IS 4420 Final Project

By: Cygen Stanley, Junruiyi Xie, Bao Nguyen, Ahmed Alhaimi

1. Project Overview

General description or introduction of the company where the group project is being conducted, which includes: business goals of the company and products/services provided by the company etc.

We are an online shopping catalog with delivery options. We will provide a platform that allows customers to see what products we have online and add them to their shopping carts to order. This system will allow us to keep track of inventory and see what products have the highest turnover. We can track our users' experiences and behaviors to improve our system further. We offer home delivery and a special discount when you shop if you are a rewards member. Our system will allow us to analyze our users' buying preferences and work to improve their experience..

2. User Requirements

Through observation, interviews, and/or discussion develop at least **20 user requirements** for your database**.** User requirements must be specific, and may take the form of user stories Links to an external site.

Customer Account Management

1. As a customer, I want to be able to create an account that tracks my orders.
2. As a customer, I want to access my account and update my information to receive accurate orders.
3. As a customer, I want to add one or multiple delivery addresses in the customer profile so that I can easily choose where I want my order delivered.

Shopping Experience

4. As a user, I want to see what products I can buy and add them to my cart.
5. As a customer, I want to see how much each product costs.
6. As a customer, I want to wishlist multiple items.
7. As a customer, I want to add my favorite items to my collections.
8. As a customer, I want to apply my rewards member discount (coupon) to the order.

Order Processing

9. As a customer, I want to add delivery instructions (notes) so that drivers can successfully have my order delivered.
10. As a customer, I want to receive a confirmation email after placing an order.
11. As a customer, I want to receive a confirmation SMS after placing an order.

12. As a customer, I want to see when my order arrives.

Post-Purchase Activities

13. As a customer, I want to see my past orders.
14. As a customer, I want to leave a review on products I have purchased in order that I can share my experience with other customers.

Inventory Management

15. As inventory manager, I want to see what product is low.
16. As inventory manager, I want to see how much is left of each product.
17. As Inventory manager, I want to categorize products (produce, dairy, bakery, meat, etc.) so that customers can browse.
18. As Inventory manager, I want to track which supplier provides which product so that I can manage vendor relationships.
19. As inventory manager, I want to update the database about when new shipments arrive so that inventory counts are up to date.

Delivery Management

20. As delivery driver, I want to mark orders as delivered so that the database is updated and customers can know.

Administration

21. As User admin, I want to see what product is sold the most.
22. As a marketing director, I want to know which products are popular so that I can set promotional prices.
23. As a customer service specialist, I want to look up customer's past orders by order_id to precisely assist them.

. Consider who, what, where, when, why, and how. For example:

- What will the system accomplish?
- What functions will it perform?
- Who will need access to the data?
- What data will be tracked?
- What goals will it support?
- What questions will the system be able to answer?

3. Business Rules

Develop business rules as described in Chapter 2 of the Hoffer text to drive the design of your Conceptual Model**.** The quantity of business rules will depend on the design of your database.

Customers Management

1. Each customer must have a unique Customer_ID (Primary Key)
2. Each customer profile must contain Customer_ID, First_name, Last_name, email, address, payment information, and phone number (NOT NULL)
3. A customer can have multiple saved delivery addresses
4. Each customer email address must be unique
5. Only registered customer can place an order

Employee Management

6. Each employee must have a unique Employee_ID (NOT NULL) (Primary Key)
7. An employee must be assigned to a specific role (inventory manager, delivery driver, customer service specialist etc.) (NOT NULL)
8. Each delivery driver can have multiple orders per day

Product Management

9. Each product must have a unique product _ID (Primary Key)
10. Each product must belong to at least one category (Mandatory many)
11. Product price must be greater than zero
12. Products must have descriptions and images

Inventory Management

13. A supplier provides one or more products (Mandatory many)
14. Each supplier must have a unique supplier_ID (Primary Key)
15. A supplier can have supplier_name as attribute
16. Inventory quantities cannot be less than zero
17. Inventory quantities must be updated when new supplies from suppliers arrive

Order Processing

18. A customer can place more than one order (Optional many)
19. Each order must have at least one product to be placed (Mandatory many)
20. Order status must be updated at each stage (placed, processing, out for delivery, delivered)

Payment Processing

21. A payment must have a payment_ID

22. A payment can have one payment method (credit card, PayPal, Venmo) as attribute

Delivery Management

23. Delivery time slots must be available for customers to select from
24. Delivery status must be updated in real-time
25. Delivery instructions must be accessible to delivery drivers

Rewards Program

26. A reward is associated with exactly one customer
27. Rewards points are earned based on order total
28. Reward points can only be used once in a order to be placed

Customer Feedback

29. A review must be associated with a specific product and customer
30. A customer can place only one review on a single item

