You are a data generation assistant helping to populate a complete e-commerce database called “impactdb” with realistic and coherent sample data.

Objective

Generate **a full SQL script** named that inserts a large, consistent, and realistic dataset into all the tables of the “impactdb” schema (whose structure is exactly the one defined below).

All “INSERT” statements **must respect the datatypes, constraints, foreign keys, and checks** exactly as they are in the schema.

Database Structure (to respect)

The tables (and their relationships) are defined exactly as follows:

```
CREATE TABLE client (
  client_id VARCHAR(10), -- ID du client Alphanumérique 10
  email VARCHAR(100) NOT NULL, -- Adresse email du client Texte 100
  first_name VARCHAR(50) NOT NULL, -- Prénom du client Texte 50
  last_name VARCHAR(50) NOT NULL, -- Nom de famille du client Texte 50
  phone_number VARCHAR(20) NOT NULL, -- Numéro de téléphone du client (non spécifié dans le PDF, taille estimée)
  password VARCHAR(255) NOT NULL, -- Mot de passe du client Texte 255
  fidelity_status BOOLEAN NOT NULL DEFAULT FALSE, -- Statut programme fidélité Booléen 1
  subscribtion_date TIMESTAMP -- Date d'inscription du client Date 10
);

CREATE TABLE comm_preference_type (
  type_id SMALLINT, -- Identifiant interne (non spécifié dans le PDF)
  label VARCHAR(50) NOT NULL, -- Nom ou libellé (non spécifié dans le PDF)
  description VARCHAR(255) -- Description (non spécifié dans le PDF)
);

CREATE TABLE brand (
  brand_id SMALLINT, -- Identifiant de la marque (non spécifié dans le PDF)
```

```
  name VARCHAR(50) NOT NULL -- Marque de l'article Texte 50
);

CREATE TABLE category (
  category_id SMALLINT, -- Identifiant de catégorie (non spécifié dans le PDF)
  name VARCHAR(50) NOT NULL, -- Catégorie principale Texte 50
  category_parent_id SMALLINT -- Sous-catégorie Texte 50 (référence parent)
);

CREATE TABLE discount_code (
  code_id VARCHAR(10), -- ID code de réduction Alphanumérique 10
  code VARCHAR(10) NOT NULL, -- Code du bon de réduction Alphanumérique 10
  value DECIMAL(5, 2) NOT NULL, -- Valeur du code de réduction Numérique 5,2
  type VARCHAR(10) NOT NULL, -- Type de réduction (fixe/%) Texte 10
  start_date TIMESTAMP NOT NULL, -- Date de début de validité Date 10
  end_date TIMESTAMP NOT NULL -- Date de fin de validité Date 10
);

CREATE TABLE support_ticket (
  ticket_id VARCHAR(12), -- ID du ticket support Alphanumérique 12
  subject VARCHAR(100) NOT NULL, -- Sujet du ticket Texte 100
  description VARCHAR(1000), -- Description du ticket Texte 1000
  status VARCHAR(20) NOT NULL, -- Statut du ticket Texte 20
  creation_date TIMESTAMP NOT NULL, -- Date de création du ticket Date 10
  client_id VARCHAR(10) NOT NULL -- ID du client Alphanumérique 10
);

CREATE TABLE ticket_history (
  ticket_id VARCHAR(12), -- ID du ticket support Alphanumérique 12
  line_id SMALLINT, -- Ligne interne (non spécifié dans le PDF)
  date_time TIMESTAMP NOT NULL, -- Date de l'échange Date 10
```

```
  exchange_content VARCHAR(2000) NOT NULL -- Contenu de l'échange Texte (non spécifié, 2000)
);

CREATE TABLE location (
  location_id VARCHAR(8), -- ID de l'entrepôt/magasin Alphanumérique 8
  address_line VARCHAR(200) NOT NULL, -- Ligne d'adresse Texte 200
  postal_code_id INT NOT NULL
);
CREATE TABLE postal_code(
        postal_code_id INT,
  postal_code VARCHAR(10) NOT NULL, -- Code postal Alphanumérique 10
  city VARCHAR(50) NOT NULL , -- Ville Texte 50
  country_id INT NOT NULL
);
CREATE TABLE country(
        country_id INT,
  country_name VARCHAR(50) NOT NULL-- Pays Texte 50
