```
// 1. USERS COLLECTION
// ================================
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["email", "password", "firstName", "lastName", "role",
"status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        email: {
          bsonType: "string",
          pattern: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$"
        },
        password: { bsonType: "string", minLength: 8 },
        firstName: { bsonType: "string", minLength: 1 },
        lastName: { bsonType: "string", minLength: 1 },
        phone: { bsonType: "string" },
        role: {
          bsonType: "string",
          enum: ["super_admin", "store_manager", "sales_staff",
"inventory_manager", "accountant"]
        },
        permissions: {
          bsonType: "object",
          properties: {
            canAccessAllStores: { bsonType: "bool" },
            canManageUsers: { bsonType: "bool" },
            canManageProducts: { bsonType: "bool" },
            canCreateInvoices: { bsonType: "bool" },
            canManageStock: { bsonType: "bool" },
            canViewReports: { bsonType: "bool" },
            canManageSettings: { bsonType: "bool" }
          }
        },
        assignedStores: {
          bsonType: "array",
          items: { bsonType: "objectId" }
        },
        status: {
          bsonType: "string",
          enum: ["active", "inactive", "suspended"]
        },
        lastLogin: { bsonType: "date" },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
```

```
      }
    }
  }
});

// Users Indexes
db.users.createIndex({ "email": 1 }, { unique: true });
db.users.createIndex({ "role": 1, "status": 1 });
db.users.createIndex({ "assignedStores": 1 });

// ================================
// 2. STORES COLLECTION
// ================================
db.createCollection("stores", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "address", "status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        name: { bsonType: "string", minLength: 1 },
        code: { bsonType: "string" }, // Unique store identifier
        description: { bsonType: "string" },
        address: {
          bsonType: "object",
          required: ["street", "city", "state", "zipCode", "country"],
          properties: {
            street: { bsonType: "string" },
            city: { bsonType: "string" },
            state: { bsonType: "string" },
            zipCode: { bsonType: "string" },
            country: { bsonType: "string" }
          }
        },
        contact: {
          bsonType: "object",
          properties: {
            phone: { bsonType: "string" },
            email: { bsonType: "string" },
            website: { bsonType: "string" }
          }
        },
        settings: {
          bsonType: "object",
          properties: {
            currency: { bsonType: "string", "default": "USD" },
            taxRate: { bsonType: "number", minimum: 0, maximum: 1 },
```

```
            language: { bsonType: "string", "default": "en" },
            timezone: { bsonType: "string" },
            businessHours: {
              bsonType: "object",
              properties: {
                monday: { bsonType: "object", properties: { open: {
bsonType: "string" }, close: { bsonType: "string" } } },
                tuesday: { bsonType: "object", properties: { open: {
bsonType: "string" }, close: { bsonType: "string" } } },
                wednesday: { bsonType: "object", properties: { open: {
bsonType: "string" }, close: { bsonType: "string" } } },
                thursday: { bsonType: "object", properties: { open: {
bsonType: "string" }, close: { bsonType: "string" } } },
                friday: { bsonType: "object", properties: { open: {
bsonType: "string" }, close: { bsonType: "string" } } },
                saturday: { bsonType: "object", properties: { open: {
bsonType: "string" }, close: { bsonType: "string" } } },
                sunday: { bsonType: "object", properties: { open: {
bsonType: "string" }, close: { bsonType: "string" } } }
              }
            }
          }
        },
        managerId: { bsonType: "objectId" }, // Reference to store manager
        status: {
          bsonType: "string",
          enum: ["active", "inactive", "maintenance"]
