01-01 | 12:51:37
        9 | 2015-01-01 | 12:52:01
       10 | 2015-01-01 | 13:00:15
       11 | 2015-01-01 | 13:02:59
       12 | 2015-01-01 | 13:04:41
       13 | 2015-01-01 | 13:11:55
       14 | 2015-01-01 | 13:14:19
       15 | 2015-01-01 | 13:33:00
       16 | 2015-01-01 | 13:34:07
       17 | 2015-01-01 | 13:53:00
       18 | 2015-01-01 | 13:57:08
       19 | 2015-01-01 | 13:59:09
       20 | 2015-01-01 | 14:03:08
       21 | 2015-01-01 | 14:14:29
       22 | 2015-01-01 | 14:16:26
       23 | 2015-01-01 | 14:19:03
       24 | 2015-01-01 | 14:23:01
       25 | 2015-01-01 | 14:44:44
       26 | 2015-01-01 | 14:54:26
       27 | 2015-01-01 | 15:11:17
       28 | 2015-01-01 | 15:35:46
       29 | 2015-01-01 | 15:41:01
--More--
```

- **pizzas:** This table has a total of 96 rows, the pizza_type_id column was removed in order to normalize the table and avoid redundancies, but it was not removed in the original table as the data was written in the same cell, so a table was created temp to store all the data from the original table and then added to the pizzas table all the data from the temp table except for the data in the pizza_type_id column.

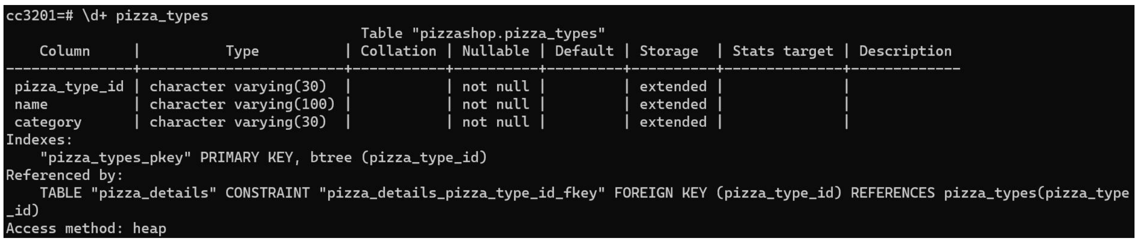


```
cc3201=# \d+ pizzas
                                              Table "pizzashop.pizzas"
  Column  |         Type          | Collation | Nullable | Default | Storage  | Stats target | Description
----------+-----------------------+-----------+----------+---------+----------+--------------+-------------
 pizza_id | character varying(30) |           | not null |         | extended |              |
 size     | character varying(3)  |           | not null |         | extended |              |
 price    | numeric(10,2)         |           |          |         | main     |              |
Indexes:
    "pizzas_pkey" PRIMARY KEY, btree (pizza_id)
Check constraints:
    "pizzas_price_check" CHECK (price >= 0::numeric)
Referenced by:
    TABLE "order_details" CONSTRAINT "order_details_pizza_id_fkey" FOREIGN KEY (pizza_id) REFERENCES pizzas(pizza_id)
    TABLE "pizza_details" CONSTRAINT "pizza_details_pizza_id_fkey" FOREIGN KEY (pizza_id) REFERENCES pizzas(pizza_id)
Access method: heap
```

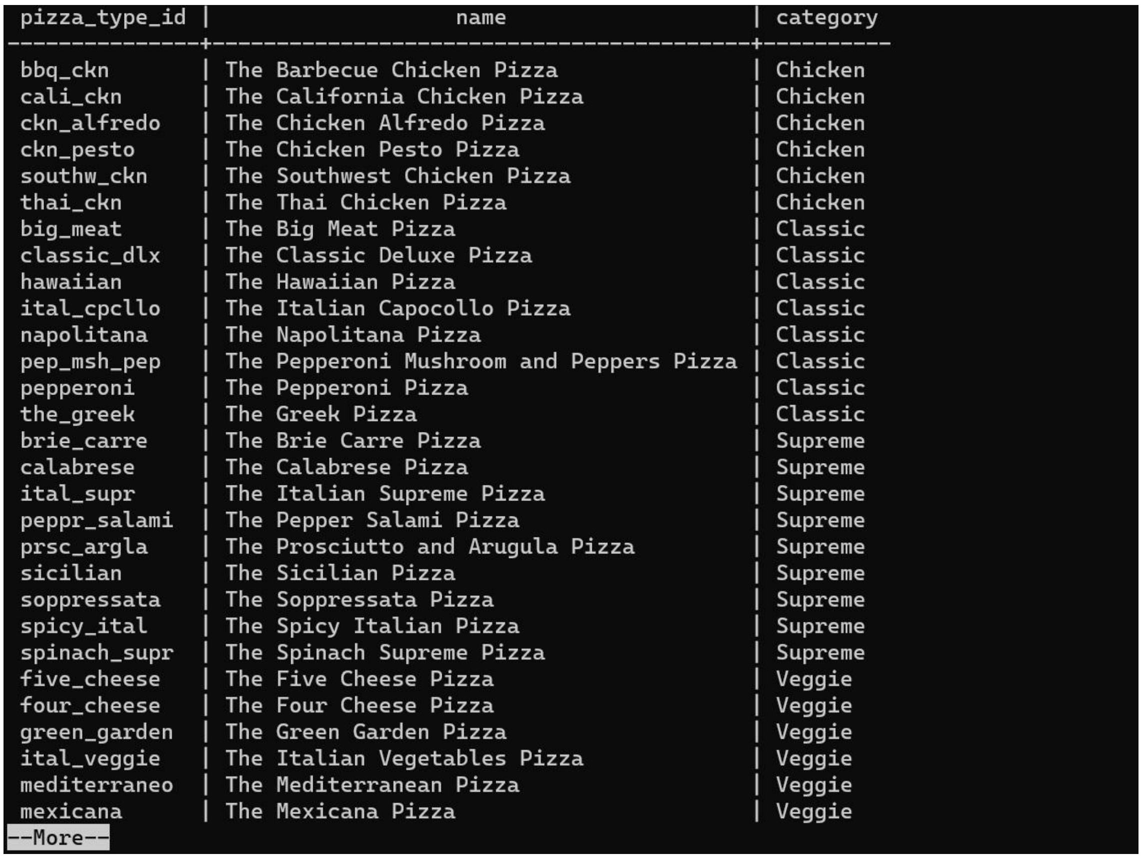


```
    pizza_id     | size | price
-----------------+------+-------
 bbq_ckn_s       | S    | 12.75
 bbq_ckn_m       | M    | 16.75
 bbq_ckn_l       | L    | 20.75
 cali_ckn_s      | S    | 12.75
 cali_ckn_m      | M    | 16.75
 cali_ckn_l      | L    | 20.75
 ckn_alfredo_s   | S    | 12.75
 ckn_alfredo_m   | M    | 16.75
 ckn_alfredo_l   | L    | 20.75
 ckn_pesto_s     | S    | 12.75
 ckn_pesto_m     | M    | 16.75
 ckn_pesto_l     | L    | 20.75
 southw_ckn_s    | S    | 12.75
 southw_ckn_m    | M    | 16.75
 southw_ckn_l    | L    | 20.75
 thai_ckn_s      | S    | 12.75
 thai_ckn_m      | M    | 16.75
 thai_ckn_l      | L    | 20.75
 big_meat_s      | S    | 12.00
 big_meat_m      | M    | 16.00
 big_meat_l      | L    | 20.50
 classic_dlx_s   | S    | 12.00
 classic_dlx_m   | M    | 16.00
 classic_dlx_l   | L    | 20.50
 hawaiian_s      | S    | 10.50
 hawaiian_m      | M    | 13.25
 hawaiian_l      | L    | 16.50
 ital_cpcllo_s   | S    | 12.00
 ital_cpcllo_m   | M    | 16.00
--More--
```

- **pizza_types:** This table has a total of 32 rows, the ingredients column was removed from the original table as it caused problems with the "," of the ingredients, as it is a table with few data, that column was manually removed and the name data was modified line 13 of the original table since it had the same formatting problem as the ingredients data (the commas and quotes were removed from the data), and after that all you had to do was copy and paste the data in the table of the database.

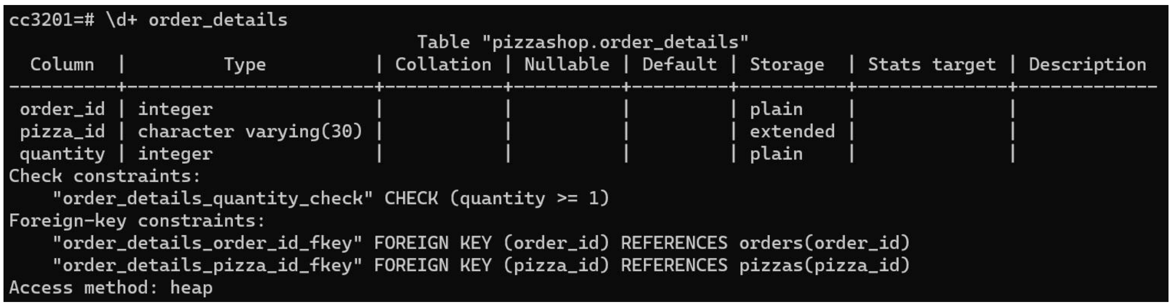


