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('purchase','sale') NOT NULL,
  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)

```
-- 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)

```
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

```
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

```
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)

```
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)

```
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
```

---

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);
```

4. Notes / Deviations

- Invoice `type` currently allows 'purchase' but application logic now only creates 'sale'; consider constraining or removing unused enum value.
- 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.
- Add partial indexes or fulltext (MySQL 8: `FULLTEXT(customer_name)`) for faster fuzzy search if needed.

5. Quick Data Integrity Checks

```
-- Orphan sales linked to invoices (should be zero due to FK cascade):
SELECT s.id FROM sales s LEFT JOIN invoice_sale is2 ON is2.sale_id = s.id
WHERE is2.id IS NULL; -- context dependent

-- Invoice vs summed sales discrepancy report:
SELECT i.id, i.total_amount, SUM(s.total_amount) AS sales_sum,
(i.total_amount - SUM(s.total_amount)) AS diff
FROM invoices i
LEFT JOIN invoice_sale isp ON isp.invoice_id = i.id
LEFT JOIN sales s ON s.id = isp.sale_id
GROUP BY i.id
HAVING diff != 0;
```

End of file.