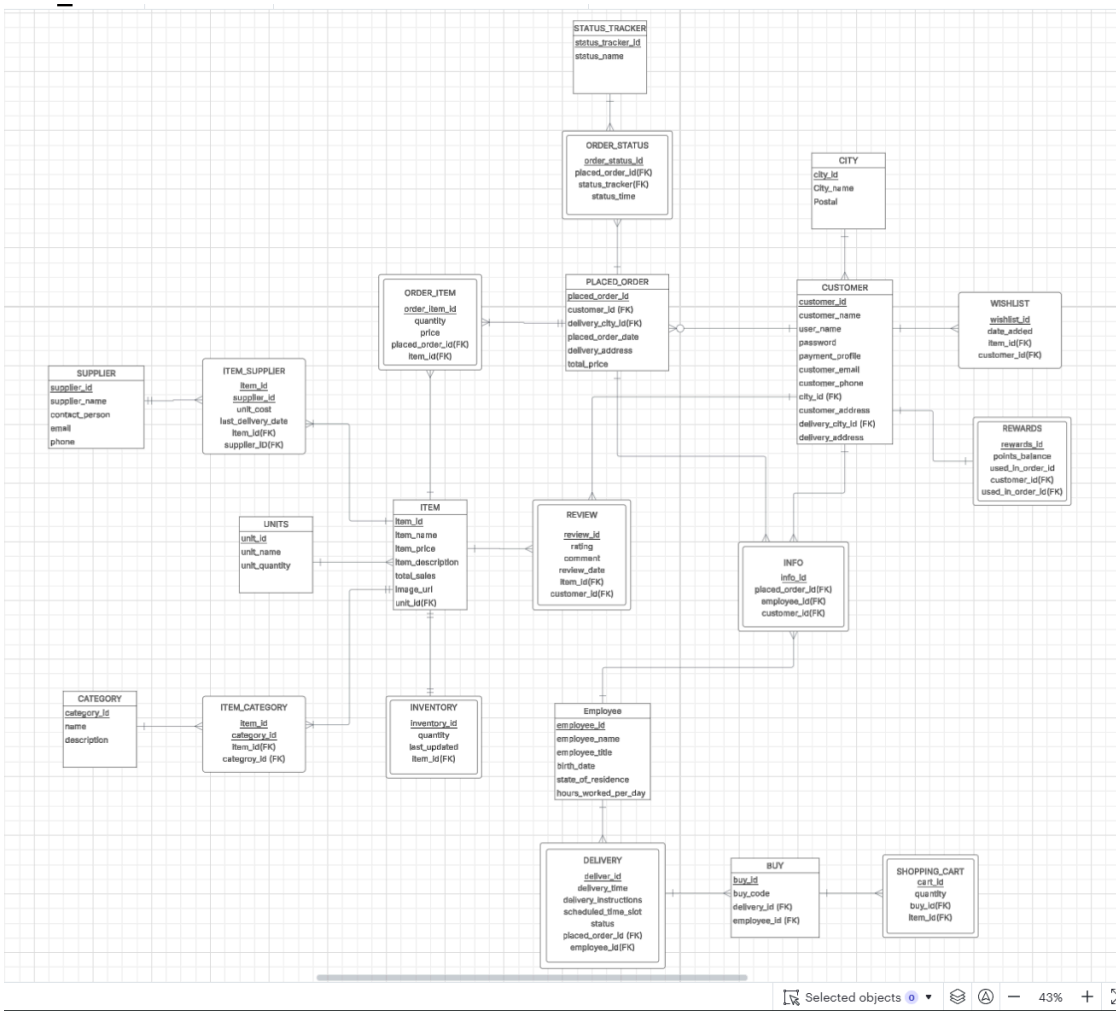
4. Data Outputs / Business Questions

Create a list of at least 10 outputs for various stakeholders who may want to retrieve data from the database**.**

1. See the total sales for the day
2. See what products sell the most in descending order and list the supplier name for each product.
3. See inventory levels of all products and include the category name for each product.
4. Which product has the lowest level of inventory in terms of quantity Which product has the lowest level of inventory in terms of quantity? Also, show the name of the supplier and the last delivery date from that supplier for that product.
5. Keep track of how many employees work more than 8 hours a day and come from CA.
6. See how many orders were placed each day and show the customer names who placed those orders.
7. Which product has the highest sales?
8. See the list of employees who are born after 2000 and come from Utah
9. See the list of employees who are older than 30 years old and have the last name Sam.
10. See the list of products that have less than 100 quantities. And, show the names of the suppliers and the categories those products belong to.
11. Show all product reviews and include the customer name who wrote the review and the product category of the reviewed item.

5. Conceptual Model

Develop an ERD using Lucid Chart based on User Requirements created in section 2 and Business Rules created in section 3**.** The number of entity types will vary based on User Requirements / Business Rules established**. A project will not be accepted with fewer than 10 entity types (do not shoot for the minimum).**



https://lucid.app/lucidchart/6d7b3ddb-5618-4ba7-9965-b4df550f627d/edit?viewport_loc=-2128%2C-73%2C5269%2C4125%2C0_0&invitationId=inv_34418879-7fb7-40df-bb2f-be556e368063

UPDATED:

Detailed ERD Design for Grocery Store Database

Entities and Attributes

1. CUSTOMER (Strong Entity)

- **Primary Key:** customer_id (INT)
- **Attributes:**
  - customer_name (VARCHAR, NOT NULL)
  - user_name (VARCHAR, NOT NULL, UNIQUE)
  - password (VARCHAR, NOT NULL)
  - payment_profile (VARCHAR, NOT NULL)
  - customer_email (VARCHAR, NOT NULL, UNIQUE)
  - customer_phone (VARCHAR, NOT NULL)
  - customer_address (VARCHAR, NOT NULL)
  - delivery_address (VARCHAR)
- **Foreign Keys:**
  - city_id (references CITY)
  - delivery_city_id (references CITY)
- **Entity Type:** Strong Entity

2. EMPLOYEE (Strong Entity)

- **Primary Key:** employee_id (INT)
- **Attributes:**
  - employee_name (VARCHAR, NOT NULL)
  - employee_title (VARCHAR, NOT NULL)
  - birth_date (DATE, NOT NULL)
  - state_of_residence (VARCHAR, NOT NULL)
  - hours_worked_per_day (DECIMAL, NOT NULL)
- **Entity Type:** Strong Entity

3. ITEM (Strong Entity)

- **Primary Key:** item_id (INT)
- **Attributes:**
  - item_name (VARCHAR, NOT NULL)
  - item_price (DECIMAL, NOT NULL, > 0)
  - item_description (TEXT)
  - total_sales (DECIMAL, DEFAULT 0)

    - image_url (VARCHAR, NOT NULL)
- **Foreign Keys:**
    - unit_id (references UNITS)
- **Entity Type:** Strong Entity

4. UNITS (Strong Entity)

- **Primary Key:** unit_id (INT)
- **Attributes:**
    - unit_name (VARCHAR, NOT NULL)
    - unit_quantity (DECIMAL)
- **Entity Type:** Strong Entity

5. CITY (Strong Entity)

- **Primary Key:** city_id (INT)
- **Attributes:**
    - city_name (VARCHAR, NOT NULL)
    - postal (VARCHAR, NOT NULL)
- **Entity Type:** Strong Entity

6. STATUS_TRACKER (Strong Entity)

- **Primary Key:** status_tracker_id (INT)
- **Attributes:**
    - status_name (VARCHAR, NOT NULL)
- **Entity Type:** Strong Entity

7. PLACED_ORDER (Strong Entity)

- **Primary Key:** placed_order_id (INT)
- **Attributes:**
    - placed_order_date (DATETIME, NOT NULL)
    - delivery_address (VARCHAR, NOT NULL)
    - total_price (DECIMAL, NOT NULL)
- **Foreign Keys:**
    - customer_id (references CUSTOMER)
    - delivery_city_id (references CITY)
- **Entity Type:** Strong Entity

8. ORDER_STATUS (Weak Entity)

- **Primary Key:** order_status_id (INT)
- **Attributes:**
    - status_time (DATETIME, NOT NULL)
- **Foreign Keys:**
    - placed_order_id (references PLACED_ORDER)