```
cc3201=# \d+ pizza_types
                                        Table "pizzashop.pizza_types"
    Column     |          Type          | Collation | Nullable | Default | Storage  | Stats target | Description
---------------+------------------------+-----------+----------+---------+----------+--------------+-------------
 pizza_type_id | character varying(30)  |           | not null |         | extended |              |
 name          | character varying(100) |           | not null |         | extended |              |
 category      | character varying(30)  |           | not null |         | extended |              |
Indexes:
    "pizza_types_pkey" PRIMARY KEY, btree (pizza_type_id)
Referenced by:
    TABLE "pizza_details" CONSTRAINT "pizza_details_pizza_type_id_fkey" FOREIGN KEY (pizza_type_id) REFERENCES pizza_types(pizza_type
_id)
Access method: heap
```

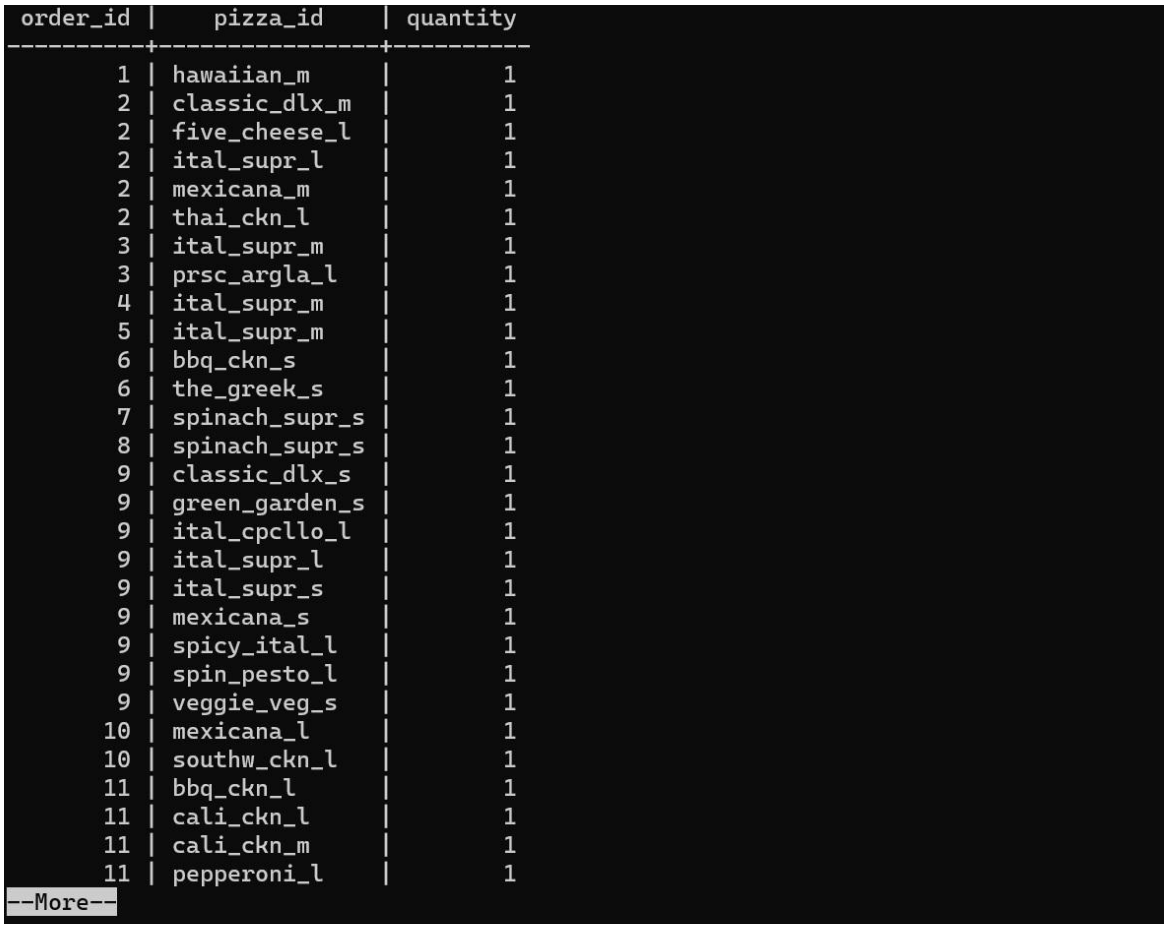


```
 pizza_type_id |                   name                     | category
---------------+--------------------------------------------+----------
 bbq_ckn       | The Barbecue Chicken Pizza                 | Chicken
 cali_ckn      | The California Chicken Pizza               | Chicken
 ckn_alfredo   | The Chicken Alfredo Pizza                  | Chicken
 ckn_pesto     | The Chicken Pesto Pizza                    | Chicken
 southw_ckn    | The Southwest Chicken Pizza                | Chicken
 thai_ckn      | The Thai Chicken Pizza                     | Chicken
 big_meat      | The Big Meat Pizza                         | Classic
 classic_dlx   | The Classic Deluxe Pizza                   | Classic
 hawaiian      | The Hawaiian Pizza                         | Classic
 ital_cpcllo   | The Italian Capocollo Pizza                | Classic
 napolitana    | The Napolitana Pizza                       | Classic
 pep_msh_pep   | The Pepperoni Mushroom and Peppers Pizza   | Classic
 pepperoni     | The Pepperoni Pizza                        | Classic
 the_greek     | The Greek Pizza                            | Classic
 brie_carre    | The Brie Carre Pizza                       | Supreme
 calabrese     | The Calabrese Pizza                        | Supreme
 ital_supr     | The Italian Supreme Pizza                  | Supreme
 peppr_salami  | The Pepper Salami Pizza                    | Supreme
 prsc_argla    | The Prosciutto and Arugula Pizza           | Supreme
 sicilian      | The Sicilian Pizza                         | Supreme
 soppressata   | The Soppressata Pizza                      | Supreme
 spicy_ital    | The Spicy Italian Pizza                    | Supreme
 spinach_supr  | The Spinach Supreme Pizza                  | Supreme
 five_cheese   | The Five Cheese Pizza                      | Veggie
 four_cheese   | The Four Cheese Pizza                      | Veggie
 green_garden  | The Green Garden Pizza                     | Veggie
 ital_veggie   | The Italian Vegetables Pizza               | Veggie
 mediterraneo  | The Mediterranean Pizza                    | Veggie
 mexicana      | The Mexicana Pizza                         | Veggie
--More--
```

- **order_details:** This table has a total of 48.620 rows, the order_details_id column was removed as it was useless data and the same methodology was used as for the pizzas table for the same reasons as above, a temporary table was created, all data was copied to that table and then copied the data from that table to order_details except for order_details_id.

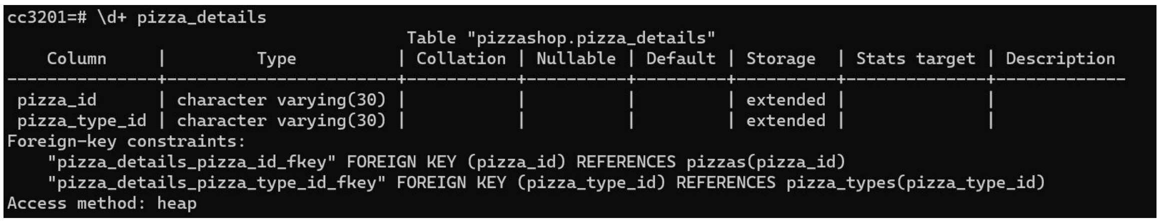


```
cc3201=# \d+ order_details
                                      Table "pizzashop.order_details"
  Column  |         Type          | Collation | Nullable | Default | Storage  | Stats target | Description
----------+-----------------------+-----------+----------+---------+----------+--------------+-------------
 order_id | integer               |           |          |         | plain    |              |
 pizza_id | character varying(30) |           |          |         | extended |              |
 quantity | integer               |           |          |         | plain    |              |
Check constraints:
    "order_details_quantity_check" CHECK (quantity >= 1)
Foreign-key constraints:
    "order_details_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
    "order_details_pizza_id_fkey" FOREIGN KEY (pizza_id) REFERENCES pizzas(pizza_id)
Access method: heap
```

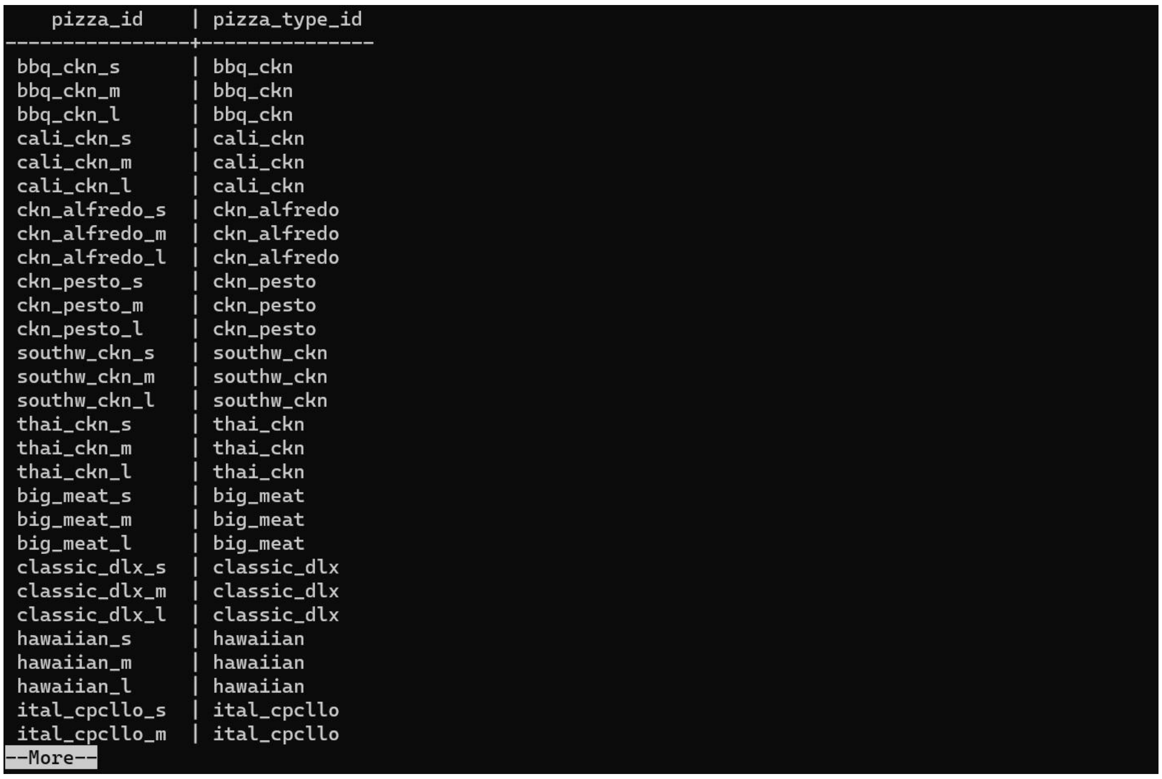


```
 order_id |    pizza_id     | quantity
----------+-----------------+----------
        1 | hawaiian_m      |        1
        2 | classic_dlx_m   |        1
        2 | five_cheese_l   |        1
        2 | ital_supr_l     |        1
        2 | mexicana_m      |        1
        2 | thai_ckn_l      |        1
        3 | ital_supr_m     |        1
        3 | prsc_argla_l    |        1
        4 | ital_supr_m     |        1
        5 | ital_supr_m     |        1
        6 | bbq_ckn_s       |        1
        6 | the_greek_s     |        1
        7 | spinach_supr_s  |        1
        8 | spinach_supr_s  |        1
        9 | classic_dlx_s   |        1
        9 | green_garden_s  |        1
        9 | ital_cpcllo_l   |        1
        9 | ital_supr_l     |        1
        9 | ital_supr_s     |        1
        9 | mexicana_s      |        1
        9 | spicy_ital_l    |        1
        9 | spin_pesto_l    |        1
        9 | veggie_veg_s    |        1
       10 | mexicana_l      |        1
       10 | southw_ckn_l    |        1
       11 | bbq_ckn_l       |        1
       11 | cali_ckn_l      |        1
       11 | cali_ckn_m      |        1
       11 | pepperoni_l     |        1
--More--
```

- **pizza_details:** This table has a total of 96 rows, which are taken from the original table pizzas, where the same methodology was followed, a temporary table was created, the pizza data was copied to that temporary table and later only the data of pizza_id and pizza_type_id to the pizza_details table.



