



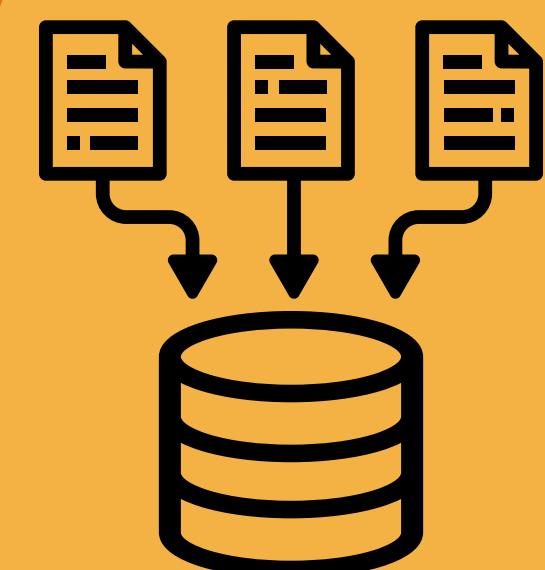
Pizzahut



By: Mohammed Hibban



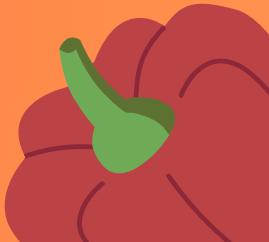
# Pizza Sales Project





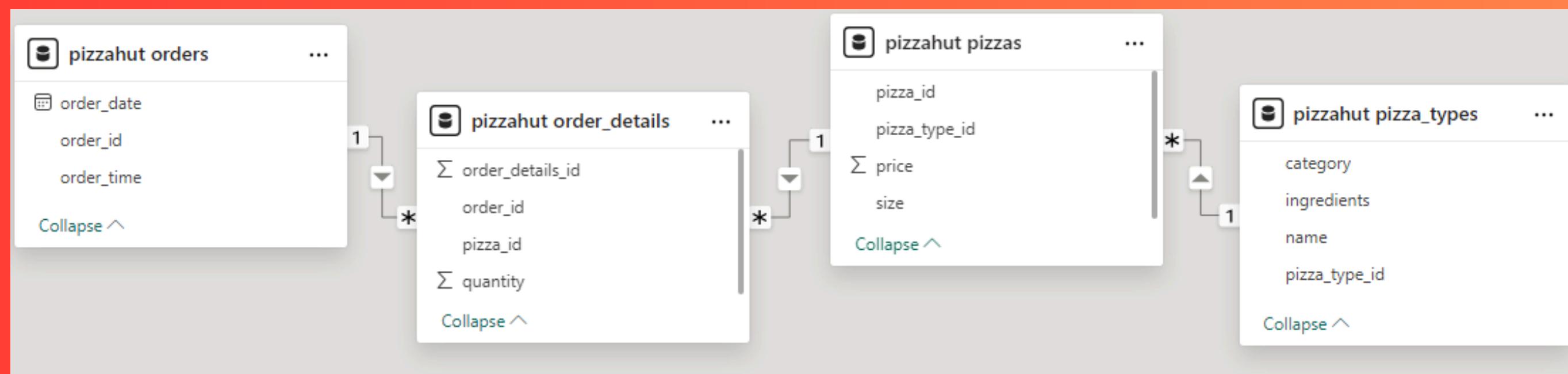
# INTRODUCTION

Hi, I am Mohammed Hibban and in this project I have utilised SQL queries to solve questions that were related to Pizza sales.



# Database and Schema

*In this database we have 4 tables as follows*



# Retrieve total number of orders placed

```
22
23 •   SELECT
24     COUNT(order_id) AS total_order
25   FROM
26     orders;
27
70  Calculating the total number of orders...
<-->
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	total_order
▶	21350

Result 4

# Calculate the total revenue generated from Pizza sales

```
28      -- Calculate the total revenue generated from pizza sales.  
29  
30 •   SELECT  
31     ROUND(SUM(order_details.quantity * pizzas.price),  
32             2) AS total_sales  
33   FROM  
34     order_details  
35   JOIN  
36     pizzas ON pizzas.pizza_id = order_details.pizza_id;  
37
```

Result Grid | Filter Rows:  Export: Wrap Cell Content:

total_sales
817860.05

# Identify the highest priced Pizzas