        },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
      }
    }
  }
});

// Stores Indexes
db.stores.createIndex({ "code": 1 }, { unique: true });
db.stores.createIndex({ "status": 1 });
db.stores.createIndex({ "managerId": 1 });

// ================================
// 3. CATEGORIES COLLECTION
// ================================
db.createCollection("categories", {
  validator: {
    $jsonSchema: {
```

```
      bsonType: "object",
      required: ["name", "status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        name: { bsonType: "string", minLength: 1 },
        description: { bsonType: "string" },
        parentId: { bsonType: "objectId" }, // For nested categories
        status: {
          bsonType: "string",
          enum: ["active", "inactive"]
        },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" }
      }
    }
  }
});

// Categories Indexes
db.categories.createIndex({ "name": 1 }, { unique: true });
db.categories.createIndex({ "parentId": 1 });
db.categories.createIndex({ "status": 1 });

// ================================
// 4. PRODUCTS COLLECTION
// ================================
db.createCollection("products", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "sku", "categoryId", "price", "cost", "status",
"createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        name: { bsonType: "string", minLength: 1 },
        description: { bsonType: "string" },
        sku: { bsonType: "string", minLength: 1 }, // Stock Keeping Unit
        barcode: { bsonType: "string" },
        categoryId: { bsonType: "objectId" },
        brand: { bsonType: "string" },
        model: { bsonType: "string" },
        price: { bsonType: "number", minimum: 0 },
        cost: { bsonType: "number", minimum: 0 },
        currency: { bsonType: "string", "default": "USD" },
        dimensions: {
          bsonType: "object",
          properties: {
```

```
            length: { bsonType: "number" },
            width: { bsonType: "number" },
            height: { bsonType: "number" },
            weight: { bsonType: "number" },
            unit: { bsonType: "string" }
          }
        },
        images: {
          bsonType: "array",
          items: {
            bsonType: "object",
            properties: {
              url: { bsonType: "string" },
              alt: { bsonType: "string" },
              isPrimary: { bsonType: "bool" }
            }
          }
        },
        specifications: {
          bsonType: "object" // Flexible object for product-specific specs
        },
        supplier: {
          bsonType: "object",
          properties: {
            name: { bsonType: "string" },
            contact: { bsonType: "string" },
            leadTime: { bsonType: "number" }, // Days
            minimumOrder: { bsonType: "number" }
          }
        },
        status: {
          bsonType: "string",
          enum: ["active", "inactive", "discontinued"]
        },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
      }
    }
  }
});

// Products Indexes
db.products.createIndex({ "sku": 1 }, { unique: true });
db.products.createIndex({ "barcode": 1 });
db.products.createIndex({ "categoryId": 1 });
```

```
db.products.createIndex({ "name": "text", "description": "text" }); // Text search
db.products.createIndex({ "status": 1 });
db.products.createIndex({ "price": 1 });

// ================================
// 5. INVENTORY COLLECTION
// ================================
db.createCollection("inventory", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["productId", "storeId", "quantity", "updatedAt"],
      properties: {
        _id: { bsonType: "objectId" },
        productId: { bsonType: "objectId" },
        storeId: { bsonType: "objectId" },
        quantity: { bsonType: "number", minimum: 0 },
        reservedQuantity: { bsonType: "number", minimum: 0, "default": 0 },
        minStockLevel: { bsonType: "number", minimum: 0, "default": 0 },
        maxStockLevel: { bsonType: "number", minimum: 0 },
        reorderPoint: 1,
        location: 1
      }
    }
  }).toArray();
}

// Function to get sales summary for dashboard
function getSalesSummary(storeId, startDate, endDate) {
  return db.invoices.aggregate([
    {
      $match: {