```
cc3201=# \d+ pizza_details
                                        Table "pizzashop.pizza_details"
    Column     |         Type          | Collation | Nullable | Default | Storage  | Stats target | Description
---------------+-----------------------+-----------+----------+---------+----------+--------------+-------------
 pizza_id      | character varying(30) |           |          |         | extended |              |
 pizza_type_id | character varying(30) |           |          |         | extended |              |
Foreign-key constraints:
    "pizza_details_pizza_id_fkey" FOREIGN KEY (pizza_id) REFERENCES pizzas(pizza_id)
    "pizza_details_pizza_type_id_fkey" FOREIGN KEY (pizza_type_id) REFERENCES pizza_types(pizza_type_id)
Access method: heap
```

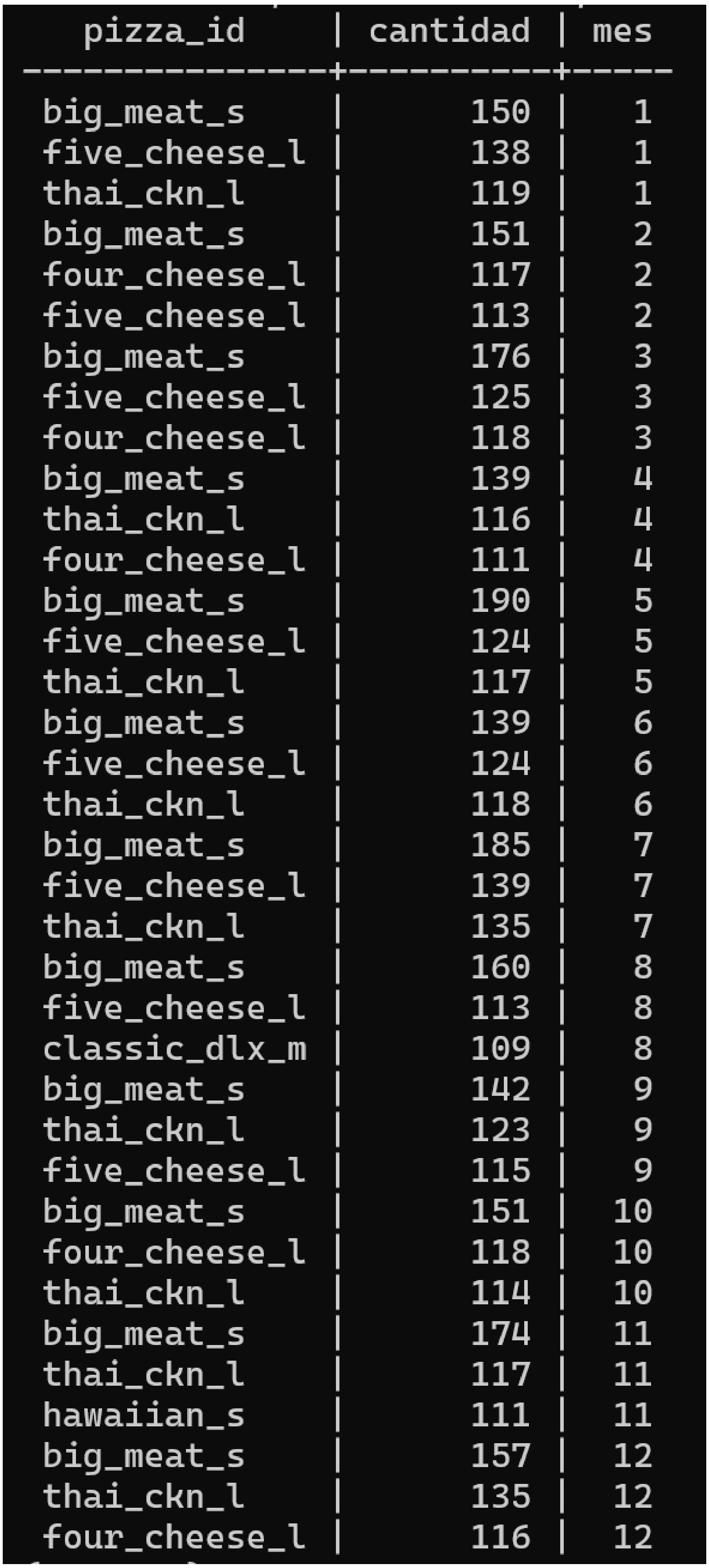


```
    pizza_id     | pizza_type_id
-----------------+---------------
 bbq_ckn_s       | bbq_ckn
 bbq_ckn_m       | bbq_ckn
 bbq_ckn_l       | bbq_ckn
 cali_ckn_s      | cali_ckn
 cali_ckn_m      | cali_ckn
 cali_ckn_l      | cali_ckn
 ckn_alfredo_s   | ckn_alfredo
 ckn_alfredo_m   | ckn_alfredo
 ckn_alfredo_l   | ckn_alfredo
 ckn_pesto_s     | ckn_pesto
 ckn_pesto_m     | ckn_pesto
 ckn_pesto_l     | ckn_pesto
 southw_ckn_s    | southw_ckn
 southw_ckn_m    | southw_ckn
 southw_ckn_l    | southw_ckn
 thai_ckn_s      | thai_ckn
 thai_ckn_m      | thai_ckn
 thai_ckn_l      | thai_ckn
 big_meat_s      | big_meat
 big_meat_m      | big_meat
 big_meat_l      | big_meat
 classic_dlx_s   | classic_dlx
 classic_dlx_m   | classic_dlx
 classic_dlx_l   | classic_dlx
 hawaiian_s      | hawaiian
 hawaiian_m      | hawaiian
 hawaiian_l      | hawaiian
 ital_cpcllo_s   | ital_cpcllo
 ital_cpcllo_m   | ital_cpcllo
--More--
```

**Queries to use:**

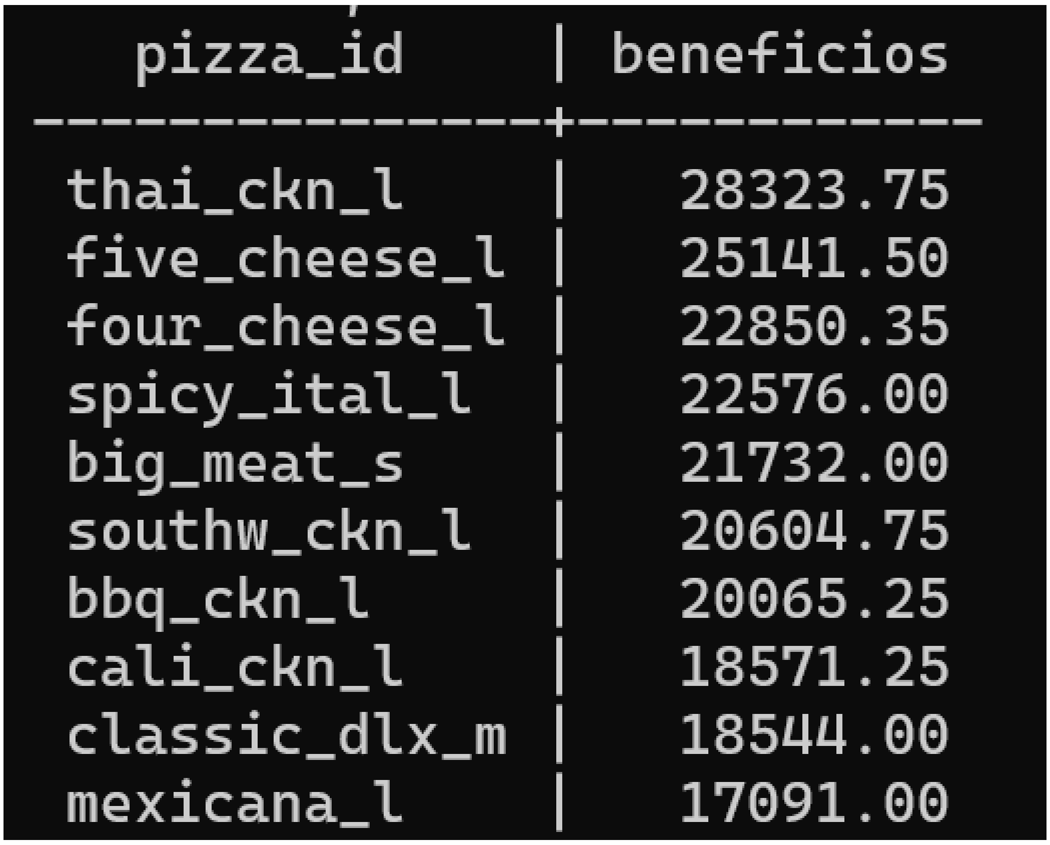
1) What are the 3 best-selling types of pizza each month?

```
SELECT DISTINCT pizza_id, cantidad, mes
FROM (
    SELECT pizza_id, sum(quantity) AS cantidad, EXTRACT(MONTH FROM
date) AS mes,
           ROW_NUMBER() OVER (PARTITION BY EXTRACT(MONTH FROM
date) ORDER BY sum(quantity) DESC) AS num
    FROM order_details
    JOIN orders ON order_details.order_id = orders.order_id
    GROUP BY pizza_id, EXTRACT(MONTH FROM date)
) AS subconsulta
WHERE num <= 3
ORDER BY mes, cantidad DESC;

-- Planning Time: 0.607 ms
-- Execution Time: 242.247 ms
```


```
   pizza_id      | cantidad | mes
-----------------+----------+-----
 big_meat_s      |      150 |   1
 five_cheese_l   |      138 |   1
 thai_ckn_l      |      119 |   1
 big_meat_s      |      151 |   2
 four_cheese_l   |      117 |   2
 five_cheese_l   |      113 |   2
 big_meat_s      |      176 |   3
 five_cheese_l   |      125 |   3
 four_cheese_l   |      118 |   3
 big_meat_s      |      139 |   4
 thai_ckn_l      |      116 |   4
 four_cheese_l   |      111 |   4
 big_meat_s      |      190 |   5
 five_cheese_l   |      124 |   5
 thai_ckn_l      |      117 |   5
 big_meat_s      |      139 |   6
 five_cheese_l   |      124 |   6
 thai_ckn_l      |      118 |   6
 big_meat_s      |      185 |   7
 five_cheese_l   |      139 |   7
 thai_ckn_l      |      135 |   7
 big_meat_s      |      160 |   8
 five_cheese_l   |      113 |   8
 classic_dlx_m   |      109 |   8
 big_meat_s      |      142 |   9
 thai_ckn_l      |      123 |   9
 five_cheese_l   |      115 |   9
 big_meat_s      |      151 |  10
 four_cheese_l   |      118 |  10
 thai_ckn_l      |      114 |  10
 big_meat_s      |      174 |  11
 thai_ckn_l      |      117 |  11
 hawaiian_s      |      111 |  11
 big_meat_s      |      157 |  12
 thai_ckn_l      |      135 |  12
 four_cheese_l   |      116 |  12
```

**2)** What are the pizzas with the greatest benefits?

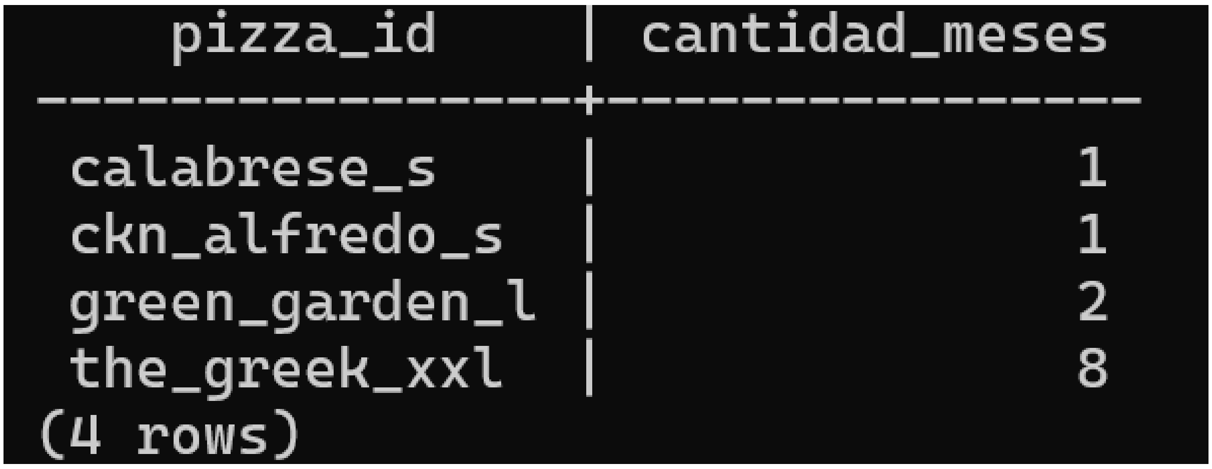
```
SELECT pizza_id, price * cantidad as BENEFICIOS
FROM pizzas, (
    SELECT pizza_id AS nombre, count(pizza_id) AS cantidad
    FROM order_details Group by pizza_id
) as subconsulta
WHERE nombre = pizza_id
Order by BENEFICIOS desc
LIMIT 10 ;

