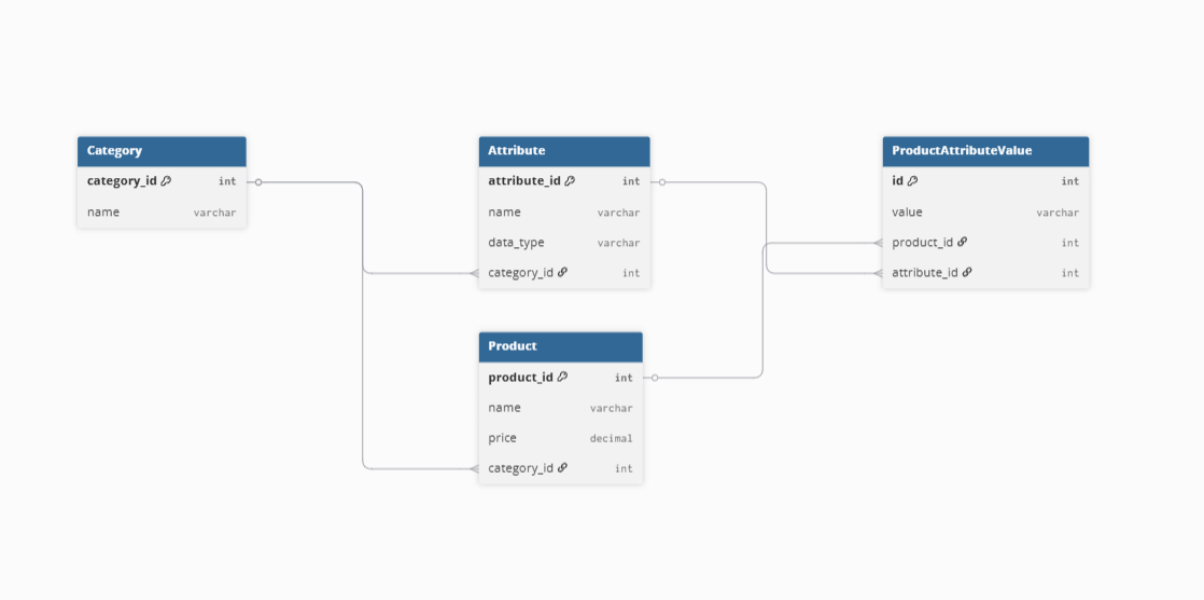
1. ER DIAGRAM:



JUSTIFICATION:

The proposed database design follows a **normalized relational schema** with an **Entity–Attribute–Value (EAV) extension** to support category-specific product attributes.

Dynamic Categories & Attributes

- Each product belongs to a Category, and each category defines its own Category_Attribute list (e.g., Smartphones → RAM, Battery; Watches → Dial Color, Strap Type).

Data Integrity

- Foreign keys ensure all products, variants, and attributes reference valid categories, brands, and option sets.

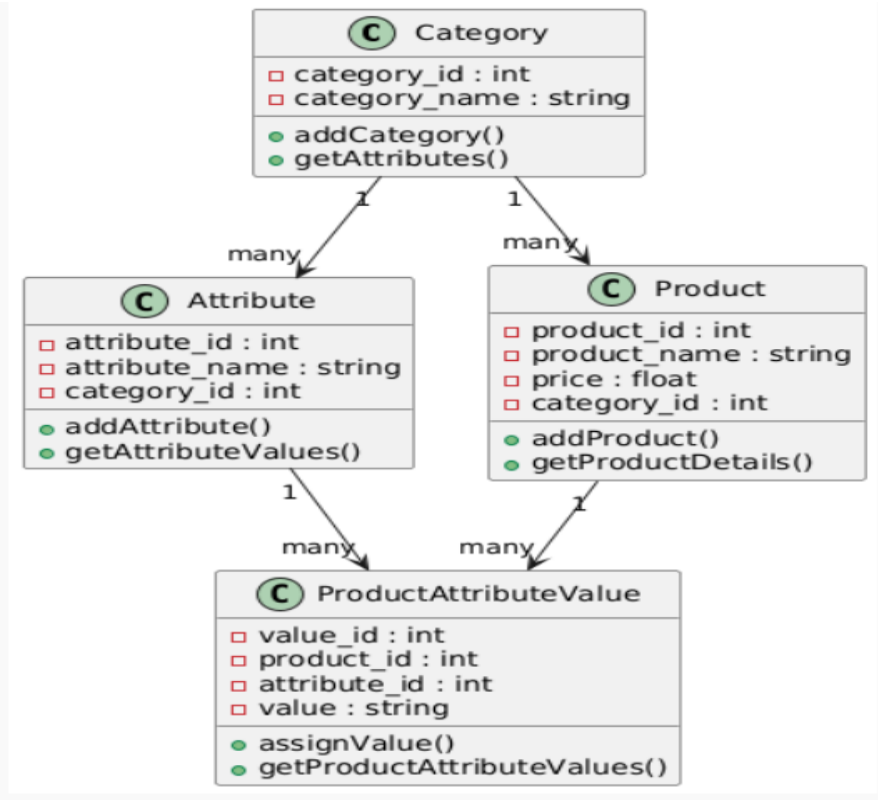
Product & Variant Separation

- Product stores general product info (brand, description).
- Product_Variant allows variant-level details (e.g., size, color, price, stock).

Scalability & Performance

- Core entities (Category, Product, Variant) remain relational for fast queries.
- Custom attributes are stored in an EAV table, enabling unlimited category-specific attributes without performance issues.

2. CLASS DIAGRAM:



JUSTIFICATION:

The class diagram is designed based on the database schema, mapping tables into logical classes with clear responsibilities.

- **Category & Category Attribute**: Handle product categories and their custom attributes (flexibility to add new fields like RAM, Dial Color).
- **Product & Product Variant**: Separate main product details from variant-level details (price, stock, size/color). This ensures scalability for both simple and variant products.
- **Product Attribute Value**: Stores actual values for category-specific attributes, keeping the model dynamic.