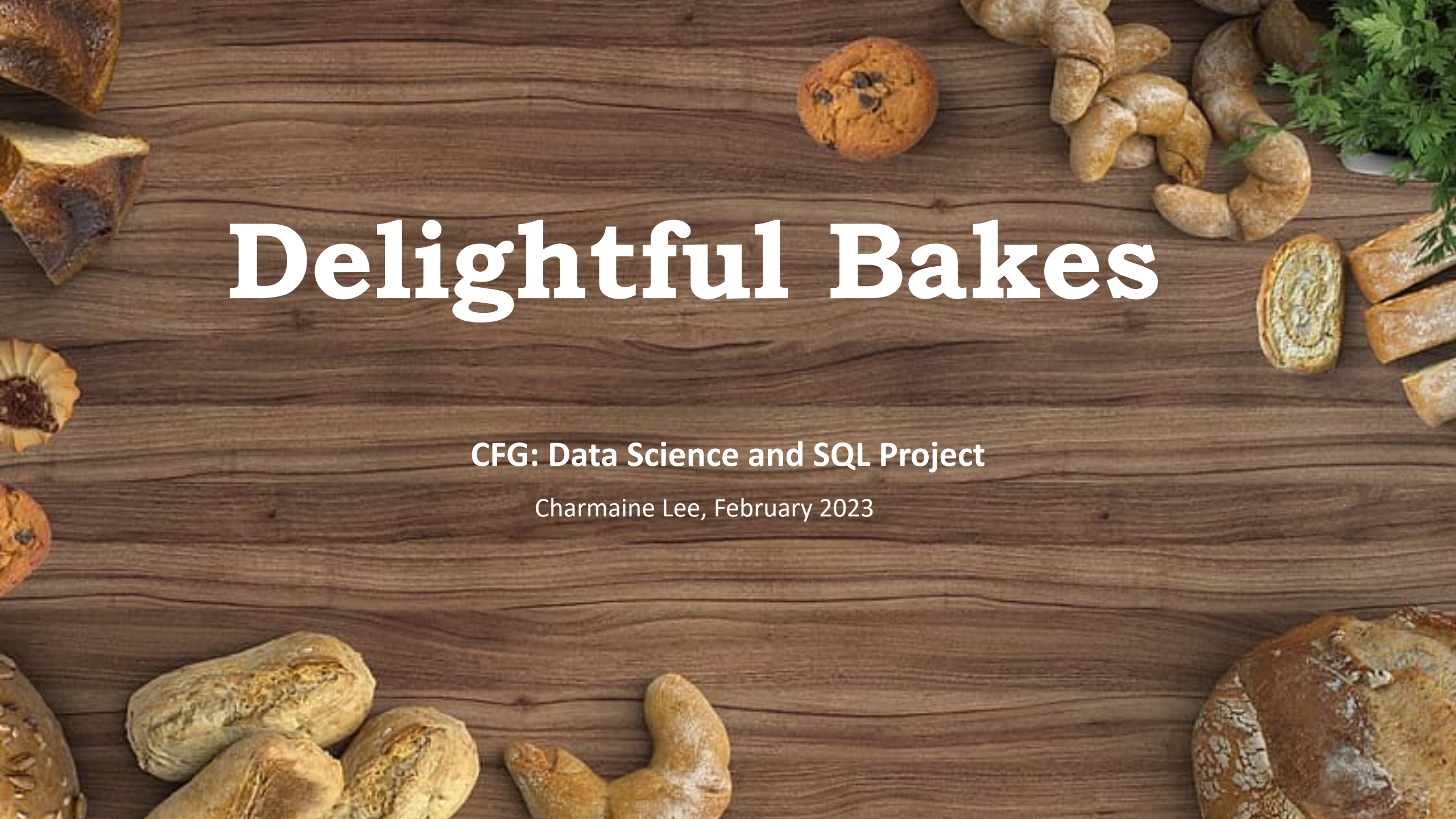


Delightful Bakes

CFG: Data Science and SQL Project

Charmaine Lee, February 2023

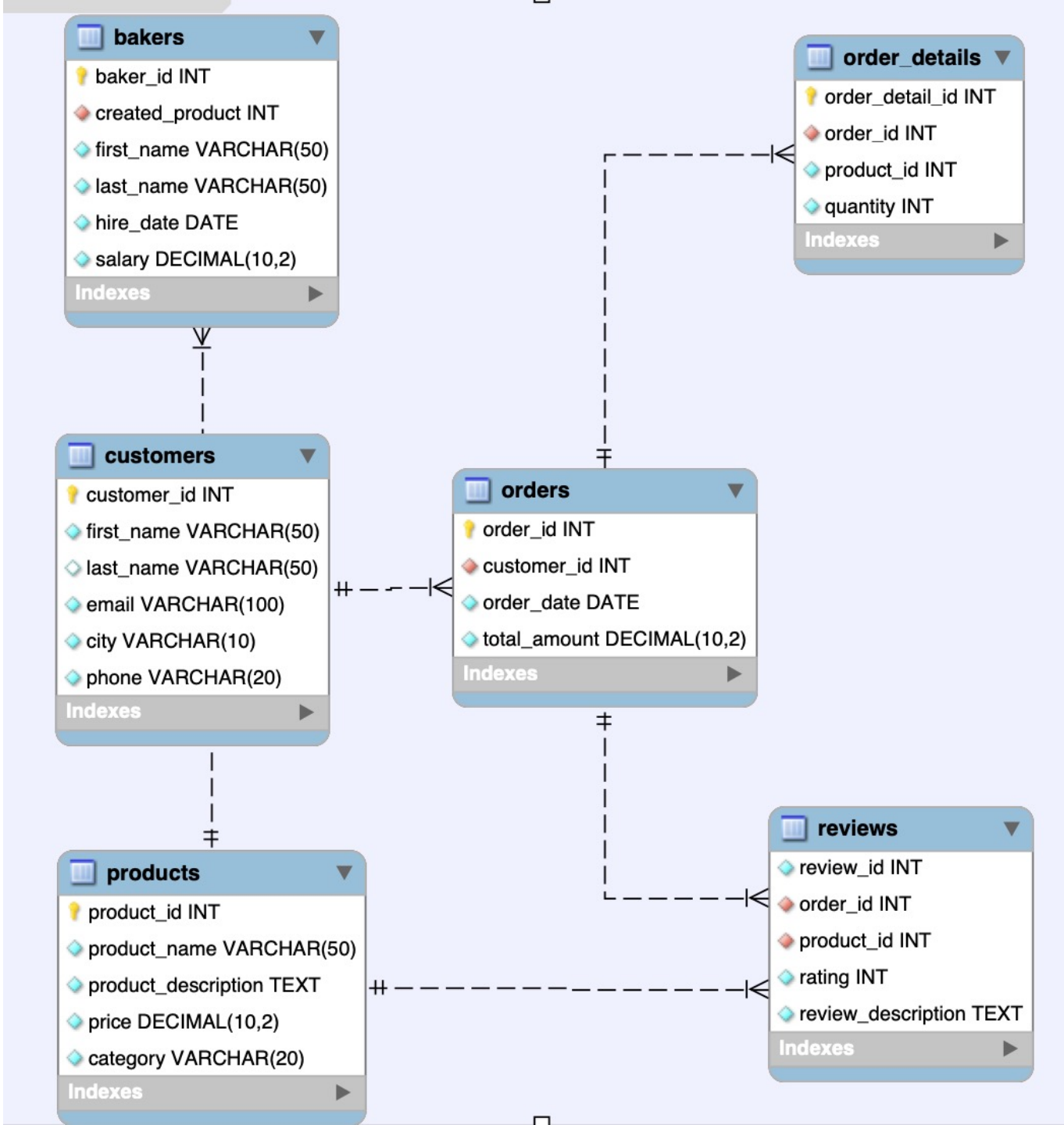
About:

In an alternate world, I would open a bakery with the essentials and specialties that are from my country, Singapore.

The Database contains sales and products information for our bakery.

It is expected to be used as analyzing our sales as a small independent bakery and analyze how our products perform with consumers.