-- Planning Time: 0.338 ms
-- Execution Time: 93.703 ms
```



```
   pizza_id       | beneficios
------------------+-------------
 thai_ckn_l       |   28323.75
 five_cheese_l    |   25141.50
 four_cheese_l    |   22850.35
 spicy_ital_l     |   22576.00
 big_meat_s       |   21732.00
 southw_ckn_l     |   20604.75
 bbq_ckn_l        |   20065.25
 cali_ckn_l       |   18571.25
 classic_dlx_m    |   18544.00
 mexicana_l       |   17091.00
```

**3)** For how many months was each type of pizza the least sold?

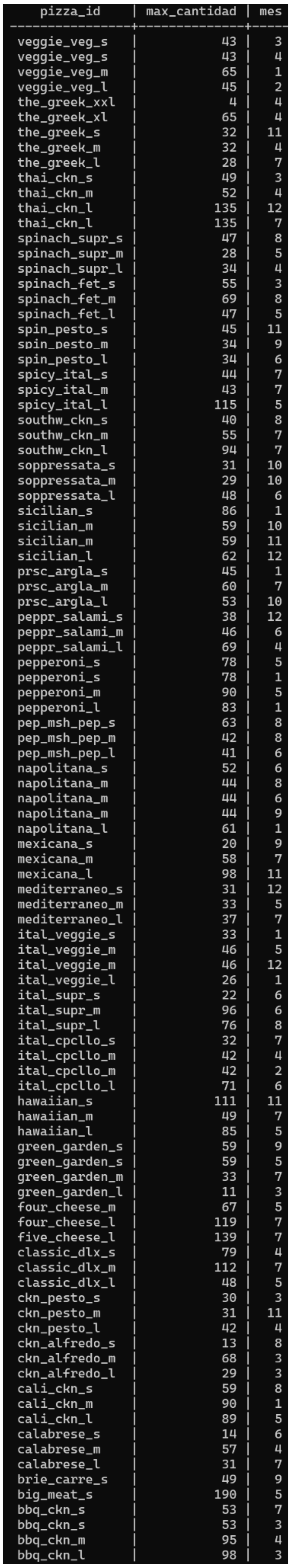
```
SELECT pizza_id, COUNT(*) AS cantidad_meses
FROM (
    SELECT DISTINCT pizza_id, cantidad, mes
    FROM (
        SELECT pizza_id, sum(quantity) AS cantidad, EXTRACT(MONTH
FROM date) AS mes,
               ROW_NUMBER() OVER (PARTITION BY EXTRACT(MONTH FROM
date) ORDER BY sum(quantity) ASC) AS num
        FROM order_details
        JOIN orders ON order_details.order_id = orders.order_id
        GROUP BY pizza_id, EXTRACT(MONTH FROM date)
    ) AS subconsulta
    WHERE num = 1
) AS pizzas_menos_vendidas
GROUP BY pizza_id;

 -- Planning Time: 0.534 ms
 -- Execution Time: 221.879 ms
```



```
    pizza_id      | cantidad_meses
------------------+----------------
 calabrese_s      |              1
 ckn_alfredo_s    |              1
 green_garden_l   |              2
 the_greek_xxl    |              8
(4 rows)
```

**4)** In which month did each type of pizza reach its peak sales?

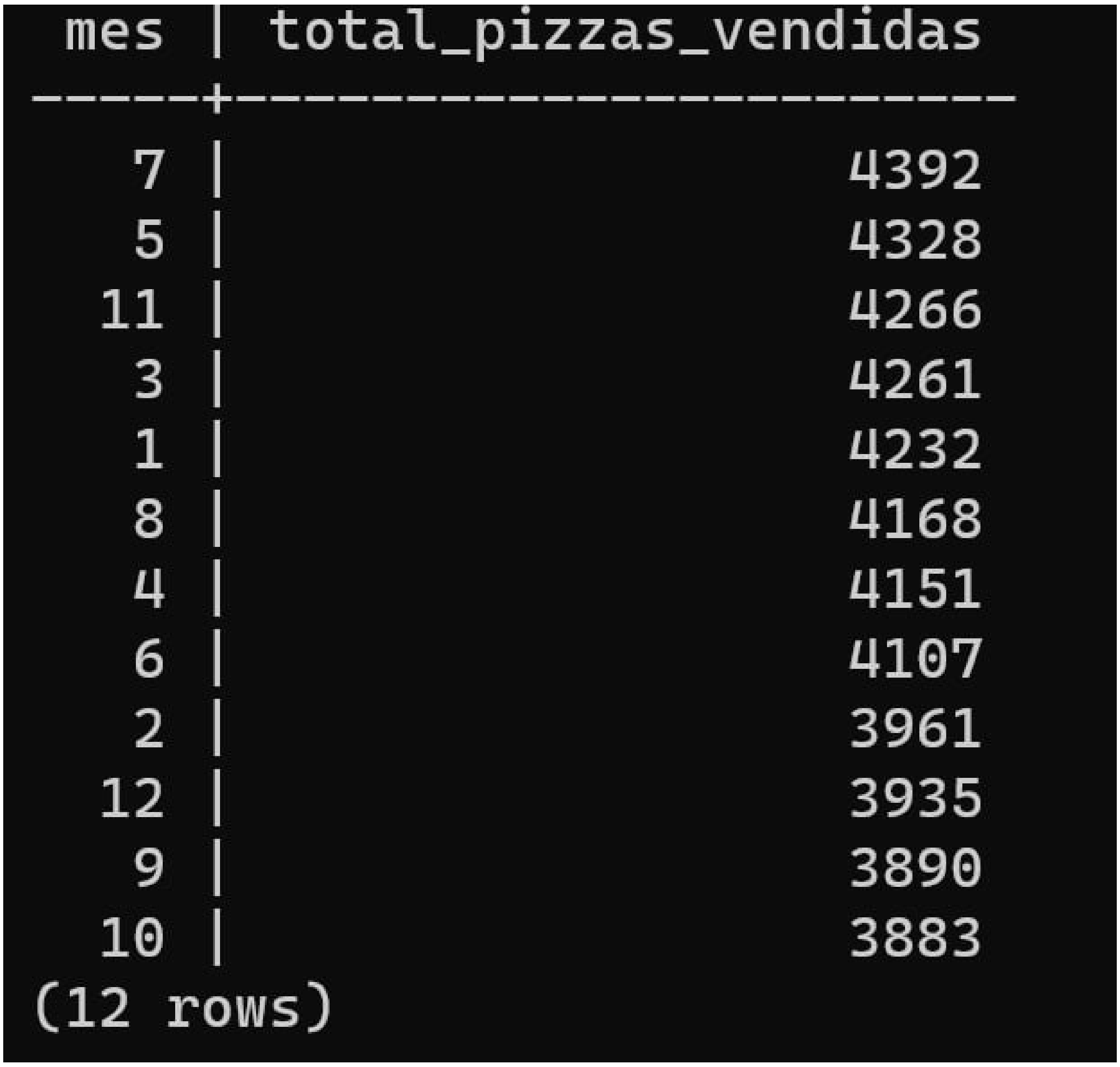
```
SELECT pizza_id, max_cantidad, mes
FROM (
    SELECT pizza_id, MAX(cantidad) AS max_cantidad, mes
    FROM (
        SELECT pizza_id, sum(quantity) AS cantidad, EXTRACT(MONTH
FROM date) AS mes,
               MAX(sum(quantity)) OVER (PARTITION BY pizza_id
ORDER BY EXTRACT(MONTH FROM date)) AS max_cantidad_por_pizza
        FROM order_details
        JOIN orders ON order_details.order_id = orders.order_id
        GROUP BY pizza_id, EXTRACT(MONTH FROM date)
    ) AS subconsulta
    WHERE cantidad = max_cantidad_por_pizza
    GROUP BY pizza_id, mes
) AS max_pizzas
WHERE (pizza_id, max_cantidad) IN (
    SELECT pizza_id, MAX(max_cantidad) FROM (
        SELECT pizza_id, MAX(cantidad) AS max_cantidad, mes
        FROM (
            SELECT pizza_id, sum(quantity) AS cantidad,
EXTRACT(MONTH FROM date) AS mes,
                   MAX(sum(quantity)) OVER (PARTITION BY pizza_id
ORDER BY EXTRACT(MONTH FROM date)) AS max_cantidad_por_pizza
            FROM order_details
            JOIN orders ON order_details.order_id =
orders.order_id
            GROUP BY pizza_id, EXTRACT(MONTH FROM date)
        ) AS subconsulta
        WHERE cantidad = max_cantidad_por_pizza
        GROUP BY pizza_id, mes
    ) AS max_pizzas_otro
    GROUP BY pizza_id
)
ORDER BY pizza_id DESC, max_cantidad DESC;

