Perfect 👍 You want a **complete list of all tables** in the e-commerce database design with their **properties** and **descriptions**.
Here's a structured breakdown:

📌 Relational Database Design for Mid-Level E-Commerce

---

1. users

Holds customer, admin, and seller details.

Column	Type	Description
id	BIGINT (PK)	Unique user identifier
name	VARCHAR(150)	Full name of the user
email	VARCHAR(150), UNIQUE	User login/communication email
password_hash	VARCHAR(255)	Hashed password for authentication
phone	VARCHAR(20)	User contact number
role	ENUM(customer, admin, seller)	Defines user role
created_at	TIMESTAMP	Record creation time
updated_at	TIMESTAMP	Last update time

---

2. categories

Supports recursive categories (subcategory to N levels).

Column	Type	Description
id	BIGINT (PK)	Category ID
name	VARCHAR(150)	Category name
parent_id	BIGINT (FK → categories.id)	Points to parent category (NULL for root)
description	TEXT	Details about the category
created_at	TIMESTAMP	Record creation time
updated_at	TIMESTAMP	Last update time

---

3. brands

Stores product brands.

Column	Type	Description
id	BIGINT (PK)	Brand ID
name	VARCHAR(150)	Brand name
description	TEXT	Brand description
created_at	TIMESTAMP	Record creation time
updated_at	TIMESTAMP	Last update time

---

4. sizes

Standard product sizes.

Column	Type	Description
id	BIGINT (PK)	Size ID
name	VARCHAR(50)	Size name (e.g., S, M, L, XL)
created_at	TIMESTAMP	Record creation time

---

5. products

Main product details.

Column	Type	Description
id	BIGINT (PK)	Product ID
category_id	BIGINT (FK → categories.id)	Product category
brand_id	BIGINT (FK → brands.id)	Product brand
name	VARCHAR(255)	Product name
sku	VARCHAR(100), UNIQUE	Stock keeping unit
description	TEXT	Product details
base_price	DECIMAL(10,2)	Base price
stock_quantity	INT	Total stock available
weight	DECIMAL(8,2)	Weight (optional for shipping)
is_seasonal	BOOLEAN	Seasonal product flag
seasonal_start_date	DATE	Seasonal start date
seasonal_end_date	DATE	Seasonal end date
status	ENUM(active, inactive)	Product availability
created_at	TIMESTAMP	Record creation time
updated_at	TIMESTAMP	Last update time

---

6. product_images

Stores multiple images for each product.

Column	Type	Description
id	BIGINT (PK)	Image ID
product_id	BIGINT (FK → products.id)	Related product
image_url	VARCHAR(255)	Image path/URL
is_primary	BOOLEAN	Marks main image
created_at	TIMESTAMP	Record creation time

---

7. product_variants

Stores size/color variations of a product.

Column	Type	Description
id	BIGINT (PK)	Variant ID
product_id	BIGINT (FK → products.id)	Related product
size_id	BIGINT (FK → sizes.id)	Product size
color	VARCHAR(50)	Product color
additional_price	DECIMAL(10,2)	Extra cost for variant
stock_quantity	INT	Available stock
created_at	TIMESTAMP	Record creation time

---

8. wishlists

Stores user’s saved products.

Column	Type	Description
id	BIGINT (PK)	Wishlist ID
user_id	BIGINT (FK → users.id)	Related user
product_id	BIGINT (FK → products.id)	Related product
created_at	TIMESTAMP	Record creation time

---

9. orders

Stores purchase orders.

Column	Type	Description
id	BIGINT (PK)	Order ID
user_id	BIGINT (FK → users.id)	Related customer
total_amount	DECIMAL(10,2)	Subtotal amount
discount_amount	DECIMAL(10,2)	Total discount applied
shipping_fee	DECIMAL(10,2)	Shipping charge
final_amount	DECIMAL(10,2)	Total after discounts/shipping
status	ENUM(pending, paid, shipped, delivered, cancelled)	Order status
created_at	TIMESTAMP	Order date
updated_at	TIMESTAMP	Last update

---

10. order_items

Stores items in an order.

Column	Type	Description
id	BIGINT (PK)	Item ID
order_id	BIGINT (FK → orders.id)	Related order
product_id	BIGINT (FK → products.id)	Related product
variant_id	BIGINT (FK → product_variants.id)	Specific variant
quantity	INT	Number of units
price_at_purchase	DECIMAL(10,2)	Price at purchase time
discount_applied	DECIMAL(10,2)	Discount per item
created_at	TIMESTAMP	Record creation time

---

11. payments

Stores payment transactions.

Column	Type	Description
id	BIGINT (PK)	Payment ID
order_id	BIGINT (FK → orders.id)	Related order
payment_method	ENUM(card, mobile_banking, COD)	Payment method
transaction_id	VARCHAR(150)	Payment gateway transaction ID
amount	DECIMAL(10,2)	Paid amount

Column	Type	Description
status	ENUM(pending, success, failed)	Payment status
created_at	TIMESTAMP	Payment date

---

12. coupons

Coupon details for discounts.