```
41 •   SELECT
42       pizza_types.name, pizzas.price
43   FROM
44       pizza_types
45   JOIN
46       pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
47   ORDER BY pizzas.price DESC
48   LIMIT 1;
```

49

< [REDACTED]

Result Grid | Filter Rows:  Export: Wrap Cell Content: Fetch rows:

	name	price
▶	The Greek Pizza	35.95

# Identify the most common Pizza size ordered

```
55 •   SELECT
56       pizzas.size,
57       COUNT(order_details.order_details_id) AS order_count
58   FROM
59       pizzas
60       JOIN
61       order_details ON pizzas.pizza_id = order_details.pizza_id
62   GROUP BY pizzas.size
63   ORDER BY order_count DESC;
```

←

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	size	order_count
▶	L	18526
	M	15385
	S	14137
	XL	544
	XXL	28

# List the top 5 most ordered pizza types along with their quantities

```
67 •   SELECT
68     pizza_types.name, SUM(order_details.quantity) AS quantity
69   FROM
70     pizza_types
71       JOIN
72       pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
73       JOIN
74       order_details ON order_details.pizza_id = pizzas.pizza_id
75   GROUP BY pizza_types.name
76   ORDER BY quantity DESC
77   LIMIT 5;
```

Result Grid | Filter Rows:  | Export: | Wrap Cell Content: | Fetch rows:

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

Join the necessary table to find the total quantity of each pizza category ordered.

```
80 •   SELECT
81       pizza_types.category,
82       SUM(order_details.quantity) AS quantity
83   FROM
84       pizza_types
85       JOIN
86       pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
87       JOIN
88       order_details ON order_details.pizza_id = pizzas.pizza_id
89   GROUP BY pizza_types.category
90   ORDER BY quantity;
```



Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	category	quantity
▶	Chicken	11050
	Veggie	11649
	Supreme	11987
	Classic	14888



# Determine the distribution of orders by hour of the day.

```
93 •   SELECT  
94       HOUR(order_time), COUNT(order_id)  
95   FROM  
96       orders  
97   GROUP BY HOUR(order_time);
```

ox

Result Grid | Filter Rows:  Export: Wrap Cell Content:

	HOUR(order_time)	COUNT(order_id)
▶	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
	10	8
	9	1

Join the relevant tables to find out the category wise distribution of Pizzas

```
100 •   SELECT
101       category, COUNT(name)
102   FROM
103       pizza_types
104   GROUP BY category;
```

Result Grid | Filter Rows:  Export: Wrap Cell Content:

	category	COUNT(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

# Group the orders by date and calculate the average number of pizzas ordered per day

```
107 •   SELECT
108      ROUND(AVG(quantity), 2) as Avg_pizza_ordered_per_day
109  FROM
110  (
111      SELECT
112          orders.order_date, SUM(order_details.quantity) AS quantity
113      FROM
114          orders
115      JOIN order_details ON orders.order_id = order_details.order_id
116      GROUP BY orders.order_date) AS order_quantity;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
<a href="#">Avg_pizza_ordered_per_day</a>		138.47		

# Determine the top 3 most ordered pizza types based on their revenue.

```
119 •  SELECT
120      pizza_types.name,
121      SUM(order_details.quantity * pizzas.price) AS revenue
122  FROM
123      pizza_types
124      JOIN
125      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
126      JOIN
127      order_details ON order_details.pizza_id = pizzas.pizza_id
128  GROUP BY pizza_types.name
129  ORDER BY revenue DESC
130  LIMIT 3;
```

```
119
120
121
122
123
124
125
126
127
128
129
130
```

Result Grid | Filter Rows:  Export: Wrap Cell Content: Fetch rows:

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

# Calculate the percentage contribution of each pizza type by total revenue.

```
133 •   SELECT
134     pizza_types.category,
135     ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
136                     ROUND(SUM(order_details.quantity * pizzas.price),2)
137                 FROM
138                     order_details
139                     JOIN
140                         pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,2) AS revenue
141   FROM
142     pizza_types
143     JOIN
144         pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
145     JOIN
146         order_details ON order_details.pizza_id = pizzas.pizza_id
147 GROUP BY pizza_types.category
148 ORDER BY revenue DESC;
```

Result Grid | Filter Rows:  Export: Wrap Cell Content:

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

# Analyse the cumulative revenue generated overtime.

```
152 •   select order_date,sum(revenue)over(order by order_date)
153     from
154     (select orders.order_date,
155      sum(order_details.quantity*pizzas.price)as revenue
156      from order_details join pizzas
157      on order_details.pizza_id=pizzas.pizza_id
158      join orders
159      on orders.order_id=order_details.order_id
160      group by orders.order_date)as sales;
```

```
161
```

```
162
```

```
<
```

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	order_date	sum(revenue)over(order by order_date)
▶	2015-01-01	2713.8500000000004
	2015-01-02	5445.75
	2015-01-03	8108.15
	2015-01-04	9863.6
	2015-01-05	11929.55
	2015-01-06	14258.5
	2015-01-07	16633.5
	2015-01-08	19008.5
	2015-01-09	21383.5
	2015-01-10	23758.5
	2015-01-11	26133.5
	2015-01-12	28508.5
	2015-01-13	30883.5
	2015-01-14	33258.5
	2015-01-15	35633.5
	2015-01-16	38008.5
	2015-01-17	40383.5
	2015-01-18	42758.5
	2015-01-19	45133.5
	2015-01-20	47508.5
	2015-01-21	50383.5
	2015-01-22	53258.5
	2015-01-23	56133.5
	2015-01-24	59008.5
	2015-01-25	61883.5
	2015-01-26	64258.5
	2015-01-27	66633.5
	2015-01-28	69008.5
	2015-01-29	71383.5
	2015-01-30	73758.5
	2015-01-31	76133.5
	2015-02-01	78508.5
	2015-02-02	80883.5
	2015-02-03	83258.5
	2015-02-04	85633.5
	2015-02-05	88008.5
	2015-02-06	90383.5
	2015-02-07	92758.5
	2015-02-08	95133.5
	2015-02-09	97508.5
	2015-02-10	100383.5
	2015-02-11	102758.5
	2015-02-12	105133.5
	2015-02-13	107508.5
	2015-02-14	110383.5
	2015-02-15	112758.5
	2015-02-16	115133.5
	2015-02-17	117508.5
	2015-02-18	120383.5
	2015-02-19	122758.5
	2015-02-20	125133.5
	2015-02-21	127508.5
	2015-02-22	130383.5
	2015-02-23	132758.5
	2015-02-24	135133.5
	2015-02-25	137508.5
	2015-02-26	140383.5
	2015-02-27	142758.5
	2015-02-28	145133.5
	2015-02-29	147508.5
	2015-03-01	150383.5
	2015-03-02	152758.5
	2015-03-03	155133.5
	2015-03-04	157508.5
	2015-03-05	160383.5
	2015-03-06	162758.5
	2015-03-07	165133.5
	2015-03-08	167508.5
	2015-03-09	170383.5
	2015-03-10	172758.5
	2015-03-11	175133.5
	2015-03-12	177508.5
	2015-03-13	180383.5
	2015-03-14	182758.5
	2015-03-15	185133.5
	2015-03-16	187508.5
	2015-03-17	190383.5
	2015-03-18	192758.5
	2015-03-19	195133.5
	2015-03-20	197508.5
	2015-03-21	200383.5
	2015-03-22	202758.5
	2015-03-23	205133.5
	2015-03-24	207508.5
	2015-03-25	210383.5
