SQL Queries Reference

Comprehensive raw SQL (MySQL 8.x style) for the actual schema (migrations) and the main runtime data operations inferred from controllers / Eloquent usage.

> NOTE: Adjust ENGINE / CHARSET / precision as needed for your RDBMS (defaulting to InnoDB + utf8mb4). Decimal scale kept from migrations.

1. Schema (DDL) (Domain Tables Only)

1.1 branches

```
CREATE TABLE branches (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  address TEXT NULL,
  contact_number VARCHAR(255) NULL,
  status VARCHAR(50) NOT NULL DEFAULT 'active',
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.2 categories

```
CREATE TABLE categories (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT NULL,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.3 suppliers

```
CREATE TABLE suppliers (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  phone VARCHAR(255) NULL,
  address VARCHAR(255) NULL,
  contract_start_date DATE NULL,
  contract_end_date DATE NULL,
  status VARCHAR(50) NOT NULL DEFAULT 'active',
  payment_terms VARCHAR(255) NULL,
```

```
  branch_id BIGINT UNSIGNED NULL,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  CONSTRAINT fk_suppliers_branch FOREIGN KEY (branch_id) REFERENCES
branches(id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.4 customers

```
CREATE TABLE customers (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  phone VARCHAR(255) NULL,
  address TEXT NULL,
  branch_id BIGINT UNSIGNED NOT NULL,
  loyalty_points INT NOT NULL DEFAULT 0,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  CONSTRAINT fk_customers_branch FOREIGN KEY (branch_id) REFERENCES
branches(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.5 products

```
CREATE TABLE products (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT NULL,
  price DECIMAL(10,2) NOT NULL,
  stock_quantity INT NOT NULL,
  is_active TINYINT(1) NOT NULL DEFAULT 1,
  supplier_id BIGINT UNSIGNED NOT NULL,
  category_id BIGINT UNSIGNED NOT NULL,
  branch_id BIGINT UNSIGNED NOT NULL,
  low_stock_notified TINYINT(1) NOT NULL DEFAULT 0,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  CONSTRAINT fk_products_supplier FOREIGN KEY (supplier_id) REFERENCES
suppliers(id) ON DELETE CASCADE,
  CONSTRAINT fk_products_category FOREIGN KEY (category_id) REFERENCES
categories(id) ON DELETE CASCADE,
  CONSTRAINT fk_products_branch FOREIGN KEY (branch_id) REFERENCES
branches(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.6 sales

```
CREATE TABLE sales (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  customer_id BIGINT UNSIGNED NULL,
  customer_name VARCHAR(255) NOT NULL,
  product_id BIGINT UNSIGNED NOT NULL,
  branch_id BIGINT UNSIGNED NOT NULL,
  tax DECIMAL(10,2) NOT NULL,
  quantity INT NOT NULL,
  discount DECIMAL(10,2) NOT NULL,
  total_amount DECIMAL(10,2) NOT NULL,
  status VARCHAR(50) NOT NULL DEFAULT 'pending',
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  CONSTRAINT fk_sales_customer FOREIGN KEY (customer_id) REFERENCES
customers(id) ON DELETE SET NULL,
  CONSTRAINT fk_sales_product FOREIGN KEY (product_id) REFERENCES
products(id) ON DELETE CASCADE,
  CONSTRAINT fk_sales_branch FOREIGN KEY (branch_id) REFERENCES
branches(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.7 invoices

```
CREATE TABLE invoices (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  invoice_number VARCHAR(255) NOT NULL UNIQUE,
  type ENUM('sale') NOT NULL, -- application now only supports 'sale'
  customer_id BIGINT UNSIGNED NULL,
  customer_name VARCHAR(255) NULL,
  due_date DATE NULL,
  total_amount DECIMAL(10,2) NOT NULL,
  tax_amount DECIMAL(10,2) NULL,
  discount DECIMAL(10,2) NULL,
  status ENUM('pending','paid','canceled') NOT NULL DEFAULT 'pending',
  notes TEXT NULL,
  branch VARCHAR(255) NULL,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  CONSTRAINT fk_invoices_customer FOREIGN KEY (customer_id) REFERENCES
customers(id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.8 invoice_sale (pivot)

```
CREATE TABLE invoice_sale (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  invoice_id BIGINT UNSIGNED NOT NULL,
  sale_id BIGINT UNSIGNED NOT NULL,
```

