Watch Marketplace Web Application Database Schema Documentation

Database design sector

January 3, 2026

Contents

1 Introduction

This document provides comprehensive documentation for the database schema of the **Watch Marketplace Web Application**. The application is a multi-actor e-commerce platform where:

- **Customers** can browse, favorite, and purchase watches
- **Sellers** can list and manage their watch inventory
- **Administrators** have full oversight and management capabilities

1.1 Technical Stack

- **RDBMS:** PostgreSQL
- **Framework:** Django
- **Normalization:** Third Normal Form (3NF)
- **Image Storage:** URL-based (external storage)

1.2 Key Features

- Multi-role user management (Customer, Seller, Admin)
- Guest browsing and purchasing capability
- Real-time chat between customers and sellers
- Product approval workflow
- Payment processing with multiple methods
- Review and rating system with image uploads
- Comprehensive admin analytics and insights

2 Core User Management

2.1 users

Purpose: Central table for all user authentication and basic profile information.

Attribute	Type	Description
user_id	SERIAL (PK)	Auto-incrementing unique identifier for each user
email	VARCHAR(255)	User's email address (must be unique, used for login)
password_hash	VARCHAR(255)	Encrypted password hash for authentication
first_name	VARCHAR(100)	User's first name
last_name	VARCHAR(100)	User's last name
contact_number	VARCHAR(20)	Phone number for contact purposes
role	VARCHAR(20)	User's role: CUSTOMER, SELLER, or ADMIN
is_active	BOOLEAN	Account activation status (default: TRUE)
is_verified	BOOLEAN	Email verification status (default: FALSE)
created_at	TIMESTAMP	Account creation timestamp
updated_at	TIMESTAMP	Last profile update timestamp
last_login	TIMESTAMP	Most recent login timestamp

2.2 customers

Purpose: Extended profile information specific to customer accounts.

Attribute	Type	Description
customer_id	SERIAL (PK)	Unique identifier for customer
user_id	INTEGER (FK)	References users table (one-to-one relationship)
is_guest	BOOLEAN	Indicates if this is a guest checkout account
created_at	TIMESTAMP	Customer account creation timestamp

2.3 sellers

Purpose: Extended profile information and verification details for seller accounts.

Attribute	Type	Description
seller_id	SERIAL (PK)	Unique identifier for seller
user_id	INTEGER (FK)	References users table (one-to-one relationship)
cnic	VARCHAR(15)	National ID card number (unique, required for verification)
verification_status	VARCHAR(20)	Status: PENDING, VERIFIED, or REJECTED
verification_date	TIMESTAMP	Date when seller was verified by admin
created_at	TIMESTAMP	Seller account creation timestamp

2.4 administrators

Purpose: Extended profile information for admin accounts with elevated privileges.

Attribute	Type	Description
admin_id	SERIAL (PK)	Unique identifier for administrator
user_id	INTEGER (FK)	References users table (one-to-one relationship)
access_level	VARCHAR(20)	Admin privilege level: STANDARD or SUPER
created_at	TIMESTAMP	Admin account creation timestamp

3 Address Management

3.1 addresses

Purpose: Store multiple shipping and billing addresses for users. Supports both customer purchases and seller store locations.

Attribute	Type	Description
address_id	SERIAL (PK)	Unique identifier for each address
user_id	INTEGER (FK)	References users table (allows multiple addresses per user)
address_type	VARCHAR(20)	Type: SHIPPING, BILLING, or BOTH
address_line1	VARCHAR(255)	Primary address line (street, building)
address_line2	VARCHAR(255)	Secondary address line (apartment, suite)
city	VARCHAR(100)	City name
state_province	VARCHAR(100)	State or province name
postal_code	VARCHAR(20)	ZIP or postal code
country	VARCHAR(100)	Country (default: Pakistan)
is_default	BOOLEAN	Marks the default address for quick selection
created_at	TIMESTAMP	Address creation timestamp
updated_at	TIMESTAMP	Last address modification timestamp

Note: Users can have multiple addresses. One can be marked as default for convenience.

4 Store Management

4.1 stores

Purpose: Seller's storefront with branding, metrics, and contact information. Each seller has one store.