    - status_tracker_id (references STATUS_TRACKER)
- **Entity Type:** Weak Entity (existence dependent on PLACED_ORDER)

9. ORDER_ITEM (Weak Entity)

- **Primary Key:** order_item_id (INT)
- **Attributes:**
    - quantity (DECIMAL, NOT NULL, > 0)
    - price (DECIMAL, NOT NULL, > 0)
- **Foreign Keys:**
    - placed_order_id (references PLACED_ORDER)
    - item_id (references ITEM)
- **Entity Type:** Weak Entity (existence dependent on PLACED_ORDER)

10. DELIVERY (Weak Entity)

- **Primary Key:** delivery_id (INT)
- **Attributes:**
    - delivery_time (DATETIME)
    - delivery_instructions (VARCHAR)
    - scheduled_time_slot (DATETIME, NOT NULL)
    - status (VARCHAR, NOT NULL)
- **Foreign Keys:**
    - placed_order_id (references PLACED_ORDER)
    - employee_id (references EMPLOYEE)
- **Entity Type:** Weak Entity (existence dependent on PLACED_ORDER)

11. INFO (Weak Entity)

- **Primary Key:** info_id (INT)
- **Foreign Keys:**
    - placed_order_id (references PLACED_ORDER)
    - employee_id (references EMPLOYEE)
    - customer_id (references CUSTOMER)
- **Entity Type:** Weak Entity (associative entity connecting order, employee, and customer)

12. SHOPPING_CART (Weak Entity)

- **Primary Key:** cart_id (INT)
- **Attributes:**
    - quantity (DECIMAL, NOT NULL, > 0)
- **Foreign Keys:**
    - buy_id (references BUY)
    - item_id (references ITEM)
- **Entity Type:** Weak Entity (existence dependent on BUY)

13. BUY (Strong Entity)

- **Primary Key:** buy_id (INT)
- **Attributes:**
  - buy_code (VARCHAR, NOT NULL)
- **Foreign Keys:**
  - delivery_id (references DELIVERY)
  - employee_id (references EMPLOYEE)
- **Entity Type:** Strong Entity

14. WISHLIST (Associative Entity)

- **Primary Key:** wishlist_id (INT)
- **Attributes:**
  - date_added (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- **Foreign Keys:**
  - item_id (references ITEM)
  - customer_id (references CUSTOMER)
- **Entity Type:** Associative Entity (connects CUSTOMER and ITEM)

15. CATEGORY (Strong Entity)

- **Primary Key:** category_id (INT)
- **Attributes:**
  - name (VARCHAR, NOT NULL)
  - description (TEXT)
- **Entity Type:** Strong Entity

16. ITEM_CATEGORY (Associative Entity)

- **Primary Key:** Composite (item_id, category_id)
- **Foreign Keys:**
  - item_id (references ITEM)
  - category_id (references CATEGORY)
- **Entity Type:** Associative Entity (connects ITEM and CATEGORY)

17. SUPPLIER (Strong Entity)

- **Primary Key:** supplier_id (INT)
- **Attributes:**
  - supplier_name (VARCHAR, NOT NULL)
  - contact_person (VARCHAR, NOT NULL)
  - email (VARCHAR, NOT NULL)
  - phone (VARCHAR, NOT NULL)
- **Entity Type:** Strong Entity

18. ITEM_SUPPLIER (Associative Entity)

- **Primary Key:** Composite (item_id, supplier_id)
- **Attributes:**
  - unit_cost (DECIMAL, NOT NULL)
  - last_delivery_date (DATE)
- **Foreign Keys:**
  - item_id (references ITEM)
  - supplier_id (references SUPPLIER)
- **Entity Type:** Associative Entity (connects ITEM and SUPPLIER)

19. INVENTORY (Weak Entity)

- **Primary Key:** inventory_id (INT)
- **Attributes:**
  - quantity (DECIMAL, NOT NULL, ≥ 0)
  - last_updated (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- **Foreign Keys:**
  - item_id (references ITEM)
- **Entity Type:** Weak Entity (existence dependent on ITEM)

20. REWARDS (Weak Entity)

- **Primary Key:** rewards_id (INT)
- **Attributes:**
  - points_balance (INT, DEFAULT 0)
  - used_in_order_id (INT)
- **Foreign Keys:**
  - customer_id (references CUSTOMER)
  - used_in_order_id (references PLACED_ORDER)
- **Entity Type:** Weak Entity (existence dependent on CUSTOMER)

21. REVIEW (Weak Entity)

- **Primary Key:** review_id (INT)
- **Attributes:**
  - rating (INT, NOT NULL)
  - comment (TEXT)
  - review_date (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- **Foreign Keys:**
  - customer_id (references CUSTOMER)
  - item_id (references ITEM)
- **Entity Type:** Weak Entity (existence dependent on both CUSTOMER and ITEM)

Relationships and Cardinality Constraints

1. **CUSTOMER to CITY** (Many-to-One)

    - Cardinality: Many CUSTOMERs can be in one CITY
    - Participation: Total participation (a CUSTOMER must be associated with a CITY)
2. **CUSTOMER to PLACED_ORDER** (One-to-Many, Optional)
    - Cardinality: One CUSTOMER can place many PLACED_ORDERs
    - Participation: Partial participation (a CUSTOMER may not have any PLACED_ORDER)
3. **ITEM to UNITS** (Many-to-One)
    - Cardinality: Many ITEMs can use one UNIT type
    - Participation: Total participation (an ITEM must have a UNIT)
4. **PLACED_ORDER to ORDER_ITEM** (One-to-Many, Mandatory)
    - Cardinality: One PLACED_ORDER has many ORDER_ITEMs
    - Participation: Total participation (a PLACED_ORDER must have at least one ORDER_ITEM)
    - Constraint: Minimum one ORDER_ITEM per PLACED_ORDER
5. **PLACED_ORDER to ORDER_STATUS** (One-to-Many)
    - Cardinality: One PLACED_ORDER can have many ORDER_STATUS updates
    - Participation: Total participation (a PLACED_ORDER must have at least one ORDER_STATUS)
6. **ORDER_STATUS to STATUS_TRACKER** (Many-to-One)
    - Cardinality: Many ORDER_STATUS records can reference one STATUS_TRACKER
    - Participation: Total participation (an ORDER_STATUS must have a STATUS_TRACKER)
7. **ITEM to ORDER_ITEM** (One-to-Many)
    - Cardinality: One ITEM can appear in many ORDER_ITEMs
    - Participation: Partial participation (an ITEM may not be in any ORDER_ITEM)
8. **PLACED_ORDER to DELIVERY** (One-to-One)
    - Cardinality: One PLACED_ORDER has exactly one DELIVERY
    - Participation: Total participation (a PLACED_ORDER must have a DELIVERY)
9. **EMPLOYEE to DELIVERY** (One-to-Many)
    - Cardinality: One EMPLOYEE can handle many DELIVERYs
    - Participation: Partial participation (an EMPLOYEE may not be assigned to any DELIVERY)

