🛒E-COMMERCE DATABASE SYSTEM🛍️

In this system we are using **relational database principles.** Our system can manage users, products, orders, inventory and categories. Shows the relation between tables.

Made with love by :

Dev Dipankar Bera, Neel Panajkar, Atharva Gangrade,

Ahmed Abdullatef , Mujtaba Ali Khan

COURSE: ITCS 3160-Database Design and Implementation

INSTRUCTOR: Dr. Azhar Ahamad

TA'S: Sharvary Dumbre and Thanmai Bvsma

Date: May 2, 2025

Project Overview

Project description :

- Our project is on E-Commerce Database system, it is designed to manage users, products, inventory, and categories for an limited online shopping platform. Our systems will show the customer history, placement of the order, and tracking of inventory.

Objectives:

- To create a relational database for managing E-Commerce operations.
- Also we to perform CREATE, READ , UPDATE , DELETE operations, joins, and aggregate functions.
- Ensure data integrity and scalability through proper schema design.

Scope:

- Manage products catalog with categories
- Manage customers
- Order processing system

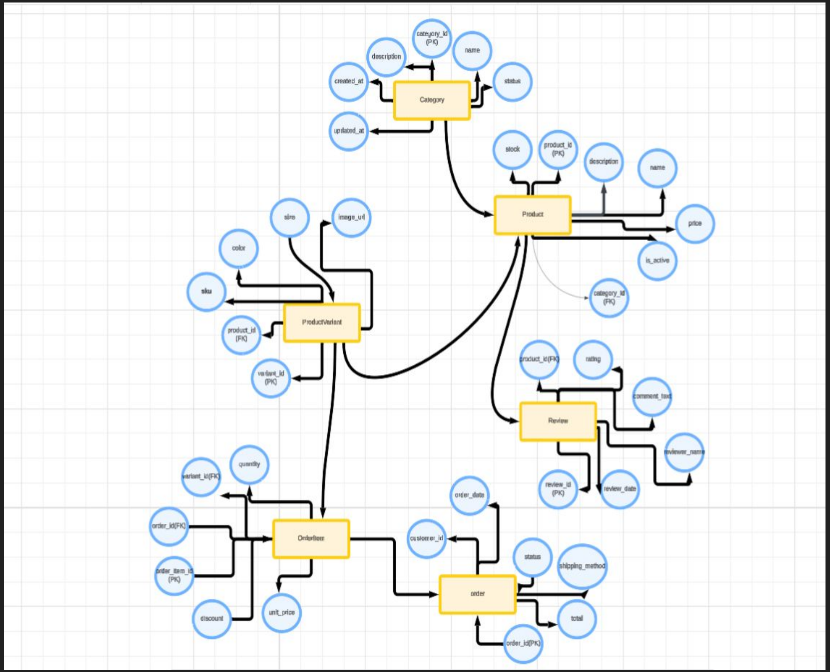
Requirements Gathering

🗂️**Data Requirements:** We need to create tables for Category, Product, ProductVariant, Order, OrderItem, and Review. These cover all core aspects of the system, including inventory, order tracking, and user feedback.

⚙️**Functional Requirements:**

- Customers can browse product categories and variants
- Customers can place orders with specific product variants
- System tracks stock and calculates total with discounts
- Admins can insert/update/delete categories and products
- Orders are processed with shipment method and status
- Customers can leave product reviews and ratings

ER Diagram



Clickable link below:

Here is the link of lucidchart to view the above chart properly

And

The Google document for the complete information about our table relationships.

Category

- Represents product categories (e.g., Electronics, Clothing).
- Contains attributes like category_id, name, description, status, etc.

Product

- Each product belongs to a category (category_id is a foreign key).
- Contains attributes like product_id, name, price, stock, etc.

ProductVariant

- Represents specific versions of a product (e.g., different sizes/colors).
- Linked to Product via product_id (foreign key).
- Contains attributes like variant_id, sku, color, "size", etc.

Order

- Represents a customer's purchase.
- Contains attributes like order_id, customer_id, order_date, status, etc.

OrderItem

- Connects orders and product variants (many-to-many relationship).
- Each record indicates which variant was ordered and in what quantity.
- Foreign keys: order_id and variant_id.

Review

- Captures customer feedback for products.
- Linked to Product via product_id.
- Contains attributes like rating, comment_text, reviewer_name, and review_date.

Logical Design

1. Category

- **PK**: category_id
- name
- description
- status
- created_at
- updated_at

2. Product

- **PK**: product_id
- name
- description
- price
- stock
- **FK**: category_id → Category(category_id)
- is_active

3. ProductVariant

- **PK**: variant_id
- **FK**: product_id → Product(product_id)
- sku
- color
- size
- image_url

4. Order

- **PK**: order_id
- customer_id
- order_date
- status
- shipping_method
- total

5. OrderItem

- **PK**: order_item_id
- **FK**: order_id → Order(order_id)
- **FK**: variant_id → ProductVariant(variant_id)
- quantity
- unit_price
- discount

6. Review

- **PK**: review_id
- **FK**: product_id → Product(product_id)
- rating
- comment_text
- reviewer_name
- review_date

Normalization to 3NF

1NF (First Normal Form)

- Each table has a primary key
- All attributes contain atomic (indivisible) values
- No repeating groups or arrays (e.g., colors, sizes in `ProductVariant` are separate rows or columns)

2NF (Second Normal Form)

- All non-key attributes are fully functionally dependent on the entire primary key
- Example: In `OrderItem`, `quantity`, `unit_price`, and `discount` depend on the full `order_item_id` (PK)
- No partial dependencies in any table

3NF (Third Normal Form)