Attribute	Type	Description
store_id	SERIAL (PK)	Unique identifier for store
seller_id	INTEGER (FK)	References sellers table (one-to-one relationship)
store_name	VARCHAR(255)	Display name of the store (must be unique)
store_slug	VARCHAR(255)	URL-friendly version of store name (for SEO)
store_bio	TEXT	Store description and seller introduction
store_logo_url	VARCHAR(500)	URL to store logo image
store_banner_url	VARCHAR(500)	URL to store banner/header image
store_contact	VARCHAR(20)	Store-specific contact number
store_email	VARCHAR(255)	Store-specific email address
store_rating	DECIMAL(3,2)	Average rating (0.00 to 5.00)
total_reviews	INTEGER	Count of reviews received
total_sales	DECIMAL(12,2)	Cumulative sales revenue
total_orders	INTEGER	Count of completed orders
is_active	BOOLEAN	Store operational status
created_at	TIMESTAMP	Store creation timestamp
updated_at	TIMESTAMP	Last store update timestamp

Business Logic: Metrics (ratings, sales, orders) are automatically updated through triggers or application logic when orders are completed or reviews are submitted.

5 Product Catalog

5.1 product_categories

Purpose: Hierarchical categorization system for organizing watches by type.

Attribute	Type	Description
category_id	SERIAL (PK)	Unique identifier for category
category_name	VARCHAR(100)	Display name (e.g., "Luxury Watches")
category_description	TEXT	Detailed category description
parent_category_id	INTEGER (FK)	Self-referencing for subcategories (NULL for top-level)
is_active	BOOLEAN	Category visibility status

5.2 brands

Purpose: Normalized table for watch brand information to avoid duplication.

Attribute	Type	Description
brand_id	SERIAL (PK)	Unique identifier for brand
brand_name	VARCHAR(100)	Brand name (e.g., "Rolex", "Omega")
brand_description	TEXT	Brand history and information
brand_logo_url	VARCHAR(500)	URL to brand logo image
country_of_origin	VARCHAR(100)	Brand's country of origin

5.3 products

Purpose: Core product catalog containing all watch listings with detailed specifications and approval workflow.

Attribute	Type	Description
Identification		
product_id	SERIAL (PK)	Unique identifier for product
seller_id	INTEGER (FK)	References sellers table
store_id	INTEGER (FK)	References stores table
brand_id	INTEGER (FK)	References brands table
category_id	INTEGER (FK)	References product_categories table
Basic Information		
model_name	VARCHAR(255)	Watch model name
reference_number	VARCHAR(100)	Manufacturer's reference number
year_manufactured	INTEGER	Year the watch was made (1800 to current year + 1)
Condition & Authentication		
condition	VARCHAR(20)	NEW, LIKE_NEW, EXCELLENT, GOOD, FAIR, PARTS_ONLY
has_box	BOOLEAN	Whether original box is included
has_papers	BOOLEAN	Whether original papers/certificates included
has_warranty	BOOLEAN	Whether warranty is available
warranty_months	INTEGER	Duration of warranty in months

pieces	INTEGER	The number of available pieces this particular model has left. Set status to sold out when this becomes zero.
Pricing		
price	DECIMAL(12,2)	Current selling price
original_price	DECIMAL(12,2)	Original retail price (for discount calculation)
currency	VARCHAR(3)	Currency code (default: PKR)
Status Management		
status	VARCHAR(20)	DRAFT, ACTIVE, SOLD, RESERVED, INACTIVE
approval_status	VARCHAR(20)	PENDING, APPROVED, REJECTED, REQUIRES_CHANGES
approved_by	INTEGER (FK)	References administrators table
approved_at	TIMESTAMP	Timestamp of approval
rejection_reason	TEXT	Admin's reason for rejection (if applicable)
Technical Specifications		
specifications	JSONB	Flexible JSON field for detailed specs
description	TEXT	Detailed product description
case_material	VARCHAR(100)	Material (e.g., "Stainless Steel", "Gold")
movement_type	VARCHAR(100)	Type (e.g., "Automatic", "Quartz")
case_diameter_mm	DECIMAL(5,2)	Watch case diameter in millimeters
water_resistance	VARCHAR(50)	Water resistance rating (e.g., "100m")
Engagement Metrics		
view_count	INTEGER	Number of times product was viewed
favorite_count	INTEGER	Number of times added to favorites
Timestamps		
created_at	TIMESTAMP	Product listing creation timestamp
updated_at	TIMESTAMP	Last modification timestamp

