**QUESTION ONE:**

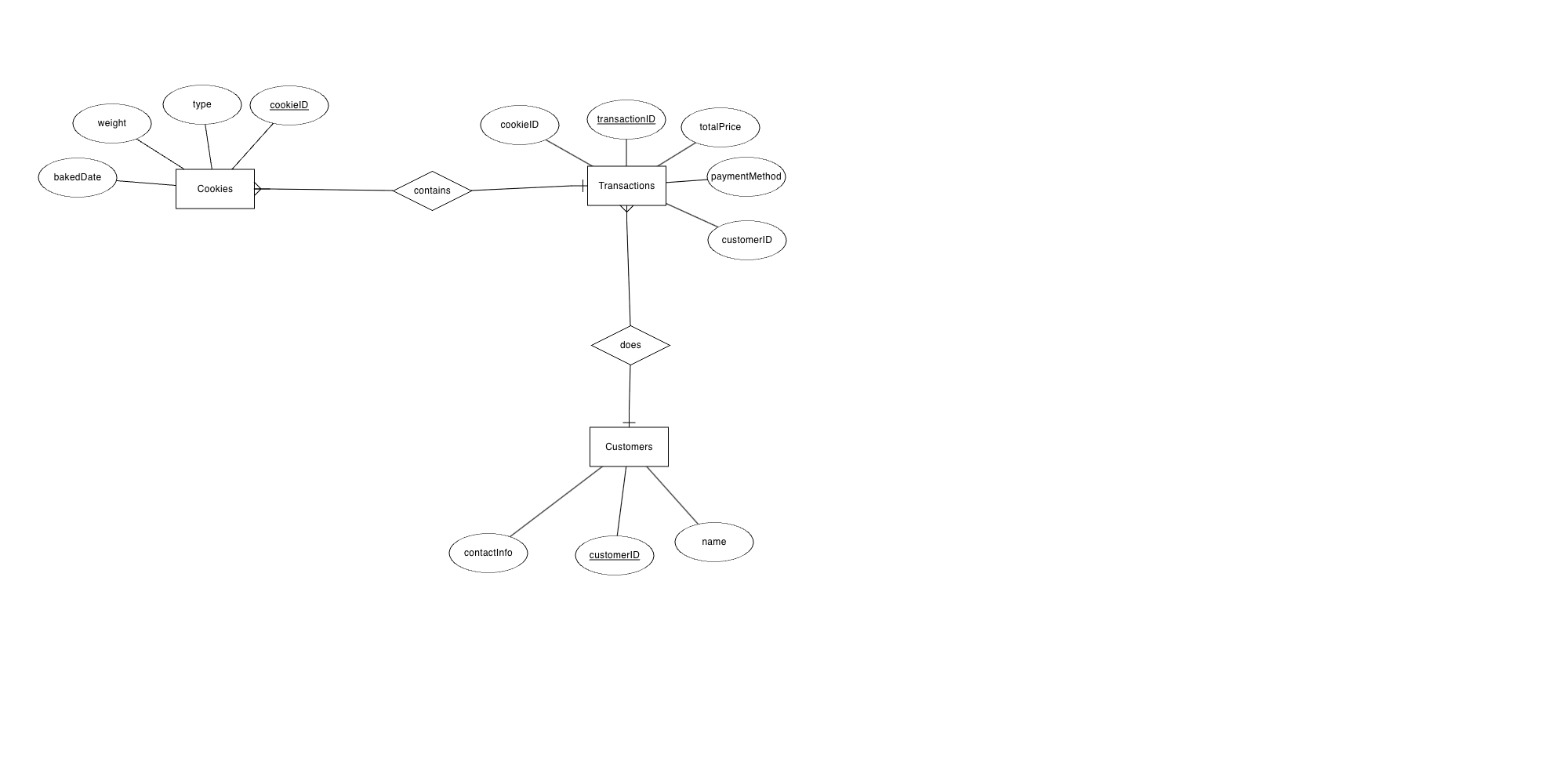


Figure: Entity Relationship diagram for Yasmine’s business

The diagram is made using **erdplus.com**. In this ERD, the relationships are depicted by lines connecting the entities, with cardinality (one-to-many, many-to-one) indicated by **crow's feet** or other notations.

**Entities and Attributes**:

1. Cookies:
   1. cookieID (Primary Key)
   2. type
   3. weight
   4. bakedDate
2. Transactions:
   1. transactionID (Primary Key)
   2. cookieID (Foreign Key referencing Cookie)
   3. customerID (Foreign Key referencing Customer)
   4. totalPrice
   5. paymentMethod
3. Customers:
   1. customerID (Primary Key)
   * name
   * contactInfo

**Relationships:**

* **One-to-Many relationship between Cookies and Transactions**: Each transaction can have many cookies. But a cookie with a unique ID belongs to a single transaction.
* **Many-to-One relationship between Transaction and Customer:** A customer can do many transactions. But a transaction belongs to a single unique customer.

