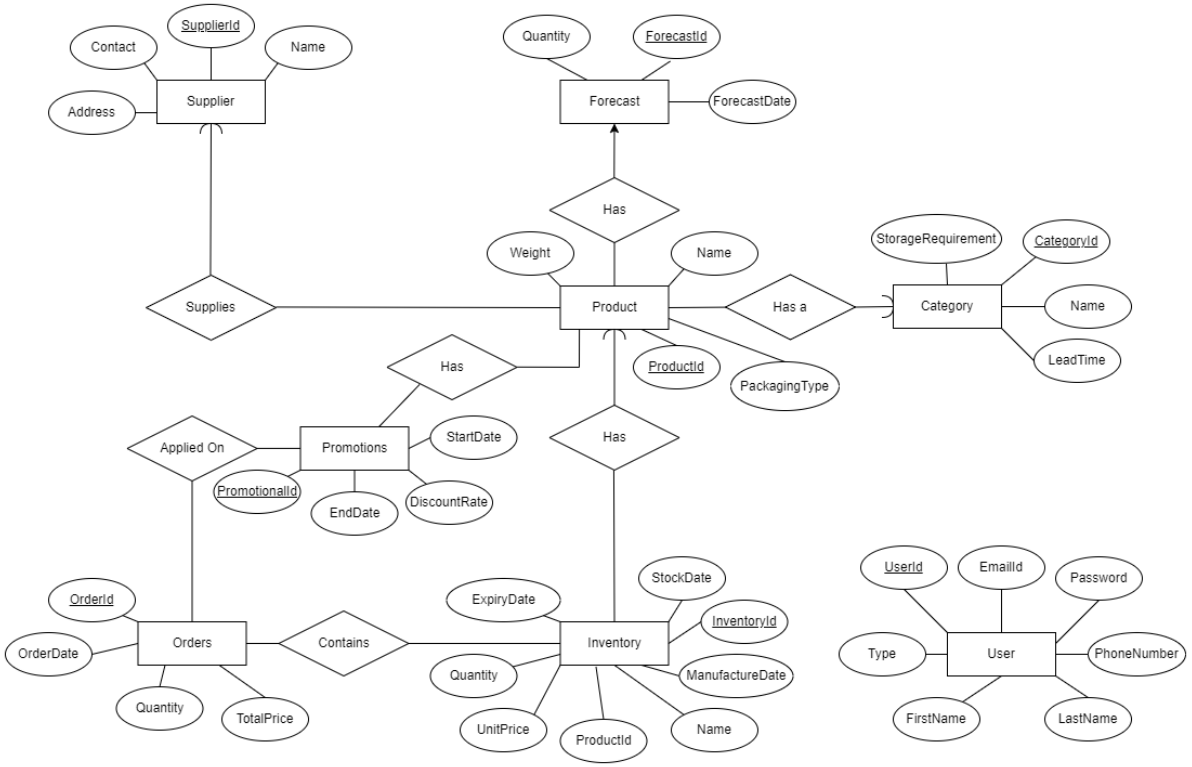
CS 411 – Project Task 1 (PT1) Task 2

Title	*Description*
Team Name	SelectStar
Project Title	Data-Driven Forecasting and Visualization of Fresh Food Demand in a Retail Store
Team Members	Mihir Shah (mihirss3@illinois.edu) Mohit Badve (mbadve2@illinois.edu) Prajakta Pikale (ppikale2@illinois.edu) Riya Tendulkar (rtend@illinois.edu)

Table of Contents

1. ER Diagram



https://drive.google.com/file/d/1vM0hk3YxSCf1iwk2rn18pSDwNqSXlmjh/view?usp=drive_link

2. Entity and Attribute Descriptions and Assumptions

1. **User**
    a. *Description:*
        i. The user has an admin functionality where he can load data into the database or can view it.
        ii. The user will also have access to view the dashboard.
        iii. The user is not a part of the environment, i.e. he/she is not the customer or buyer and has no relation to the other entities of the database and hence is presented as a separate entity altogether.
        iv. However, since we need to create a table for the user to ensure they can login and view/add data, we have included them as an entity in the ER.