        storeId: ObjectId(storeId),
        status: { $in: ["paid", "partial_paid"] },
        createdAt: { $gte: startDate, $lte: endDate }
      }
    },
    {
      $group: {
        _id: null,
        totalSales: { $sum: "$total" },
        totalInvoices: { $sum: 1 },
        averageOrderValue: { $avg: "$total" }
      }
    }
```

```
  ]).toArray()[0];
}

// Function to get top selling products
function getTopSellingProducts(storeId, limit = 10) {
  return db.invoices.aggregate([
    { $match: { storeId: ObjectId(storeId), status: { $in: ["paid",
"partial_paid"] } } },
    { $unwind: "$items" },
    {
      $group: {
        _id: "$items.productId",
        totalQuantity: { $sum: "$items.quantity" },
        totalRevenue: { $sum: "$items.total" },
        productName: { $first: "$items.productName" },
        sku: { $first: "$items.sku" }
      }
    },
    { $sort: { totalQuantity: -1 } },
    { $limit: limit }
  ]).toArray();
}

// Function to get customer analytics
function getCustomerAnalytics(customerId) {
  return db.invoices.aggregate([
    { $match: { customerId: ObjectId(customerId) } },
    {
      $group: {
        _id: null,
        totalOrders: { $sum: 1 },
        totalSpent: { $sum: "$total" },
        averageOrderValue: { $avg: "$total" },
        lastOrderDate: { $max: "$createdAt" }
      }
    },
    {
      $lookup: {
        from: "customers",
        localField: "_id",
        foreignField: "_id",
        as: "customer"
      }
    }
  ]).toArray()[0];
}
```

```
// ================================
// ADDITIONAL COLLECTIONS FOR ADVANCED FEATURES
// ================================

// ================================
// 14. NOTIFICATIONS COLLECTION
// ================================
db.createCollection("notifications", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["userId", "type", "title", "message", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        userId: { bsonType: "objectId" },
        storeId: { bsonType: "objectId" },
        type: {
          bsonType: "string",
          enum: ["info", "warning", "error", "success", "low_stock",
"overdue_invoice", "system"]
        },
        title: { bsonType: "string", minLength: 1 },
        message: { bsonType: "string", minLength: 1 },
        data: { bsonType: "object" }, // Additional context data
        isRead: { bsonType: "bool", "default": false },
        readAt: { bsonType: "date" },
        priority: {
          bsonType: "string",
          enum: ["low", "medium", "high", "urgent"],
          "default": "medium"
        },
        expiresAt: { bsonType: "date" },
        createdAt: { bsonType: "date" }
      }
    }
  }
});

// Notifications Indexes
db.notifications.createIndex({ "userId": 1, "isRead": 1, "createdAt": -1
});
db.notifications.createIndex({ "type": 1 });
db.notifications.createIndex({ "expiresAt": 1 }, { expireAfterSeconds: 0
});

// ================================
// 15. DISCOUNTS COLLECTION
```

```
// ================================
db.createCollection("discounts", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "type", "value", "status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        name: { bsonType: "string", minLength: 1 },
        description: { bsonType: "string" },
        code: { bsonType: "string" }, // Discount code for customers
        type: {
          bsonType: "string",
          enum: ["percentage", "fixed_amount", "buy_x_get_y"]
        },
        value: { bsonType: "number", minimum: 0 },
        conditions: {
          bsonType: "object",
          properties: {
            minimumAmount: { bsonType: "number", minimum: 0 },
            maximumAmount: { bsonType: "number", minimum: 0 },
            applicableProducts: {
              bsonType: "array",
              items: { bsonType: "objectId" }
            },
            applicableCategories: {
              bsonType: "array",
              items: { bsonType: "objectId" }
            },
            applicableCustomers: {
              bsonType: "array",
              items: { bsonType: "objectId" }
            }
          }
        },
        validFrom: { bsonType: "date" },
        validTo: { bsonType: "date" },
        usageLimit: { bsonType: "number", minimum: 0 },