Workflow: Sellers create products in DRAFT status. Once submitted, they go to PENDING approval. Admins can APPROVE or REJECT listings. Only APPROVED products with ACTIVE status are visible to customers.

6 Image Management

6.1 images

Purpose: Central repository for all image URLs used throughout the application. Tracks uploader and provides thumbnail support.

Attribute	Type	Description
image_id	SERIAL (PK)	Unique identifier for image
image_url	VARCHAR(500)	Full-size image URL (stored externally)
thumbnail_url	VARCHAR(500)	Thumbnail version URL for faster loading
alt_text	VARCHAR(255)	Alternative text for accessibility
image_order	INTEGER	Default ordering preference
uploaded_by	INTEGER (FK)	References users table (tracks who uploaded)
created_at	TIMESTAMP	Upload timestamp

6.2 product_images

Purpose: Junction table linking products to their images with ordering and primary image designation.

Attribute	Type	Description
product_image_id	SERIAL (PK)	Unique identifier
product_id	INTEGER (FK)	References products table
image_id	INTEGER (FK)	References images table
is_primary	BOOLEAN	Marks the main product image for thumbnails
display_order	INTEGER	Order in which images should be displayed

Note: One product can have multiple images. One image must be marked as primary (is_primary = TRUE).

6.3 review_images

Purpose: Junction table linking customer-uploaded photos to their reviews.

Attribute	Type	Description
review_image_id	SERIAL (PK)	Unique identifier
review_id	INTEGER (FK)	References reviews table
image_id	INTEGER (FK)	References images table
display_order	INTEGER	Order for displaying review images

Image Upload Flow:

1. User uploads image to external storage (AWS S3, Cloudinary, etc.)
2. URL is returned and stored in `images` table
3. Junction table (`product_images` or `review_images`) links image to entity

7 Order Management

7.1 orders

Purpose: Master table for all customer orders containing status tracking, pricing, and delivery information.

Attribute	Type	Description
Identification		
order_id	SERIAL (PK)	Unique identifier for order
order_number	VARCHAR(50)	Human-readable order number (e.g., "ORD-2024-00001")
customer_id	INTEGER (FK)	References customers table
store_id	INTEGER (FK)	References stores table
Status		
order_status	VARCHAR(20)	PENDING, CONFIRMED, PROCESSING, SHIPPED, DELIVERED, CANCELLED, REFUNDED
Pricing Breakdown		
subtotal	DECIMAL(12,2)	Sum of all order items
advance_payment	BOOLEAN	if the product meets a certain price range (code logic check), advance payment is required from the customer
advance_amount	DECIMAL(12,2)	what is the advance amount if it is required (calculated from the subtotal)?
tax_amount	DECIMAL(12,2)	Applied tax amount
shipping_cost	DECIMAL(12,2)	Delivery charges
total_amount	DECIMAL(12,2)	Final amount (subtotal + tax + shipping)
currency	VARCHAR(3)	Currency code (default: PKR)
Addresses		
shipping_address_id	INTEGER (FK)	References addresses table
billing_address_id	INTEGER (FK)	References addresses table
Shipping & Tracking		
tracking_number	VARCHAR(100)	Courier tracking number
carrier	VARCHAR(100)	Shipping carrier name (e.g., "TCS", "Leopards")
expected_delivery_date	DATE	Estimated delivery date
actual_delivery_date	DATE	Actual delivery date (filled upon delivery)
Notes		
customer_notes	TEXT	Special instructions from customer
admin_notes	TEXT	Internal notes (not visible to customer)
Timestamps		
created_at	TIMESTAMP	Order creation timestamp
updated_at	TIMESTAMP	Last status update timestamp

7.2 order_items

Purpose: Line items within an order. Stores product snapshot at time of purchase to preserve historical data.