10. **CUSTOMER to WISHLIST** (One-to-Many)
    - Cardinality: One CUSTOMER can have many WISHLIST entries
    - Participation: Partial participation (a CUSTOMER may not have any WISHLIST entries)
11. **ITEM to ITEM_CATEGORY** (One-to-Many, Mandatory)
    - Cardinality: One ITEM can be in many ITEM_CATEGORY entries
    - Participation: Total participation (an ITEM must belong to at least one CATEGORY)
    - Constraint: Minimum one CATEGORY per ITEM
12. **CATEGORY to ITEM_CATEGORY** (One-to-Many)
    - Cardinality: One CATEGORY can have many ITEM_CATEGORY entries
    - Participation: Partial participation (a CATEGORY may not have any ITEMs)
13. **SUPPLIER to ITEM_SUPPLIER** (One-to-Many, Mandatory)
    - Cardinality: One SUPPLIER can be in many ITEM_SUPPLIER entries
    - Participation: Total participation (a SUPPLIER must provide at least one ITEM)
    - Constraint: Minimum one ITEM per SUPPLIER
14. **ITEM to ITEM_SUPPLIER** (One-to-Many)
    - Cardinality: One ITEM can have many ITEM_SUPPLIER entries
    - Participation: Total participation (an ITEM must have at least one SUPPLIER)
15. **ITEM to INVENTORY** (One-to-One)
    - Cardinality: One ITEM has exactly one INVENTORY record
    - Participation: Total participation (an ITEM must have an INVENTORY record)
16. **CUSTOMER to REWARDS** (One-to-One)
    - Cardinality: One CUSTOMER has exactly one REWARDS record
    - Participation: Total participation (a CUSTOMER must have a REWARDS record)
17. **DELIVERY to BUY** (One-to-Many)
    - Cardinality: One DELIVERY can have many BUY records
    - Participation: Partial participation (a DELIVERY may not have any BUY records)
18. **BUY to SHOPPING_CART** (One-to-Many)
    - Cardinality: One BUY can have many SHOPPING_CART items

    - Participation: Total participation (a BUY must have at least one SHOPPING_CART item)
19. **ITEM to REVIEW** (One-to-Many)
    - Cardinality: One ITEM can have many REVIEWs
    - Participation: Partial participation (an ITEM may not have any REVIEWs)
20. **CUSTOMER to REVIEW** (One-to-Many, Constrained)
    - Cardinality: One CUSTOMER can have many REVIEWs
    - Participation: Partial participation (a CUSTOMER may not have any REVIEWs)
    - Constraint: A CUSTOMER can only write one REVIEW per ITEM

Simplified but Comprehensive ERD Design

Entity Types and Relationships

1. CUSTOMER (Strong Entity)

- **Primary Key:** customer_id (INT, auto-increment)
- **Attributes:**
    - customer_name (VARCHAR(100), NOT NULL)
    - user_name (VARCHAR(50), NOT NULL, UNIQUE)
    - password (VARCHAR(255), NOT NULL)
    - payment_profile (VARCHAR(100), NOT NULL)
    - customer_email (VARCHAR(100), NOT NULL, UNIQUE)
    - customer_phone (VARCHAR(20), NOT NULL)
    - customer_address (VARCHAR(200), NOT NULL)
- **Foreign Key:** city_id (references CITY)
- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to CITY:** Many-to-One (Many customers can be in one city)
- **Cardinality to PLACED_ORDER:** One-to-Many (A customer can place multiple orders)
- **Participation in PLACED_ORDER:** Optional (A customer may not have placed any orders)

2. CITY (Strong Entity)

- **Primary Key:** city_id (INT, auto-increment)
- **Attributes:**
    - city_name (VARCHAR(50), NOT NULL)
    - postal (VARCHAR(10), NOT NULL)

- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to CUSTOMER:** One-to-Many (A city can have many customers)

3. ITEM (Strong Entity)

- **Primary Key:** item_id (INT, auto-increment)
- **Attributes:**
  - item_name (VARCHAR(100), NOT NULL)
  - item_price (DECIMAL(10,2), NOT NULL, CHECK (item_price > 0))
  - item_description (TEXT)
  - image_url (VARCHAR(255), NOT NULL)
- **Foreign Key:** unit_id (references UNITS)
- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to UNITS:** Many-to-One (Many items use one unit type)
- **Cardinality to ORDER_ITEM:** One-to-Many (An item can be in multiple order items)
- **Cardinality to INVENTORY:** One-to-One (An item has exactly one inventory record)
- **Cardinality to ITEM_CATEGORY:** One-to-Many (An item can belong to multiple categories)
- **Participation in ITEM_CATEGORY:** Mandatory (An item must belong to at least one category)
- **Cardinality to ITEM_SUPPLIER:** One-to-Many (An item can be supplied by multiple suppliers)
- **Participation in ITEM_SUPPLIER:** Mandatory (An item must have at least one supplier)

4. UNITS (Strong Entity)

- **Primary Key:** unit_id (INT, auto-increment)
- **Attributes:**
  - unit_name (VARCHAR(20), NOT NULL)
  - unit_quantity (DECIMAL(10,2))
- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to ITEM:** One-to-Many (A unit type can be used for many items)

5. CATEGORY (Strong Entity)

- **Primary Key:** category_id (INT, auto-increment)
- **Attributes:**
  - name (VARCHAR(50), NOT NULL)
  - description (TEXT)
- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to ITEM_CATEGORY:** One-to-Many (A category can contain many items)

6. INVENTORY (Weak Entity)

- **Primary Key:** inventory_id (INT, auto-increment)
- **Foreign Key:** item_id (references ITEM)
- **Attributes:**
    - quantity (DECIMAL(10,2), NOT NULL, CHECK (quantity >= 0))
    - last_updated (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- **Entity Type:** Weak Entity (existence depends on ITEM)
- **Cardinality to ITEM:** One-to-One (An inventory record belongs to exactly one item)
- **Participation:** Total (Every item must have an inventory record)