        usedCount: { bsonType: "number", minimum: 0, "default": 0 },
        storeId: { bsonType: "objectId" },
        status: {
          bsonType: "string",
          enum: ["active", "inactive", "expired"]
        },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
```

```
      }
    }
  }
});

// Discounts Indexes
db.discounts.createIndex({ "code": 1 }, { unique: true, sparse: true });
db.discounts.createIndex({ "storeId": 1, "status": 1 });
db.discounts.createIndex({ "validFrom": 1, "validTo": 1 });

// ================================
// 16. SESSIONS COLLECTION (for user session management)
// ================================
db.createCollection("sessions", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["userId", "token", "createdAt", "expiresAt"],
      properties: {
        _id: { bsonType: "objectId" },
        userId: { bsonType: "objectId" },
        token: { bsonType: "string", minLength: 1 },
        refreshToken: { bsonType: "string" },
        ipAddress: { bsonType: "string" },
        userAgent: { bsonType: "string" },
        isActive: { bsonType: "bool", "default": true },
        lastActivity: { bsonType: "date" },
        createdAt: { bsonType: "date" },
        expiresAt: { bsonType: "date" }
      }
    }
  }
});

// Sessions Indexes
db.sessions.createIndex({ "token": 1 }, { unique: true });
db.sessions.createIndex({ "userId": 1, "isActive": 1 });
db.sessions.createIndex({ "expiresAt": 1 }, { expireAfterSeconds: 0 });

// ================================
// 17. AUDIT_TRAIL COLLECTION (for compliance)
// ================================
db.createCollection("audit_trail", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
```

```
      required: ["tableName", "recordId", "action", "userId",
"timestamp"],
      properties: {
        _id: { bsonType: "objectId" },
        tableName: { bsonType: "string", minLength: 1 },
        recordId: { bsonType: "objectId" },
        action: {
          bsonType: "string",
          enum: ["INSERT", "UPDATE", "DELETE"]
        },
        oldValues: { bsonType: "object" },
        newValues: { bsonType: "object" },
        changedFields: {
          bsonType: "array",
          items: { bsonType: "string" }
        },
        userId: { bsonType: "objectId" },
        userName: { bsonType: "string" },
        storeId: { bsonType: "objectId" },
        ipAddress: { bsonType: "string" },
        timestamp: { bsonType: "date" }
      }
    }
  }
});

// Audit Trail Indexes
db.audit_trail.createIndex({ "tableName": 1, "recordId": 1, "timestamp":
-1 });
db.audit_trail.createIndex({ "userId": 1, "timestamp": -1 });
db.audit_trail.createIndex({ "timestamp": -1 });

// ================================
// DATABASE VIEWS FOR COMMON QUERIES
// ================================

// View for current inventory levels with product details
db.createView("inventory_with_products", "inventory", [
  {
    $lookup: {
      from: "products",
      localField: "productId",
      foreignField: "_id",
      as: "product"
    }
  },
  { $unwind: "$product" },
```

```
  {
    $lookup: {
      from: "stores",
      localField: "storeId",
      foreignField: "_id",
      as: "store"
    }
  },
  { $unwind: "$store" },
  {
    $project: {
      productId: 1,
      storeId: 1,
      quantity: 1,
      reservedQuantity: 1,
      minStockLevel: 1,
      maxStockLevel: 1,
      reorderPoint: 1,
      location: 1,
      "product.name": 1,
      "product.sku": 1,
      "product.price": 1,
      "product.cost": 1,
      "product.status": 1,
      "store.name": 1,
      "store.code": 1,
      updatedAt: 1
    }
  }
]);

// View for invoice summaries with customer details
db.createView("invoice_summaries", "invoices", [
  {
    $lookup: {
      from: "customers",
      localField: "customerId",
      foreignField: "_id",
      as: "customer"
    }
  },
  { $unwind: "$customer" },
  {
    $lookup: {
      from: "stores",
      localField: "storeId",
      foreignField: "_id",
```

```
      as: "store"
    }
  },
  { $unwind: "$store" },
  {
    $project: {
      invoiceNumber: 1,
      customerId: 1,
      storeId: 1,
      subtotal: 1,
      taxAmount: 1,
      total: 1,
      status: 1,
      paymentStatus: 1,
      dueDate: 1,
      createdAt: 1,
      "customer.firstName": 1,