);



CREATE TABLE client_address (
  client_address_id VARCHAR(10), -- ID de l'adresse de livraison Alphanumérique 10
  is_default BOOLEAN NOT NULL DEFAULT FALSE, -- Statut adresse par défaut Booléen 1
  location_id VARCHAR(8) NOT NULL, -- ID de l'entrepôt/magasin Alphanumérique 8
  client_id VARCHAR(10) NOT NULL-- ID du client Alphanumérique 10
);

CREATE TABLE product (
  product_id VARCHAR(15), -- Référence de l'article Alphanumérique 15
  name VARCHAR(100) NOT NULL, -- Nom de l'article Texte 100
```

```
  description VARCHAR(1000), -- Description de l'article Texte 1000
  category_id SMALLINT NOT NULL, -- Catégorie principale Texte 50
  brand_id SMALLINT NOT NULL-- Marque de l'article Texte 50
);

CREATE TABLE warehouse (
  warehouse_id VARCHAR(8), -- ID de l'entrepôt/magasin Alphanumérique 8
  name VARCHAR(50) NOT NULL, -- Nom de l'entrepôt/magasin Texte 50
  location_id VARCHAR(8) NOT NULL -- ID de localisation (référence au même type)
);

CREATE TABLE article (
  SKU_code VARCHAR(15), -- Référence de l'article Alphanumérique 15
  size VARCHAR(5) NOT NULL, -- Taille de l'article Alphanumérique 5
  color VARCHAR(30) NOT NULL, -- Couleur de l'article Texte 30
  product_id VARCHAR(15) NOT NULL -- Référence de l'article Alphanumérique 15
);

CREATE TABLE price (
  SKU_code VARCHAR(15), -- Référence de l'article Alphanumérique 15
  start_date TIMESTAMP NOT NULL, -- Date de début de validité Date 10
  sale_price DECIMAL(8, 2) NOT NULL, -- Prix de base Numérique 8,2
  end_date TIMESTAMP NOT NULL -- Date de fin de validité Date 10
);

CREATE TABLE order_ (
  order_id VARCHAR(12), -- ID de la commande Alphanumérique 12
  date_ TIMESTAMP NOT NULL, -- Date de la commande Date 10
  total_amount DECIMAL(10, 2) NOT NULL, -- Montant total de la commande Numérique 10,2
  status VARCHAR(20) NOT NULL DEFAULT 'PENDING', -- Statut de la commande Texte 20
  client_address_id VARCHAR(10) NOT NULL, -- ID de l'adresse de livraison Alphanumérique 10
```

```
  client_id VARCHAR(10) NOT NULL -- ID du client Alphanumérique 10
);

CREATE TABLE transaction (
  transaction_id VARCHAR(20), -- ID de la transaction Alphanumérique 20
  amount DECIMAL(10, 2) NOT NULL, -- Montant de la transaction Numérique 10,2
  payment_type VARCHAR(30) NOT NULL, -- Type de paiement Texte 30
  order_id VARCHAR(12) NOT NULL -- ID de la commande Alphanumérique 12
);

CREATE TABLE return_ (
  return_id VARCHAR(10), -- ID du retour Alphanumérique 10
  return_date TIMESTAMP NOT NULL, -- Date de retour Date 10
  reason VARCHAR(150), -- Raison du retour Texte 150
  refund_amount DECIMAL(10, 2) NOT NULL, -- Montant du remboursement Numérique 10,2
  order_id VARCHAR(12) NOT NULL -- ID de la commande Alphanumérique 12
);

CREATE TABLE review (
  review_id VARCHAR(10), -- ID de l'avis Alphanumérique 10
  rating SMALLINT NOT NULL, -- Note (sur 5) Numérique 1
  content VARCHAR(500), -- Contenu de l'avis Texte 500
  status VARCHAR(20) NOT NULL DEFAULT 'PENDING', -- Statut de l'avis Texte 20
  product_id VARCHAR(15) NOT NULL, -- Référence de l'article Alphanumérique 15
  client_id VARCHAR(10) -- ID du client Alphanumérique 10
);

CREATE TABLE client_preference (
  client_id VARCHAR(10), -- ID du client Alphanumérique 10
  type_id SMALLINT -- Type de préférence (non spécifié dans le PDF)
);
```

```
CREATE TABLE stock (
   SKU_code VARCHAR(15), -- Référence de l'article Alphanumérique 15
   warehouse_id VARCHAR(8), -- ID de l'entrepôt/magasin Alphanumérique 8
   quantity SMALLINT NOT NULL -- Quantité en stock Numérique 5
);

CREATE TABLE order_line (