	2015-03-26	212758.5
	2015-03-27	215133.5
	2015-03-28	217508.5
	2015-03-29	220383.5
	2015-03-30	222758.5
	2015-03-31	225133.5
	2015-04-01	227508.5
	2015-04-02	230383.5
	2015-04-03	232758.5
	2015-04-04	235133.5
	2015-04-05	237508.5
	2015-04-06	240383.5
	2015-04-07	242758.5
	2015-04-08	245133.5
	2015-04-09	247508.5
	2015-04-10	250383.5
	2015-04-11	252758.5
	2015-04-12	255133.5
	2015-04-13	257508.5
	2015-04-14	260383.5
	2015-04-15	262758.5
	2015-04-16	265133.5
	2015-04-17	267508.5
	2015-04-18	270383.5
	2015-04-19	272758.5
	2015-04-20	275133.5
	2015-04-21	277508.5
	2015-04-22	280383.5
	2015-04-23	282758.5
	2015-04-24	285133.5
	2015-04-25	287508.5
	2015-04-26	290383.5
	2015-04-27	292758.5
	2015-04-28	295133.5
	2015-04-29	297508.5
	2015-04-30	300383.5
	2015-05-01	302758.5
	2015-05-02	305133.5
	2015-05-03	307508.5
	2015-05-04	310383.5
	2015-05-05	312758.5
	2015-05-06	315133.5
	2015-05-07	317508.5
	2015-05-08	320383.5
	2015-05-09	322758.5
	2015-05-10	325133.5
	2015-05-11	327508.5
	2015-05-12	330383.5
	2015-05-13	332758.5
	2015-05-14	335133.5
	2015-05-15	337508.5
	2015-05-16	340383.5
	2015-05-17	342758.5
	2015-05-18	345133.5
	2015-05-19	347508.5
	2015-05-20	350383.5
	2015-05-21	352758.5
	2015-05-22	355133.5
	2015-05-23	357508.5
	2015-05-24	360383.5
	2015-05-25	362758.5
	2015-05-26	365133.5
	2015-05-27	367508.5
	2015-05-28	370383.5
	2015-05-29	372758.5
	2015-05-30	375133.5
	2015-05-31	377508.5
	2015-06-01	380383.5
	2015-06-02	382758.5
	2015-06-03	385133.5
	2015-06-04	387508.5
	2015-06-05	390383.5
	2015-06-06	392758.5
	2015-06-07	395133.5
	2015-06-08	397508.5
	2015-06-09	400383.5
	2015-06-10	402758.5
	2015-06-11	405133.5
	2015-06-12	407508.5
	2015-06-13	410383.5
	2015-06-14	412758.5
	2015-06-15	415133.5
	2015-06-16	417508.5
	2015-06-17	420383.5
	2015-06-18	422758.5
	2015-06-19	425133.5
	2015-06-20	427508.5
	2015-06-21	430383.5
	2015-06-22	432758.5
	2015-06-23	435133.5
	2015-06-24	437508.5
	2015-06-25	440383.5
	2015-06-26	442758.5
	2015-06-27	445133.5
	2015-06-28	447508.5
	2015-06-29	450383.5
	2015-06-30	452758.5
	2015-07-01	455133.5
	2015-07-02	457508.5
	2015-07-03	460383.5
	2015-07-04	462758.5
	2015-07-05	465133.5
	2015-07-06	467508.5
	2015-07-07	470383.5
	2015-07-08	472758.5
	2015-07-09	475133.5
	2015-07-10	477508.5
	2015-07-11	480383.5
	2	

# Determine the top 3 most ordered pizza type based on revenue for each pizza category

```
163 •   select name,revenue from
164   (select category,name,revenue,
165    rank() over(partition by category order by revenue desc)as rn
166    from
167   (select pizza_types.category,pizza_types.name,
168    sum(order_details.quantity*pizzas.price)as revenue
169    from pizza_types join pizzas
170    on pizza_types.pizza_type_id=pizzas.pizza_type_id
171    join order_details
172    on order_details.pizza_id=pizzas.pizza_id
173    group by pizza_types.category,pizza_types.name) as a)as b
174   where rn<=3;
```

Result Grid | Filter Rows:  Export: Wrap Cell Content:

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5
	The Classic Deluxe Pizza	38180.5
	The Hawaiian Pizza	32273.25
	The Pepperoni Pizza	30161.75
	The Spicy Italian Pizza	34831.25
	The Italian Supreme Pizza	33476.75
	The Sicilian Pizza	30940.5
	The Four Cheese Pizza	32265.70000000065

*Thank You*