```
  line_total DECIMAL(12,2) NULL,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  UNIQUE KEY uq_invoice_sale (invoice_id, sale_id),
  CONSTRAINT fk_invoice_sale_invoice FOREIGN KEY (invoice_id) REFERENCES
invoices(id) ON DELETE CASCADE,
  CONSTRAINT fk_invoice_sale_sale FOREIGN KEY (sale_id) REFERENCES
sales(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.9 stock_notifications

```
CREATE TABLE stock_notifications (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  product_id BIGINT UNSIGNED NOT NULL,
  type VARCHAR(255) NOT NULL,
  message TEXT NOT NULL,
  is_read TINYINT(1) NOT NULL DEFAULT 0,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  CONSTRAINT fk_stock_notifications_product FOREIGN KEY (product_id)
REFERENCES products(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1.10 users

```
CREATE TABLE users (
  id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  role ENUM('1','2') NOT NULL DEFAULT '1', -- 1=cashier, 2=admin
  branch_id BIGINT UNSIGNED NULL,
  created_at TIMESTAMP NULL,
  updated_at TIMESTAMP NULL,
  CONSTRAINT fk_users_branch FOREIGN KEY (branch_id) REFERENCES
branches(id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

---

2. Core Data Manipulation Queries (DML) from Controllers (Domain Only)

Below each logical action shows the approximate SQL generated by Eloquent (simplified). Parameter placeholders use `?`.

2.1 Invoice Creation (store)

Eloquent (InvoiceController@store):

```
$invoice = Invoice::create([
  'invoice_number' => $invoiceNumber,
  'type' => $request->type, // 'sale'
  'branch' => $branchName,
  'customer_id' => $request->customer_id,
  'customer_name' => $customerName ?? $request->customer_name,
  'notes' => $request->notes,
  'due_date' => $request->due_date,
  'total_amount' => $request->total_amount,
  'tax_amount' => $request->tax_amount ?? 0,
  'discount' => $request->discount ?? 0,
  'status' => $request->status,
]);

if ($request->filled('sales')) {
    $syncData = collect($request->sales)
       ->mapWithKeys(fn ($id) => [$id => ['line_total' => null]])
       ->toArray();
    $invoice->sales()->sync($syncData);
}
```

```
-- Generate unique number (done in PHP). Insert invoice
INSERT INTO invoices (invoice_number, type, branch, customer_id,
customer_name, notes, due_date, total_amount, tax_amount, discount, status,
created_at, updated_at)
VALUES (?, 'sale', ?, ?, ?, ?, ?, ?, ?, ?, ?, NOW(), NOW());

-- Attach sales (pivot entries)
INSERT INTO invoice_sale (invoice_id, sale_id, line_total, created_at,
updated_at)
VALUES
  (?, ?, NULL, NOW(), NOW()),
  ... -- one row per selected sale
ON DUPLICATE KEY UPDATE line_total = VALUES(line_total);
```

2.2 Invoice Index (listing + filters)

Eloquent (InvoiceController@index core chain):

```
$query = Invoice::query()-
>with(['customer:id,name','sales:id,total_amount,status']);
if ($user->role === 1 && $branchName) { $query->where('branch',
$branchName); }
if ($status = $request->get('status')) { $query->where('status', $status);
}
if ($search = $request->get('q')) {
```

```
  $query->where(fn($q) => $q->where('invoice_number','like',"%$search%")
                            -
>orWhere('customer_name','like',"%$search%"));
}
$paginated = $query->latest()->paginate(10);
```

```
SELECT * FROM invoices
WHERE 1=1
  [AND branch = ?               -- if cashier role with branch]
  [AND status = ?               -- optional filter]
  [AND (invoice_number LIKE ? OR customer_name LIKE ?)]
ORDER BY invoices.created_at DESC
LIMIT 10 OFFSET ?;  -- pagination

-- Counts for summary
SELECT COUNT(*) FROM invoices WHERE status='pending';
SELECT COUNT(*) FROM invoices WHERE status='paid';
SELECT COUNT(*) FROM invoices WHERE status='canceled';
```

2.3 Invoice Show

Eloquent (InvoiceController@show):

```
$invoice =
Invoice::with(['customer:id,name','sales:id,total_amount,status'])
  ->findOrFail($id);
```

```
SELECT * FROM invoices WHERE id = ? LIMIT 1;  -- plus eager-loaded sales &
customer
SELECT s.id, s.total_amount, s.status
FROM sales s
JOIN invoice_sale isp ON isp.sale_id = s.id
WHERE isp.invoice_id = ?;
```

