DnA Project Phase 1

Team 25

Team Members

- Amol Vijayachandran - 2023101093
- Ananth Rajesh - 2023101030
- Atharv Sanjaykumar Bhatt - 2023101089
- Siddharth Mago - 2023101088

Online Marketplace Database

1 Mini-World Description

The Online Marketplace Inventory Management System is designed to serve as a comprehensive platform that connects sellers with customers while managing product listings, inventory, orders, and customer feedback. The system caters to various users, including regular sellers, premium sellers, customers, and marketplace administrators.

The primary purpose of this database is to:

- Enable efficient product listing and inventory management
- Facilitate order processing and tracking
- Manage customer relationships and feedback
- Support promotional activities and discount programs
- Provide analytics for business decision-making

User Types and Their Interactions

- **Sellers (including Premium Sellers):**
  - Manage product listings and inventory
  - Process orders
  - View sales analytics

- Participate in promotional campaigns
- Interact with customer reviews
- Refer other sellers to the platform

- **Customers:**
  - Browse and search products
  - Place and track orders
  - Write product reviews
  - Participate in promotions
  - Maintain their profile and order history
- **Administrators:**
  - Manage seller accounts
  - Overview marketplace performance
  - Monitor transaction patterns
  - Manage promotional campaigns
  - Handle platform-wide inventory analytics

2 Database Requirements

2.1 Entity Types and Attributes

1. **Seller**
   - Primary Key: `seller_id` (string, format: S_XXXXX)
   - Attributes:
     - `name` (string, required)
     - `address` (composite: street, city, state, zip)
     - `join_date` (date, required)
     - `rating` (float, range: 0.0-5.0)
     - `referral_id` (foreign key to Seller)
2. **Premium_Seller** (subclass of Seller)
   - Primary Key: `seller_id` (inherited)
   - Attributes:
     - `premium_since` (date, required)
     - `tier` (string, domain: {'SILVER', 'GOLD', 'PLATINUM'})
     - `commission_rate` (float, range: 0.01-0.10)
3. **Product**

- Primary Keys: `product_id` (string, format: P_XXXXX) and `SKU` (string)
- Attributes:
  - `name` (string, required)
  - `price` (float, > 0)
  - `stock` (integer, ≥ 0)
  - `images` (multi-valued)
  - `avg_rating` (derived from Product_Review)
  - `seller_id` (foreign key to Seller)
  - `category_id` (foreign key to Category)

4. **Category**
   - Primary Key: `category_id` (string, format: C_XXXXX)
   - Attributes:
     - `name` (string, required)
     - `parent_category_id` (self-referential foreign key)

5. **Customer**
   - Primary Key: `customer_id` (string, format: CUST_XXXXX)
   - Attributes:
     - `name` (string, required)
     - `email` (string, unique, required)
     - `address` (composite: street, city, state, zip)
     - `join_date` (date, required)

2.2 Weak Entity Types

1. **Order_Item** (weak to Order)
   - Partial Key: `item_number` (integer)
   - Identifying Owner: `order_id`
   - Attributes:
     - `product_id` (foreign key to Product)
     - `quantity` (integer, > 0)
     - `unit_price` (float, > 0)

2. **Product_Review** (weak to Customer and Product)
   - Partial Key: `review_id`
   - Identifying Owners: `customer_id`, `product_id`
   - Attributes:
     - `rating` (integer, 1-5)
     - `comment` (string)
     - `review_date` (date)

2.3 Relationship Types and Constraints

1. **Sells** (Seller to Product)
   - Cardinality: 1:M
   - Participation: Total participation from Product
2. **Belongs_To** (Product to Category)
   - Cardinality: M:N
   - Constraint: Each product must belong to at least one category
3. **Places** (Customer to Order)
   - Cardinality: 1:M
   - Participation: Partial for Customer, Total for Order
4. **Contains** (Order to Order_Item)
   - Cardinality: 1:M
   - Participation: Total participation from Order_Item
5. **Refers** (Seller to Seller)
   - Cardinality: 1:M
   - Same entity type in distinct roles (referrer and referee)

2.4 N-ary Relationship

Promotion (between Seller, Product, Customer, and Discount_Code)

- Constraints:
  - Each promotion must involve at least one seller
  - Each promotion must include at least one product
  - Promotions can have multiple discount codes
  - Customers can participate in multiple promotions

3 Functional Requirements

3.1 Retrieval Operations

- **Selection Queries**
  - Find all products with stock below the threshold
  - List premium sellers with ratings above 4.5
  - Get all orders in a specific status

  - Find products in a price range

  Selection Query Examples

  ```
  -- Get all premium sellers with rating > 4.5
  SELECT s.*, ps.*
  FROM Seller s
  JOIN Premium_Seller ps ON s.seller_id = ps.seller_id
  WHERE s.rating > 4.5;
  ```