      "customer.lastName": 1,
      "customer.companyName": 1,
      "customer.email": 1,
      "customer.type": 1,
      "store.name": 1,
      "store.code": 1,
      itemCount: { $size: "$items" }
    }
  }
]);

// View for sales analytics
db.createView("sales_analytics", "invoices", [
  {
    $match: {
      status: { $in: ["paid", "partial_paid"] }
    }
  },
  {
    $group: {
      _id: {
        storeId: "$storeId",
        year: { $year: "$createdAt" },
        month: { $month: "$createdAt" },
        day: { $dayOfMonth: "$createdAt" }
      },
      dailySales: { $sum: "$total" },
      dailyOrders: { $sum: 1 },
      averageOrderValue: { $avg: "$total" }
    }
```

```
  },
  {
    $lookup: {
      from: "stores",
      localField: "_id.storeId",
      foreignField: "_id",
      as: "store"
    }
  },
  { $unwind: "$store" },
  {
    $project: {
      storeId: "$_id.storeId",
      storeName: "$store.name",
      storeCode: "$store.code",
      date: {
        $dateFromParts: {
          year: "$_id.year",
          month: "$_id.month",
          day: "$_id.day"
        }
      },
      dailySales: 1,
      dailyOrders: 1,
      averageOrderValue: 1
    }
  },
  { $sort: { date: -1 } }
]);

// ================================
// STORED PROCEDURES (MongoDB Functions)
// ================================

// Function to process an invoice and update inventory
db.system.js.save({
  _id: "processInvoice",
  value: function(invoiceId) {
    var invoice = db.invoices.findOne({ _id: ObjectId(invoiceId) });
    if (!invoice || invoice.status !== 'draft') {
      return { success: false, message: "Invoice not found or already
processed" };
    }

    // Check inventory availability
    for (var i = 0; i < invoice.items.length; i++) {
      var item = invoice.items[i];
```

```
  var inventory = db.inventory.findOne({
    productId: item.productId,
    storeId: invoice.storeId
  });

  if (!inventory || inventory.quantity < item.quantity) {
    return {
      success: false,
      message: "Insufficient stock for product: " + item.productName
    };
  }
}

// Update inventory and create stock movements
for (var i = 0; i < invoice.items.length; i++) {
  var item = invoice.items[i];

  // Update inventory
  db.inventory.updateOne(
    { productId: item.productId, storeId: invoice.storeId },
    {
      $inc: { quantity: -item.quantity },
      $set: {
        lastSaleDate: new Date(),
        updatedAt: new Date()
      }
    }
  );

  // Create stock movement record
  db.stock_movements.insertOne({
    productId: item.productId,
    storeId: invoice.storeId,
    type: "out",
    quantity: -item.quantity,
    reason: "Sale",
    reference: invoice.invoiceNumber,
    referenceId: invoice._id,
    unitCost: item.unitPrice,
    totalCost: item.total,
    createdAt: new Date(),
    createdBy: invoice.createdBy
  });
}

// Update invoice status
db.invoices.updateOne(
```

```
      { _id: ObjectId(invoiceId) },
      {
        $set: {
          status: "sent",
          updatedAt: new Date()
        }
      }
    );

    return { success: true, message: "Invoice processed successfully" };
  }
});

// Function to generate restock suggestions
db.system.js.save({
  _id: "generateRestockSuggestions",
  value: function(storeId) {
    return db.inventory.aggregate([
      { $match: { storeId: ObjectId(storeId) } },
      { $match: { $expr: { $lte: ["$quantity", "$reorderPoint"] } } },
      {
        $lookup: {
          from: "products",
          localField: "productId",
          foreignField: "_id",
          as: "product"
        }
      },
      { $unwind: "$product" },
      {
        $lookup: {
          from: "stock_movements",
          let: { productId: "$productId", storeId: "$storeId" },
          pipeline: [
            {
              $match: {
                $expr: {
                  $and: [
                    { $eq: ["$productId", "$productId"] },
                    { $eq: ["$storeId", "$storeId"] },
                    { $eq: ["$type", "out"] },
                    { $gte: ["$createdAt", new Date(Date.now() - 30 * 24 *
60 * 60 * 1000)] }
                  ]
                }
              }
            },
```

```
            {
              $group: {
                _id: null,