   SKU_code VARCHAR(15), -- Référence de l'article Alphanumérique 15
   order_id VARCHAR(12), -- ID de la commande Alphanumérique 12
   quantity_ordered SMALLINT NOT NULL, -- Quantité d'article commandée Numérique 4
   unit_price_paid DECIMAL(10, 2) NOT NULL -- Prix payé unitaire Numérique 10,2
);

CREATE TABLE applicable_category (
   category_id SMALLINT, -- Catégorie principale Texte 50
   code_id VARCHAR(10) -- ID code de réduction Alphanumérique 10
);

CREATE TABLE returned_item (
   SKU_code VARCHAR(15), -- Référence de l'article Alphanumérique 15
   return_id VARCHAR(10), -- ID du retour Alphanumérique 10
   quantity_returned SMALLINT NOT NULL -- Quantité retournée Numérique 4
);



-- Client
ALTER TABLE Client
   ADD CONSTRAINT PK_Client PRIMARY KEY (client_id),
   ADD CONSTRAINT UK_Client_Email UNIQUE (email);
```

```
-- comm_preference_type

ALTER TABLE comm_preference_type
  ADD CONSTRAINT PK_CommPreferenceType PRIMARY KEY (type_id),
  ADD CONSTRAINT UK_CommPreferenceType_Label UNIQUE (label);

-- TABLE: brand
ALTER TABLE brand
  ADD CONSTRAINT PK_Brand PRIMARY KEY (brand_id),
  ADD CONSTRAINT UK_Brand_Name UNIQUE (name);

-- TABLE: category
ALTER TABLE category
  ADD CONSTRAINT PK_Category PRIMARY KEY (category_id);
ALTER TABLE category
  ADD CONSTRAINT UK_Category_Name UNIQUE (name),
  ADD CONSTRAINT FK_Category_ParentCategory FOREIGN KEY (category_parent_id)
    REFERENCES category (category_id)
    ON DELETE SET NULL -- si la catégorie parente est supprimée, la sous-catégorie devient principale (NULL)
    ON UPDATE CASCADE;


-- TABLE: discount_code
ALTER TABLE discount_code
  ADD CONSTRAINT PK_DiscountCode PRIMARY KEY (code_id),
  ADD CONSTRAINT UK_DiscountCode_Code UNIQUE (code),
  ADD CONSTRAINT CHK_DiscountCode_Dates CHECK (end_date >= start_date),
  ADD CONSTRAINT CHK_DiscountCode_ValuePositive CHECK (value >= 0),
  ADD CONSTRAINT CHK_DiscountCode_Type CHECK (type IN ('PERCENT', 'FIXED'));
```

```
-- TABLE: country
ALTER TABLE country
   ADD CONSTRAINT PK_Country PRIMARY KEY (country_id),
   ADD CONSTRAINT UK_Country_Name UNIQUE (country_name);



-- TABLE: postal_code
ALTER TABLE postal_code
   ADD CONSTRAINT PK_PostalCode PRIMARY KEY (postal_code_id),
   ADD CONSTRAINT UK_PostalCode_PostalCodeCountry UNIQUE (postal_code, country_id),
   ADD CONSTRAINT FK_PostalCode_Country FOREIGN KEY (country_id)
      REFERENCES country (country_id)
      ON DELETE RESTRICT
      ON UPDATE CASCADE;


-- ----------------------------------------
-- TABLE: location
-- ----------------------------------------
ALTER TABLE location
   ADD CONSTRAINT PK_Location PRIMARY KEY (location_id),
   ADD CONSTRAINT FK_Location_PostalCode FOREIGN KEY (postal_code_id)
      REFERENCES postal_code (postal_code_id)
      ON DELETE RESTRICT -- protecting the locations
      ON UPDATE CASCADE;



-- TABLE: support_ticket
ALTER TABLE support_ticket
```

```
  ADD CONSTRAINT PK_SupportTicket PRIMARY KEY (ticket_id),
  ADD CONSTRAINT FK_SupportTicket_Client FOREIGN KEY (client_id)
    REFERENCES Client (client_id)
    ON DELETE CASCADE -- si le client est supprimé, l'historique de ses tickets n'est plus pertinent, no come back
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_SupportTicket_Status CHECK (status IN ('OPEN', 'IN_PROGRESS', 'CLOSED', 'RESOLVED'));


-- TABLE: warehouse
ALTER TABLE warehouse
  ADD CONSTRAINT PK_Warehouse PRIMARY KEY (warehouse_id),
  ADD CONSTRAINT UK_Warehouse_Name UNIQUE (name),
  ADD CONSTRAINT FK_Warehouse_Location FOREIGN KEY (location_id)
    REFERENCES location (location_id)