ER Diagram

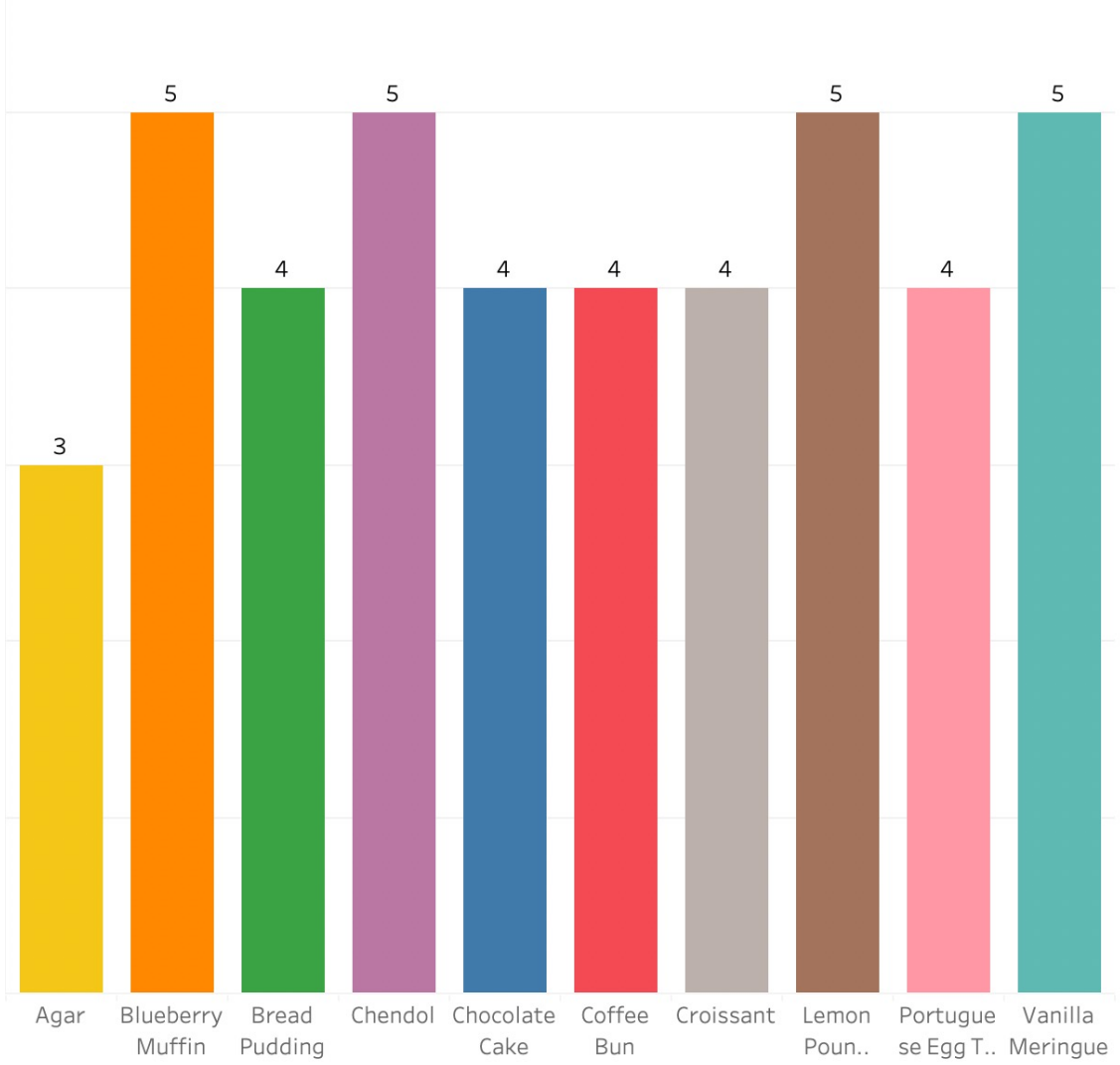

bakers
baker_id INT
created_product INT
first_name VARCHAR(50)
last_name VARCHAR(50)
hire_date DATE
salary DECIMAL(10,2)
Indexes
order_details
order_detail_id INT
order_id INT
product_id INT
quantity INT
Indexes
customers
customer_id INT
first_name VARCHAR(50)
last_name VARCHAR(50)
email VARCHAR(100)
city VARCHAR(10)
phone VARCHAR(20)
Indexes
orders
order_id INT
customer_id INT
order_date DATE
total_amount DECIMAL(10,2)
Indexes
products
product_id INT
product_name VARCHAR(50)
product_description TEXT
price DECIMAL(10,2)
category VARCHAR(20)
Indexes
reviews
review_id INT
order_id INT
product_id INT
rating INT
review_description TEXT
Indexes

Demo time

VIEW 1: Creates a view with inner joins, to find what are our customer's reviews on our products.