Attribute	Type	Description

order_item_id	SERIAL (PK)	Unique identifier for line item
order_id	INTEGER (FK)	References orders table
product_id	INTEGER (FK)	References products table
product_name	VARCHAR(255)	Product name at time of purchase (snapshot)
product_price	DECIMAL(12,2)	Price at time of purchase (snapshot)
quantity	INTEGER	Quantity ordered (typically 1 for unique watches)
subtotal	DECIMAL(12,2)	Line item total (price × quantity)
created_at	TIMESTAMP	Item added to order timestamp

Why Snapshot? If product is deleted or price changes, order history remains accurate.

8 Payment Management

8.1 payment_methods

Purpose: Reference table for available payment methods in the system.

Attribute	Type	Description
payment_method_id	SERIAL (PK)	Unique identifier
method_name	VARCHAR(50)	Name (e.g., "CREDIT_CARD", "CASH_ON_DELIVERY")
method_description	TEXT	Description of payment method
is_active	BOOLEAN	Whether method is currently available

Pre-populated values: CASH_ON_DELIVERY, CREDIT_CARD, BANK_TRANSFER, DIGITAL_WALLET

8.2 payments

Purpose: Transaction records for all payment attempts and completions linked to orders.

Attribute	Type	Description
payment_id	SERIAL (PK)	Unique identifier for payment
order_id	INTEGER (FK)	References orders table
payment_method_id	INTEGER (FK)	References payment_methods table
payment_tier	VARCHAR(20)	Service level: STANDARD, EXPRESS, or PREMIUM
payment_status	VARCHAR(20)	PENDING, PROCESSING, COMPLETED, FAILED, REFUNDED, CANCELLED
amount	DECIMAL(12,2)	Payment amount
currency	VARCHAR(3)	Currency code (default: PKR)
transaction_id	VARCHAR(255)	Gateway transaction ID (unique)
payment_gateway	VARCHAR(50)	Gateway name (e.g., "Stripe", "JazzCash")
gateway_response	JSONB	Full API response from payment gateway
payment_initiated_at	TIMESTAMP	When payment was initiated
payment_completed_at	TIMESTAMP	When payment was confirmed
created_at	TIMESTAMP	Record creation timestamp
updated_at	TIMESTAMP	Last update timestamp

Payment Flow:

1. Payment record created with PENDING status
2. Gateway processes payment
3. Status updated to COMPLETED or FAILED
4. Gateway response stored in JSONB for auditing

9 Reviews and Ratings

9.1 reviews

Purpose: Customer feedback and ratings for stores and products with moderation capabilities.

Attribute	Type	Description
review_id	SERIAL (PK)	Unique identifier for review
store_id	INTEGER (FK)	References stores table
customer_id	INTEGER (FK)	References customers table
order_id	INTEGER (FK)	References orders table (optional, for verification)
product_id	INTEGER (FK)	References products table (optional)
rating	DECIMAL(2,1)	Star rating from 1.0 to 5.0
title	VARCHAR(255)	Review headline
description	TEXT	Detailed review text
helpful_count	INTEGER	Number of "helpful" votes
not_helpful_count	INTEGER	Number of "not helpful" votes
is_verified_purchase	BOOLEAN	TRUE if linked to completed order
is_approved	BOOLEAN	Admin moderation status (default: TRUE)
is_featured	BOOLEAN	Marked as featured review by admin
created_at	TIMESTAMP	Review submission timestamp
updated_at	TIMESTAMP	Last edit timestamp

Business Rules:

- One customer can only review an order once (UNIQUE constraint on customer_id + order_id)
- Reviews linked to orders are automatically marked as verified purchases
- Store ratings are recalculated when reviews are added/updated

10 Chat and Messaging

10.1 chat_conversations

Purpose: Container for message threads between customers and sellers about specific products.

