Problems

1. **Get the total quantity of all ingredients in stock.**

```
SELECT SUM(quantity_in_stock) AS total_quantity FROM ingredients;
```

2. **Retrieve the names of customers who made a transaction in November 2024.**

```
SELECT c.name
FROM customers c
JOIN transactions t ON c.customer_id = t.customer_id
WHERE MONTH(transaction_date) = 11 AND YEAR(transaction_date) = 2024;
```

3. **Count the number of transactions per customer.**

```
SELECT c.name, COUNT(t.transaction_id) AS transaction_count
FROM customers c JOIN transactions t ON c.customer_id = t.customer_id
GROUP BY c.customer_id;
```

4. **Get the total revenue generated from all transactions.**

```
SELECT SUM(total_price) AS total_revenue FROM transactions;
```

5. **Find the average price of all products.**

```
SELECT AVG(price) AS average_price FROM products;
```

6. **Write a query to select the names of all products and their corresponding ingredients.**

```
SELECT p.name AS product_name, i.name AS ingredient_name
FROM products p
JOIN product_ingredients pi ON p.product_id = pi.product_id
JOIN ingredients i ON pi.ingredient_id = i.ingredient_id;
```

7. **Show details of products along with their prices, ordered by price in descending order.**

```
SELECT name, price FROM products ORDER BY price DESC;
```

8. **Find the top 5 most expensive services.**

```
SELECT name, price FROM services ORDER BY price DESC LIMIT 5;
```

9. **Find the average transaction value for November 2024.**

```
SELECT AVG(total_price) AS average_transaction_value
FROM transactions
WHERE MONTH(transaction_date) = 11 AND YEAR(transaction_date) = 2024;
```

10. **Write a query to select the names and contact info of all customers who have made a transaction after '2024-01-01'.**

```
SELECT c.name, c.contact_info

FROM customers c

JOIN transactions t ON c.customer_id = t.customer_id

WHERE t.transaction_date > '2024-01-01';
```

11. **Write a query to find the total sales (total_price) for each customer by joining the transactions table with the customers table. Display the customer_id, name, and the total amount they spent.**

```
SELECT c.customer_id, c.name, SUM(t.total_price) AS total_spent

FROM customers c

JOIN transactions t ON c.customer_id = t.customer_id

GROUP BY c.customer_id, c.name;
```

12. **Write a query to find all products that require more than one ingredient. Display the product name and the number of ingredients used for each.**

```
SELECT p.name AS product_name, COUNT(pi.ingredient_id) AS ingredient_count

FROM products p

JOIN product_ingredients pi ON p.product_id = pi.product_id

GROUP BY p.name

HAVING COUNT(pi.ingredient_id) > 1;
```

13. **Write a query to list all products that have low stock (i.e., their ingredient quantity is less than 10).**

```
SELECT DISTINCT products.name AS product_name FROM products

INNER JOIN product_ingredients ON products.product_id =

product_ingredients.product_id
```

```
INNER JOIN ingredients ON product_ingredients.ingredient_id =
ingredients.ingredient_id
WHERE ingredients.quantity_in_stock < 10;
```

14. **Write a query to find the most expensive product in the products table. Display the product name and price.**

```
SELECT name, price
FROM products
WHERE price = (SELECT MAX(price) FROM products);
```

15. **Write a query to find the total sales (total_price) per product. Display the product name and the total sales amount.**

```
SELECT p.name AS product_name, SUM(t.total_price) AS total_sales
FROM products p
JOIN product_ingredients pi ON p.product_id = pi.product_id
JOIN transactions t ON pi.product_id = t.product_id
GROUP BY p.name;
```

16. **Join transactions and customers to view customer names with their transaction details.**

```
SELECT transactions.transaction_id, transactions.transaction_date,
customers.name, transactions.total_price
FROM transactions
INNER JOIN customers ON transactions.customer_id =
customers.customer_id;
```

17. **Get the details of the product with the highest price.**

```
SELECT * FROM products ORDER BY price DESC LIMIT 1;
```

18. **Select all products where the price is between 50 and 150.**

```
SELECT name, price FROM products WHERE price BETWEEN 50 AND 150;
```

19. **Find the minimum and maximum prices of products.**

```
 SELECT MIN(price) AS minimum_price, MAX(price) AS maximum_price
 FROM products;
```

20. **Get the products that are priced higher than the average price of products in their category.**

```
SELECT p.name, p.price
FROM products p
WHERE p.price > (
  SELECT AVG(price) FROM products WHERE category = p.category);
```

21. **Count the number of customers who have made at least one transaction.**

```
SELECT COUNT(DISTINCT customer_id) AS customer_count FROM
transactions;
```

22. **List each product's name along with the total quantity of ingredients required for that product, ordered by product name.**

```
SELECT p.name AS product_name, SUM(pi.quantity) AS total_ingredients
FROM products p
JOIN product_ingredients pi ON p.product_id = pi.product_id
GROUP BY p.name
ORDER BY p.name;
```

23. **Retrieve all products that cost more than the average price of products.**

```
SELECT name, price FROM products WHERE price > (SELECT AVG(price)
FROM products);
```

24. **Get the names and descriptions of services that do not have "event" in their name.**

```
SELECT name, description FROM services WHERE name NOT LIKE
'%event%';
```

25. **Find the name of customers who made transactions in November 2024.**

```
SELECT DISTINCT c.name
FROM customers c
JOIN transactions t ON c.customer_id = t.customer_id
WHERE MONTH(t.transaction_date) = 11 AND YEAR(t.transaction_date) = 2024;
```

26. **Get the name and quantity of all ingredients used in the product 'Carbonara'.**

```
SELECT i.name, pi.quantity
FROM ingredients i
JOIN product_ingredients pi ON i.ingredient_id = pi.ingredient_id
JOIN products p ON pi.product_id = p.product_id
WHERE p.name = 'Carbonara';
```

27. **Get the name and date of transactions for a specific customer, say, 'John Doe'.**

```
SELECT t.transaction_id, t.transaction_date
FROM transactions t
JOIN customers c ON t.customer_id = c.customer_id
WHERE c.name = 'John Doe';
```

28. **Get the average price of all products.**

```
SELECT AVG(price) AS average_price
FROM products;
```

29. **Find transactions with a total price greater than 500.**

```
SELECT transaction_id, total_price
FROM transactions
WHERE total_price > 500;
```

30. **Write a queary that inserts a new transaction that happened on 2024-11-09 by**

customer_id 5 that totaled ₱988.00

```
INSERT INTO transactions (transaction_date, costumer_id, total_price)

VALUES (‘2024-11-09’, 5, 988.00);
```