DATAWITHDANNY.COM

START YOUR SQL ENGINES

PIZZA RUNNER

CASE STUDY #2

8 WEEK SQL CHALLENGE

8WEEKSQLCHALLENGE.COM

Introduction

- Danny was scrolling through his Instagram feed when something really caught his eye - “80s Retro Styling and Pizza Is The Future!”

- Danny was sold on the idea, but he knew that pizza alone was not going to help him get seed funding to expand his new Pizza Empire - so he had one more genius idea to combine with it - he was going to *Uberize* it - and so Pizza Runner was launched!

- Danny started by recruiting “runners” to deliver fresh pizza from Pizza Runner Headquarters (otherwise known as Danny’s house) and also maxed out his credit card to pay freelance developers to build a mobile app to accept orders from customers.

Problem Statement

Design and analyze a data-driven solution for *Pizza Runner*, a startup combining 80s retro styling with on-demand pizza delivery. The system must manage and analyze customer orders, runner deliveries, and operational performance to support business expansion and decision-making.

Case Study Questions

1.How many pizzas were ordered?

2.How many unique customer orders were made?

3.How many successful orders were delivered by each runner?

4.How many of each type of pizza was delivered?

5.How many Vegetarian and Meat lovers were ordered by each customer?

6.What was the maximum number of pizzas delivered in a single order?

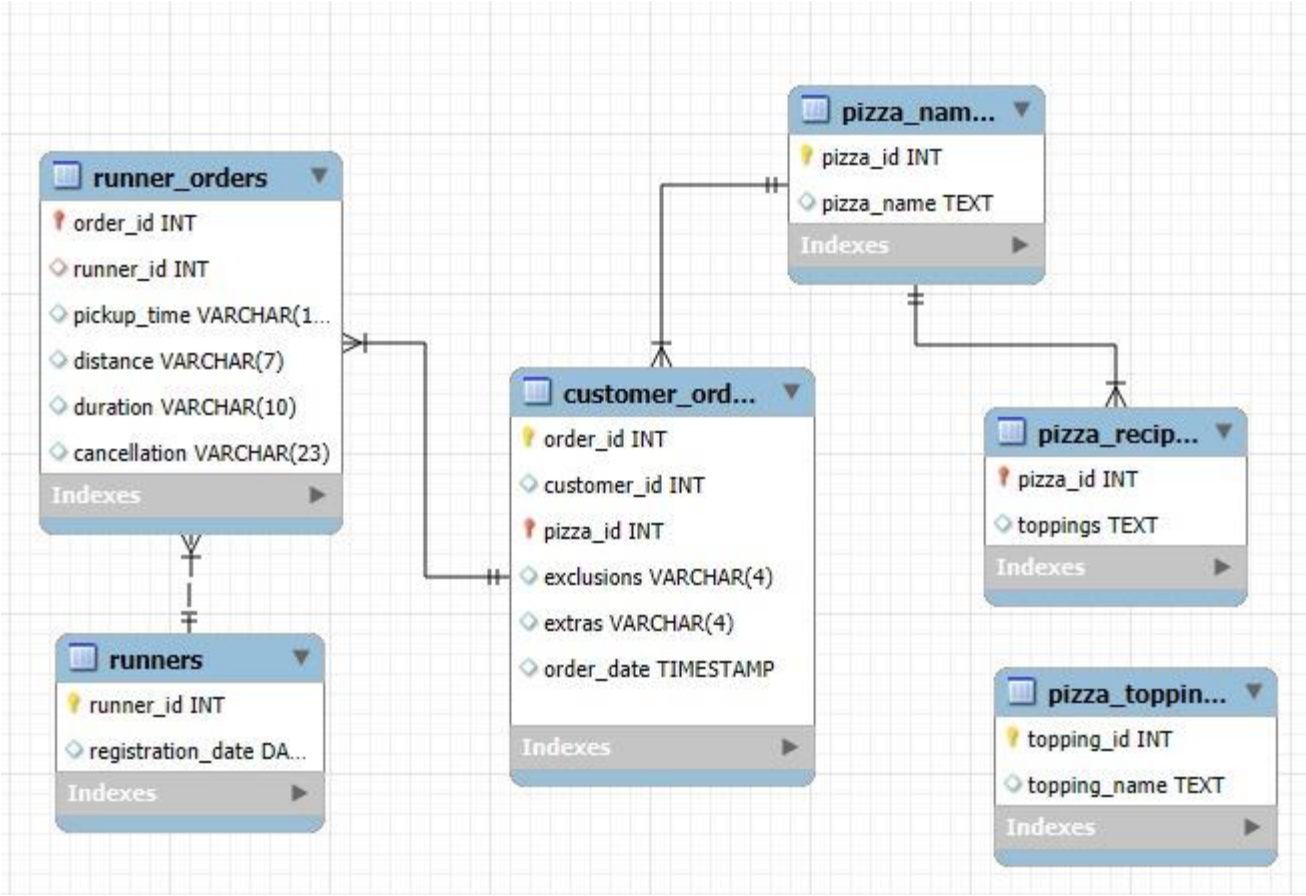
7.How many pizzas were delivered that had both exclusions and extras?