Highest rated (5 out of 5) are:

Blueberry muffin;
Chendol;
Lemon Pound Cake;
Vanilla Meringue



VIEW 2: Creates a view with left joins to show all customers, even if they don't have any orders.

```sql
CREATE VIEW cust_orders AS
SELECT
  c.customer_id,
  o.order_id,
  o.order_date,
  o.total_amount
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;

SELECT * FROM cust_orders;
```

	customer_id	order_id	order_date	total_amou...
▶	1	1	2022-12-01	50.97
	2	2	2022-12-02	23.97
	3	3	2022-12-03	37.95
	4	4	2022-12-04	56.93
	5	5	2022-12-05	28.91
	6	6	2023-02-12	20.47
	7	7	2023-02-13	15.99
	8	8	2023-02-14	10.49
	9	9	2023-02-15	25.97
	10	10	2023-02-16	18.99

STORED FUNCTIONS

SCENARIO:

Every month, Delightful Bakes gives *10% off* the total cost of purchase. The following function can be run during the sale to find the final price the customer needs to pay.

```
DELIMITER //
CREATE FUNCTION calculate_discount(total_amount FLOAT(2))
RETURNS FLOAT(2)
DETERMINISTIC
BEGIN
  DECLARE final_price FLOAT(2);
  SET final_price = total_amount * 0.9;
  RETURN final_price;
END//
DELIMITER ;
```

To demonstrate, to calculate final bill after 10% discount:

```
SELECT calculate_discount(total_amount) AS final_price
FROM orders;
```

final_price
45.873
21.573
34.155
51.237
26.019
18.423

SUBQUERY

To analyse which baker produced the most popular product.

```sql
WITH baker_rank AS (
    SELECT
        p.product_id,
        p.product_name,
        SUM(od.quantity) AS total_quantity
    FROM products p
    JOIN order_details od ON p.product_id = od.product_id
    GROUP BY p.product_id, p.product_name
    ORDER BY total_quantity DESC)

SELECT
    b.baker_id,
    p.product_name,
    total_quantity
FROM bakers b
JOIN products p ON b.baker_id = p.product_id
JOIN baker_rank br ON p.product_id = br.product_id
```

baker_id	product_name	total_quantity
1	Chocolate Cake	4
2	Blueberry Muffin	7
3	Coffee Bun	5
4	Vanilla Meringue	3
5	Bread Pudding	4
6	Agar	2
7	Chendol	4
8	Portuguese Egg Tart	1
9	Lemon Pound Cake	3
10	Croissant	2

TRIGGER

I have created a trigger that updates the the last_ordered_on column in 'customers' table whenever *a new order* is inserted into the 'orders' table.

```
DELIMITER //
CREATE TRIGGER update_last_order_date
    AFTER INSERT ON orders
    FOR EACH ROW
BEGIN
    UPDATE customers
    SET last_order_date = NEW.order_date
    WHERE customer_id = NEW.customer_id;
END//
DELIMITER ;
```

TRIGGER

Insert a new order into the 'orders' table and then check the updated value of the last_order_date column in the 'customers' table to demonstrate.

```
INSERT INTO orders (order_id, customer_id, order_date, total_amount)
VALUES (1000, 1, '2023-03-01', '25.99');

SELECT * FROM customers WHERE customer_id = 1;
```

customer_id	first_name	last_name	email	city	phone	last_order_date
1	John	Doe	johndoe@email.com	NY	123-456-7890	2023-03-01

QUERIES WITH GROUP BY

This query will find the names of customers who have made 3 or more orders and show the total number of orders they have made.