    b. *Assumptions:*
        i. Email is unique for each user, ensuring no two users can register with the same email.
        ii. Admins manage various data, while analysts primarily use the system for data analysis.
        iii. Phone numbers are optional but assumed to be valid if provided.

c. Attributes and their Justification:

Attribute	Description	Comments
UserId	An auto generated unique id for every user of our application (analysts and administrators)	It is a Primary Key being 'unique not null' for each user.
Email	User provided email which is used to login to the application, notifications and password recovery.	It is Unique for each user.
Password	Password set by user which adheres to certain password requirements of the application. This along with email is used to authenticate and login to the application.	It is stored in an encrypted format to avoid credentials exposure.
Type	There are two types of users of our application - analysts and administrators. • Admins manage users, suppliers, products, categories, inventories, orders and promotional offers data. Ideally, most of this data will be inserted in an automated way, but admins simulate that in our application. • Analysts view and use dashboards, generate and download forecasted data.	It is an enum with only two values: analysts and admin.
FirstName	First name of the user	
LastName	Last name of the user	
PhoneNumber	PhoneNumber of the user	

2. Product

a. Description:

i. A product stores the information about the type of products in the store. Some examples would be – Eggs, Apples, Milk, etc.

b. Assumptions:

i. Weight is measured in a standardized unit (e.g., kilograms or grams).

ii. Price is not an attribute of product, as it is highly volatile and instead considered as an inventory attribute.

iii. Inventory is a product in stock.

c. Attributes and their Justification:

Attribute	Description	Comments
ProductId	An auto generated unique id for every product.	It is a Primary Key being 'unique not null' for each product.
Name	Name of the product.	

PackagingType	Packaging method of the product e.g. Canned, Sliced, Frozen	
Weight	Weight of the product	

3. **Inventory**
    a. *Description:*
        i. Inventory is essential in storing the details of the "Products" currently available in the store. This includes the quantity, manufacture, expiry, price of the product currently available in the store.

    b. *Assumptions:*
        i. LeadTime is assumed to be an average based on the category, not for individual products.

    c. *Why an entity instead of merging with Product table:*
        i. This is created as a separate entity because the "Product" table contains information about the product which will remain same for all instances of the product.
        ii. To store information, which is different for each instance of the product, we need to store the data separately.
        iii. For example, "Milk" has weight as 1ltr and is produced by Whole Foods. This information will be stored in the Product table. Now for a particular carton of the milk, it will be manufactured on 10/09/2024 and expires on 12/09/2024 and has a price of $2. This information will change with each carton (instance) of the product and hence is stored separately.

    d. *Attributes and their Justification:*

Attribute	Description	Comments
InventoryId	An auto generated unique id for every inventory.	It is a Primary Key being 'unique not null' for each inventory.
StockDate	Date when the product was stocked in the inventory	
UnitPrice	Sales price of the product for that inventory	
ManufactureDate	Mfg. Date of the product	
ExpiryDate	Exp.Date of the product	
Quantity	Number of units of the same product stored in that inventory	

4. **Supplier**
    a. *Description:*

i. A supplier provides products and is a manufacturer of sorts. Supplier is essential to forecast which particular supplier's products are sold more and if more products need to be ordered from a particular supplier.

b. Assumptions:

i. The Contact field refers to a valid contact method (e.g., phone number or email).

ii. Address and name are assumed to be accurate for business identification and communication.

c. Why an entity instead of merging with Product table:

i. Supplier and Product have a "one-to-many" relation, i.e. a product will have one supplier but a single supplier can provide multiple products.

ii. The supplier is not merged with the product table because the supplier's name, address and contact information will be repeated several times and create unnecessary redundancy. To avoid that, a separate entity Supplier is created.

d. Attributes and their Justification:

Attribute	Description	Comments
SupplierId	An auto generated unique id for every product supplier.	It is a Primary Key being 'unique not null' for each supplier.
Name	Name of the supplier	
Address	Address of the supplier	
Contact	Contact of the supplier	

5. Category

a. Description:

i. Category is used to store the "type" of product and all the properties associated with that specific type of product.

b. Assumptions:

i. LeadTime is assumed to be an average based on the category, not for individual products.