- No transitive dependencies (non-key attribute depending on another non-key attribute)
- Example: In `Product`, `price` and `stock` depend only on `product_id` (PK), not on `category_id` or any other field
- All attributes are directly dependent on the PK and only the PK

PK : Orange
FK: Yellow

Physical Model

1. Sample Data

```
SELECT * FROM Category;

SELECT * FROM Product;

SELECT * FROM ProductVariant;

SELECT * FROM "Order";

SELECT * FROM OrderItem;

SELECT * FROM Review;
```

3. Product Sales & Ratings Summary

```
SELECT
    p.name AS product_name,
    c.name AS category_name,
    SUM(oi.quantity) AS total_units_sold,
    SUM(oi.quantity * oi.unit_price) AS total_revenue,
    ROUND(AVG(r.rating), 2) AS average_rating
FROM Product p
JOIN Category c ON p.category_id = c.category_id
JOIN ProductVariant pv ON p.product_id = pv.product_id
JOIN OrderItem oi ON pv.variant_id = oi.variant_id
JOIN "Order" o ON oi.order_id = o.order_id
LEFT JOIN Review r ON p.product_id = r.product_id
GROUP BY p.name, c.name
ORDER BY total_revenue DESC;
```

4. Total Revenue by Category

```
SELECT
    c.name AS category_name,
    SUM(oi.quantity * oi.unit_price) AS
category_revenue
FROM Category c
JOIN Product p ON c.category_id =
p.category_id
JOIN ProductVariant pv ON p.product_id =
pv.product_id
JOIN OrderItem oi ON pv.variant_id =
oi.variant_id
GROUP BY c.name
ORDER BY category_revenue DESC;
```

2. 2–3 CRUD operations

```
INSERT INTO Category VALUES (11, 'Extra', 'Backup',
'Active', SYSDATE, SYSDATE);

UPDATE Product SET price = 149 WHERE product_id =
5;

DELETE FROM Review WHERE review_id = 210;
```

5. Detailed Order Item Breakdown with Discounted Total

```
SELECT
    o.order_id,
    p.name AS product_name,
    oi.quantity,
    oi.unit_price,
    oi.discount,
    (oi.quantity * (oi.unit_price - oi.discount)) AS total_price_after_discount
FROM "Order" o
JOIN OrderItem oi ON o.order_id = oi.order_id
JOIN ProductVariant pv ON oi.variant_id = pv.variant_id
JOIN Product p ON pv.product_id = p.product_id
ORDER BY o.order_id;
```

***Screenshots for all queries and code are provided in a word document :**
https://docs.google.com/document/d/1rJdnosHE57aPn5co5wyQpj1-2bZZKA0W0vFQ5YmMd-s/edit?usp=sharing

Testing and Validation Plan

1. Full Table Join

```
SELECT
    o.order_id,
    o.order_date,
    c.name AS category_name,
    p.name AS product_name,
    pv.sku,
    oi.quantity,
    oi.unit_price,
    r.rating,
    r.comment_text
FROM "Order" o
JOIN OrderItem oi ON o.order_id = oi.order_id
JOIN ProductVariant pv ON oi.variant_id =
pv.variant_id
JOIN Product p ON pv.product_id =
p.product_id
JOIN Category c ON p.category_id =
c.category_id
LEFT JOIN Review r ON p.product_id =
r.product_id;
```

2. Revenue Summary per Product

```
SELECT
    p.name AS product_name,
    SUM(oi.quantity * oi.unit_price) AS total_revenue
FROM Product p
JOIN ProductVariant pv ON p.product_id = pv.product_id
JOIN OrderItem oi ON pv.variant_id = oi.variant_id
GROUP BY p.name
ORDER BY total_revenue DESC;
```

3. Top Rated Products

```
SELECT
    p.name AS product_name,
    ROUND(AVG(r.rating), 2) AS average_rating
FROM Product p
JOIN Review r ON p.product_id = r.product_id
GROUP BY p.name
HAVING AVG(r.rating) >= 4.0
ORDER BY average_rating DESC;
```

🏆 Bonus Section

1. Full Relationship Join

```
SELECT
    o.order_id,
    c.name AS category_name,
    p.name AS product_name,
    pv.sku,
    oi.quantity,
    oi.unit_price,
    r.rating,
    r.comment_text
FROM "Order" o
JOIN OrderItem oi ON o.order_id = oi.order_id
JOIN ProductVariant pv ON oi.variant_id = pv.variant_id
JOIN Product p ON pv.product_id = p.product_id
JOIN Category c ON p.category_id = c.category_id
LEFT JOIN Review r ON p.product_id = r.product_id;
```

🏆 Bonus Section

2. Join with Aggregation

```
SELECT
    o.order_id,
    SUM(oi.quantity * oi.unit_price) AS total_order_amount
FROM "Order" o
JOIN OrderItem oi ON o.order_id = oi.order_id
GROUP BY o.order_id
ORDER BY total_order_amount DESC;
```

*** Screenshots for all queries are provided in a word document :**
https://docs.google.com/document/d/1rJdnosHE57aPn5co5wyQpj1-2bZZKA0W0vFQ5YmMd-s/edit?usp=sharing

Conclusion

Our project successfully demonstrates the design and implementation of a relational database system tailored for modern e-commerce operations.

- It manages essential entities such as users, products, orders, inventory, and categories.
- The system supports robust CRUD operations, enabling smooth data manipulation and maintenance.
- We ensured data integrity through appropriate use of primary and foreign keys.
- Functionalities like order tracking, customer management, and inventory updates are handled efficiently.
- The design follows normalization principles (up to 3NF) to support scalability and maintainability.

*** Screenshots for all queries are provided in a word document :**
https://docs.google.com/document/d/1rJdnosHE57aPn5co5wyQpj1-2bZZKA0W0vFQ5YmMd-s/edit?usp=sharing

Thank You