-- Planning Time: 1.023 ms
-- Execution Time: 462.481 ms
```



```
    pizza_id     | max_cantidad | mes
-----------------+--------------+----
 veggie_veg_s    |           43 |   3
 veggie_veg_s    |           43 |   4
 veggie_veg_m    |           65 |   1
 veggie_veg_l    |           45 |   2
 the_greek_xxl   |            4 |   4
 the_greek_xl    |           65 |   4
 the_greek_s     |           32 |  11
 the_greek_m     |           32 |   4
 the_greek_l     |           28 |   7
 thai_ckn_s      |           49 |   3
 thai_ckn_m      |           52 |   4
 thai_ckn_l      |          135 |  12
 thai_ckn_l      |          135 |   7
 spinach_supr_s  |           47 |   8
 spinach_supr_m  |           28 |   5
 spinach_supr_l  |           34 |   4
 spinach_fet_s   |           55 |   3
 spinach_fet_m   |           69 |   8
 spinach_fet_l   |           47 |   5
 spin_pesto_s    |           45 |  11
 spin_pesto_m    |           34 |   9
 spin_pesto_l    |           34 |   6
 spicy_ital_s    |           44 |   7
 spicy_ital_m    |           43 |   7
 spicy_ital_l    |          115 |   5
 southw_ckn_s    |           40 |   8
 southw_ckn_m    |           55 |   7
 southw_ckn_l    |           94 |   7
 soppressata_s   |           31 |  10
 soppressata_m   |           29 |  10
 soppressata_l   |           48 |   6
 sicilian_s      |           86 |   1
 sicilian_m      |           59 |  10
 sicilian_m      |           59 |  11
 sicilian_l      |           62 |  12
 prsc_argla_s    |           45 |   1
 prsc_argla_m    |           60 |   7
 prsc_argla_l    |           53 |  10
 peppr_salami_s  |           38 |  12
 peppr_salami_m  |           46 |   6
 peppr_salami_l  |           69 |   4
 pepperoni_s     |           78 |   5
 pepperoni_s     |           78 |   1
 pepperoni_m     |           90 |   5
 pepperoni_l     |           83 |   1
 pep_msh_pep_s   |           63 |   8
 pep_msh_pep_m   |           42 |   8
 pep_msh_pep_l   |           41 |   6
 napolitana_s    |           52 |   6
 napolitana_m    |           44 |   8
 napolitana_m    |           44 |   6
 napolitana_m    |           44 |   9
 napolitana_l    |           61 |   1
 mexicana_s      |           20 |   9
 mexicana_m      |           58 |   7
 mexicana_l      |           98 |  11
 mediterraneo_s  |           31 |  12
 mediterraneo_m  |           33 |   5
 mediterraneo_l  |           37 |   7
 ital_veggie_s   |           33 |   1
 ital_veggie_m   |           46 |   5
 ital_veggie_m   |           46 |  12
 ital_veggie_l   |           26 |   1
 ital_supr_s     |           22 |   6
 ital_supr_m     |           96 |   6
 ital_supr_l     |           76 |   8
 ital_cpcllo_s   |           32 |   7
 ital_cpcllo_m   |           42 |   4
 ital_cpcllo_m   |           42 |   2
 ital_cpcllo_l   |           71 |   6
 hawaiian_s      |          111 |  11
 hawaiian_m      |           49 |   7
 hawaiian_l      |           85 |   5
 green_garden_s  |           59 |   9
 green_garden_s  |           59 |   5
 green_garden_m  |           33 |   7
 green_garden_l  |           11 |   3
 four_cheese_m   |           67 |   5
 four_cheese_l   |          119 |   7
 five_cheese_l   |          139 |   7
 classic_dlx_s   |           79 |   4
 classic_dlx_m   |          112 |   7
 classic_dlx_l   |           48 |   5
 ckn_pesto_s     |           30 |   3
 ckn_pesto_m     |           31 |  11
 ckn_pesto_l     |           42 |   4
 ckn_alfredo_s   |           13 |   8
 ckn_alfredo_m   |           68 |   3
 ckn_alfredo_l   |           29 |   3
 cali_ckn_s      |           59 |   8
 cali_ckn_m      |           90 |   1
 cali_ckn_l      |           89 |   5
 calabrese_s     |           14 |   6
 calabrese_m     |           57 |   4
 calabrese_l     |           31 |   7
 brie_carre_s    |           49 |   9
 big_meat_s      |          190 |   5
 bbq_ckn_s       |           53 |   7
 bbq_ckn_s       |           53 |   3
 bbq_ckn_m       |           95 |   4
 bbq_ckn_l       |           98 |   3
```

**5)** What are the historical monthly sales figures for the pizza shop?

```
SELECT mes, SUM(cantidad) AS total_pizzas_vendidas
FROM (
    SELECT EXTRACT(MONTH FROM date) AS mes, SUM(quantity) AS
cantidad
    FROM order_details
    JOIN orders ON order_details.order_id = orders.order_id
    GROUP BY EXTRACT(MONTH FROM date)
) AS resumen_meses
GROUP BY mes
ORDER BY total_pizzas_vendidas DESC;

-- Planning Time: 0.475 ms
-- Execution Time: 217.786 ms
```

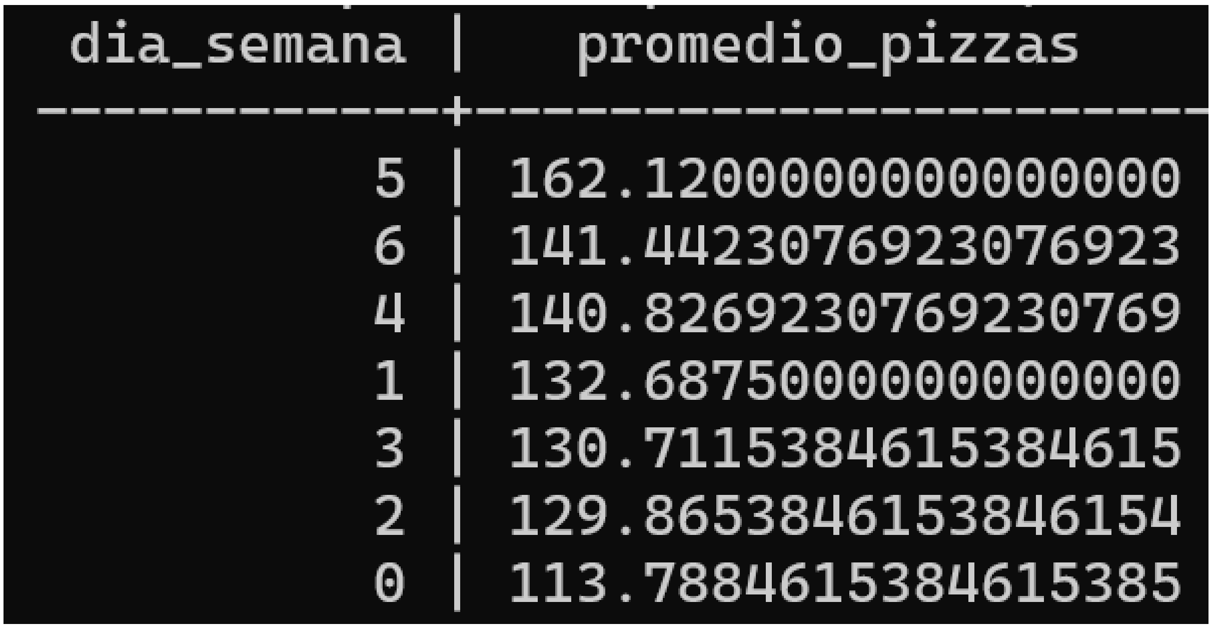


```
 mes | total_pizzas_vendidas
-----+-----------------------
   7 |                  4392
   5 |                  4328
  11 |                  4266
   3 |                  4261
   1 |                  4232
   8 |                  4168
   4 |                  4151
   6 |                  4107
   2 |                  3961
  12 |                  3935
   9 |                  3890
  10 |                  3883
(12 rows)
```

6) What is the average number of pizzas sold for each day of the week? In order to know what is the day of the week with the most sales.

```
SELECT EXTRACT(DOW FROM date) AS dia_semana,
AVG(total_pizzas_vendidas) AS promedio_pizzas
FROM (
    SELECT orders.date, COUNT(order_details.pizza_id) AS
total_pizzas_vendidas
    FROM orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.date
) AS resumen_dias
GROUP BY dia_semana
ORDER BY promedio_pizzas DESC;
```

```
-- Planning Time: 0.411 ms
-- Execution Time: 201.009 ms
```



```
  dia_semana |    promedio_pizzas
 ------------+----------------------
           5 | 162.1200000000000000
           6 | 141.4423076923076923
           4 | 140.8269230769230769
           1 | 132.6875000000000000
           3 | 130.7115384615384615
           2 | 129.8653846153846154
           0 | 113.7884615384615385
```

**Optimizations:**

Since these queries will be frequently used for data analysis, we decided to create materialized views for each of the queries.

```
--Query 1
CREATE MATERIALIZED VIEW consulta_1 AS
SELECT DISTINCT pizza_id, cantidad, mes
FROM (
    SELECT pizza_id, SUM(quantity) AS cantidad, EXTRACT(MONTH FROM
date) AS mes,
           ROW_NUMBER() OVER (PARTITION BY EXTRACT(MONTH FROM
date) ORDER BY SUM(quantity) DESC) AS num
    FROM order_details
    JOIN orders ON order_details.order_id = orders.order_id
    GROUP BY pizza_id, EXTRACT(MONTH FROM date)
) AS subconsulta
WHERE num <= 3
ORDER BY mes, cantidad DESC;
```

```
--To refresh view
REFRESH MATERIALIZED VIEW consulta_1;
```

```
--To use query
SELECT * FROM consulta_1;
```

```
--Query 2
CREATE MATERIALIZED VIEW consulta_2 AS
SELECT pizza_id, price * cantidad AS beneficios
FROM pizzas, (
    SELECT pizza_id AS nombre, COUNT(pizza_id) AS cantidad
    FROM order_details
    GROUP BY pizza_id
) AS subconsulta
WHERE nombre = pizza_id
ORDER BY beneficios DESC
LIMIT 10;
```

```
--To refresh view
REFRESH MATERIALIZED VIEW consulta_2;
```

```
--To use query
SELECT * FROM consulta_2;
```

```
--Query 3
CREATE MATERIALIZED VIEW consulta_3 AS
SELECT pizza_id, COUNT(*) AS cantidad_meses
FROM (
    SELECT DISTINCT pizza_id, cantidad, mes
    FROM (
        SELECT pizza_id, SUM(quantity) AS cantidad, EXTRACT(MONTH
FROM date) AS mes,
               ROW_NUMBER() OVER (PARTITION BY EXTRACT(MONTH FROM
date) ORDER BY SUM(quantity) ASC) AS num
        FROM order_details
        JOIN orders ON order_details.order_id = orders.order_id
        GROUP BY pizza_id, EXTRACT(MONTH FROM date)
    ) AS subconsulta
    WHERE num = 1
) AS pizzas_menos_vendidas
GROUP BY pizza_id;
```

```
--To refresh view
REFRESH MATERIALIZED VIEW consulta_3;
```

```
--To use Query
SELECT * FROM consulta_3;
```

```
--Query 4
CREATE MATERIALIZED VIEW consulta_4 AS
SELECT pizza_id, max_cantidad, mes
FROM (
    SELECT pizza_id, MAX(cantidad) AS max_cantidad, mes
    FROM (
        SELECT pizza_id, sum(quantity) AS cantidad, EXTRACT(MONTH
FROM date) AS mes,
               MAX(sum(quantity)) OVER (PARTITION BY pizza_id
ORDER BY EXTRACT(MONTH FROM date)) AS max_cantidad_por_pizza
        FROM order_details
        JOIN orders ON order_details.order_id = orders.order_id
        GROUP BY pizza_id, EXTRACT(MONTH FROM date)
    ) AS subconsulta
    WHERE cantidad = max_cantidad_por_pizza
    GROUP BY pizza_id, mes
) AS max_pizzas
WHERE (pizza_id, max_cantidad) IN (
    SELECT pizza_id, MAX(max_cantidad) FROM (
        SELECT pizza_id, MAX(cantidad) AS max_cantidad, mes
        FROM (
            SELECT pizza_id, sum(quantity) AS cantidad,
EXTRACT(MONTH FROM date) AS mes,
                   MAX(sum(quantity)) OVER (PARTITION BY pizza_id
ORDER BY EXTRACT(MONTH FROM date)) AS max_cantidad_por_pizza
            FROM order_details
            JOIN orders ON order_details.order_id =
orders.order_id
            GROUP BY pizza_id, EXTRACT(MONTH FROM date)
        ) AS subconsulta
        WHERE cantidad = max_cantidad_por_pizza
        GROUP BY pizza_id, mes
    ) AS max_pizzas_otro
    GROUP BY pizza_id
)
ORDER BY pizza_id DESC, max_cantidad DESC;
```