c. Why an entity instead of merging with Product table:

i. Category and Product have a "one-to-many" relation, i.e. a product will have one category but a single category can provide multiple products.

ii. Category is not merged with the product table because the attributes of category will be repeated several times and create unnecessary redundancy. To avoid that, a separate entity Category is created.

d. Attributes and their Justification:

Attribute	Description	Comments
CategoryId	An auto generated unique id for every product category.	It is a Primary Key being 'unique not null' for each category.
Name	Name of the category e.g. fruits, vegetables, milk, cheese.	
LeadTime	The usual expiry period of the category. e.g. Milk usually should be consumed within a week.	
StorageRequirements	The usual temperature, space, light and moisture conditions of the storage required for a particular category.	

6. **Promotional Offer**
   a. *Description:*
      i. Promotional offers are utilized to predict the demand during these offers. During certain promotional offers, the demands are increased and hence we have added this as an entity.

   b. *Assumptions:*
      i. We are assuming that a single promotional offer can be applied on multiple orders and an order can have multiple promotional offers applied to it.

   c. *Attributes and their Justification:*

Attribute	Description	Comments
PromotionalOfferId	An auto generated unique id for every promotional offer.	It is a Primary Key being 'unique not null' for each promotional offer.
ProductId	Product identified by ProductId reference	It is a Foreign Key reference to ProductId of Product Entity.
StartDate	Date from which the offer starts	
EndDate	Date after which the offer ends	
DiscountRate	Total discount in percentage	

7. **Orders**
   a. *Description:*
      i. Order entity contains the description of the orders which were placed. It will have information about the product that was purchased, the total cost, date, offer, etc.

   b. *Assumptions:*
      i. Orders have multiple products/inventories, and promotional offers.

ii. TotalPrice is the final cost after considering any promotional discounts.

c. Attributes and their Justification:

Attribute	Description	Comments
OrderId	An auto generated unique id for every order.	It is a Primary Key being 'unique not null' for each order.
OrderDate	Date on which the order is placed	
PromotionalOfferId	PromotionalOffer identified by PromotionalOfferId reference	It is a Foreign Key reference to PromotionalOfferId of PromotionalOffer Entity.
TotalPrice	Order Price	

8. Forecast

a. Description:

i. This entity is used to store the forecast predictions for the products.

b. Assumptions:

i. Forecasts are linked to specific products, predicting their future demand.

c. Attributes and their Justification:

Attribute	Description	Comments
ForecastId	An auto generated unique id for every forecast.	It is a Primary Key being 'unique not null' for each order.
ProductId	Product identified by ProductId reference.	It is a Foreign Key reference to ProductId of Product Entity.
ForecastDate	Date on which the forecast is made.	
ForecastQuantity	Quantity forecasted for the product	

3. Relationships and their Cardinalities

1. Supplier — Supplies — Product

- **Relationship:** A supplier supplies many products but a product can have only one supplier.
- **Cardinality:** One-to-Many
- **Description:** Each supplier can supply multiple products, but each product is supplied by only one supplier. A single supplier can provide many different fresh food products,

such as fruits, vegetables, dairy, and meats. For example, a supplier named "Green Farms" might deliver apples, oranges, and various vegetables to a retail store. This relationship is crucial for maintaining a diverse inventory of fresh foods, ensuring that stores can meet customer demand for a variety of healthy options.

2. Product — Has — Forecast

- **Relationship:** Each product can have one or more forecasts, and each forecast is associated with one product.
- **Cardinality:** One-to-Many
- **Description:** Each fresh food product can have several sales forecasts associated with it based on historical data. For instance, the sales forecast for strawberries might vary across different seasons, predicting higher demand in summer months for barbecues and gatherings. This relationship enables the retail store to anticipate the right amount of inventory needed to meet fluctuating customer preferences throughout the year.

3. Product — Has — Inventory

- **Relationship:** A product can have one or more inventory records, and each inventory record is for one product.
- **Cardinality:** One-to-Many
- **Description:** Each fresh food product can have multiple inventory records corresponding to different days, reflecting varying stock levels and prices. For example, organic spinach may have an inventory record for each day it is stocked, showing that there are 200 units available today at a price of $2.50, while tomorrow's record might indicate 180 units at $2.75. This approach allows for accurate tracking of inventory changes over time, helping to minimize waste and ensure that fresh products are consistently available, thereby supporting sustainability efforts in the store.