- **Projection Queries**
  - Get product names and their current stock levels
  - List seller names and their ratings
  - View customer names and their total order counts
  - Display category names and product counts

  Projection Query Examples

  ```
  -- Get product names and their average ratings
  SELECT p.name, AVG(pr.rating) as avg_rating
  FROM Product p
  LEFT JOIN Product_Review pr ON p.product_id = pr.product_id
  GROUP BY p.name;
  ```

- **Aggregate Functions**
  - Calculate the average rating per seller
  - Find total sales volume per category
  - Determine maximum order value per customer
  - Compute average order processing time

  Aggregate Function Examples

  ```
  -- Get total sales by seller
  SELECT s.name, SUM(oi.quantity * oi.unit_price) as total_sales
  FROM Seller s
  JOIN Product p ON s.seller_id = p.seller_id
  JOIN Order_Item oi ON p.product_id = oi.product_id
  GROUP BY s.seller_id;
  ```

- **Search Operations**

    - Search products by name or description
    - Find sellers by location
    - Search orders by date range
    - Look up customers by email pattern

  Search Function Example

```
-- Search products by partial name match
SELECT * FROM Product
WHERE name LIKE '%search_term%';
```

3.2 Analysis Reports

- **Seller Performance Analysis**
    - Sales volume over time
    - Rating trends
    - Order fulfillment rates
    - Category-wise performance
    - Cross-reference with premium status

```
--  Seller performance analysis
SELECT s.name,
       COUNT(DISTINCT o.order_id) as total_orders,
       AVG(pr.rating) as avg_rating,
       SUM(oi.quantity * oi.unit_price) as total_revenue
FROM Seller s
JOIN Product p ON s.seller_id = p.seller_id
LEFT JOIN Product_Review pr ON p.product_id = pr.product_id
JOIN Order_Item oi ON p.product_id = oi.product_id
JOIN Order o ON oi.order_id = o.order_id
GROUP BY s.seller_id;
```

- **Customer Behavior Analysis**
    - Purchase patterns
    - Category preferences
    - Review participation
    - Promotion engagement
    - Order value trends

```
-- Customer purchase patterns across categories
SELECT c.name, cat.name as category, COUNT(o.order_id)
as order_count
FROM Customer c
JOIN Order o ON c.customer_id = o.customer_id
JOIN Order_Item oi ON o.order_id = oi.order_id
JOIN Product p ON oi.product_id = p.product_id
JOIN Category cat ON p.category_id = cat.category_id
GROUP BY c.customer_id, cat.category_id;
```

3.3 Modification Operations

- **Insert Operations**
  - Add new product with constraint checking:
    * Valid `seller_id`
    * Unique SKU
    * Valid category
    * Non-negative price and stock

  Insert Operation Query Example

```
-- Insert Operation with Constraints
INSERT INTO Product (product_id, SKU, name, price, stock,
seller_id, category_id)
VALUES ('P123', 'SKU123', 'New Product', 99.99, 100, 'S1', 'C1')
WHERE EXISTS (SELECT 1 FROM Seller WHERE seller_id = 'S1')
AND EXISTS (SELECT 1 FROM Category WHERE category_id = 'C1');
```

- **Update Operations**
  - Modify product stock levels
  - Update order status
  - Change seller ratings
  - Adjust promotion details

  Update Operation Query Example

```
-- Update Operation
UPDATE Product
SET stock = stock - :quantity
WHERE product_id = :product_id
AND stock >= :quantity;
```

- **Delete Operations**
  - Remove discontinued products
  - Delete canceled orders
  - Remove expired promotions
  - Delete invalid reviews

  Delete Operation Query Example

  ```
  -- Delete Operation
  DELETE FROM Product_Review
  WHERE review_id = :review_id
  AND customer_id = :customer_id;
  ```

4 Data Constraints and Business Rules

Inventory Management

- Stock cannot be negative
- Low stock alerts at configurable thresholds
- Stock updates must be atomic

Order Processing

- Orders must have at least one item
- Order total must match item totals
- Status changes must follow valid transitions