2.4 Invoice Update

Eloquent (InvoiceController@update):

```
$invoice->update($request-
>only(['status','discount','tax_amount','total_amount','notes']));
```

```
UPDATE invoices
SET status = ?, discount = ?, tax_amount = ?, total_amount = ?, notes = ?,
updated_at = NOW()
WHERE id = ?;
```

2.5 Invoice Delete

```
-- Pivot rows removed by ON DELETE CASCADE
DELETE FROM invoices WHERE id = ?;
```

2.6 Sales Index (filters & role scope)

Eloquent (SaleController@index excerpt):

```
$query = Sale::with(['customer','branch','product']);
if (! $user->isAdmin()) { $query->where('branch_id',$user->branch_id); }
// apply optional filters: status, product_id, customer name/id, date range
$sales = $query->latest()->paginate(15);
```

```
SELECT * FROM sales
WHERE 1=1
  [AND branch_id = ?               -- non-admin restriction]
  [AND status = ?]
  [AND product_id = ?]
  [AND (customer_name LIKE ? OR EXISTS(
        SELECT 1 FROM customers c WHERE c.id = sales.customer_id AND c.name
LIKE ?))]
  [AND (id = ? OR customer_name LIKE ?)]
  [AND DATE(created_at) >= ?]
  [AND DATE(created_at) <= ?]
ORDER BY sales.created_at DESC
LIMIT 15 OFFSET ?;  -- pagination
```

2.7 Sale Create (store)

Eloquent (SaleController@store):

```
$total = $request->price * $request->quantity;
$total_amount = $total - ($total * $discount/100) + ($total * $tax/100);
Sale::create([
  'customer_id' => $request->customer_id,
  'branch_id' => $branch_id,
  'product_id' => $request->product_id,
  'customer_name' => $customer_name,
```

```
    'tax' => $tax,
    'discount' => $discount,
    'total_amount' => $total_amount,
    'status' => $request->status,
    'quantity' => $request->quantity,
  ]);
```

```
INSERT INTO sales (customer_id, branch_id, product_id, customer_name, tax,
discount, total_amount, status, quantity, created_at, updated_at)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, NOW(), NOW());
```

2.8 Sale Update (status)

```
UPDATE sales SET status = ?, updated_at = NOW() WHERE id = ?;
```

2.9 Sale Delete

```
DELETE FROM sales WHERE id = ?;
```

2.10 Low Stock Notification Creation (Artisan command, inferred)

```
INSERT INTO stock_notifications (product_id, type, message, is_read,
created_at, updated_at)
VALUES (?, 'low_stock', ?, 0, NOW(), NOW());
```

2.11 Stock Notification Fetch (dashboard)

```
SELECT * FROM stock_notifications ORDER BY created_at DESC; -- (ordering
inferred, may be implicit)
```

2.12 Authentication (registration, login, password reset, remember token)

> These queries illustrate typical authentication flows. They assume (optionally) the presence of Laravel's default `password_reset_tokens` and `sessions` tables even if not listed in the domain schema above.

Registration (create user)

```

```

```
INSERT INTO users (name, email, password, role, branch_id, created_at,
updated_at)
VALUES (?, ?, ?, ?, ?, NOW(), NOW());
```

Lookup user by email for login (password hash verified in application layer)

```
SELECT id, password, role, branch_id FROM users WHERE email = ? LIMIT 1;
```

Optional: set / rotate remember token

```
UPDATE users SET remember_token = ?, updated_at = NOW() WHERE id = ?;
```

Create session row (if using database sessions)

```
INSERT INTO sessions (id, user_id, ip_address, user_agent, payload,
last_activity)
VALUES (?, ?, ?, ?, ?, ?);  -- last_activity = UNIX timestamp
```

Touch session on request

```
UPDATE sessions SET last_activity = ? WHERE id = ?;
```

Logout (delete session)

```
DELETE FROM sessions WHERE id = ?;  -- or WHERE user_id = ? to flush all
```

Initiate password reset (insert or replace token)

```
INSERT INTO password_reset_tokens (email, token, created_at)
VALUES (?, ?, NOW())
ON DUPLICATE KEY UPDATE token = VALUES(token), created_at =
VALUES(created_at);
```