```
--To refresh view
REFRESH MATERIALIZED VIEW consulta_4;
```

```
--To use Query
SELECT * FROM consulta_4;
```

```
--Query 5
CREATE MATERIALIZED VIEW consulta_5 AS
SELECT mes, SUM(cantidad) AS total_pizzas_vendidas
FROM (
    SELECT EXTRACT(MONTH FROM date) AS mes, SUM(quantity) AS
cantidad
    FROM order_details
    JOIN orders ON order_details.order_id = orders.order_id
    GROUP BY EXTRACT(MONTH FROM date)
) AS resumen_meses
GROUP BY mes
ORDER BY total_pizzas_vendidas DESC;
```

```
--To refresh view
REFRESH MATERIALIZED VIEW consulta_5;
```

```
--To use query
SELECT * FROM consulta_5;
```

```
--Query 6
CREATE MATERIALIZED VIEW consulta_6 AS
SELECT EXTRACT(DOW FROM date) AS dia_semana,
AVG(total_pizzas_vendidas) AS promedio_pizzas
FROM (
    SELECT orders.date, COUNT(order_details.pizza_id) AS
total_pizzas_vendidas
    FROM orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.date
) AS resumen_dias
GROUP BY dia_semana
ORDER BY promedio_pizzas DESC;
```

```
--To refresh view
REFRESH MATERIALIZED VIEW consulta_6;
```

```
--To use query
SELECT * FROM consulta_6;
```

```
--View is created
CREATE MATERIALIZED VIEW consulta_n AS /* Query to optimize */
--To update the view
REFRESH MATERIALIZED VIEW consulta_n;
--To see the query
SELECT * FROM consulta_n;
```

## Milestone 4

**Optimization:**

Doing an analysis of the time taken by the optimized and non-optimized queries, we obtained the following results:

```
--Query 1 (Non-optimized)
EXPLAIN ANALIZE
    SELECT DISTINCT pizza_id, cantidad, mes
    FROM (
        SELECT pizza_id, sum(quantity) AS cantidad, EXTRACT(MONTH
FROM date) AS mes,
               ROW_NUMBER() OVER (PARTITION BY EXTRACT(MONTH FROM
date) ORDER BY sum(quantity) DESC) AS num
        FROM order_details
        JOIN orders ON order_details.order_id = orders.order_id
        GROUP BY pizza_id, EXTRACT(MONTH FROM date)
    ) AS subconsulta
    WHERE num <= 3
    ORDER BY mes, cantidad DESC;

-- Planning Time: 0.607 ms
-- Execution Time: 242.247 ms
```

```
--Query 1 (Optimized)
EXPLAIN ANALIZE
    SELECT * FROM consulta_1;

-- Planning Time: 0.123 ms
-- Execution Time: 0.096 ms
```

```
--Query 2 (Non-optimized)
EXPLAIN ANALIZE
    SELECT pizza_id, price * cantidad AS BENEFICIOS
    FROM pizzas, (
        SELECT pizza_id AS nombre, count(pizza_id) AS cantidad
        FROM order_details Group by pizza_id) AS subconsulta
    WHERE nombre = pizza_id
    Order by BENEFICIOS desc
    LIMIT 10;
```

```
-- Planning Time: 0.338 ms
-- Execution Time: 93.703 ms
```

```
--Query 2 (Optimized)
EXPLAIN ANALIZE
    SELECT * FROM consulta_2;
-- Planning Time: 0.119 ms
-- Execution Time: 0.048 ms
```

```
--Query 3 (Non-optimized)
EXPLAIN ANALIZE
    SELECT pizza_id, COUNT(*) AS cantidad_meses
    FROM (
        SELECT DISTINCT pizza_id, cantidad, mes
        FROM (
            SELECT pizza_id, sum(quantity) AS cantidad,
EXTRACT(MONTH FROM date) AS mes,
                   ROW_NUMBER() OVER (PARTITION BY EXTRACT(MONTH
FROM date) ORDER BY sum(quantity) ASC) AS num
            FROM order_details
            JOIN orders ON order_details.order_id =
orders.order_id
            GROUP BY pizza_id, EXTRACT(MONTH FROM date)
        ) AS subconsulta
        WHERE num = 1
    ) AS pizzas_menos_vendidas
    GROUP BY pizza_id;

 -- Planning Time: 0.534 ms
 -- Execution Time: 221.879 ms
```

```
--Query 3 (Optimized)
EXPLAIN ANALIZE
    SELECT * FROM consulta_3;
-- Planning Time: 0.147 ms
-- Execution Time: 0.037 ms
```

```
--Query 4 (Non-optimized)
EXPLAIN ANALIZE
```

```
    SELECT pizza_id, max_cantidad, mes
    FROM (
        SELECT pizza_id, MAX(cantidad) AS max_cantidad, mes
        FROM (
            SELECT pizza_id, sum(quantity) AS cantidad,
EXTRACT(MONTH FROM date) AS mes,
                   MAX(sum(quantity)) OVER (PARTITION BY pizza_id
ORDER BY EXTRACT(MONTH FROM date)) AS max_cantidad_por_pizza
            FROM order_details
            JOIN orders ON order_details.order_id =
orders.order_id
            GROUP BY pizza_id, EXTRACT(MONTH FROM date)
        ) AS subconsulta
        WHERE cantidad = max_cantidad_por_pizza
        GROUP BY pizza_id, mes
    ) AS max_pizzas
    WHERE (pizza_id, max_cantidad) IN (
        SELECT pizza_id, MAX(max_cantidad) FROM (
            SELECT pizza_id, MAX(cantidad) AS max_cantidad, mes
            FROM (
                SELECT pizza_id, sum(quantity) AS cantidad,
EXTRACT(MONTH FROM date) AS mes,
                       MAX(sum(quantity)) OVER (PARTITION BY
pizza_id ORDER BY EXTRACT(MONTH FROM date)) AS
max_cantidad_por_pizza
                FROM order_details
                JOIN orders ON order_details.order_id =
orders.order_id
                GROUP BY pizza_id, EXTRACT(MONTH FROM date)
            ) AS subconsulta
            WHERE cantidad = max_cantidad_por_pizza
            GROUP BY pizza_id, mes
        ) AS max_pizzas_otro
        GROUP BY pizza_id
    )
    ORDER BY pizza_id DESC, max_cantidad DESC;

-- Planning Time: 1.023 ms
-- Execution Time: 462.481 ms
```

```
--Query 4 (Optimized)
```

```sql
EXPLAIN ANALIZE
    SELECT * FROM consulta_4;
-- Planning Time: 0.121 ms
-- Execution Time: 0.164 ms
```

```sql
--Query 5 (Non-optimized)
EXPLAIN ANALIZE
    SELECT mes, SUM(cantidad) AS total_pizzas_vendidas
    FROM (
        SELECT EXTRACT(MONTH FROM date) AS mes, SUM(quantity) AS
cantidad
        FROM order_details
        JOIN orders ON order_details.order_id = orders.order_id
        GROUP BY EXTRACT(MONTH FROM date)
    ) AS resumen_meses
    GROUP BY mes
    ORDER BY total_pizzas_vendidas DESC;

-- Planning Time: 0.475 ms
-- Execution Time: 217.786 ms
```

```sql
--Query 5 (Optimized)
EXPLAIN ANALIZE
    SELECT * FROM consulta_5;
-- Planning Time: 0.126 ms
-- Execution Time: 0.025 ms
```