Column	Type	Description
id	BIGINT (PK)	Coupon ID
code	VARCHAR(50), UNIQUE	Coupon code
description	TEXT	Coupon details
discount_type	ENUM(percentage, fixed)	Type of discount
discount_value	DECIMAL(10,2)	Value of discount
min_purchase_amount	DECIMAL(10,2)	Minimum order value
max_discount_amount	DECIMAL(10,2)	Maximum discount (optional)
valid_from	DATE	Start date
valid_to	DATE	End date
usage_limit	INT	Max number of uses
status	ENUM(active, inactive)	Status
created_at	TIMESTAMP	Creation time

---

13. coupon_usages

Tracks coupon usage.

Column	Type	Description
id	BIGINT (PK)	Usage ID
coupon_id	BIGINT (FK → coupons.id)	Related coupon
user_id	BIGINT (FK → users.id)	Who used the coupon
order_id	BIGINT (FK → orders.id)	Order where coupon applied
used_at	TIMESTAMP	When coupon was used

---

14. discounts

Special discounts on categories/products.

Column	Type	Description
id	BIGINT (PK)	Discount ID
product_id	BIGINT NULL (FK → products.id)	Discounted product
category_id	BIGINT NULL (FK → categories.id)	Discounted category
discount_type	ENUM(percentage, fixed)	Discount type
discount_value	DECIMAL(10,2)	Value of discount
valid_from	DATE	Start date
valid_to	DATE	End date
created_at	TIMESTAMP	Creation time

---

15. shipping_methods

Defines shipping options.

Column	Type	Description
id	BIGINT (PK)	Shipping ID
name	VARCHAR(150)	Shipping name
description	TEXT	Details
fee	DECIMAL(10,2)	Shipping fee
is_free_shipping	BOOLEAN	Free shipping flag
created_at	TIMESTAMP	Record creation time

---

16. order_shipping

Stores shipping details for each order.

Column	Type	Description
id	BIGINT (PK)	Shipping record ID
order_id	BIGINT (FK → orders.id)	Related order
shipping_method_id	BIGINT (FK → shipping_methods.id)	Selected method
address	TEXT	Delivery address
tracking_number	VARCHAR(100)	Courier tracking number
created_at	TIMESTAMP	Creation time

---

17. reviews

Product reviews and ratings.

Column	Type	Description
id	BIGINT (PK)	Review ID
user_id	BIGINT (FK → users.id)	Reviewer
product_id	BIGINT (FK → products.id)	Reviewed product
rating	INT (1–5)	Star rating
comment	TEXT	Customer review
created_at	TIMESTAMP	Review date

---

18. inventory_logs

Tracks stock changes.

Column	Type	Description
id	BIGINT (PK)	Log ID
product_id	BIGINT (FK → products.id)	Related product
variant_id	BIGINT NULL (FK → product_variants.id)	Related variant
change_type	ENUM(in, out)	Stock in/out
quantity_changed	INT	Amount changed
note	TEXT	Reason (sale, restock, return)
created_at	TIMESTAMP	Date of log

---

19. notifications

System notifications for users.

Column	Type	Description
id	BIGINT (PK)	Notification ID
user_id	BIGINT (FK → users.id)	Receiver
message	TEXT	Notification content
status	ENUM(unread, read)	Read status
created_at	TIMESTAMP	Notification time

✅ This is a **full professional e-commerce schema** with **all features you requested + standard tables** (sizes, brands, reviews, logs, notifications).

In **Laravel 12**, you can manage **soft delete** easily using the `SoftDeletes` trait.
Soft deleting means instead of permanently removing a record from the database, Laravel will just set a `deleted_at` timestamp. That way you can restore it later if needed.

Steps to Manage Soft Delete in Laravel 12

1. Add `deleted_at` column in migration

When creating a table (for example `products`), add the `softDeletes()` column:

```
public function up(): void
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->decimal('price', 10, 2);
        $table->integer('stock_quantity');
        $table->timestamps();
        $table->softDeletes(); // 👈 adds deleted_at column
    });
}
```

Then run:

```
php artisan migrate
```

---

2. Enable SoftDeletes in Model

In your model (e.g., `Product.php`):

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Product extends Model
{
    use SoftDeletes; // 👈 enables soft delete

    protected $fillable = ['name', 'price', 'stock_quantity'];

    protected $dates = ['deleted_at']; // optional in Laravel 12 (auto-
detected)
}
```

---

3. Using Soft Deletes in Queries

- **Delete a record (soft delete):**

```
$product = Product::find(1);
$product->delete(); // sets deleted_at timestamp
```

- **Get all records except soft deleted:**

```
$products = Product::all();
```

- **Get only soft deleted records:**

```
$trashed = Product::onlyTrashed()->get();
```

- **Get both active and soft deleted records:**

```
$allProducts = Product::withTrashed()->get();
```

- **Restore a soft deleted record:**

```
$product = Product::withTrashed()->find(1);
$product->restore(); // sets deleted_at to NULL
```

- **Permanently delete a record:**

```
$product = Product::withTrashed()->find(1);
$product->forceDelete(); // removes row from DB permanently
```

✅ This is the **standard way** to manage soft deletes in **Laravel 12**.

Do you want me to also give you a **real-world example for your ecommerce project** (like soft delete for `orders`, `products`, and `users`)?