Validate password reset token

```
SELECT email FROM password_reset_tokens WHERE email = ? AND token = ? LIMIT
1;
```

Complete password reset (update hash & remove token)

```
UPDATE users SET password = ?, updated_at = NOW() WHERE email = ?;
DELETE FROM password_reset_tokens WHERE email = ?;
```

(Optional) Invalidate all sessions for a user after password change

```
DELETE FROM sessions WHERE user_id = ?;  -- if enforcing global logout
```

2.13 Product Index (listing with relations)

Eloquent (ProductController@index):

```
Product::with(['category','supplier','branch'])->paginate(10);
```

```
SELECT p.*
FROM products p
LEFT JOIN categories c ON c.id = p.category_id
LEFT JOIN suppliers s ON s.id = p.supplier_id
LEFT JOIN branches b ON b.id = p.branch_id
ORDER BY p.id DESC
LIMIT 10 OFFSET ?; -- pagination (Page * 10)
```

2.14 Product Store (create)

Eloquent (ProductController@store):

```
Product::create($request->all());
```

```
INSERT INTO products (name, price, category_id, stock_quantity,
description, supplier_id, branch_id, is_active, created_at, updated_at)
VALUES (?, ?, ?, ?, ?, ?, ?, COALESCE(?, 1), NOW(), NOW());
```

2.15 Product Delete

Eloquent (ProductController@destroy):

```
Product::destroy($id);
```

```
DELETE FROM products WHERE id = ?;
```

2.16 Product Search (name / description / category.name / supplier.name)

Eloquent (ProductController@search core):

```
$q = Product::with(['category','supplier','branch']);
if ($queryString) {
  $q->where(fn($q2) => $q2->where('name','like',"%$queryString%")
     ->orWhere('description','like',"%$queryString%")
     ->orWhereHas('category', fn($c)=>$c-
>where('name','like',"%$queryString%"))
     ->orWhereHas('supplier', fn($s)=>$s-
>where('name','like',"%$queryString%"))
  );
}
$q->paginate(10);
```

```
SELECT p.*
FROM products p
LEFT JOIN categories c ON c.id = p.category_id
LEFT JOIN suppliers s ON s.id = p.supplier_id
LEFT JOIN branches b ON b.id = p.branch_id
WHERE (
      p.name LIKE ?
   OR p.description LIKE ?
   OR c.name LIKE ?
   OR s.name LIKE ?
)
ORDER BY p.id DESC
LIMIT 10 OFFSET ?; -- pagination
```

2.17 Product Index (Branch-scoped POS view)

Eloquent (ProductController@indexBranch):

```
Product::with(['category','supplier','branch'])
  ->where('branch_id', $branchId)
  ->paginate(10);
```

```
SELECT p.*
FROM products p
```

```
LEFT JOIN categories c ON c.id = p.category_id
LEFT JOIN suppliers s ON s.id = p.supplier_id
WHERE p.branch_id = ?
ORDER BY p.id DESC
LIMIT 10 OFFSET ?;
```

2.18 Low Stock Products (threshold, active only)

Eloquent (ProductController@getLowStock):

```
Product::with(['category','supplier','branch'])
  ->where('stock_quantity','<=',$threshold)
  ->where('is_active', true)
  ->get();
```

```
SELECT p.*
FROM products p
LEFT JOIN categories c ON c.id = p.category_id
LEFT JOIN suppliers s ON s.id = p.supplier_id
LEFT JOIN branches b ON b.id = p.branch_id
WHERE p.stock_quantity <= ?
  AND p.is_active = 1;
```

> NOTE: Product search uses multiple OR predicates; consider composite/covering indexes or FULLTEXT (MySQL 8+) on (name, description). For high cardinality supplier/category filters, separate WHERE clauses may improve selectivity before OR expansions.