```sql
--Query 6 (Non-optimized)
EXPLAIN ANALIZE
    SELECT
        EXTRACT(DOW FROM date) AS dia_semana,
        AVG(total_pizzas_vendidas) AS promedio_pizzas
    FROM (
        SELECT orders.date, COUNT(order_details.pizza_id) AS
total_pizzas_vendidas
        FROM orders
        JOIN order_details ON orders.order_id =
order_details.order_id
        GROUP BY orders.date
    ) AS resumen_dias
    GROUP BY dia_semana
    ORDER BY promedio_pizzas DESC;
-- Planning Time: 0.411 ms
-- Execution Time: 201.009 ms
```

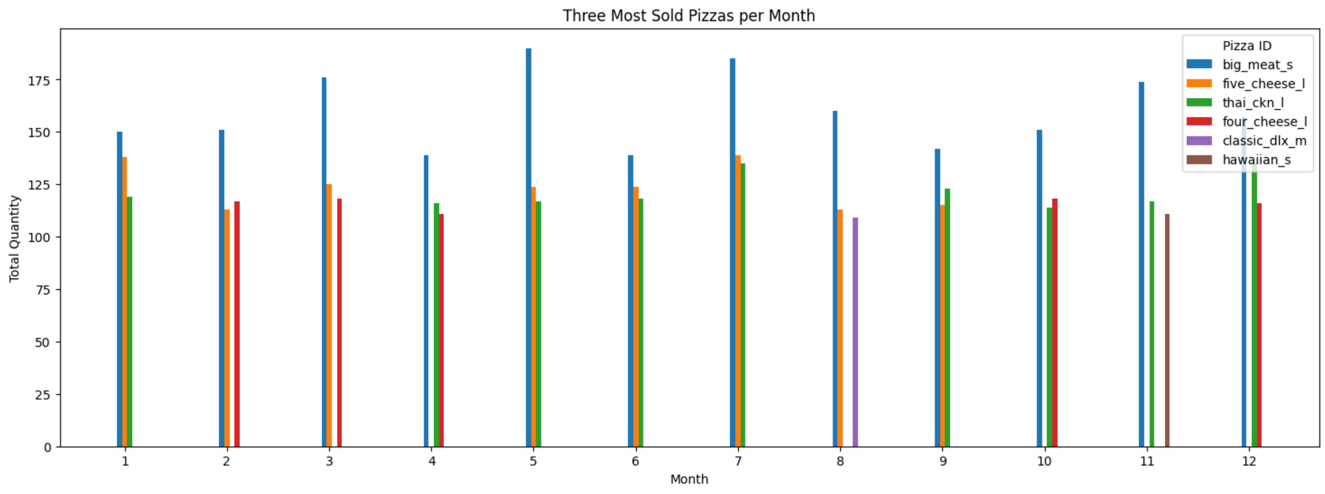
```
--Query 6 (Optimized)
EXPLAIN ANALIZE
    SELECT * FROM consulta_6;
-- Planning Time: 0.131 ms
-- Execution Time: 0.034 ms
```

Based on the following comparison results, we can conclude that there is a significant difference in query times between materialized views and non-materialized views. The most noticeable difference lies in the execution time, as materialized views considerably reduce the query execution time compared to non-materialized views.

**Analysis Results:**

Having made a study with the data obtained from the consultations, we obtained the following results:

Query 1:
The results table displays the quantity of pizzas sold per month for the three most sold pizza types of each month. Each bar represents the “pizza_id” indicating the type of pizza of each “month” based on the “quantity” of pizza sold in that month.



By analyzing the data, we can observe some trends and patterns:

1. **Best-selling type of pizza**: The type of pizza "big_meat_s" is the best-selling type of pizza in all months.

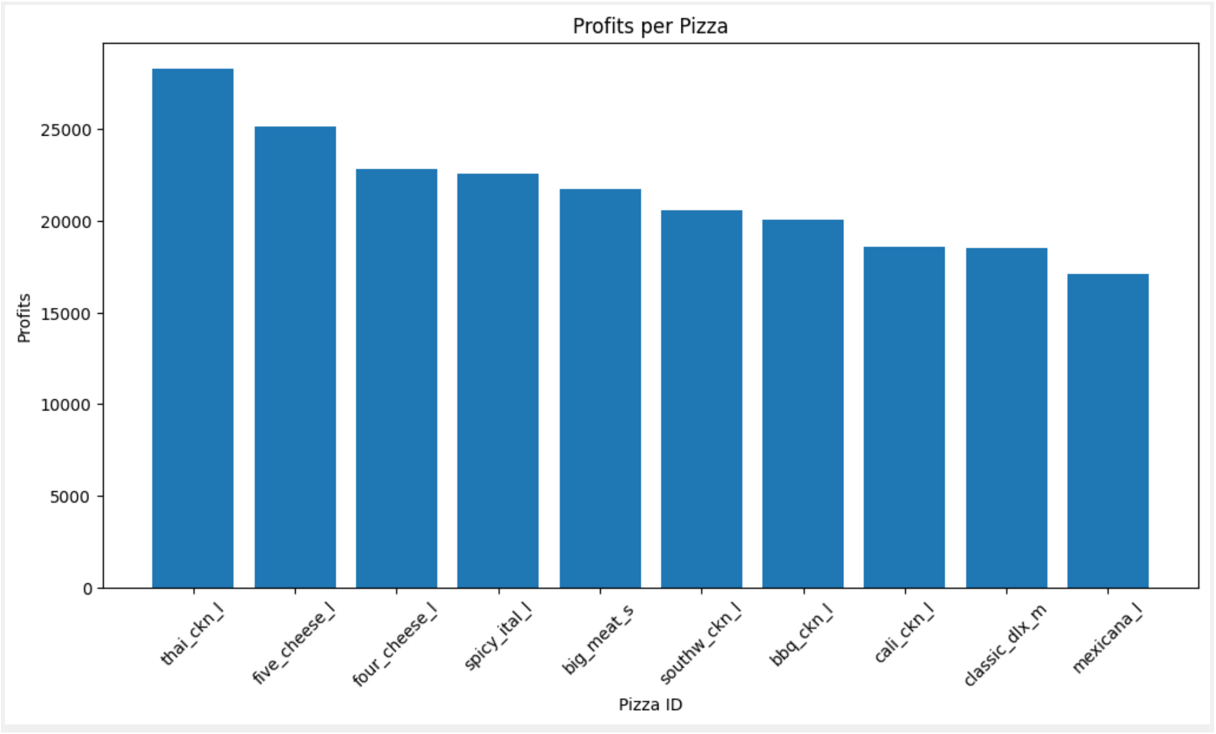
2. **Variation in monthly sales**: The number of pizzas sold varies from month to month. For example, in month 3, 176 "big_meat_s" pizzas were sold, but in month 4 it dropped to 139. This variation may be influenced by seasonal factors, promotions, or changes in customer demand.

3. **Preferences by type of pizza**: Throughout the months, it can be observed that certain types of pizza are consistently popular. In addition to the "big_meat_s", the "five_cheese_l" and "thai_ckn_l" pizzas also have a good number of sales in several months.

4. **Stable sales**: Some pizzas, such as "four_cheese_l" and "classic_dlx_m", have relatively stable sales over the months. This could indicate a loyal customer base or consistent demand for those specific flavors.

In general, these data provide an overview of the monthly sales of different types of pizza. For a deeper analysis, other factors can be taken into account, such as prices, promotions, sales locations, and marketing efforts. This would help to better understand the patterns and make informed decisions about sales strategy and pizza inventory.

Query 2:
The results table shows the profits generated by different types of pizza. Each row represents a type of pizza with two columns: "pizza_id" indicating the type of pizza and "beneficios" representing the profits generated in monetary terms.



When analyzing the data, we can draw the following conclusions:

1. **Most profitable pizza type**: The "thai_ckn_l" pizza type has generated the highest profit with a total of 28,323.75 monetary units. This indicates that this pizza is popular among customers and has a high profit margin.

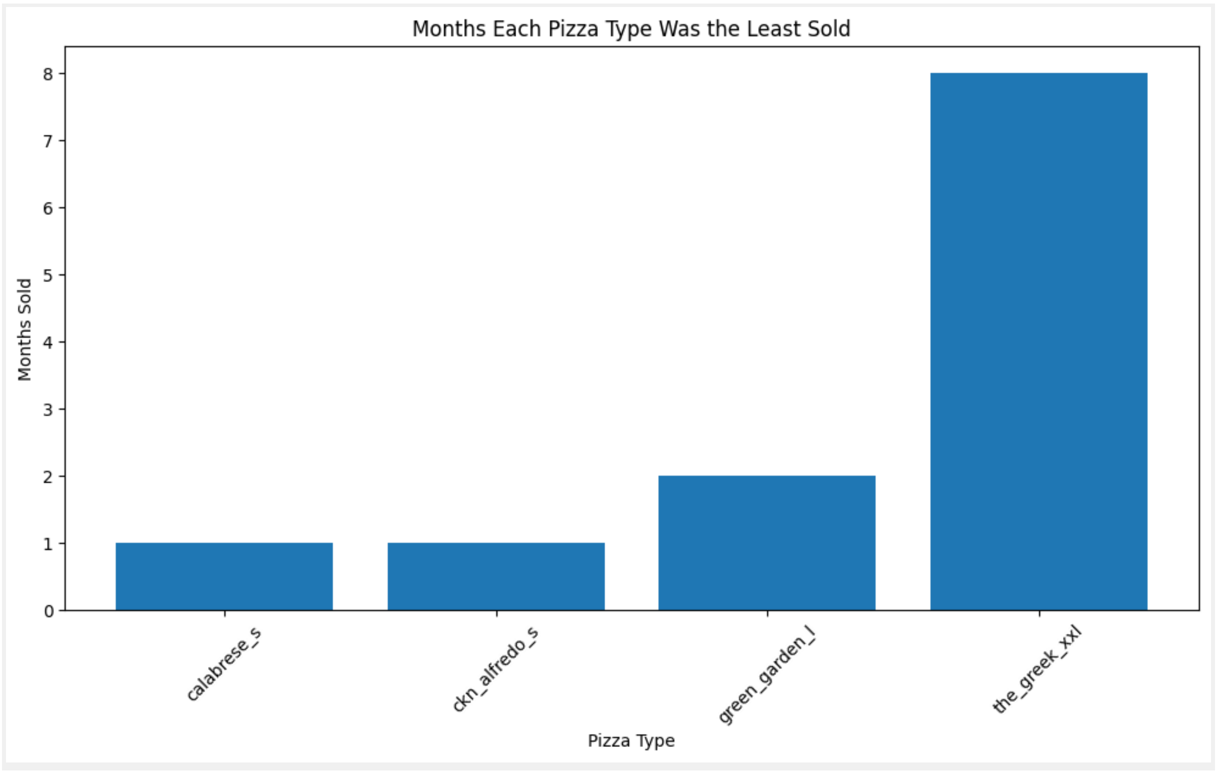
2. **Profitability of different pizza types**: The top five most profitable pizza types in descending order are: "thai_ckn_l" (28,323.75), "five_cheese_l" (25,141.50), "four_cheese_l" (22,850.35), "spicy_ital_l" (22,576.00), and "big_meat_s" (21,732.00). These pizza types generate the highest profits for the business.

3. **Variation in profits**: There is a significant difference in profits generated by each pizza type. For example, the "mexicana_l" pizza type has generated 17,091.00 monetary units, indicating a lower profit compared to the most profitable pizza types.

4. **Relationship between sales and profits**: It is interesting to note that the "big_meat_s" pizza type is the second highest in sales according to the table above but is not among the most profitable pizza types in terms of profits. This may indicate that, although it has good demand, the profit margin per unit is lower compared to other pizza types.

In general, this data provides information about the profitability of different pizza types. This can help the company make informed decisions about pricing strategy, promotions, and focus on the most profitable pizza types. It is also important to consider other factors, such as costs associated with the production of each pizza type, to get a complete picture of profitability and make effective business decisions.

Query 3:
The results table shows the number of months in which each type of pizza was the least sold. Each row represents a type of pizza with two columns: "pizza_id" indicating the type of pizza and "cantidad_meses" representing the number of months in which that pizza had the lowest number of sales.



When analyzing the data, we can draw the following conclusions:

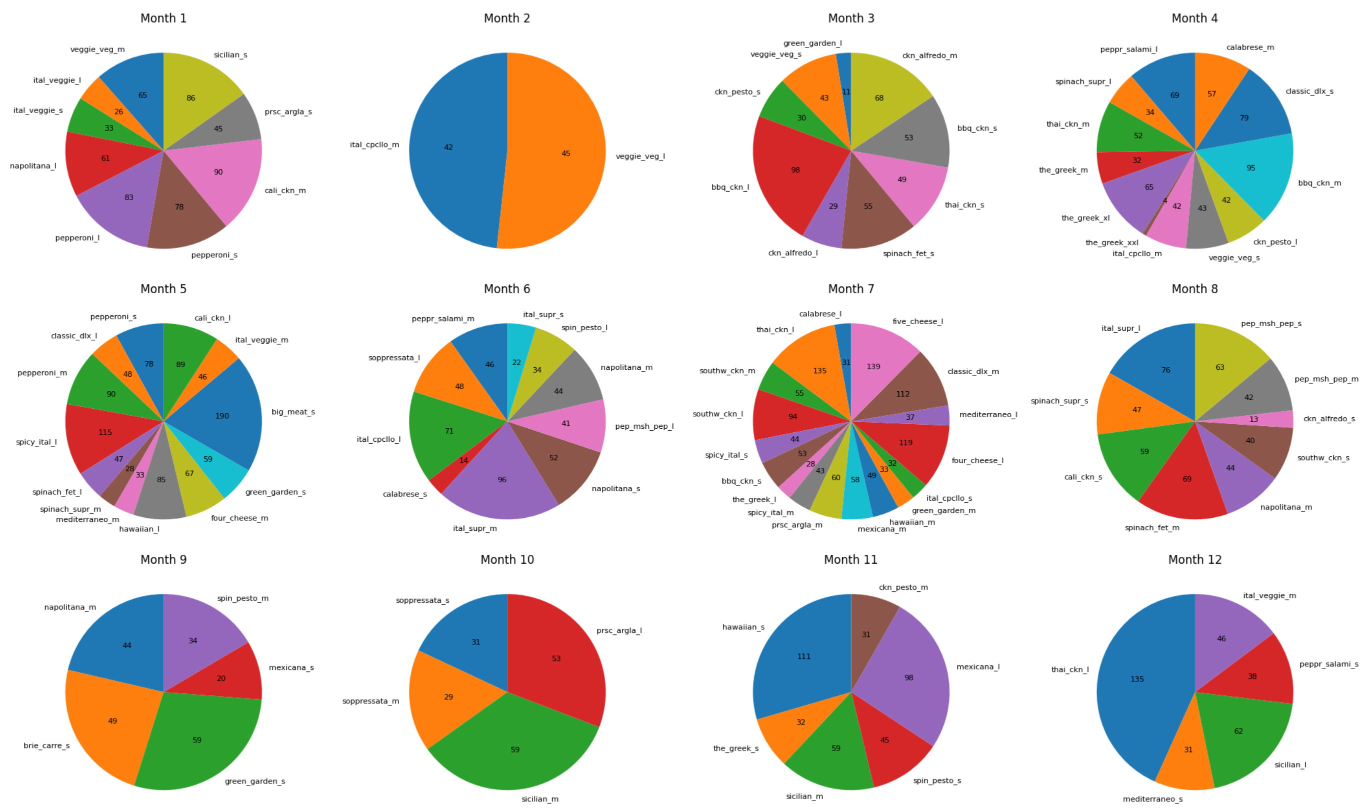
1. **Least sold pizza in fewer months**: The "calabrese_s" and "ckn_alfredo_s" pizzas were the least sold in only 1 month each. This means that these pizzas did not have high demand during most of the months.

2. **Least sold pizza in 2 months**: The "green_garden_l" pizza was the least sold in 2 months. Although it performed better than the pizzas mentioned earlier, it still was not popular during most of the months.

3. **Least sold pizza in 8 months**: The "the_greek_xxl" pizza had the worst performance as it was the least sold in 8 months. This indicates that this pizza was not popular and had very low demand compared to other types of pizza.