7. PLACED_ORDER (Strong Entity)

- **Primary Key:** placed_order_id (INT, auto-increment)
- **Foreign Key:** customer_id (references CUSTOMER)
- **Attributes:**
    - placed_order_date (DATETIME, NOT NULL)
    - delivery_address (VARCHAR(200), NOT NULL)
    - total_price (DECIMAL(10,2), NOT NULL)
    - status (VARCHAR(20), NOT NULL) /* e.g., placed, processing, delivered */
- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to CUSTOMER:** Many-to-One (Many orders can be placed by one customer)
- **Cardinality to ORDER_ITEM:** One-to-Many (An order contains multiple items)
- **Participation in ORDER_ITEM:** Mandatory (An order must contain at least one item)

8. ORDER_ITEM (Weak Entity)

- **Primary Key:** order_item_id (INT, auto-increment)
- **Foreign Keys:**
    - placed_order_id (references PLACED_ORDER)
    - item_id (references ITEM)
- **Attributes:**
    - quantity (DECIMAL(10,2), NOT NULL, CHECK (quantity > 0))
    - price (DECIMAL(10,2), NOT NULL)
- **Entity Type:** Weak Entity (existence depends on PLACED_ORDER and ITEM)
- **Cardinality to PLACED_ORDER:** Many-to-One (Many order items belong to one order)
- **Cardinality to ITEM:** Many-to-One (Many order items can reference one item)

9. EMPLOYEE (Strong Entity)

- **Primary Key:** employee_id (INT, auto-increment)
- **Attributes:**

    - employee_name (VARCHAR(100), NOT NULL)
    - employee_title (VARCHAR(50), NOT NULL)
    - birth_date (DATE, NOT NULL)
    - state_of_residence (VARCHAR(2), NOT NULL)
    - hours_worked_per_day (DECIMAL(4,2), NOT NULL)
- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to Delivery:** One-to-Many (An employee can handle multiple deliveries)

10. SUPPLIER (Strong Entity)

- **Primary Key:** supplier_id (INT, auto-increment)
- **Attributes:**
    - supplier_name (VARCHAR(100), NOT NULL)
    - contact_person (VARCHAR(100), NOT NULL)
    - email (VARCHAR(100), NOT NULL)
    - phone (VARCHAR(20), NOT NULL)
- **Entity Type:** Strong Entity (exists independently)
- **Cardinality to ITEM_SUPPLIER:** One-to-Many (A supplier can provide many items)
- **Participation in ITEM_SUPPLIER:** Mandatory (A supplier must provide at least one item)

11. ITEM_CATEGORY (Associative Entity)

- **Primary Key:** Composite (item_id, category_id)
- **Foreign Keys:**
    - item_id (references ITEM)
    - category_id (references CATEGORY)
- **Entity Type:** Associative Entity (represents many-to-many relationship)
- **Cardinality:** Many-to-Many (Items can belong to multiple categories and categories can contain multiple items)

12. ITEM_SUPPLIER (Associative Entity)

- **Primary Key:** Composite (item_id, supplier_id)
- **Foreign Keys:**
    - item_id (references ITEM)
    - supplier_id (references SUPPLIER)
- **Attributes:**
    - last_delivery_date (DATE)
    - unit_cost (DECIMAL(10,2), NOT NULL)
- **Entity Type:** Associative Entity (represents many-to-many relationship)
- **Cardinality:** Many-to-Many (Items can be supplied by multiple suppliers and suppliers can provide multiple items)

Summary of Relationships

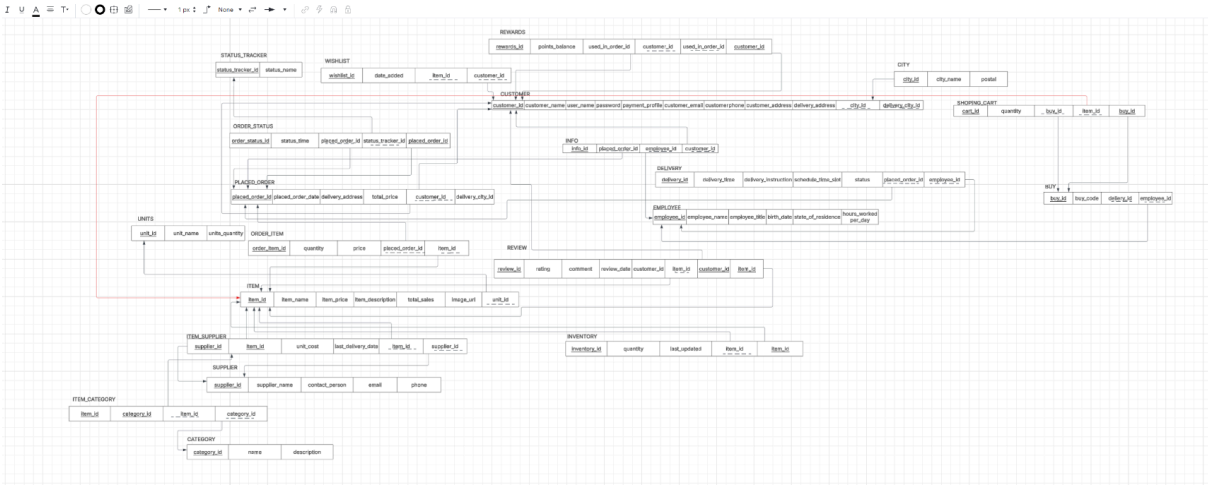
1. **CUSTOMER to CITY:** Many-to-One
    - A customer belongs to one city
    - A city can have many customers
2. **CUSTOMER to PLACED_ORDER:** One-to-Many (Optional)
    - A customer can place zero, one, or many orders
    - Each order belongs to exactly one customer
3. **ITEM to UNITS:** Many-to-One
    - An item is measured in one unit type
    - A unit type can be used for many items
4. **ITEM to INVENTORY:** One-to-One
    - Each item has exactly one inventory record
    - Each inventory record belongs to exactly one item
5. **PLACED_ORDER to ORDER_ITEM:** One-to-Many (Mandatory)
    - An order contains one or more order items
    - Each order item belongs to exactly one order
6. **ITEM to ORDER_ITEM:** One-to-Many
    - An item can appear in multiple order items
    - Each order item references exactly one item
7. **ITEM to ITEM_CATEGORY:** One-to-Many (Mandatory)
    - An item must belong to at least one category
    - Each item-category entry refers to exactly one item
8. **CATEGORY to ITEM_CATEGORY:** One-to-Many
    - A category can contain multiple items
    - Each item-category entry refers to exactly one category
9. **SUPPLIER to ITEM_SUPPLIER:** One-to-Many (Mandatory)
    - A supplier must provide at least one item
    - Each item-supplier entry refers to exactly one supplier
10. **ITEM to ITEM_SUPPLIER:** One-to-Many (Mandatory)
    - An item must be supplied by at least one supplier
    - Each item-supplier entry refers to exactly one item