                totalSold: { $sum: { $abs: "$quantity" } }
              }
            }
          ],
          as: "salesData"
        }
      },
      {
        $project: {
          productId: 1,
          productName: "$product.name",
          sku: "$product.sku",
          currentStock: "$quantity",
          reorderPoint: 1,
          maxStockLevel: 1,
          monthlySales: { $ifNull: [{ $arrayElemAt:
["$salesData.totalSold", 0] }, 0] },
          suggestedOrder: {
            $subtract: [
              "$maxStockLevel",
              "$quantity"
            ]
          },
          supplier: "$product.supplier"
        }
      },
      { $match: { suggestedOrder: { $gt: 0 } } },
      { $sort: { monthlySales: -1 } }
    ]).toArray();
  }
});

// ================================
// SECURITY AND PERFORMANCE OPTIMIZATIONS
// ================================

// Create compound indexes for better query performance
db.invoices.createIndex({ "storeId": 1, "status": 1, "createdAt": -1 });
db.invoices.createIndex({ "customerId": 1, "status": 1, "createdAt": -1
});
db.stock_movements.createIndex({ "storeId": 1, "type": 1, "createdAt": -1
});
db.inventory.createIndex({ "storeId": 1, "quantity": 1, "minStockLevel": 1
});
```

```
// Create partial indexes for active records only
db.users.createIndex(
  { "email": 1, "status": 1 },
  { partialFilterExpression: { "status": "active" } }
);

db.products.createIndex(
  { "name": "text", "sku": 1 },
  { partialFilterExpression: { "status": "active" } }
);

// Create TTL indexes for temporary data
db.sessions.createIndex({ "createdAt": 1 }, { expireAfterSeconds: 86400
}); // 24 hours
db.activity_logs.createIndex({ "timestamp": 1 }, { expireAfterSeconds:
7776000 }); // 90 days

// ================================
// SAMPLE QUERIES FOR COMMON OPERATIONS
// ================================

/*
// Get dashboard data for a store
db.invoice_summaries.aggregate([
  { $match: { storeId: ObjectId("STORE_ID"), status: { $in: ["paid",
"partial_paid"] } } },
  {
    $group: {
      _id: { $dateToString: { format: "%Y-%m-%d", date: "$createdAt" } },
      dailySales: { $sum: "$total" },
      orderCount: { $sum: 1 }
    }
  },
  { $sort: { "_id": -1 } },
  { $limit: 30 }
]);

// Get low stock alerts
db.inventory_with_products.find({
  storeId: ObjectId("STORE_ID"),
  $expr: { $lte: ["$quantity", "$minStockLevel"] },
  "product.status": "active"
});

// Get customer purchase history
db.invoice_summaries.find({
```

```
  customerId: ObjectId("CUSTOMER_ID")
}).sort({ createdAt: -1 });

// Get product sales performance
db.invoices.aggregate([
  { $match: { storeId: ObjectId("STORE_ID"), status: { $in: ["paid",
"partial_paid"] } } },
  { $unwind: "$items" },
  {
    $group: {
      _id: "$items.productId",
      totalQuantity: { $sum: "$items.quantity" },
      totalRevenue: { $sum: "$items.total" },
      orderCount: { $sum: 1 }
    }
  },
  {
    $lookup: {
      from: "products",
      localField: "_id",
      foreignField: "_id",
      as: "product"
    }
  },
  { $unwind: "$product" },
  { $sort: { totalRevenue: -1 } }
]);
*/

print("✅ ERP System MongoDB Schema Created Successfully!");
print("📊 Collections: 17 main collections + 3 views");
print("🔐 Indexes: Optimized for performance and security");
print("⚡ Functions: Utility functions for common operations");
print("🛡️  Validation: Schema validation rules applied");
print("📈 Ready for your ERP system implementation!");Point: { bsonType:
"number", minimum: 0 },
        location: {
          bsonType: "object",
          properties: {
            aisle: { bsonType: "string" },
            shelf: { bsonType: "string" },
            bin: { bsonType: "string" }
          }
        },
        lastRestockDate: { bsonType: "date" },
        lastSaleDate: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
```

```
        updatedBy: { bsonType: "objectId" }
      }
    }
  }
});

// Inventory Indexes
db.inventory.createIndex({ "productId": 1, "storeId": 1 }, { unique: true
});
db.inventory.createIndex({ "storeId": 1 });
db.inventory.createIndex({ "quantity": 1 });
db.inventory.createIndex({ "minStockLevel": 1, "quantity": 1 }); // For
low stock alerts

// ================================
// 6. CUSTOMERS COLLECTION