In summary, the table provides information about the pizza types that had the lowest number of sales in different months. This data can be useful in identifying which pizzas have low demand and may require additional strategies to improve their sales, such as special promotions, ingredient modifications, or changes in marketing.

Query 4:
The results table shows the peak sales point of each pizza type in a specific month. It is separated by month, where each color represents a "pizza_id" indicating the type of pizza, and its maximum sales quantity represents its peak of sales.



Upon analyzing the data, we can draw the following conclusions:

1. **Pizza with the highest sales in a single month**: The pizza "big_meat_s" reached the peak sales point with 190 units in month 5. This indicates that this pizza had high demand in that particular month.

2. **Variation in the peak sales points**: Some pizzas, such as "thai_ckn_l" and "five_cheese_l," reached their peak sales points in month 7 with 135 and 139 units, respectively. This indicates a strong demand for these pizzas in that specific month.

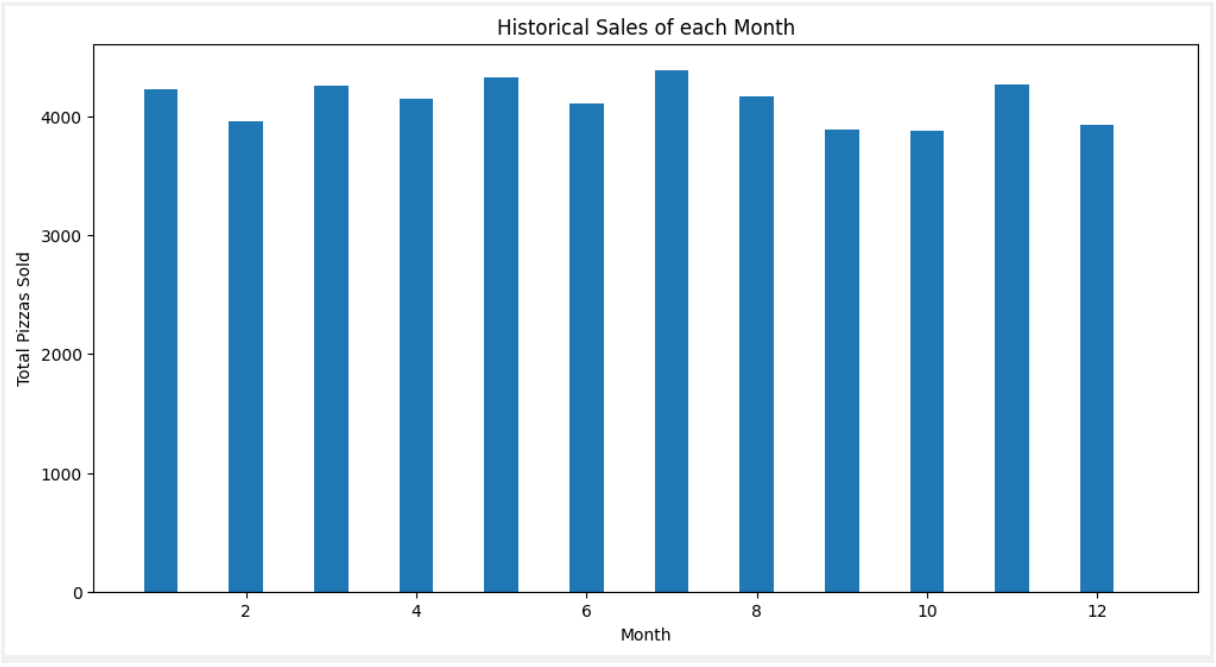
3. **Different successful months for different pizzas**: Each type of pizza had a specific month in which it reached its peak sales point. Some pizzas, like "mexicana_l," "ital_supr_m," and "the_greek_xl," had their peak sales points in month 11.

4. **Variation in peak sales points**: The peak sales points vary widely among different pizzas. Some pizzas reach higher peak sales points, like "big_meat_s" with 190 units, while others reach lower peak sales points, like "green_garden_l" with only 11 units.

In summary, the table provides information about the peak sales points achieved by each type of pizza in different months. This can help identify the most successful months for each type of pizza and provide valuable insights for sales planning and strategy.

Query 5:

The results table shows the total number of pizzas sold each month. The table consists of two columns: "month" which indicates the corresponding month, and "total_pizzas_sold" which represents the overall quantity of pizzas sold in that month.



Upon analyzing the data, we can draw the following conclusions:

1. **Months with higher sales**: The months with the highest total pizza sales were month 7, with a total of 4,392 pizzas sold, followed by month 5 with 4,328 and month 11 with 4,266. These months show a high demand for pizzas.

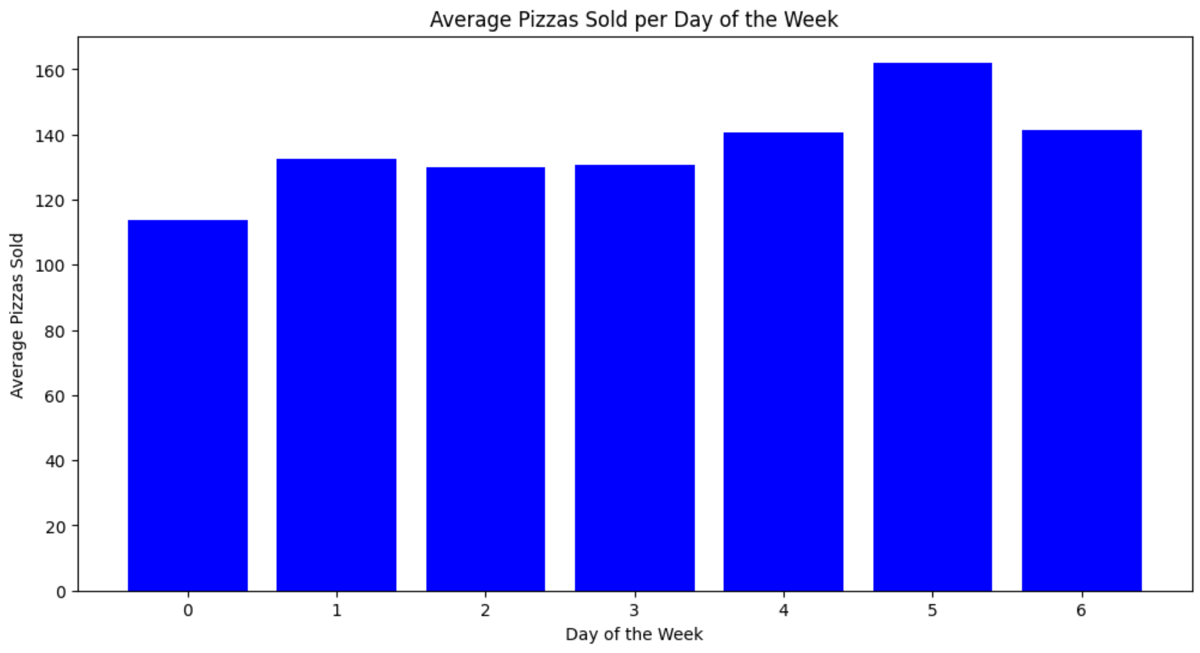
2. **Months with lower sales**: The months with the lowest total pizza sales were month 9, with a total of 3,890 pizzas sold, followed by month 10 with 3,883, and month 12 with 3,935. These months exhibit relatively lower demand compared to others.

3. **Variation in monthly sales**: There is variation in pizza sales from one month to another. Months 7, 5, 11, and 3 showed higher sales, while months 9, 10, and 12 exhibited lower sales.

4. **Sales trends throughout the year**: If we observe the table as a whole, we can notice that sales tend to be higher in the early months (1, 2, 3) and during the summer months (5, 6, 7, 8). However, each year and location may have different seasonal patterns.

In summary, these data provide an overview of monthly pizza sales. Analyzing these trends can help businesses plan their inventory, promotions, and marketing strategies to maximize sales during the stronger months and address challenges during slower months.

Query 6:
The results table displays the average number of pizzas sold per day of the week. The table consists of two columns: "dia_semana" indicating the day of the week, and "promedio_pizzas" representing the average pizzas sold on that day.



Upon analyzing the data, we can draw the following conclusions:

1. **Day of the week with the highest average**: Friday (day 5) has the highest average of pizzas sold, with an average of 162.12 pizzas. This indicates that Fridays are generally the days with the highest pizza demand.

2. **Day of the week with the lowest average**: Sunday (day 0) has the lowest average of pizzas sold, with an average of 113.79 pizzas. This suggests that Sundays may have the lowest pizza demand.

3. **Variation in daily averages**: There is variation in the average number of pizzas sold from one day to another. Fridays, Saturdays (day 6), and Thursdays (day 4) show the highest averages, while Sundays show the lowest average.

4. **Demand patterns during the week**: The average number of pizzas sold suggests that demand gradually increases as the week progresses, peaking on Fridays and Saturdays. Subsequently, demand decreases on Sundays and shows a downward trend as the start of the next week approaches.

These data provide valuable insights into pizza demand patterns based on the days of the week. This information can be useful for workforce planning, inventory management, and implementing marketing strategies focused on days with higher demand.

**Conclusions:**
In summary, analyzing the different query results yields several conclusions:

- Most sold and profitable pizza types vary. Some, like "big_meat_s" and "thai_ckn_l," are both top sellers and most profitable, while others may have high sales but lower profits.

- Sales and profits show variations throughout the year and across weekdays. Certain months and weekdays exhibit higher pizza demand, aiding marketing strategies and inventory management.

- Some pizzas maintain consistent sales over months, while others may experience low sales in specific months. This suggests customer preferences and the need to adjust strategies for improved sales of certain pizza types.

- Overall, these findings provide valuable insights into sales and profitability patterns for different pizza types. Businesses can use this information to make informed decisions on inventory management, promotional planning, and enhancing profitability in the pizza industry.