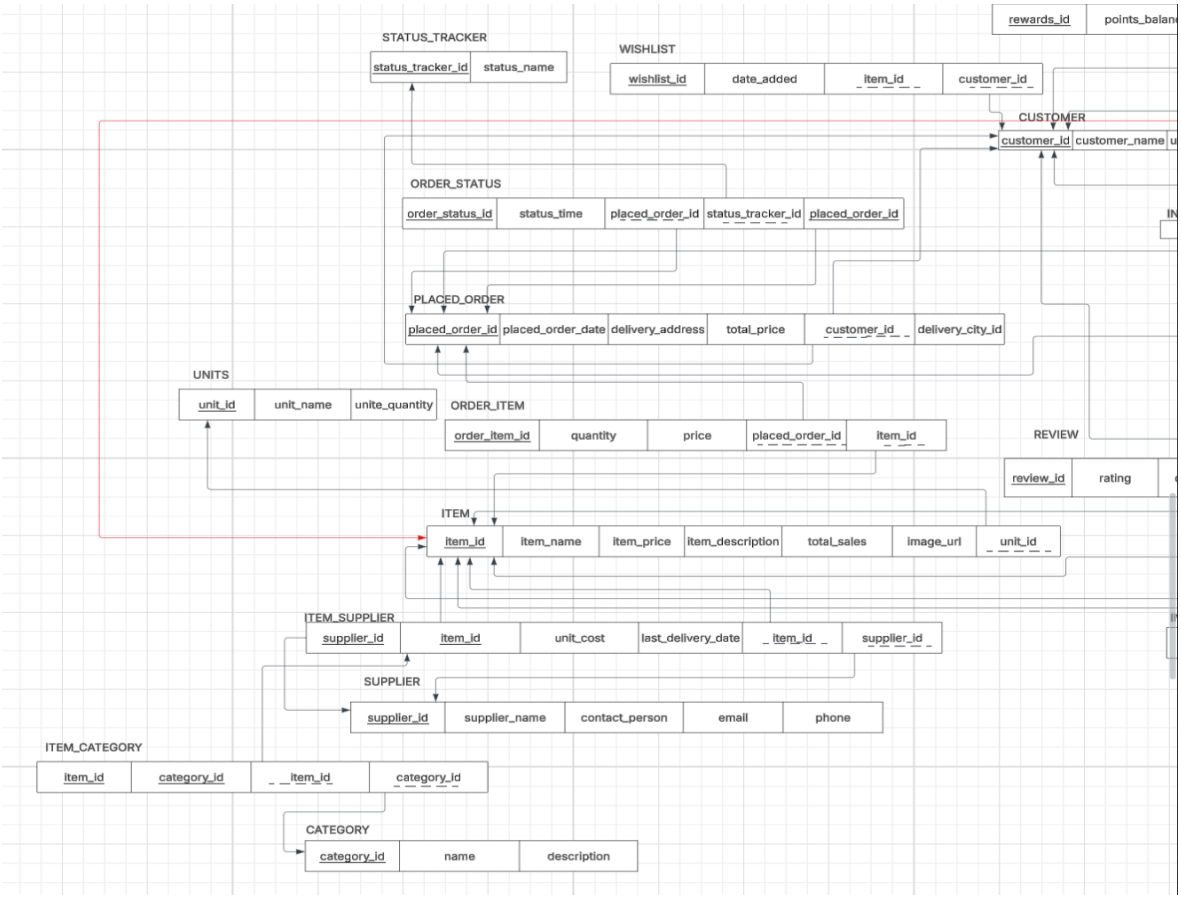
This design provides a streamlined yet comprehensive structure that satisfies your project requirements while maintaining simplicity. It includes 12 entities (exceeding the

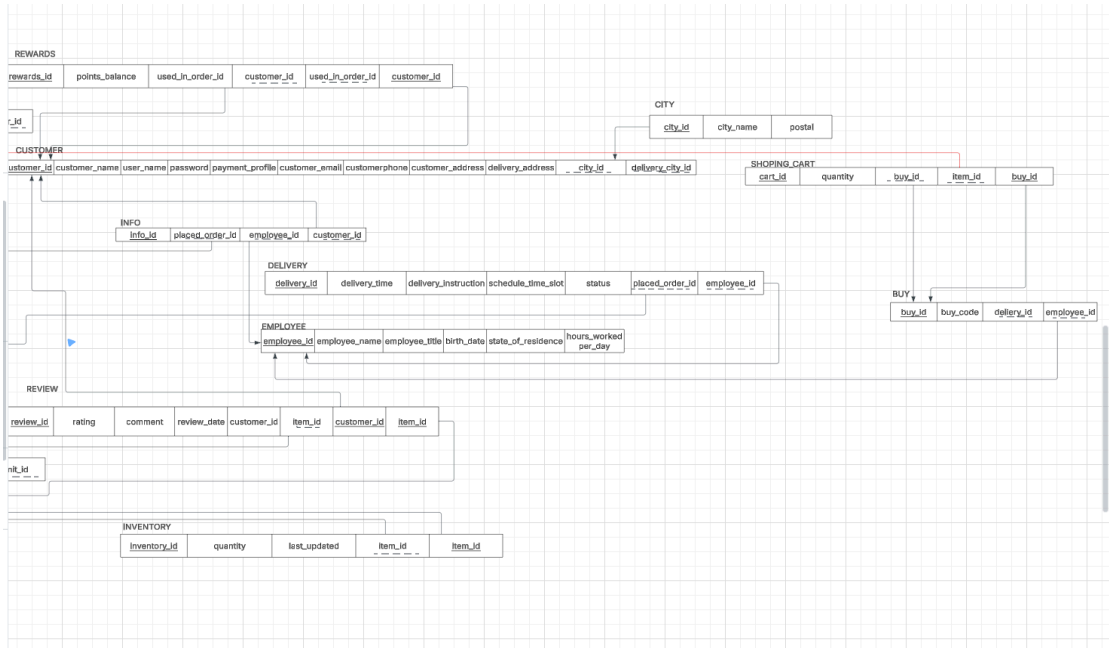
minimum 10 required), defines all key relationships with appropriate cardinality and participation constraints, and supports the business rules and questions you've outlined.

