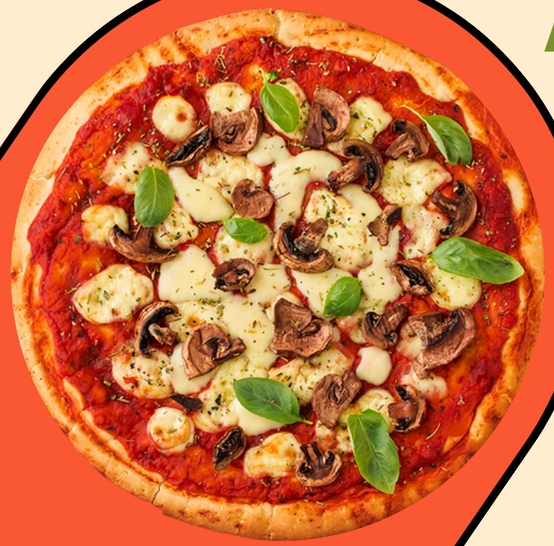
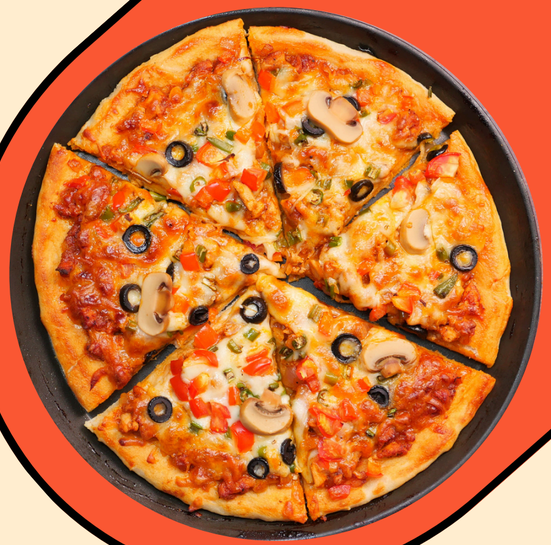
HTTPS://WWW.LINKEDIN.COM/IN/SIDDIQUI-ADDY/

-by Adeel Siddiqui

INDIAN DATA CLUB

PIZZA ANALYTICS CHALLENGE





MINI PROJECT FOR IDC 21 DAY SQL CHALLENGE

ABOUT THE PROJECT

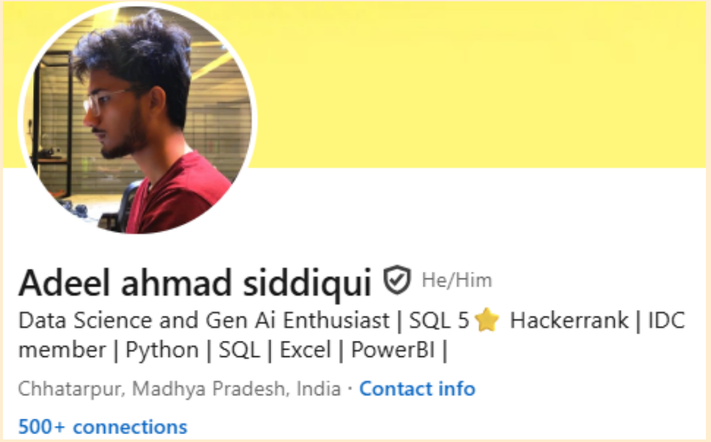
In this mini project, titled **Great Pizza Analytics**, I stepped into the role of a data analyst for **IDC Pizza**, a restaurant scenario created as part of the IDC 21 Day SQL Challenge. Using SQL, I analyzed raw pizza sales data to answer a series of business-driven questions.