4. Product — Has — Promotions

- **Relationship:** A product can be involved in one or more promotions, and a promotion can apply to multiple products.
- **Cardinality:** Many-to-Many
- **Description:** Each fresh food product can be associated with multiple promotions, and each promotion can apply to several products. For instance, a promotion offering a 20% discount on all fruits may include products like apples, bananas, and strawberries. Conversely, a specific product, such as organic spinach, may participate in different promotions throughout the month, such as a "Buy One, Get One Free" deal and a seasonal discount for spring sales. This flexibility in promotions encourages customer engagement and boosts sales by allowing retailers to effectively market their products, leading to increased visibility and revenue for both the store and the suppliers.

5. Promotions — Applied On — Orders

- **Relationship:** One or more promotions can be applied to an order, and an order can benefit from multiple promotions.
- **Cardinality:** Many-to-Many
- **Description:** Many promotions can be applied to different customer orders. For example, a customer may buy fresh vegetables and fruits during a store promotion offering a discount on orders above a certain amount. This flexibility encourages customers to purchase more items, enhancing sales while promoting healthy eating choices.

6. **Orders — Contains — Inventory**

- **Relationship:** An order can contain one or more inventory items, and each inventory item can be part of one or more orders.
- **Cardinality:** Many-to-Many
- **Description:** An order can include one or more inventory items, and each inventory item can be part of multiple orders. For example, if a customer places an order for groceries, it may contain several inventory items such as organic carrots, whole grain bread, and almond milk. Each of these inventory items could also appear in multiple customer orders, as they are commonly purchased products. This relationship allows for flexible order fulfilment, ensuring that customers can easily obtain their desired products while enabling the retailer to efficiently manage stock levels. Additionally, it helps in tracking purchasing trends and popular items, which can inform future inventory decisions and promotions.

7. **Category — Has — Product**

- **Relationship:** A category can have one or more products, and each product belongs to exactly one category.
- **Cardinality:** One-to-Many
- **Description:** A category can contain one or more products, while each product is assigned to exactly one category. For instance, a "Fruits" category may include various products such as apples, bananas, and oranges. Each of these fruits belongs to the "Fruits" category and cannot be assigned to another category simultaneously. This structure helps in organizing products efficiently, making it easier for customers to find items they are looking for within specific categories. Additionally, having a clear categorization of products supports better inventory management, enabling stores to identify which categories are performing well and which may need additional stock or promotional efforts.

4. Decomposition to 3NF

- **Functional Dependencies:**
  i. SupplierId → Name, Address, Contact
  ii. CategoryId → Name, LeadTime, StorageRequirements
  iii. InventoryId → ProductId, StockDate, UnitPrice, ManufactureDate, ExpiryDate, Quantity
  iv. PromotionalOfferId → ProductId, StartDate, EndDate, DiscountRate
  v. OrderId → OrderDate, TotalPrice, PromotionalOfferId
  vi. ForecastId → ProductId, ForecastDate, ForecastQuantity
  vii. ProductId → Name, PackagingType, Weight, CategoryId, SupplierId
  viii. ProductId → SupplierId, Supplier.Name, Supplier.Contact, Supplier.Address,
  ix. ProductId → CategoryId, Category.PackagingType, Category.LeadTime, Category.Name, Category.StorageRequirements

- *Table 1: Forecast*

  - 1NF: All the attributes of Forecast table are atomic.
  - 2NF: All attributes are dependent on the primary key (from FD vi.)
  - 3NF: There are no transitive dependencies, hence the relation is in 3NF.

- *Table 2: Order*
  - 1NF: All attributes of Order table are atomic.
  - 2NF: All attributes are dependent on the primary key (from FD v.)
  - 3NF: There are no transitive dependencies, hence the relation is in 3NF.

- *Table 3: Inventory*
  - 1NF: All attributes of Inventory table are atomic.
  - 2NF: All attributes are dependent on the primary key (from FD iii.)
  - 3NF: There are no transitive dependencies, hence the relation is in 3NF.