8.What was the total volume of pizzas ordered for each hour of the day?

9.What was the volume of orders for each day of the week?

10. What is the successful delivery percentage for each runner?

Entity Relationship Diagram



Creating database

```sql
CREATE DATABASE pizza_runner;
```

```sql
-- Table: runners
CREATE TABLE runners (
  runner_id INTEGER PRIMARY KEY,
  registration_date DATE
);
```

Creating runners Table

```sql
-- Insert runners
INSERT INTO runners VALUES
  (1, '2021-01-01'),
  (2, '2021-01-03'),
  (3, '2021-01-08'),
  (4, '2021-01-15');
```

Insert Values in runners Table

```sql
-- Table: customer_orders
CREATE TABLE customer_orders (
  order_id INTEGER,
  customer_id INTEGER,
  pizza_id INTEGER,
  exclusions VARCHAR(4),
  extras VARCHAR(4),
  order_date TIMESTAMP,
  PRIMARY KEY (order_id, pizza_id),
  FOREIGN KEY (pizza_id) REFERENCES pizza_names(pizza_id)
);
```

Creating customer_order Table

```sql
-- Insert customer orders
INSERT INTO customer_orders VALUES
  (1, 101, 1, '', '', '2020-01-01 18:05:02'),
  (2, 101, 1, '', '', '2020-01-01 19:00:52'),
  (3, 102, 1, '', '', '2020-01-02 23:51:23'),
  (3, 102, 2, '', NULL, '2020-01-02 23:51:23'),
  (4, 103, 1, '4', '', '2020-01-04 13:23:46'),
  (5, 104, 1, NULL, '1', '2020-01-08 21:00:29'),
  (6, 101, 2, NULL, NULL, '2020-01-08 21:03:13'),
  (7, 105, 2, NULL, '1', '2020-01-08 21:20:29'),
  (8, 102, 1, NULL, NULL, '2020-01-09 23:54:33'),
  (9, 103, 1, '4', '1,5', '2020-01-10 11:22:59'),
  (10, 104, 1, '2,6', '1,4', '2020-01-11 18:34:49');
```

Insert Values in customers_orders Table

```sql
-- Table: runner_orders
CREATE TABLE runner_orders (
  order_id INTEGER PRIMARY KEY,
  runner_id INTEGER,
  pickup_time VARCHAR(19),
  distance VARCHAR(7),
  duration VARCHAR(10),
  cancellation VARCHAR(23),
  FOREIGN KEY (order_id) REFERENCES customer_orders(order_id),
  FOREIGN KEY (runner_id) REFERENCES runners(runner_id)
);
```

Creating runner_orders Table

```sql
-- Insert runner orders
INSERT INTO runner_orders VALUES
  (1, 1, '2020-01-01 18:15:34', '20km', '32 minutes', ''),
  (2, 1, '2020-01-01 19:10:54', '20km', '27 minutes', ''),
  (3, 1, '2020-01-03 00:12:37', '13.4km', '20 mins', NULL),
  (4, 2, '2020-01-04 13:53:03', '23.4', '40', NULL),
  (5, 3, '2020-01-08 21:10:57', '10', '15', NULL),
  (6, 3, NULL, NULL, NULL, 'Restaurant Cancellation'),
  (7, 2, '2020-01-08 21:30:45', '25km', '25mins', NULL),
  (8, 2, '2020-01-10 00:15:02', '23.4 km', '15 minute', NULL),
  (9, 2, NULL, NULL, NULL, 'Customer Cancellation'),
  (10, 1, '2020-01-11 18:50:20', '10km', '10minutes', NULL);
```

Insert Values in runner_orders Table

```sql
-- Table: pizza_names
CREATE TABLE pizza_names (
  pizza_id INTEGER PRIMARY KEY,
  pizza_name TEXT
);
```

Creating pizza_names Table

```sql
-- Insert pizza names
INSERT INTO pizza_names VALUES
  (1, 'Meatlovers'),
  (2, 'Vegetarian');
```

Insert Values in pizza_names Table

```sql
-- Table: pizza_recipes
CREATE TABLE pizza_recipes (
  pizza_id INTEGER PRIMARY KEY,
  toppings TEXT,
  FOREIGN KEY (pizza_id) REFERENCES pizza_names(pizza_id)
);
```

Creating pizza_recipes Table

```sql
-- Insert pizza recipes
INSERT INTO pizza_recipes VALUES
  (1, '1,2,3,4,5,6,8,10'),
  (2, '4,6,7,9,11,12');
```

Insert Values in pizza_recipes Table

```sql
-- Table: pizza_toppings
CREATE TABLE pizza_toppings (
  topping_id INTEGER PRIMARY KEY,
  topping_name TEXT
);
```