2.19 Dashboard Index (aggregate eager loads)

Eloquent (DashboardController@index excerpts):

```
if (Cache::add('low-stock-scan-lock', true, 20)) {
  Artisan::queue('stock:check-low');
}
$sales = Sale::with(['product','customer','branch'])->get();
$products = Product::with(['category','supplier','branch'])->get();
$customers = Customer::all();
$branches = Branch::all();
$suppliers = Supplier::all();
$stockNotifications = StockNotification::all();
$users = User::all();
```

```
-- Background (non-blocking) low stock scan trigger: (Artisan queued
command)
```

```
-- INSERT INTO jobs (...) VALUES (... 'stock:check-low' ...);  -- internal
to Laravel queue

-- Sales with relations
SELECT s.* FROM sales s ORDER BY s.id DESC; -- plus separate SELECT for
related product/customer/branch due to eager loading

-- Products with relations
SELECT p.* FROM products p ORDER BY p.id DESC; -- categories, suppliers,
branches loaded in separate queries

-- Customers
SELECT * FROM customers ORDER BY id DESC;

-- Branches
SELECT * FROM branches ORDER BY id DESC;

-- Suppliers
SELECT * FROM suppliers ORDER BY id DESC;

-- Stock Notifications
SELECT * FROM stock_notifications ORDER BY id DESC;

-- Users
SELECT * FROM users ORDER BY id DESC;
```

> NOTE: Dashboard loads entire tables (no pagination). Consider: counts via `SELECT COUNT(*)`, recent N records, or caching if dataset grows. Eager loading issues: many separate SELECTs; could aggregate counts instead for performance.

2.20 Customer Index (with counts & filters)

Eloquent (CustomerController@index excerpt):

```
$q = Customer::withCount(['sales','invoices'])->with('branch');
if (! $user->isAdmin()) { $q->where('branch_id',$user->branch_id); }
if ($branch = $request->get('branch')) { $q->where('branch_id',$branch); }
if ($search = $request->get('q')) {
  $q->where(fn($qq)=>$qq->where('name','like',"%$search%")
      ->orWhere('email','like',"%$search%")
      ->orWhere('phone','like',"%$search%"));
}
if ($minSales = $request->get('min_sales')) { $q-
>having('sales_count','>=',$minSales); }
if ($minInv = $request->get('min_invoices')) { $q-
>having('invoices_count','>=',$minInv); }
$customers = $q->orderByDesc('id')->paginate(15);
```

```
SELECT c.*, COUNT(s.id) AS sales_count, COUNT(DISTINCT i.id) AS
invoices_count
FROM customers c
LEFT JOIN sales s ON s.customer_id = c.id
LEFT JOIN invoices i ON i.customer_id = c.id
WHERE 1=1
  [AND c.branch_id = ?]            -- branch filter / role restriction
  [AND (c.name LIKE ? OR c.email LIKE ? OR c.phone LIKE ?)]
GROUP BY c.id
  [HAVING sales_count >= ?]
  [HAVING invoices_count >= ?]
ORDER BY c.id DESC
LIMIT 15 OFFSET ?;
```

2.21 Customer Store

Eloquent:

```
Customer::create([
  'name'=>$r->name,'email'=>$r->email,'phone'=>$r->phone,
  'address'=>$r->address,'loyalty_points'=>$r->loyalty_points ?? 0,
  'branch_id'=>$user->branch_id,
]);
```

```
INSERT INTO customers
(name,email,phone,address,loyalty_points,branch_id,created_at,updated_at)
VALUES (?,?,?,?,?,?,NOW(),NOW());
```

2.22 Customer Update

Eloquent: `$customer->update($request->only('name','email','phone','address','loyalty_points'));`

```
UPDATE customers SET name=?, email=?, phone=?, address=?, loyalty_points=?,
updated_at=NOW()
WHERE id=?;
```

2.23 Customer Delete (with guard)

Eloquent:

```
if ($customer->sales()->count()>0 || $customer->invoices()->count()>0) { /*
block */ }
```

```
$customer->delete();
```

```
DELETE FROM customers WHERE id = ?; -- only executed if no related
sales/invoices
```

2.24 Customer Show

Eloquent: `Customer::with(['sales','invoices'])->findOrFail($id);`

```
SELECT * FROM customers WHERE id=? LIMIT 1;
SELECT * FROM sales WHERE customer_id=?;
SELECT * FROM invoices WHERE customer_id=?;
```

2.25 Supplier Index (filters)

Eloquent (SupplierController@index):

```
$q = Supplier::with('branch');
if ($status=$r->get('status')) $q->where('status',$status);
if ($branch=$r->get('branch')) $q->where('branch_id',$branch);
if ($search=$r->get('q')) {
  $q->where(fn($qq)=>$qq->where('name','like',"%$search%")
    ->orWhere('email','like',"%$search%")
    ->orWhere('phone','like',"%$search%"));
}
$suppliers=$q->orderByDesc('id')->paginate(10);
```

```
SELECT * FROM suppliers
WHERE 1=1
  [AND status = ?]
  [AND branch_id = ?]
  [AND (name LIKE ? OR email LIKE ? OR phone LIKE ?)]
ORDER BY id DESC
LIMIT 10 OFFSET ?;
```