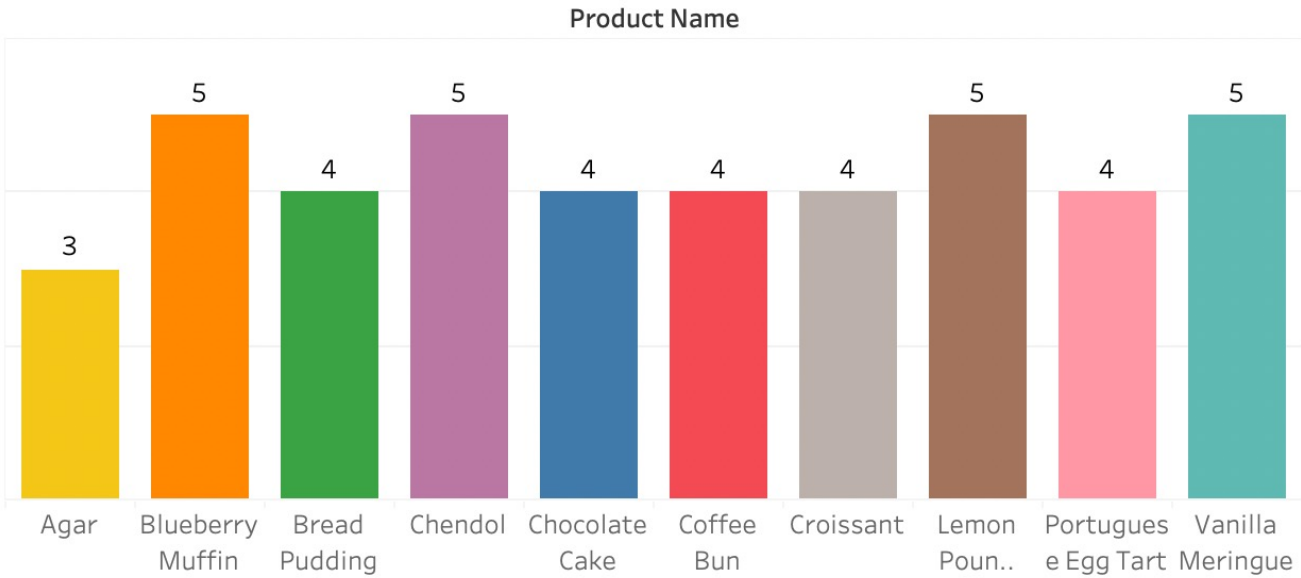
```sql
SELECT
  c.customer_id,
  COUNT(o.order_id) AS total_orders
FROM
  customers c
  JOIN orders o ON c.customer_id = o.customer_id
GROUP BY
  c.customer_id
HAVING
  total_orders > 1;
```

	customer_id	total_orders
▶	1	2

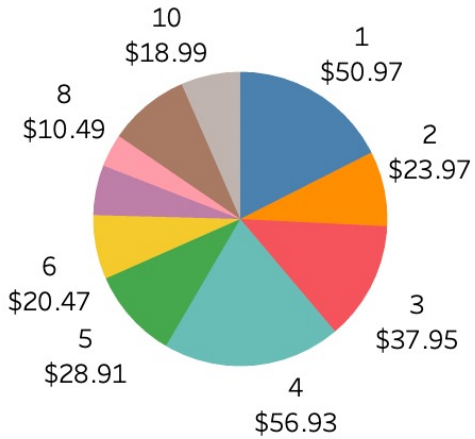
Delightful Bakes Dashboard

By: Charmaine Lee

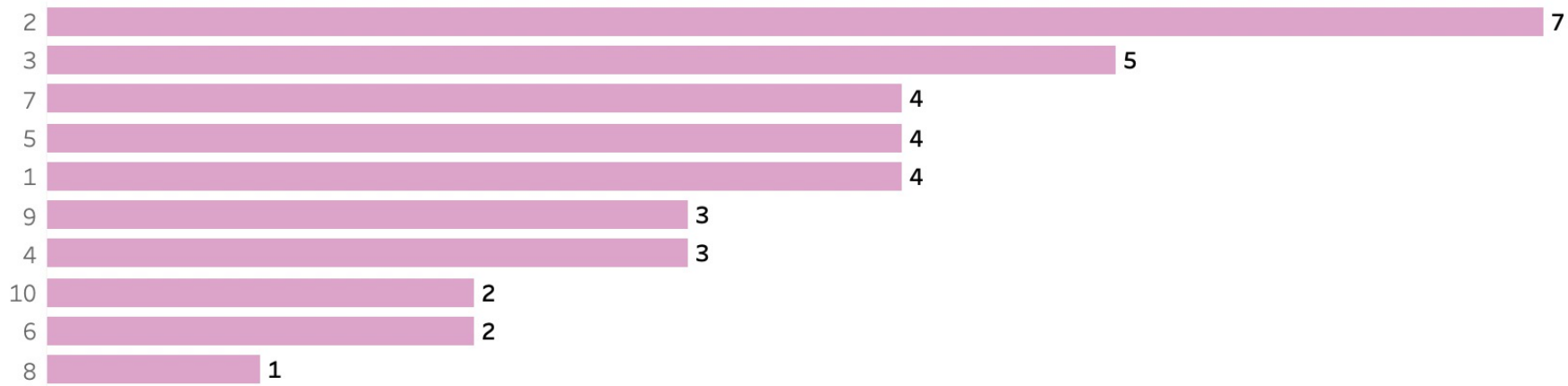
Product Popularity



Spent Amt



Baker that sold the most products



Practicality into coding

Data visualization improvements

Reflections

Finding business problems