Attribute	Type	Description
conversation_id	SERIAL (PK)	Unique identifier for conversation
customer_id	INTEGER (FK)	References customers table
seller_id	INTEGER (FK)	References sellers table
product_id	INTEGER (FK)	References products table (optional)
status	VARCHAR(20)	ACTIVE, CLOSED, or ARCHIVED
last_message_at	TIMESTAMP	Timestamp of most recent message
created_at	TIMESTAMP	Conversation start timestamp
updated_at	TIMESTAMP	Last activity timestamp

Note: UNIQUE constraint on (customer_id, seller_id, product_id) ensures one conversation per product inquiry.

10.2 messages

Purpose: Individual messages within conversations with read receipt tracking.

Attribute	Type	Description
message_id	SERIAL (PK)	Unique identifier for message
conversation_id	INTEGER (FK)	References chat_conversations table
sender_id	INTEGER (FK)	References users table
message_text	TEXT	Message content
message_type	VARCHAR(20)	TEXT, IMAGE, or SYSTEM
attachment_url	VARCHAR(500)	URL for image attachments
is_read	BOOLEAN	Read status (default: FALSE)
read_at	TIMESTAMP	When message was read
created_at	TIMESTAMP	Message sent timestamp

Chat Flow:

1. Customer clicks "Contact Seller" on a product
2. System creates or retrieves conversation between customer and seller
3. Messages are added with sender_id tracking who sent each message
4. Read receipts updated when recipient views conversation

11 Additional Features

11.1 favorites

Purpose: Wishlist functionality allowing customers to save products for later.

Attribute	Type	Description
favorite_id	SERIAL (PK)	Unique identifier
customer_id	INTEGER (FK)	References customers table
product_id	INTEGER (FK)	References products table
created_at	TIMESTAMP	When product was favorited

Note: UNIQUE constraint on (customer_id, product_id) prevents duplicate favorites.

11.2 admin_activity_log

Purpose: Audit trail for all administrative actions for security and accountability.

Attribute	Type	Description
log_id	SERIAL (PK)	Unique identifier
admin_id	INTEGER (FK)	References administrators table
action_type	VARCHAR(50)	Type of action (e.g., "APPROVE_PRODUCT", "BAN_USER")
entity_type	VARCHAR(50)	Type of entity affected (e.g., "PRODUCT", "USER")
entity_id	INTEGER	ID of affected entity
description	TEXT	Detailed description of action
ip_address	INET	IP address of admin at time of action
user_agent	TEXT	Browser/device information
created_at	TIMESTAMP	Action timestamp

11.3 notifications

Purpose: System-wide notification delivery for user alerts and updates.

Attribute	Type	Description
notification_id	SERIAL (PK)	Unique identifier
user_id	INTEGER (FK)	References users table
notification_type	VARCHAR(50)	Type (e.g., "ORDER_SHIPPED", "NEW_MESSAGE")
title	VARCHAR(255)	Notification headline
message	TEXT	Detailed notification message
link_url	VARCHAR(500)	Optional link to related page
is_read	BOOLEAN	Read status (default: FALSE)
read_at	TIMESTAMP	When notification was read
created_at	TIMESTAMP	Notification creation timestamp

12 Entity Relationships Overview

12.1 One-to-One Relationships

- **users ↔ customers:** Each user can be a customer
- **users ↔ sellers:** Each user can be a seller
- **users ↔ administrators:** Each user can be an admin
- **sellers ↔ stores:** Each seller has one store

12.2 One-to-Many Relationships

- **users → addresses:** One user can have multiple addresses
- **stores → products:** One store can list many products
- **sellers → products:** One seller can create many products
- **customers → orders:** One customer can place many orders
- **stores → orders:** One store can receive many orders
- **orders → order_items:** One order contains many items
- **stores → reviews:** One store can receive many reviews
- **customers → reviews:** One customer can write many reviews
- **customers → chat_conversations:** One customer can have multiple conversations
- **sellers → chat_conversations:** One seller can have multiple conversations

12.3 Many-to-Many Relationships (via Junction Tables)

- **products ↔ images:** Via `product_images`
- **reviews ↔ images:** Via `review_images`
- **customers ↔ products:** Via `favorites` (wishlist)

13 Key Design Decisions

13.1 Normalization (3NF)

The schema is designed in Third Normal Form to:

- **Eliminate redundancy:** Brands, categories, and payment methods are in separate tables
- **Maintain data integrity:** Foreign key constraints prevent orphaned records
- **Enable flexible queries:** Normalized structure allows efficient JOINs
- **Reduce update anomalies:** Changing a brand name updates one record, not thousands

13.2 Snapshot Pattern

The `order_items` table stores product name and price at time of purchase rather than just referencing the product. This ensures:

- Historical accuracy if products are deleted
- Price history preservation when prices change
- Legal compliance for transaction records

13.3 Soft Deletes

Many tables include `is_active` flags rather than hard deletes:

- Preserves referential integrity
- Enables data recovery
- Maintains audit trails
- Allows "deactivation" without data loss

13.4 JSONB for Flexibility

Used in `specifications` and `gateway_response` fields:

- Accommodates varying product specifications
- Stores complex gateway responses without schema changes
- Queryable in PostgreSQL with JSON operators
- Maintains schema flexibility for future expansion

13.5 URL-Based Image Storage

Images stored as URLs pointing to external storage (S3, Cloudinary):

- Reduces database size
- Improves performance
- Enables CDN usage
- Simplifies backup and scaling
- Central `images` table prevents URL duplication

14 Indexes for Performance

Critical indexes have been defined on:

- Foreign key columns (customer_id, seller_id, product_id, etc.)
- Frequently filtered columns (status, approval_status, is_read)
- Search columns (store_name, email)
- Date/timestamp columns used in sorting (created_at, updated_at)
- Composite indexes for common query patterns

These indexes significantly improve:

- JOIN operation performance
- WHERE clause filtering
- ORDER BY sorting
- Search functionality

15 Security Considerations

15.1 Authentication & Authorization

- Passwords stored as hashes (never plain text)
- Role-based access control via `users.role`
- Admin privilege levels (STANDARD vs SUPER)
- Verification workflow for sellers

15.2 Data Integrity

- CHECK constraints on enums (status values, ratings)
- NOT NULL constraints on critical fields
- UNIQUE constraints prevent duplicates (emails, CNIC)
- Foreign key constraints with CASCADE/RESTRICT rules

15.3 Audit Trail

- `admin_activity_log` tracks all admin actions
- Timestamp fields (created_at, updated_at) on all tables
- IP address and user agent logging
- Payment gateway responses stored for reconciliation

16 Integration with Django

16.1 Django Compatibility Features

- **SERIAL:** Django's AutoField equivalent
- **Timestamps:** Compatible with auto_now and auto_now_add
- **Naming conventions:** Snake_case follows Django ORM standards
- **Foreign keys:** Align with Django's ForeignKey syntax

16.2 Suggested Django Models Structure

```
myapp/
 models/
    __init__.py
    user.py          # User, Customer, Seller, Admin
    store.py         # Store
    product.py       # Product, Category, Brand
    order.py         # Order, OrderItem
    payment.py       # Payment, PaymentMethod
    review.py        # Review
    chat.py          # ChatConversation, Message
    image.py         # Image, ProductImage, ReviewImage
    notification.py  # Notification, Favorite, etc.
```

16.3 Recommended Django Packages

- **django-rest-framework:** RESTful API development
- **django-filter:** Advanced filtering capabilities
- **django-storages:** S3/Cloudinary integration for images
- **celery:** Asynchronous task processing (emails, notifications)
- **channels:** WebSocket support for real-time chat
- **stripe/razorpay:** Payment gateway integration

17 Common Query Patterns

17.1 Customer Queries

Browse active, approved products with images:

```
SELECT p.*, pi.image_url
FROM products p
JOIN product_images pi ON p.product_id = pi.product_id
WHERE p.status = 'ACTIVE'
  AND p.approval_status = 'APPROVED'
  AND pi.is_primary = TRUE;
```

Get customer order history:

```
SELECT o.*, s.store_name, oi.product_name, oi.product_price
FROM orders o
JOIN stores s ON o.store_id = s.store_id
JOIN order_items oi ON o.order_id = oi.order_id
WHERE o.customer_id = ?
ORDER BY o.created_at DESC;
```