- *Table 4: Promotions*
  - 1NF: All attributes of Promotions table are atomic.
  - 2NF: All attributes are dependent on the primary key (from FD iv.)
  - 3NF: There are no transitive dependencies, hence the relation is in 3NF.

- **3NF:**

  Previously, we had Product table which contained Category, Supplier attributes as ProductId determines everything.

  - The relation of table Product was as follows-

  Product (ProductId, Name, Weight, CategoryId, CategoryName, LeadTime, PackagingType, StorageRequirement, SupplierId, SupplierName, Contact, Address)

  - But we followed the procedure of decomposition to 3NF.

  ProductId → Name, PackagingType, Weight

  ProductId → SupplierId, Supplier.Name, Supplier.Contact, Supplier.Address

  ProductId → CategoryId, Category.PackagingType, Category.LeadTime, Category.Name, Category.StorageRequirements

  SupplierId → Supplier.Name, Supplier.Address, Supplier.Contact

  CategoryId → Category. Name, Category. LeadTime, Category. StorageRequirements

- Minimal Basis

Primary Key:

Left	Middle	Right	None
ProductId	SupplierId, CategoryId	*everything else	-

(ProductId) + = {ProductId, SupplierId, Name, PackagingType, Weight, CategoryId, SupplierId, Supplier.Name, Supplier.Contact, Supplier.Address, CategoryId, Category.PackagingType, Category.LeadTime, Category.Name, Category.StorageRequirements}

Primary Key is (ProductId, CategoryId, SupplierId)

Removing redundant dependencies:

ProductId → Name, PackagingType, Weight

SupplierId → SupplierId, Supplier.Name, Supplier.Address, Supplier.Contact

CategoryId → CategoryId, Category. Name, Category. LeadTime, Category. StorageRequirements

Decomposing it into Relations:

Product(ProductId, Name, PackagingType, Weight)

Supplier(SupplierId, Name, Address, Contact)

Category(CategoryId, Name, LeadTime, StorageRequirements)

This component of relational schema is now in 3NF.

Other components of relational schema are already in 3NF.

5. Relational Schema

- **User** (UserId:INT [PK], EmailId:VARCHAR(255), Password:VARCHAR(255), Type:VARCHAR(10), FirstName:VARCHAR(50), LastName:VARCHAR(50), PhoneNumber:VARCHAR(20))
- **Supplier** (SupplierId:INT [PK], Name:VARCHAR(100), Address:VARCHAR(255), Contact:VARCHAR(100))
- **Category** (CategoryId:INT [PK], Name:VARCHAR(50), LeadTime:VARCHAR(50), StorageRequirements:VARCHAR(255))
- **Product** (ProductId:INT [PK], Name:VARCHAR(100), PackagingType:VARCHAR(50), Weight:FLOAT, CategoryId:INT [FK to Category.CategoryId], SupplierId:INT [FK to Supplier.SupplierId])

- **Inventory** (InventoryId:INT [PK], StockDate:DATE, ProductId:INT [FK to Product.ProductId], UnitPrice:FLOAT, ManufactureDate:DATE, ExpiryDate:DATE, Quantity:INT)

- **PromotionalOffer** (PromotionalOfferId:INT [PK], ProductId:INT [FK to Product.ProductId], StartDate:DATE, EndDate:DATE, DiscountRate:FLOAT)

- **Order** (OrderId:INT [PK],OrderDate:DATE, PromotionalOfferId:INT [FK to PromotionalOffer.PromotionalOfferId], TotalPrice:FLOAT)

- **Order_Has_Inventories** (OrderId:INT [PK, FK to Order.OrderId], InventoryId:INT [PK, FK to Inventory.InventoryId])

- **Order_Has_PromotionalOffers** (OrderId:INT [PK, FK to Order.OrderId], PromotionalOfferId:INT [PK, FK to PromotionalOffer.PromotionalOfferId])

- **Forecast** (ForecastId:INT [PK], ProductId:INT [FK to Product.ProductId], ForecastDate:DATE, ForecastQuantity:INT)