Creating pizza_toppings Table

```sql
-- Insert pizza toppings
INSERT INTO pizza_toppings VALUES
  (1, 'Bacon'),
  (2, 'BBQ Sauce'),
  (3, 'Beef'),
  (4, 'Cheese'),
  (5, 'Chicken'),
  (6, 'Mushrooms'),
  (7, 'Onions'),
  (8, 'Pepperoni'),
  (9, 'Peppers'),
  (10, 'Salami'),
  (11, 'Tomatoes'),
  (12, 'Tomato Sauce');
```

Insert Values in pizza_toppings Table

-- 1. How many pizzas were ordered?

```sql
SELECT
    COUNT(*) AS Total_Orders
FROM
    customer_orders
```

Result Grid

	Total_Orders
▶	11

-- 2. How many unique customer orders were made?

```sql
SELECT
    COUNT(DISTINCT order_id), COUNT(DISTINCT customer_id)
FROM
    customer_orders
```

Result Grid | Filter Rows:

	COUNT(DISTINCT order_id)	COUNT(DISTINCT customer_id)
▶	10	5

-- 3. How many successful orders were delivered by each runner?

```sql
SELECT
    COUNT(*) AS Successfull_Orders,
    runner_orders.runner_id
FROM
    runners
        JOIN
    runner_orders ON runners.runner_id = runner_orders.runner_id
WHERE
    runner_orders.cancellation IS NULL
GROUP BY runner_orders.runner_id
```

Result Grid | Filter Rows:

Successfull_Orders	runner_id
2	1
3	2
1	3

-- 4. How many of each type of pizza was delivered?

```sql
SELECT
    COUNT(*), pizza_name as Type_of_pizza_delivered
FROM
    customer_orders
        JOIN
    pizza_names ON customer_orders.pizza_id = pizza_names.pizza_id
    join runner_orders on runner_orders.order_id = customer_orders.order_id
WHERE cancellation IS NULL
GROUP BY pizza_name
```

Result Grid | Filter Rows:

COUNT(*)	Type_of_pizza_delivered
5	Meatlovers
2	Vegetarian

-- 5. How many Vegetarian and Meatlovers were ordered by each customer?

```sql
SELECT
    COUNT(*), pizza_name, customer_id
FROM
    customer_orders
        JOIN
    pizza_names ON customer_orders.pizza_id = pizza_names.pizza_id
WHERE
    pizza_name IN ('Meatlovers' , 'Vegetarian')
GROUP BY pizza_name , customer_id
```

Result Grid | Filter Rows:

COUNT(*)	pizza_name	customer_id
2	Meatlovers	101
2	Meatlovers	102
2	Meatlovers	103
2	Meatlovers	104
1	Vegetarian	102
1	Vegetarian	101
1	Vegetarian	105

-- 6. What was the maximum number of pizzas delivered in a single order?

```sql
SELECT
    COUNT(*) AS Pizza_count, customer_orders.order_id
FROM
    customer_orders
        JOIN
    runner_orders ON customer_orders.order_id = runner_orders.order_id
WHERE
    runner_orders.cancellation IS NULL
GROUP BY customer_orders.order_id
ORDER BY Pizza_count DESC
LIMIT 1;
```

Result Grid | Filter Rov

	Pizza_count	order_id
▶	2	3

-- 7. How many pizzas were delivered that had both exclusions and extras?

```sql
SELECT
    COUNT(*) as Pizzas_With_Both_Exclusions_Extras
FROM
    customer_orders
        JOIN
    runner_orders ON customer_orders.order_id = runner_orders.order_id
WHERE
    customer_orders.exclusions IS NOT NULL AND
    customer_orders.extras IS NOT NULL;
```

Result Grid | Filter Rows:

Pizzas_With_Both_Exclusions_Extras
6

-- 8. What was the total volume of pizzas ordered for each hour of the day?

```sql
SELECT
    HOUR(order_date) AS Hour_of_the_day,
    COUNT(*) AS Total_Orders
FROM
    customer_orders
GROUP BY Hour_of_the_day
```

Result Grid | Filter Rows:

Hour_of_the_day	Total_Orders
18	2
19	1
23	3
13	1
21	3
11	1

-- 9. What was the volume of orders for each day of the week?

```sql
SELECT
    DAYNAME(order_date) AS Day_of_the_week,
    COUNT(*) AS Total_Orders
FROM
    customer_orders
GROUP BY Day_of_the_week
```

Result Grid Filter Rows:

Day_of_the_week	Total_Orders
Wednesday	5
Thursday	3
Saturday	2
Friday	1

-- 10. What is the successful delivery percentage for each runner?

```sql
SELECT
    runner_id,
    count(CASE WHEN cancellation IS NULL THEN 1 END) * 100.0 / COUNT(*) AS successful_delivery_percentage
FROM
    runner_orders
GROUP BY  runner_id
```

Result Grid | Filter Rows:

runner_id	successful_delivery_percentage
1	50.00000
2	75.00000
3	50.00000

50.00000