The project demonstrates how queries involving filtering, aggregation, joins, grouping, and self-joins can be applied to uncover insights such as:

- **Which pizza categories sell the most**
- **Which pizzas were never ordered**
- **How order values vary across customers**
- **Price differences between sizes of the same pizza**

ABOUT THE ANALYST

with Batchlors in Business Management and Master's in Economics, I'm Preparing to be a Business Analyst
and progressively advancing my skills.
I'm also doing Google Advanced Data Analytics
Certification and Extra Curricular Projects
I have made several Projects, listed on my Github
and taken part in Codebasics RPC 16



HTTPS://GITHUB.COM/ADDYSIDDIQUI7

DATASETS USED

ORDERS.CSV (21,000+ ROWS)

Contains each customer order with fields like order_id, date, and time.

ORDER_DETAILS.CSV (44,000+ ROWS)

Line-item breakdown of each order, including order_details_id, order_id, pizza_id, and quantity.

PIZZAS_TYPES.CSV

Describes pizza categories and recipes with pizza_type_id, name, category, and ingredients.

PIZZAS.CSV

Defines each pizza variant with pizza_id, pizza_type_id, size, and price.

ALL ANALYSIS USING MYSQL8.0

PHASE 1: FOUNDATION & INSPECTION

01. Database schema and table design

```
-- creating new database for IDC PIZZA Data Analyis mini challenge
create database mini_challenge;
use mini_challenge;

-- creating nessary tables and their schemas
create table order_details(
order_details_id int , order_id int, pizza_id VARCHAR(30), quantity int);
create table orders(
order_id int , date DATE, time TIME);
create table pizza_types(
pizza_type_id VARCHAR(30), name VARCHAR(30), category VARCHAR(20), ingredients VARCHAR(150));
create table pizzas(
pizza_id VARCHAR(50), pizza_type_id VARCHAR(50), size VARCHAR(1), price FLOAT);
```

OUTPUT

02. List all unique pizza categories (DISTINCT)

```
select DISTINCT(category)
        from pizza_types;
```

category
Chicken
Classic
Supreme
Veggie

03. Display `pizza_type_id`, `name`, and ingredients, replacing NULL ingredients with `"Missing Data"`. Show first 5 rows.

```
select pizza_type_id, name,
        CASE when ingredients is null then "Missing Data"
        else ingredients END as INGREDIENTS
from pizza_types
limit 5;
```

pizza_type_id	name	INGREDIENTS
bbq_ckn	The Barbecue Chicken Pizza	Barbecued Chicken, Red Peppers, Green Pe
cali_ckn	The California Chicken Pizza	Chicken, Artichoke, Spinach, Garlic, Jalaper
ckn_alfredo	The Chicken Alfredo Pizza	Chicken, Red Onions, Red Peppers, Mushro
ckn_pesto	The Chicken Pesto Pizza	Chicken, Tomatoes, Red Peppers, Spinach,
southw_ckn	The Southwest Chicken Pizza	Chicken, Tomatoes, Red Peppers, Red Onio

SALES PERFORMANCE

04. Check for pizzas missing a price (IS NULL).

```
select * from pizzas
        where price is null;
```

	pizza_id	pizza_type_id	size	price

05. Orders placed on '2015-01-01' (SELECT + WHERE).

```
select * from orders
        where date = '2015-01-01';
```

	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

06. Pizzas sold in sizes 'L' or 'XL'.

```
select * from pizzas
        where size = 'L' or 'XL';
```

	pizza_id	pizza_type_id	size	price
▶	bbq_ckn_l	bbq_ckn	L	20.75
	cali_ckn_l	cali_ckn	L	20.75
	ckn_alfredo_l	ckn_alfredo	L	20.75
	ckn_pesto_l	ckn_pesto	L	20.75
	southw_ckn_l	southw_ckn	L	20.75

07. Orders placed on '2015-01-01' (SELECT + WHERE).

```
select * from pizzas
        where price >= 15.00 and price <= 17.00;
```

	pizza_id	pizza_type_id	size	price
▶	bbq_ckn_m	bbq_ckn	M	16.75
	cali_ckn_m	cali_ckn	M	16.75
	ckn_alfredo_m	ckn_alfredo	M	16.75
	ckn_pesto_m	ckn_pesto	M	16.75
	southw_ckn_m	southw_ckn	M	16.75