6. Logical Model

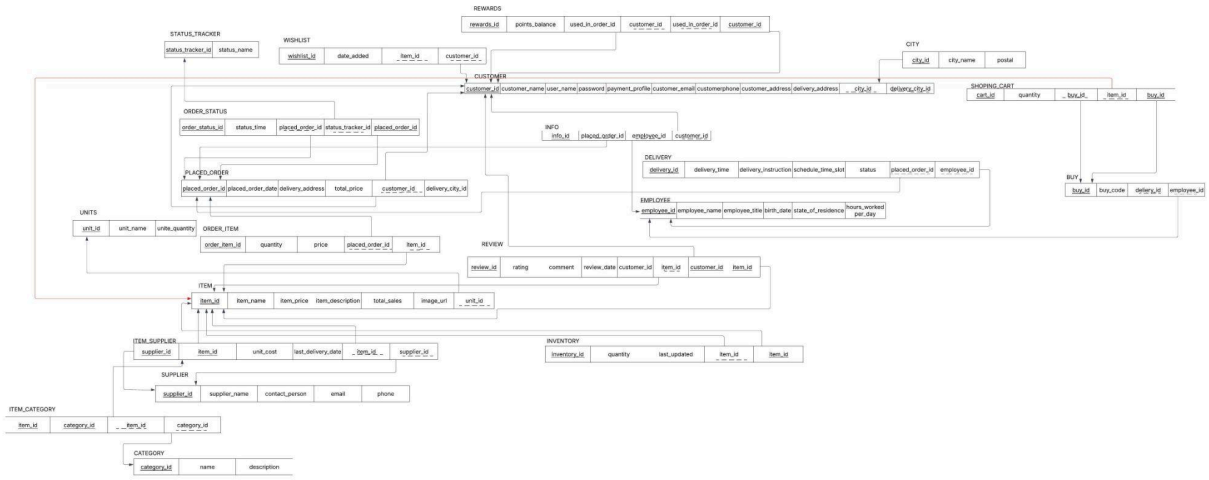
Convert the ERD created in section 5 into a relational model using Lucid Chart.







Full-scale Relational model(highest resolution, see if this is clear enough:



7. Database Implementation

Implement relations in the logical model developed in item D using DDL and DML in MySQL Server**.** Must show DDL and DML code used to create schema and load data (no SQL dump). Include sufficient data to create meaningful queries (recommend minimum of 40 rows of data in each transactional table, and 15 rows of data in all other tables).

```
create table city
(
  city_id int auto_increment primary key,
  city_name varchar(100) not null,
  postal varchar(20)
);

create table status_tracker
(
  status_tracker_id int auto_increment primary key,
  status_name varchar(50) not null
);

create table supplier
(
  supplier_id int auto_increment primary key,
  supplier_name varchar(100) not null,
  contact_person varchar(100),
  email varchar(255),
  phone varchar(20)
);

create table units
(
  unit_id int auto_increment primary key,
  unit_name varchar(50) not null,
  unit_quantity int not null
);

create table category
(
  category_id int auto_increment primary key,
  name varchar(100) not null,
  description text
);

create table customer
(
  customer_id int auto_increment primary key,
  customer_name varchar(100) not null,
```

```
  user_name varchar(50) not null,
  password varchar(255) not null,
  payment_profile varchar(255) not null,
  customer_email varchar(255) not null unique,
  customer_phone varchar(20)  not null,
  customer_address varchar(255) not null,
  delivery_address varchar(255),
  city_id int not null,
  delivery_city_id int,
  foreign key (city_id) references city(city_id),
  foreign key (delivery_city_id) references city(city_id)
);

create table employee
(
  employee_id int auto_increment primary key,
  employee_name varchar(100) not null,
  employee_title varchar(100) not null,
  birth_date date not null,
  state_of_residence varchar(100) not null,
  hours_worked_per_day decimal(4,2) not null
);

create table item
(
  item_id int auto_increment primary key,
  item_name varchar(100) not null,
  item_price decimal(10,2) not null check (item_price > 0),
  item_description text,
  total_sales int default 0,
  image_url varchar(255),
  unit_id int,
  foreign key (unit_id) references units(unit_id)
);

create table item_supplier
(
  item_id int not null,
  supplier_id int not null,
  unit_cost decimal(10,2) not null,
```

```
  last_delivery_date date,
  primary key (item_id, supplier_id),
  foreign key (item_id) references item(item_id),
  foreign key (supplier_id) references supplier(supplier_id)
);

create table item_category
(
  item_id int not null,
  category_id int not null,
  primary key (item_id, category_id),
  foreign key (item_id) references item(item_id),
  foreign key (category_id) references category(category_id)
);

create table inventory
(
  inventory_id int auto_increment primary key,
  quantity int not null,
  last_updated timestamp default current_timestamp on update current_timestamp,
  item_id int not null,
  foreign key (item_id) references item(item_id)
);


create table placed_order
(
  placed_order_id int auto_increment primary key,
  customer_id int not null,
  delivery_city_id int not null,
  placed_order_date date not null,
  delivery_address varchar(255),
  total_price decimal(10,2) not null,
  foreign key (customer_id) references customer(customer_id),
  foreign key (delivery_city_id) references city(city_id)
);

create table order_status
(
  order_status_id int auto_increment primary key,
```

```
  placed_order_id int not null,
  status_tracker_id int not null,
  status_time timestamp not null,
  foreign key (placed_order_id) references placed_order(placed_order_id),
  foreign key (status_tracker_id) references status_tracker(status_tracker_id)
);

create table order_item (
  order_item_id int auto_increment primary key,
  quantity int not null,
  price decimal(10,2) not null,
  placed_order_id int not null,
  item_id int not null,
  foreign key (placed_order_id) references placed_order(placed_order_id),
  foreign key (item_id) references item(item_id)
);

create table review
(
  review_id int auto_increment primary key,
  rating int not null,
  comment text,
  review_date date not null,
  item_id int not null,
  customer_id int not null,
  foreign key (item_id) references item(item_id),
  foreign key (customer_id) references customer(customer_id)
);

create table wishlist
(
  wishlist_id int auto_increment primary key,
  date_added date not null,
  item_id int not null,
  customer_id int not null,
  foreign key (item_id) references item(item_id),
  foreign key (customer_id) references customer(customer_id)
);

create table rewards
```

```
(
  rewards_id int auto_increment primary key,
  points_balance int not null,
  used_in_order_id int,
  customer_id int not null,
  foreign key (used_in_order_id) references placed_order(placed_order_id),
  foreign key (customer_id) references customer(customer_id)
);

create table delivery
(
  delivery_id int auto_increment primary key,
  delivery_time timestamp,
  delivery_instructions text,
  scheduled_time_slot varchar(50),
  status varchar(50),
  placed_order_id int not null,
  employee_id int not null,
  foreign key (placed_order_id) references placed_order(placed_order_id),
  foreign key (employee_id) references employee(employee_id)
);

create table buy
(
  buy_id int auto_increment primary key,
  buy_code varchar(50) not null,
  delivery_id int not null,
  employee_id int not null,
  foreign key (delivery_id) references delivery(delivery_id),
  foreign key (employee_id) references employee(employee_id)
);

create table shopping_cart
(
  cart_id int auto_increment primary key,
  quantity int not null,
  buy_id int not null,
  item_id int not null,
  foreign key (buy_id) references buy(buy_id),
  foreign key (item_id) references item(item_id)
```

```
);

create table info
(
  info_id int auto_increment primary key,
  placed_order_id int not null,
  employee_id int not null,
  customer_id int not null,
  foreign key (placed_order_id) references placed_order(placed_order_id),
  foreign key (employee_id) references employee(employee_id),
  foreign key (customer_id) references customer(customer_id)
);
```