17.2 Seller Queries

Get store analytics:

```
SELECT
    s.total_sales,
    s.total_orders,
    s.store_rating,
    COUNT(DISTINCT p.product_id) as total_products,
    COUNT(DISTINCT CASE WHEN p.status = 'SOLD'
        THEN p.product_id END) as sold_products
FROM stores s
LEFT JOIN products p ON s.store_id = p.store_id
WHERE s.seller_id = ?
GROUP BY s.store_id;
```

17.3 Admin Queries

Products pending approval:

```
SELECT p.*, s.store_name, u.email as seller_email
FROM products p
JOIN stores s ON p.store_id = s.store_id
JOIN sellers sel ON s.seller_id = sel.seller_id
JOIN users u ON sel.user_id = u.user_id
WHERE p.approval_status = 'PENDING'
ORDER BY p.created_at ASC;
```

Revenue analytics:

```
SELECT
    DATE_TRUNC('month', o.created_at) as month,
    COUNT(*) as order_count,
```

```
    SUM(o.total_amount) as revenue
FROM orders o
WHERE o.order_status = 'DELIVERED'
GROUP BY month
ORDER BY month DESC;
```

18 Future Enhancements

18.1 Potential Additions

1. **Coupons & Discounts:** Promotional code system
2. **Shipping Providers:** Integration with multiple carriers
3. **Return/Refund Management:** RMA workflow tables
4. **Product Comparisons:** Side-by-side watch comparisons
5. **Auction System:** Bid functionality for rare watches
6. **Seller Subscriptions:** Tiered seller plans (basic, premium)
7. **Multi-language Support:** Internationalization tables
8. **Analytics Events:** User behavior tracking
9. **Email Templates:** Transactional email management
10. **API Keys:** Third-party integration management

18.2 Scalability Considerations

- Partition large tables (orders, messages) by date
- Implement read replicas for reporting queries
- Cache frequently accessed data (product listings, store info)
- Use materialized views for complex analytics
- Consider message queue for asynchronous processing

19 Appendix: Sample Data Flow

19.1 Customer Purchase Flow

1. Customer browses products (queries `products`, `product_images`)
2. Adds product to favorites (inserts into `favorites`)
3. Contacts seller (creates `chat_conversation`, adds `messages`)
4. Places order (creates `orders`, `order_items`)
5. Makes payment (creates `payments`, updates `payment_status`)
6. Order fulfilled (updates `order_status` to SHIPPED, DELIVERED)
7. Leaves review (creates `reviews`, optionally `review_images`)
8. Store rating recalculated (updates `stores.store_rating`)

19.2 Seller Onboarding Flow

1. User registers (creates `users` with role='SELLER')
2. Extended profile (creates `sellers` with CNIC)
3. Admin verification (updates `verification_status` to VERIFIED)
4. Creates store (creates `stores`)
5. Lists product (creates `products`, `product_images`)
6. Admin approves (updates `approval_status` to APPROVED)
7. Product goes live (updates `status` to ACTIVE)

19.3 Admin Moderation Flow

1. Reviews pending products (queries `products` WHERE approval_status='PENDING')
2. Inspects product details and images
3. Either:
   - Approves (updates `approval_status`, logs in `admin_activity_log`)
   - Rejects (sets `rejection_reason`, sends `notification`)
   - Requests changes (updates to REQUIRES_CHANGES status)
4. Monitors platform metrics (queries `stores`, `orders`, `reviews`)
5. Manages user accounts (updates `users.is_active`)

20 Conclusion

This database schema provides a solid foundation for the Watch Marketplace web application. It balances:

- **Normalization:** Reduces redundancy while maintaining query performance
- **Flexibility:** JSONB fields and extensible design for future needs
- **Security:** Role-based access, verification workflows, audit trails
- **Scalability:** Indexed appropriately for growth
- **Business Logic:** Captures complex workflows (approvals, payments, reviews)

The schema is production-ready and can be directly implemented in PostgreSQL with Django ORM models. All major user stories for customers, sellers, and administrators are supported.

For questions or clarifications, please contact the database design team.