08. Pizzas with "Chicken" in the name.

```
select * from pizza_types
        where name like "%Chicken%";
```

pizza_type_id	name	category	ingredients
bbq_ckn	The Barbecue Chicken Pizza	Chicken	Barbecued Ch
cali_ckn	The California Chicken Pizza	Chicken	Chicken, Artic
ckn_alfredo	The Chicken Alfredo Pizza	Chicken	Chicken, Red
ckn_pesto	The Chicken Pesto Pizza	Chicken	Chicken, Toma
southw_ckn	The Southwest Chicken Pizza	Chicken	Chicken, Toma

09. Orders on '2015-02-15' or placed after 8 PM.

```
select * from orders
        where date = '2015-02-15'
        or time > '20:00:00';
```

order_id	date	time
60	2015-01-01	20:05:16
61	2015-01-01	20:08:43
62	2015-01-01	20:50:16
63	2015-01-01	20:51:42
64	2015-01-01	20:52:08

PHASE 3: SALES ANALYSIS

10. Total quantity of pizzas sold (SUM).

```
select sum(quantity) as Total_Quantity_Sold
       from order_details;
```

	Total_Quantity_Sold
▶	49574

11. Average pizza price (AVG)

```
select round(avg(price),2)
       as "Avg Pizza Price (in $$)"
from pizzas;
```

	Avg Pizza Price (in $$)
▶	16.14

12. Total order value per order (JOIN, SUM, GROUP BY).

```sql
select
 o.order_id,
 round(sum(od.quantity * p.price),0) as total_order_value
from orders o
join order_details od on o.order_id = od.order_id
join pizzas p on od.pizza_id = p.pizza_id
group by o.order_id order by total_order_value desc;
```

order_id	total_order_value
18845	419
10760	417
1096	285
6169	284
12257	277

13. Total quantity sold per pizza category (JOIN, GROUP BY).

```sql
select
    pt.category,
    sum(od.quantity) as total_quantity
from order_details od
join pizzas p on od.pizza_id = p.pizza_id
join pizza_types pt on p.pizza_type_id = pt.pizza_type_id
group by pt.category
order by total_quantity desc;
```

14. Categories with more than 5,000 pizzas sold (HAVING).

```sql
select
    pt.category,
    sum(od.quantity) as total_quantity
from order_details od
join pizzas p on od.pizza_id = p.pizza_id
join pizza_types pt on p.pizza_type_id = pt.pizza_type_id
group by pt.category
having sum(od.quantity) > 5000
order by total_quantity desc;
```

	category	total_quantity
▶	Classic	12949
	Chicken	11050
	Supreme	10530
	Veggie	10123

15. Pizzas never ordered (LEFT/RIGHT JOIN).

```sql
select
    p.pizza_id,
    p.pizza_type_id,
    p.size,
    p.price
from pizzas p
left join order_details od on p.pizza_id = od.pizza_id
where od.pizza_id is null;
```

pizza_id	pizza_type_id	size	price
big_meat_l	big_meat	L	20.5
four_cheese_s	four_cheese	S	11.75
five_cheese_m	five_cheese	M	15.5
big_meat_m	big_meat	M	16
five_cheese_s	five_cheese	S	12.5

16. Price differences between different sizes of the same pizza (SELF JOIN).

```sql
select
    p1.pizza_type_id,
    p1.size as size1,
    p1.price as price1,
    p2.size as size2,
    p2.price as price2,
    abs(p1.price - p2.price) as price_difference
from pizzas p1
join pizzas p2
    on p1.pizza_type_id = p2.pizza_type_id
   and p1.size <> p2.size
order by p1.pizza_type_id, price_difference desc;
```

pizza_type_id	size1	price1	size2	price2	price_difference
bbq_ckn	L	20.75	S	12.75	8
bbq_ckn	S	12.75	L	20.75	8
bbq_ckn	M	16.75	S	12.75	4
bbq_ckn	L	20.75	M	16.75	4
bbq_ckn	S	12.75	M	16.75	4


THANK YOU