**QUESTION 2:**

1. **To create Customers table:**

| CREATE TABLE Customers (  customerID INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,  name VARCHAR(255) NOT NULL,  contactInfo VARCHAR(255) ); |
| --- |

1. **To create Cookies Table:**

| CREATE TABLE Cookies (  cookieID INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,  type VARCHAR(255),  weight FLOAT,  bakedDate DATE ); |
| --- |

1. **To create Transactions Table:**

| CREATE TABLE Transaction (  transactionID INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,  cookieID INT,  customerID INT,  totalPrice FLOAT,  paymentMethod VARCHAR(200),  FOREIGN KEY (cookieID) REFERENCES Cookies(cookieID),  FOREIGN KEY (customerID) REFERENCES Customers(customerID) ); |
| --- |

**QUESTION 3:**

1. **For Customers table:**

| INSERT INTO Customers  (name, contactInfo)  VALUES  ('Jack Sparrow', 'jack@carribean.com'); |
| --- |

1. **For Cookies table:**

| INSERT INTO Cookies  (type, weight, bakedDate)  VALUES  ('Choc Chip', 21, '2024-04-05'); |
| --- |

1. **For Transactions table:**

| INSERT INTO Transactions  (cookieID, customerID, totalPrice, paymentMethod)  VALUES  (1, 3, 5.8, 'Apple Pay'); |
| --- |

**QUESTION 4:**

| SELECT   YEAR(bakedDate) AS year,  MONTH(bakedDate) AS month,  COUNT(\*) AS numCookies FROM   Cookies WHERE   YEAR(bakedDate) IN (YEAR(CURDATE()), YEAR(CURDATE()) - 1) GROUP BY   YEAR(bakedDate), MONTH(bakedDate) ORDER BY   YEAR(bakedDate) DESC, MONTH(bakedDate) DESC; |
| --- |

This command will retrieve the number of cookies baked in each month of the current and last year. If there are no sales of a particular type in a month, it won't show up in the result.

The hypothetical result can look like the one below which shows the number of cookies in each month of the current year and last year.

| **year** | **month** | **numCookies** |
| --- | --- | --- |
| 2024 | 12 | 20 |
| 2024 | 11 | 15 |
| 2024 | 10 | 25 |
| 2024 | 9 | 30 |
| 2024 | 8 | 18 |
| 2024 | 7 | 22 |
| 2023 | 12 | 10 |
| 2023 | 11 | 12 |
| 2023 | 10 | 20 |
| 2023 | 9 | 25 |

For example, in December 2023, 10 cookies were baked, while in November 2024, 15 cookies were baked. Months where no cookies were baked are not included in the result although we can write a query such that it shows 0 if the cookies were not sold in a particular month, which can provide a more comprehensive and accurate representation of the data for trend analysis.

**QUESTION 5:**

The four basic types of commands utilised within a MySQL table to manipulate data are:

1. **SELECT**: This command retrieves data from one or more tables based on specified criteria and returns the result set.

For example,

| SELECT \* FROM Cookies WHERE type = 'Choc Chip'; |
| --- |

This query selects all columns (\*) from the Cookies table where the type column is 'Choc Chip'.

1. **INSERT**: This command adds new rows of data to a table.

For example,

| INSERT INTO Customers (name, contactInfo) VALUES ('Elizabeth Swann', 'elizabeth@carribean.com'); |
| --- |

This command inserts a new row into the Customers table with the provided name and contact information.

1. **UPDATE**: This command modifies existing data within a table based on specified conditions.

For example,

| UPDATE Cookies SET weight = 25 WHERE type = 'Choc Chip'; |
| --- |

This command updates the weight column in the Cookies table to 25 where the type column is 'Choc Chip'.

1. **DELETE**: This command removes rows of data from a table based on specified conditions.

DELETE FROM Transactions WHERE transactionID = 1;

This command deletes the row from the Transactions table where the transactionID is 1.

**QUESTION 6:**

There are usually three levels in an analytical hierarchy which are as follows:

1. **Descriptive Analytics**: This level involves analyzing historical data to understand what happened in the past. It provides insights into trends, patterns, and relationships in the data.
2. **Predictive Analytics**: This level focuses on forecasting future outcomes based on historical data and statistical models. It uses techniques such as regression analysis, machine learning, and data mining to make predictions.
3. **Prescriptive Analytics**: This level goes beyond predicting future outcomes and recommends actions to achieve desired outcomes. It involves optimization and simulation techniques to identify the best course of action given various constraints and objectives.

These three levels encompass the spectrum of analytics, starting from understanding historical data to predicting future trends and finally recommending actions to achieve desired outcomes. Each level builds upon the previous one, with Descriptive Analytics providing the foundation for Predictive Analytics, and Predictive Analytics forming the basis for Prescriptive Analytics. This hierarchy reflects the progressive nature of analytics, where organizations move from analyzing past data to making informed decisions for the future.

**QUESTION 7:**

In MySQL, two methods commonly used to identify and interrogate the data structure of a particular table are the **DESCRIBE** command and querying the **INFORMATION\_SCHEMA**. The **DESCRIBE** command provides a quick overview of the table structure, listing the column names along with their data types and constraints. Its advantage lies in its simplicity and ease of use, making it suitable for quick reference. However, it lacks detailed information on complex data structures and relationships. On the other hand, querying the **INFORMATION\_SCHEMA** tables offers comprehensive metadata about the table, including detailed column attributes, constraints, and indexes. While this method provides more comprehensive details, its disadvantage is that it requires more complex SQL queries, potentially being more challenging for beginners.

**QUESTION 8:**

Creating the entity-relationship diagram (ERD) in Question One was a relatively straightforward process. However, the most difficult part was choosing the right tool to create the diagram. There are numerous options available for representing ERDs, each with its advantages and drawbacks. I opted for ERDplus, which employs ovals for attributes, diamonds for relationships, and rectangles for entities. This choice was influenced by familiarity with ERDplus and its recommendation during my studies in a Database Systems unit. One issue I encountered was in deciding between using crow's feet notation and simple (1, M) notation for depicting cardinality in relationships. I chose to use crow's feet notation for clarity and convention. However, an alternative approach could have been to use a tool like Lucidchart, which utilizes tables to list attributes and offers greater ease of viewing as the ERD increases in size and complexity. This approach might have provided a clearer representation of the relationships within the database schema.