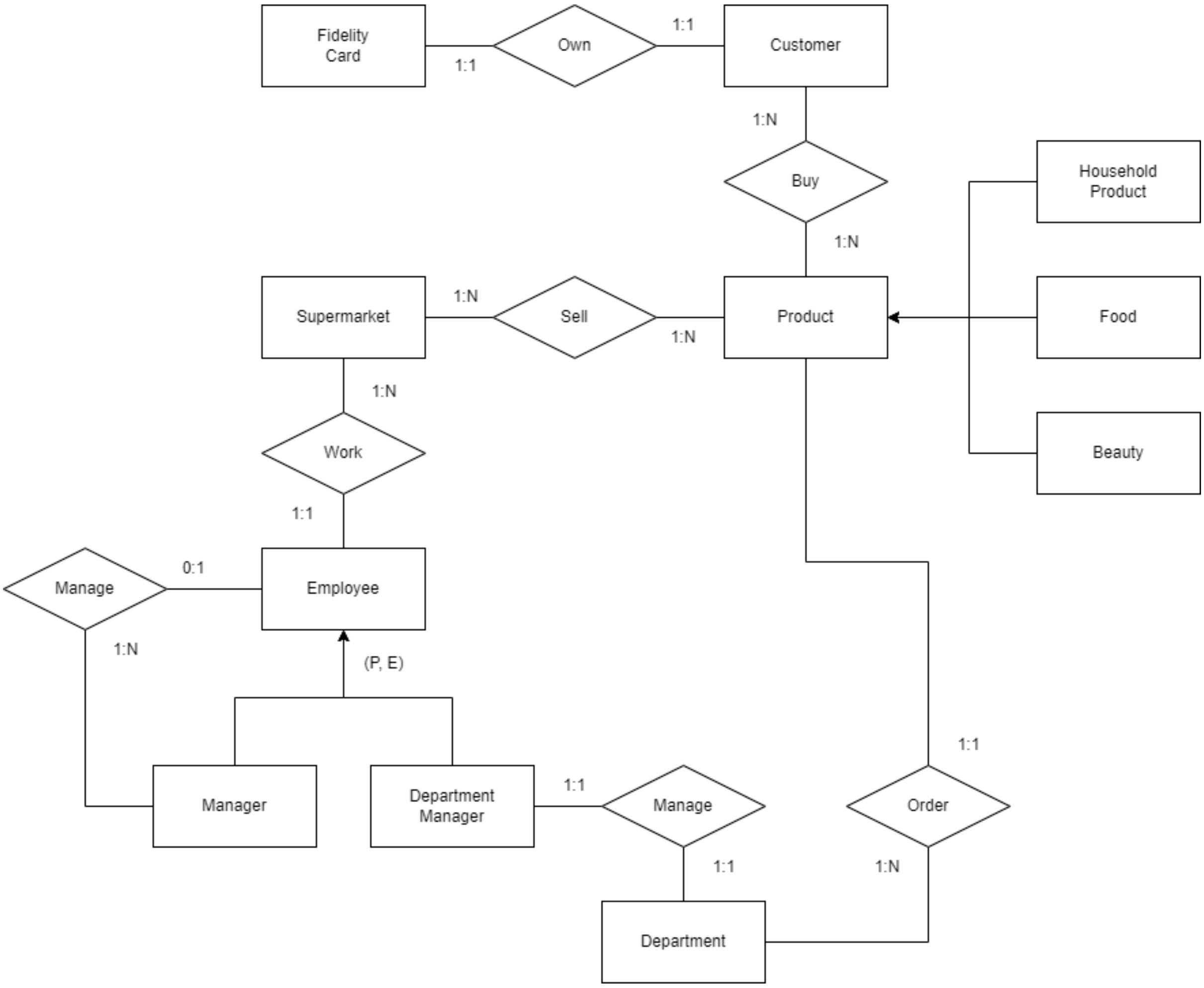
Consider the following ER Diagram



The following attributes describe the entities. The primary keys are underlined.

- **Supermarket** - Name, Location (Address, City, State), <u>Supermarket_ID</u>,
- **Employee** - Name, Surname, BirthDate, <u>Employee_ID</u>, HiringDate
    - **Manager** - (...)
    - **Department Manager** - (...)
- **Department** - Name, <u>Department_ID</u>, Description
- **Product** - Name, <u>Product_ID</u>, Price, Description, Sale (%)
    - **Household Product** - (...), Dangerous (Yes/No)
    - **Food** - (...), ExpiryDate
    - **Beauty** - (...)
- **Customer** - Name, Surname, Email, <u>Personal_ID</u>, BirthDate
- **Fidelity Card** - <u>FC_Number</u>, IssuingDate

The following attributes describe the relationships.

- **Order** - Quantity
- **Buy** - Quantity

N.B. Managers do not Manage other Managers. Managers only manage Employees and Department Managers. A manager manages all the employees in a supermarket. There's only one manager per supermarket.

Exercise 1 - SQL (1.5 PT)

(1st Exercise on table) (1.5 PT)

Exercise 2 - Neo4j (4.5 PT)

Consider the entities Employee, Manager, Department Manager, and Department from the ER model and suppose you want to store the respective data instances in a graph database. Sketch a graph model/example describing the nodes, main attributes, and edges. Either show an example graph or a graph with types. (1 PT)

Write a Cypher query to collect the number of department managers managed by a manager who manages at least 30 employees for each manager (remember that department managers are Employees). (1.5 PT)

Write a Cypher query to return the collection of departments managed by a department manager hired later than its manager and who works in a supermarket with at least 5 department managers. (2 PT)

Exercise 3 - MongoDB (5.5 PT)

Consider the Fidelity Card, Customer, Product (with all its child entities). How many collections would you define? How would you implement the relations between the concepts? Provide a simple documental representation. (1 PT)

Write a query to count the number of products bought by customers named "Fernando" and whose fidelity card was issued later than 02/01/2020. Perform the query starting from **Customer_Collection**. (1 PT)

Write a query to collect the list of all the products whose price is greater than 10€ and whose sale is equal to 10%. Return only its Product_ID. Perform the query starting from **Product_Collection**. (1.5 PT)

Write a query to count the number of products each customer bought on sale. Perform the query starting from **Customer_Collection**. (2 PT)
N.B. Only customers with a Fidelity Card can access sales.

Exercise 4 - Elasticsearch (4 PT)

Consider the Household Product entity.

4.1. Provide the complete mapping of the index (i.e., field name, field type, the structure of the mapping, etc.) (1 PT)

PUT …

Write a query to return the list of all the products whose description includes the words “Cleaning” and “Mirror”, prioritising the dangerous ones. (1.5 PT)

Write a query to return the list of all the products whose prices exceed 10.00€, whose sale is more than 10%, and whose descriptions do not include the word “Floor”. The condition on the sale attribute must not affect the final score. (1.5 PT)

Exercise 5 - Cassandra (4 PT)

Consider the Customer table. Write a Cassandra script to perform the operations listed below.

Create the Customer table, with Personal_ID as the partition key and email as the clustering key, ordering the table based on the email. (1.5 PT)

Write a CQL query to collect the customer whose e-mail is “gastani.frinzi@gmail.com”. If any further operation is required, write its CQL code. (1 PT)

Create a Custom data type named “full name” that contains both the name and the surname of the customer (1.5 PT)