8. Business Questions

In this section, we present five SQL queries addressing vital business questions for our grocery store database, delivering insights for stakeholders like store managers and marketing directors. Our queries help with operations-related problems and boost customer satisfaction. Using our MySQL schema, we leverage one-to-many, many-to-many, and one-to-one relationships to create clear, impactful results. We target key areas such as daily sales, top products with suppliers, inventory by category, stock shortages, and customer feedback to support data-driven decisions.

Query 1: Calculate the Total Sales Revenue for the Current Day

Stakeholder: Store Manager

Insight: Tracks daily revenue to monitor business performance.

Relationship Type: One-to-many (***PLACED_ORDER*** to ***CUSTOMER***)

Explanation: Joins ***placed_order*** (many) to ***customer*** (one) to ensure valid orders, summing ***total_price*** for the current day.

```sql
SELECT DATE(p.placed_order_date) AS sale_date,
       SUM(p.total_price) AS total_sales
FROM placed_order p
JOIN customer c ON p.customer_id = c.customer_id
WHERE DATE(p.placed_order_date) = CURDATE()
GROUP BY sale_date;
```

Query 2: Identify the Top Five Products by Sales Volume, Including Supplier Names

Stakeholder: Marketing Director, Procurement Manager

Insight: Highlights high-demand products and suppliers for marketing and sourcing strategies.

Relationship Type: Many-to-many (***ITEM*** to ***SUPPLIER*** via ***ITEM_SUPPLIER***)

Explanation: Uses many-to-many relationship (item to supplier via item_supplier) and one-to-many (item to order_item) to rank products by quantity sold.

```sql
SELECT i.item_name,
       SUM(oi.quantity) AS total_sold,
       s.supplier_name
FROM item i
JOIN order_item oi ON i.item_id = oi.item_id
JOIN item_supplier isup ON i.item_id = isup.item_id
JOIN supplier s ON isup.supplier_id = s.supplier_id
GROUP BY i.item_name, s.supplier_name
ORDER BY total_sold DESC
LIMIT 5;
```

Query 3: List Current Inventory Levels for All Products, Including Category Names

Stakeholder: Inventory Manager

Insight: Identifies stock levels by category for restocking and category management.

Relationship Type: Many-to-many (***ITEM*** to ***CATEGORY*** via ***ITEM_CATEGORY***)

Explanation: Combines many-to-many (item to category via item_category) and one-to-one (item to inventory) to show stock levels sorted by quantity.

```sql
SELECT i.item_name,
       inv.quantity,
       c.name AS category_name
FROM item i
JOIN inventory inv ON i.item_id = inv.item_id
JOIN item_category ic ON i.item_id = ic.item_id
JOIN category c ON ic.category_id = c.category_id
ORDER BY inv.quantity ASC;
```

Query 4: Identify the Product with the Lowest Inventory Quantity, Including Supplier Name and Most Recent Delivery Date

Stakeholder: Inventory Manager

Insight: Highlights low-stock items and category trends for inventory planning and pinpoints the product at risk of stockout, with supplier and delivery details to expedite reordering.

Relationship Type: Many-to-many (***ITEM*** to ***SUPPLIER*** via ***ITEM_SUPPLIER***)

```sql
SELECT i.item_name,
       inv.quantity,
       s.supplier_name,
       isup.last_delivery_date
FROM item i
JOIN inventory inv ON i.item_id = inv.item_id
JOIN item_supplier isup ON i.item_id = isup.item_id
JOIN supplier s ON isup.supplier_id = s.supplier_id
WHERE inv.quantity = (SELECT MIN(quantity) FROM inventory)
LIMIT 1;
```

Query 5: Display the Ten Most Recent Product Reviews, Including Customer Names and Product Category Names

Stakeholder: Marketing Director

Insight: Shows customer feedback by product and category, guiding product improvements and marketing.

Relationship Type: One-to-many (***ITEM*** to ***REVIEW***) and Many-to-many (***ITEM*** to ***CATEGORY***)

```
SELECT i.item_name,
       r.rating,
       r.comment,
       c.customer_name,
       cat.name AS category_name
FROM review r
JOIN customer c ON r.customer_id = c.customer_id
JOIN item i ON r.item_id = i.item_id
JOIN item_category ic ON i.item_id = ic.item_id
JOIN category cat ON ic.category_id = cat.category_id
ORDER BY r.review_date DESC
LIMIT 10;
```

Summary of Joins and Relationship Types

Query 1: One-to-many (***PLACED_ORDER*** to ***CUSTOMER***)

Query 2: Many-to-many (***ITEM*** to ***SUPPLIER*** via ***ITEM_SUPPLIER***), One-to-many (***ITEM*** to ***ORDER_ITEM***)

Query 3: Many-to-many (***ITEM*** to ***CATEGORY*** via ***ITEM_CATEGORY***), One-to-one (***ITEM*** to ***INVENTORY***)

Query 4: Many-to-many (***ITEM*** to ***SUPPLIER*** via ***ITEM_SUPPLIER***), One-to-one (***ITEM*** to ***INVENTORY***)

Query 5: One-to-many (***ITEM*** to ***REVIEW***, ***CUSTOMER*** to ***REVIEW***), Many-to-many (***ITEM*** to ***CATEGORY***)