2.26 Supplier Store

```
Supplier::create([...validated fields...]);
```

```
INSERT INTO suppliers
(name,email,phone,address,contract_start_date,contract_end_date,status,paym
ent_terms,branch_id,created_at,updated_at)
VALUES (?,?,?,?,?,?,?,?,?,NOW(),NOW());
```

2.27 Supplier Update

```
$supplier->update($request->all());
```

```
UPDATE suppliers SET name=?, email=?, phone=?, address=?,
contract_start_date=?, contract_end_date=?, status=?, payment_terms=?,
branch_id=?, updated_at=NOW()
WHERE id=?;
```

2.28 Supplier Delete

```
$supplier->delete();
```

```
DELETE FROM suppliers WHERE id=?;
```

2.29 Supplier Show

Eloquent: `Supplier::with(['branch','products','invoices'])->findOrFail($id);`

```
SELECT * FROM suppliers WHERE id=? LIMIT 1;
SELECT * FROM branches WHERE id = (SELECT branch_id FROM suppliers WHERE
id=?);
SELECT * FROM products WHERE supplier_id=?;
SELECT * FROM invoices WHERE supplier_id=?; -- only if relationship exists
(not shown in given code but implied earlier diagram)
```

2.30 Branch Index (with counts)

Eloquent: `Branch::withCount(['products','sales'])->paginate(10);`

```
SELECT b.*,
  (SELECT COUNT(*) FROM products p WHERE p.branch_id = b.id) AS
products_count,
  (SELECT COUNT(*) FROM sales s WHERE s.branch_id = b.id) AS sales_count
```

```
FROM branches b
ORDER BY b.id DESC
LIMIT 10 OFFSET ?;
```

2.31 Branch Store

```
Branch::create($request->all());
```

```
INSERT INTO branches
(name,address,contact_number,status,created_at,updated_at)
VALUES (?,?,?,?,NOW(),NOW());
```

2.32 Branch Update

```
$branch->update($request->all());
```

```
UPDATE branches SET name=?, address=?, contact_number=?, status=?,
updated_at=NOW()
WHERE id=?;
```

2.33 Branch Delete (guard)

Eloquent guard: counts on products or sales before delete.

```
DELETE FROM branches WHERE id=?; -- only if no products and no sales
```

2.34 Branch Show

Eloquent: `Branch::with(['products','sales'])->findOrFail($id);`

```
SELECT * FROM branches WHERE id=? LIMIT 1;
SELECT * FROM products WHERE branch_id=?;
SELECT * FROM sales WHERE branch_id=?;
```

2.35 Category Index

Eloquent: `Category::withCount('products')->paginate(10);`

```
SELECT c.*, (SELECT COUNT(*) FROM products p WHERE p.category_id=c.id) AS
products_count
FROM categories c
ORDER BY c.id DESC
LIMIT 10 OFFSET ?;
```

2.36 Category Store

```
Category::create(['name'=>$r->category_name,'description'=>$r-
>category_description]);
```

```
INSERT INTO categories (name,description,created_at,updated_at)
VALUES (?,?,NOW(),NOW());
```

2.37 Category Update

```
$category->update($request->all());
```

```
UPDATE categories SET name=?, description=?, updated_at=NOW()
WHERE id=?;
```

2.38 Category Delete (guard)

```
DELETE FROM categories WHERE id=?; -- only if no products
```

2.39 Category Show

Eloquent: `Category::with('products')->findOrFail($id);`

```
SELECT * FROM categories WHERE id=? LIMIT 1;
SELECT * FROM products WHERE category_id=?;
```

2.40 User Index (filters)

Eloquent (UserController@index):

```

```

```
$q = User::with('branch');
if ($branch=$r->get('branch')) $q->where('branch_id',$branch);
if ($role=$r->get('role')) $q->where('role',$role);
if ($search=$r->get('q')) { $q->where(fn($qq)=>$qq-
>where('name','like',"%$search%")
     ->orWhere('email','like',"%$search%")); }
$users=$q->orderByDesc('id')->paginate(10);
```

```
SELECT * FROM users
WHERE 1=1
  [AND branch_id = ?]
  [AND role = ?]
  [AND (name LIKE ? OR email LIKE ?)]
ORDER BY id DESC
LIMIT 10 OFFSET ?;
```