// ================================
db.createCollection("customers", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["type", "status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        customerCode: { bsonType: "string" }, // Auto-generated unique
code
        type: {
          bsonType: "string",
          enum: ["individual", "business"]
        },
        // Individual customer fields
        firstName: { bsonType: "string" },
        lastName: { bsonType: "string" },
        // Business customer fields
        companyName: { bsonType: "string" },
        taxId: { bsonType: "string" },
        // Common fields
        email: { bsonType: "string" },
        phone: { bsonType: "string" },
        address: {
          bsonType: "object",
          properties: {
            street: { bsonType: "string" },
            city: { bsonType: "string" },
            state: { bsonType: "string" },
            zipCode: { bsonType: "string" },
            country: { bsonType: "string" }
```

```
          }
        },
        billingAddress: {
          bsonType: "object",
          properties: {
            street: { bsonType: "string" },
            city: { bsonType: "string" },
            state: { bsonType: "string" },
            zipCode: { bsonType: "string" },
            country: { bsonType: "string" }
          }
        },
        category: {
          bsonType: "string",
          enum: ["regular", "vip", "wholesale", "retail"]
        },
        creditLimit: { bsonType: "number", minimum: 0, "default": 0 },
        paymentTerms: { bsonType: "number", "default": 30 }, // Days
        discount: { bsonType: "number", minimum: 0, maximum: 1, "default":
0 },
        status: {
          bsonType: "string",
          enum: ["active", "inactive", "blocked"]
        },
        notes: { bsonType: "string" },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
      }
    }
  }
});

// Customers Indexes
db.customers.createIndex({ "customerCode": 1 }, { unique: true });
db.customers.createIndex({ "email": 1 });
db.customers.createIndex({ "phone": 1 });
db.customers.createIndex({ "type": 1, "status": 1 });
db.customers.createIndex({ "firstName": "text", "lastName": "text",
"companyName": "text" });

// ================================
// 7. INVOICES COLLECTION
// ================================
db.createCollection("invoices", {
  validator: {
    $jsonSchema: {
```

```
      bsonType: "object",
      required: ["invoiceNumber", "customerId", "storeId", "items",
"subtotal", "total", "status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        invoiceNumber: { bsonType: "string", minLength: 1 },
        customerId: { bsonType: "objectId" },
        storeId: { bsonType: "objectId" },
        items: {
          bsonType: "array",
          minItems: 1,
          items: {
            bsonType: "object",
            required: ["productId", "quantity", "unitPrice", "total"],
            properties: {
              productId: { bsonType: "objectId" },
              productName: { bsonType: "string" },
              sku: { bsonType: "string" },
              quantity: { bsonType: "number", minimum: 0.01 },
              unitPrice: { bsonType: "number", minimum: 0 },
              discount: { bsonType: "number", minimum: 0, "default": 0 },
              total: { bsonType: "number", minimum: 0 }
            }
          }
        },
        subtotal: { bsonType: "number", minimum: 0 },
        discountAmount: { bsonType: "number", minimum: 0, "default": 0 },
        taxAmount: { bsonType: "number", minimum: 0, "default": 0 },
        taxRate: { bsonType: "number", minimum: 0, maximum: 1 },
        total: { bsonType: "number", minimum: 0 },
        currency: { bsonType: "string", "default": "USD" },
        paymentTerms: { bsonType: "number", "default": 30 },
        dueDate: { bsonType: "date" },
        status: {
          bsonType: "string",
          enum: ["draft", "sent", "paid", "partial_paid", "overdue",
"cancelled"]
        },
        paymentStatus: {
          bsonType: "string",
          enum: ["unpaid", "partial", "paid", "overpaid"]
        },
        notes: { bsonType: "string" },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
      }
```

```
    }
  }
});