    ON DELETE RESTRICT -- Protège les emplacements utilisés
    ON UPDATE CASCADE;


-- TABLE: product
ALTER TABLE product
  ADD CONSTRAINT PK_Product PRIMARY KEY (product_id),
  ADD CONSTRAINT FK_Product_Category FOREIGN KEY (category_id)
    REFERENCES category (category_id)
    ON DELETE RESTRICT -- impossible de suppremersi des produits existent
    ON UPDATE CASCADE,
  ADD CONSTRAINT FK_Product_Brand FOREIGN KEY (brand_id)
    REFERENCES brand (brand_id)
    ON DELETE RESTRICT -- ici aussi
    ON UPDATE CASCADE;
```

```
-- TABLE: article (SKU)
ALTER TABLE article
  ADD CONSTRAINT PK_Article PRIMARY KEY (SKU_code),
  ADD CONSTRAINT FK_Article_Product FOREIGN KEY (product_id)
    REFERENCES product (product_id)
    ON DELETE CASCADE -- si le produit est supprime toutes les variations disparaissent
    ON UPDATE CASCADE;


-- TABLE: client_address
ALTER TABLE client_address
  ADD CONSTRAINT PK_ClientAddress PRIMARY KEY (client_address_id),
  ADD CONSTRAINT UK_ClientAddress_Unique_Location UNIQUE (client_id, location_id),

  ADD CONSTRAINT FK_ClientAddress_Client FOREIGN KEY (client_id)
    REFERENCES Client (client_id)
    ON DELETE CASCADE -- l'adresse n'existe pas sans le client
    ON UPDATE CASCADE,
  ADD CONSTRAINT FK_ClientAddress_Location FOREIGN KEY (location_id)
    REFERENCES location (location_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_ClientAddress_IsDefault CHECK (is_default IN (0, 1));


-- TABLE: order_
ALTER TABLE order_
  ADD CONSTRAINT PK_Order PRIMARY KEY (order_id),
  ADD CONSTRAINT FK_Order_Client FOREIGN KEY (client_id)
    REFERENCES Client (client_id)
```

```
    ON DELETE RESTRICT    -- le client ne doit pas être supprimé tant qu'il a des commandes
    ON UPDATE CASCADE,
  -- impossoble de supprimer une adresse si commande en cours pour cette adresse par mesure de sécurité; RESTRICT est plus sûr.
  ADD CONSTRAINT FK_Order_ClientAddress FOREIGN KEY (client_address_id)
    REFERENCES client_address (client_address_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_Order_Status CHECK (status IN ('PENDING', 'PROCESSING', 'SHIPPED', 'DELIVERED', 'CANCELLED', 'RETURNED'));

-- TABLE: transaction
ALTER TABLE transaction
  ADD CONSTRAINT PK_Transaction PRIMARY KEY (transaction_id),
  ADD CONSTRAINT FK_Transaction_Order FOREIGN KEY (order_id)
    REFERENCES order_ (order_id)
    ON DELETE CASCADE    -- La transaction dépend entièrement de la commande.
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_Transaction_PaymentType CHECK (payment_type IN ('CARD', 'PAYPAL', 'TRANSFER', 'WALLET','REFUND'));

-- TABLE: return_
ALTER TABLE return_
  ADD CONSTRAINT PK_Return PRIMARY KEY (return_id),
  ADD CONSTRAINT FK_Return_Order FOREIGN KEY (order_id)
    REFERENCES order_ (order_id)
    ON DELETE RESTRICT -- Protège l'historique de la commande
    ON UPDATE CASCADE;
```

```
-- TABLE: review
ALTER TABLE review
  ADD CONSTRAINT PK_Review PRIMARY KEY (review_id),
  -- un avis est lié au produit et au client
  ADD CONSTRAINT FK_Review_Client FOREIGN KEY (client_id)
    REFERENCES Client (client_id)
    ON DELETE SET NULL -- si le client est supprimé l'avis reste anonyme
    ON UPDATE CASCADE,
  ADD CONSTRAINT FK_Review_Product FOREIGN KEY (product_id)
    REFERENCES product (product_id)
    ON DELETE CASCADE -- si le produit est supprimé, les avis le concernant aussi sont supp
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_Review_Rating CHECK (rating BETWEEN 1 AND 5),
  ADD CONSTRAINT CHK_Review_Status CHECK (status IN ('PENDING', 'APPROVED',
'REJECTED'));
```

```
-- TABLE: TICKET_HISTORY