2.41 User Store

```
User::create($validated);
```

```
INSERT INTO users
(name,email,password,role,branch_id,created_at,updated_at)
VALUES (?,?,?,?,?,NOW(),NOW());
```

2.42 User Update

```
$user->update($validated);
```

```
UPDATE users SET name=?, email=?, role=?, branch_id=?, updated_at=NOW()
WHERE id=?;
```

2.43 User Delete

```
DELETE FROM users WHERE id=?;
```

2.44 Stock Notification Mark As Read

Eloquent: `if($n){ $n->is_read=true; $n->save(); }`

```
UPDATE stock_notifications SET is_read=1, updated_at=NOW() WHERE id=?;
```

2.45 Stock Notification Delete

```
DELETE FROM stock_notifications WHERE id=?;
```

2.46 Customer Search (separate endpoint)

Eloquent: name/email/phone OR conditions with paginate(10).

```
SELECT * FROM customers
WHERE name LIKE ? OR email LIKE ? OR phone LIKE ?
ORDER BY id DESC
LIMIT 10 OFFSET ?;
```

2.47 Supplier Search (separate endpoint)

Similar pattern to index but dedicated route.

```
SELECT * FROM suppliers
WHERE name LIKE ? OR email LIKE ? OR phone LIKE ?
ORDER BY id DESC
LIMIT 10 OFFSET ?;
```

---

3. Suggested Optimization Indexes

(Domain tables only.)

```
CREATE INDEX invoices_status_idx ON invoices (status);
CREATE INDEX invoices_branch_idx ON invoices (branch);
CREATE INDEX invoices_created_at_idx ON invoices (created_at);

CREATE INDEX sales_branch_created_idx ON sales (branch_id, created_at
DESC);
CREATE INDEX sales_status_idx ON sales (status);
CREATE INDEX sales_product_idx ON sales (product_id);
CREATE INDEX sales_customer_name_idx ON sales (customer_name);

CREATE INDEX invoice_sale_invoice_idx ON invoice_sale (invoice_id);
CREATE INDEX invoice_sale_sale_idx ON invoice_sale (sale_id);

CREATE INDEX products_branch_idx ON products (branch_id);
```

```
CREATE INDEX products_supplier_idx ON products (supplier_id);
CREATE INDEX products_category_idx ON products (category_id);

CREATE INDEX customers_branch_idx ON customers (branch_id);
CREATE INDEX suppliers_branch_idx ON suppliers (branch_id);
CREATE INDEX users_branch_idx ON users (branch_id);
```

---

4. Notes / Deviations

- Invoice `type` column effectively constrained to 'sale'; enum simplified accordingly (update migration if not yet applied).
- Walk-in sale search for invoice creation: controller applies `(id = ? OR customer_name LIKE ?)` when no `customer_id` but `sale_search` term provided and restricts otherwise with `WHERE 1=0` to avoid large unfiltered result sets.
- Monetary columns use DECIMAL(10,2) (line_total DECIMAL(12,2) in pivot) – adjust if higher precision required.
- Cascade / set null behavior mirrors the migrations.
- `sessions.payload` and job tables left as in default Laravel scaffolding.
- Replace NOW() with `CURRENT_TIMESTAMP` if preferred; Eloquent populates timestamps automatically.
- Consider FULLTEXT on `customer_name`, `suppliers.name`, `products.name` for improved fuzzy search (MySQL 8+).

---

5. Quick Data Integrity Checks

```
-- Invoices with zero monetary fields (potential client calculation issue):
SELECT id, invoice_number, total_amount, tax_amount, discount FROM invoices
WHERE (total_amount = 0 OR total_amount IS NULL) AND created_at >= (NOW() -
INTERVAL 7 DAY);

-- Orphan pivot rows (should be zero):
SELECT isp.id FROM invoice_sale isp
LEFT JOIN invoices i ON i.id = isp.invoice_id
LEFT JOIN sales s ON s.id = isp.sale_id
WHERE i.id IS NULL OR s.id IS NULL;

-- Sales not attached to any invoice (normal for un-invoiced sales):
SELECT s.id, s.total_amount FROM sales s
LEFT JOIN invoice_sale isp ON isp.sale_id = s.id
WHERE isp.id IS NULL;
```

End of file.