User Management

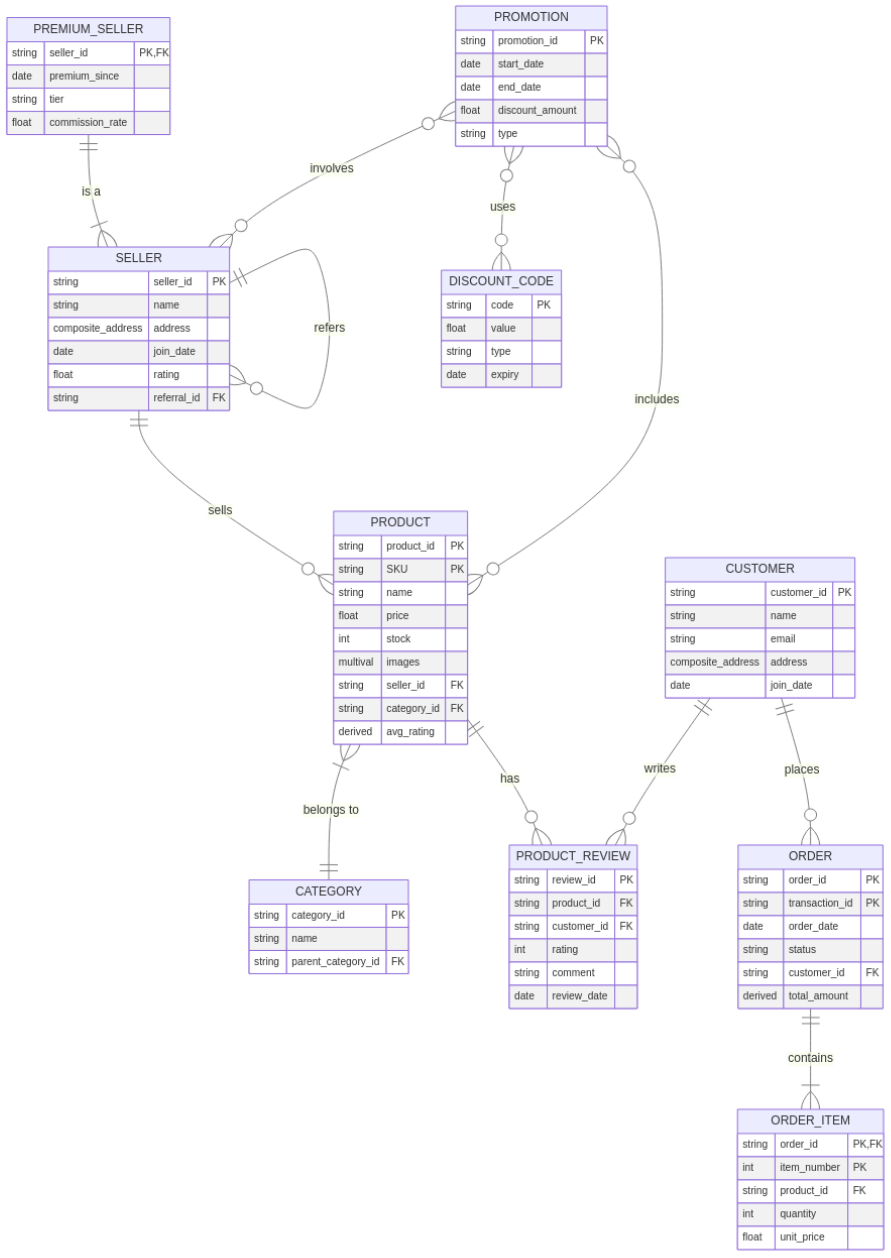
- Unique email addresses for customers
- Valid rating ranges (0-5)
- Premium seller qualifications

Review System

- One review per product per customer
- Reviews only allowed for purchased products
- Rating must be 1-5

These requirements form the foundation for implementing a robust and scalable marketplace system to manage inventory effectively, process orders, and facilitate seller-customer interactions.

5 ER Diagram



PREMIUM_SELLER

string	seller_id	PK,FK
date	premium_since	
string	tier	
float	commission_rate	

PROMOTION

string	promotion_id	PK
date	start_date	
date	end_date	
float	discount_amount	
string	type	

SELLER

string	seller_id	PK
string	name	
composite_address	address	
date	join_date	
float	rating	
string	referral_id	FK

DISCOUNT_CODE

string	code	PK
float	value	
string	type	
date	expiry	

PRODUCT

string	product_id	PK
string	SKU	PK
string	name	
float	price	
int	stock	
multival	images	
string	seller_id	FK
string	category_id	FK
derived	avg_rating	

CUSTOMER

string	customer_id	PK
string	name	
string	email	
composite_address	address	
date	join_date	

CATEGORY

string	category_id	PK
string	name	
string	parent_category_id	FK

PRODUCT_REVIEW

string	review_id	PK
string	product_id	FK
string	customer_id	FK
int	rating	
string	comment	
date	review_date	

ORDER

string	order_id	PK
string	transaction_id	PK
date	order_date	
string	status	
string	customer_id	FK
derived	total_amount	

ORDER_ITEM

string	order_id	PK,FK
int	item_number	PK
string	product_id	FK
int	quantity	
float	unit_price	