ALTER TABLE ticket_history
  ADD CONSTRAINT PK_TicketHistory PRIMARY KEY (ticket_id, line_id),
  ADD CONSTRAINT FK_TicketHistory_Ticket FOREIGN KEY (ticket_id)
    REFERENCES support_ticket (ticket_id)
    ON DELETE CASCADE  -- L'historique des lignes dépend du ticket parent.
    ON UPDATE CASCADE;

-- TABLE: PRICE (Historisation Article)
ALTER TABLE price
```

```
  ADD CONSTRAINT PK_Price PRIMARY KEY (SKU_code, start_date),

  ADD CONSTRAINT FK_Price_Article FOREIGN KEY (SKU_code)
    REFERENCES article (SKU_code)
    ON DELETE CASCADE -- les prix dépendent de l'article (SKU)
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_Price_SalePrice CHECK (sale_price > 0);

-- TABLE: CLIENT_PREFERENCE
ALTER TABLE client_preference
  ADD CONSTRAINT PK_ClientPreference PRIMARY KEY (client_id, type_id),
  ADD CONSTRAINT FK_ClientPreference_Client FOREIGN KEY (client_id)
    REFERENCES Client (client_id)
    ON DELETE CASCADE -- si le client est supprimé ses préférences le sont aussi
    ON UPDATE CASCADE,
  ADD CONSTRAINT FK_ClientPreference_Type FOREIGN KEY (type_id)
    REFERENCES comm_preference_type (type_id)
    ON DELETE RESTRICT-- si le type de préférence est supprimé on ne supprime pas le client
    ON UPDATE CASCADE;


-- TABLE: STOCK
ALTER TABLE stock
  ADD CONSTRAINT PK_Stock PRIMARY KEY (SKU_code, warehouse_id),

  ADD CONSTRAINT FK_Stock_Article FOREIGN KEY (SKU_code)
    REFERENCES article (SKU_code)
    ON DELETE CASCADE -- Si l'article(SKUç) est supprimé, l'enregistrement de stock doit l'etre
    ON UPDATE CASCADE,
  ADD CONSTRAINT FK_Stock_Warehouse FOREIGN KEY (warehouse_id)
    REFERENCES warehouse (warehouse_id)
```

```
    ON DELETE CASCADE -- Si l'entrepôt est supprimé aussi
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_Stock_Quantity CHECK (quantity >= 0);


-- TABLE: ORDER_LINE
ALTER TABLE order_line
  ADD CONSTRAINT PK_OrderLine PRIMARY KEY (SKU_code, order_id),
  ADD CONSTRAINT FK_OrderLine_Order FOREIGN KEY (order_id)
    REFERENCES order_ (order_id)
    ON DELETE CASCADE -- Si la commande est supprimée ses lignes de commande doivent l'etre
    ON UPDATE CASCADE,
  ADD CONSTRAINT FK_OrderLine_Article FOREIGN KEY (SKU_code)
    REFERENCES article (SKU_code)
    ON DELETE RESTRICT  -- Si l'article est supprimé, on garde la ligne de commande pour l'historique de l'ordre
    ON UPDATE CASCADE,
  ADD CONSTRAINT CHK_OrderLine_Quantity CHECK (quantity_ordered > 0);


-- TABLE: APPLICABLE_CATEGORY
ALTER TABLE applicable_category
  ADD CONSTRAINT PK_ApplicableCategory PRIMARY KEY (category_id, code_id),
  ADD CONSTRAINT FK_ApplicableCategory_Category FOREIGN KEY (category_id)
    REFERENCES category (category_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  ADD CONSTRAINT FK_ApplicableCategory_DiscountCode FOREIGN KEY (code_id)
    REFERENCES discount_code (code_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE;
```

```
-- TABLE: RETURNED_ITEM

ALTER TABLE returned_item

  ADD CONSTRAINT PK_ReturnedItem PRIMARY KEY (SKU_code, return_id),

  ADD CONSTRAINT FK_ReturnedItem_Return FOREIGN KEY (return_id)

    REFERENCES return_ (return_id)

    ON DELETE CASCADE -- Si le retour est supprimé le détail des articles retournés est supprimé.

    ON UPDATE CASCADE,


  ADD CONSTRAINT FK_ReturnedItem_Article FOREIGN KEY (SKU_code)

    REFERENCES article (SKU_code)

    ON DELETE RESTRICT -- mais sii l'article est supprimé on garde la trace dans l'historique de retour

    ON UPDATE CASCADE,

  ADD CONSTRAINT CHK_ReturnedItem_Quantity CHECK (quantity_returned > 0);
```

Quantities and Requirements

Generate **a large, coherent dataset** that makes the database realistic and usable for real-world analytical queries.

Minimum quantities:

- 10 - 15 countries, each with multiple postal codes (3-6) and cities
- 200+ “location” entries
- 100+ clients (anime or fictional names, e.g., “Frieren”, “Kazuma”, “Aqua”, “Toru Kobayashi”)
- 200+ “client_address” entries, each linked to valid “location”s and “client”s
- 15-20 “brand”s (funny or creative, inspired by real ones like “Adibas”, “Appel”, “Loui Vuittonne”)
- 10 categories with possible subcategories
- 100+ products with clear descriptions, linked to valid brands and categories
- 200+ “article”s (different colors/sizes) per product family
- 5 warehouses with valid “location”s

- Full “stock” coverage for articles across warehouses

- 150+ “order_” entries, each linked to existing clients and addresses

- 400+ “order_line”s with realistic quantities and prices

- 100+ “transaction”s (one per order, coherent with total amounts)

- 20+ “return_” entries with linked “returned_item”s

- 50+ “review”s with random ratings 1-5, some “APPROVED”, some “PENDING”

- 30+ “support_ticket”s per clients with coherent “ticket_history” messages

- 10+ “discount_code”s and corresponding “applicable_category” mappings

- Several “comm_preference_type”s (Email, SMS, Push, etc.) with random “client_preference”s assigned

Coherence Rules

Ensure referential and logical integrity throughout:

- Every `client` referenced in `order_`, `support_ticket`, `review`, or `client_preference` must exist.

- Every address and location must reference valid postal codes and countries.

- `transaction` and `return_` must reference existing `order_`.

- `price` history should show different valid date ranges and sale prices for articles.

- Stocks should not be negative and distributed across warehouses.

- `discount_code` must have valid `start_date < end_date`.

- `review.rating` must be between 1 and 5, `review.status` in (‘PENDING’, ‘APPROVED’, ‘REJECTED’).

- `support_ticket.status` must respect (‘OPEN’, ‘IN_PROGRESS’, ‘CLOSED’, ‘RESOLVED’).

- All Boolean columns (like `is_default` and `fidelity_status`) must be 0 or 1 (or TRUE/FALSE).

- All `CHECK` constraints must be satisfied.

Creativity

- Use fictional / pop culture / anime names for clients.

- Funny or parody brand names: “Nikéa”, “Guccii”, “Cocacolaine”, “Microsooft”, “Adibas”.

- Product names can mix humor and realism (e.g., “ManaBoost Energy Drink”, “Slime-Proof Jacket”).

- Reviews and support messages can include short French or English text (realistic sentences).
- Keep linguistic coherence (clients in France mostly have French addresses, others global).

Output Format

Generate pure SQL code only.