// Invoices Indexes
db.invoices.createIndex({ "invoiceNumber": 1 }, { unique: true });
db.invoices.createIndex({ "customerId": 1 });
db.invoices.createIndex({ "storeId": 1 });
db.invoices.createIndex({ "status": 1 });
db.invoices.createIndex({ "paymentStatus": 1 });
db.invoices.createIndex({ "dueDate": 1 });
db.invoices.createIndex({ "createdAt": -1 });

// ================================
// 8. PAYMENTS COLLECTION
// ================================
db.createCollection("payments", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["invoiceId", "amount", "method", "status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        invoiceId: { bsonType: "objectId" },
        paymentNumber: { bsonType: "string" },
        amount: { bsonType: "number", minimum: 0.01 },
        method: {
          bsonType: "string",
          enum: ["cash", "credit_card", "debit_card", "bank_transfer",
"check", "other"]
        },
        reference: { bsonType: "string" }, // Transaction reference
        status: {
          bsonType: "string",
          enum: ["pending", "completed", "failed", "cancelled"]
        },
        notes: { bsonType: "string" },
        createdAt: { bsonType: "date" },
        processedAt: { bsonType: "date" },
        processedBy: { bsonType: "objectId" }
      }
    }
  }
});

// Payments Indexes
db.payments.createIndex({ "invoiceId": 1 });
```

```
db.payments.createIndex({ "status": 1 });
db.payments.createIndex({ "method": 1 });
db.payments.createIndex({ "createdAt": -1 });

// ================================
// 9. STOCK_MOVEMENTS COLLECTION
// ================================
db.createCollection("stock_movements", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["productId", "storeId", "type", "quantity", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        productId: { bsonType: "objectId" },
        storeId: { bsonType: "objectId" },
        type: {
          bsonType: "string",
          enum: ["in", "out", "adjustment", "transfer", "damaged",
"returned"]
        },
        quantity: { bsonType: "number" }, // Can be negative for out
movements
        reason: { bsonType: "string" },
        reference: { bsonType: "string" }, // Invoice number, PO number,
etc.
        referenceId: { bsonType: "objectId" }, // Reference to invoice,
purchase order, etc.
        fromStoreId: { bsonType: "objectId" }, // For transfers
        toStoreId: { bsonType: "objectId" }, // For transfers
        unitCost: { bsonType: "number", minimum: 0 },
        totalCost: { bsonType: "number" },
        notes: { bsonType: "string" },
        createdAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
      }
    }
  }
});

// Stock Movements Indexes
db.stock_movements.createIndex({ "productId": 1, "storeId": 1 });
db.stock_movements.createIndex({ "type": 1 });
db.stock_movements.createIndex({ "createdAt": -1 });
db.stock_movements.createIndex({ "referenceId": 1 });

// ================================
```

```
// 10. PURCHASE_ORDERS COLLECTION
// ================================
db.createCollection("purchase_orders", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["poNumber", "supplier", "storeId", "items", "total",
"status", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        poNumber: { bsonType: "string", minLength: 1 },
        supplier: {
          bsonType: "object",
          required: ["name", "contact"],
          properties: {
            name: { bsonType: "string" },
            contact: { bsonType: "string" },
            address: { bsonType: "string" }
          }
        },
        storeId: { bsonType: "objectId" },
        items: {
          bsonType: "array",
          minItems: 1,
          items: {
            bsonType: "object",
            required: ["productId", "quantity", "unitCost", "total"],
            properties: {
              productId: { bsonType: "objectId" },
              productName: { bsonType: "string" },
              sku: { bsonType: "string" },
              quantity: { bsonType: "number", minimum: 0.01 },
              unitCost: { bsonType: "number", minimum: 0 },
              total: { bsonType: "number", minimum: 0 },
              receivedQuantity: { bsonType: "number", minimum: 0,
"default": 0 }
            }
          }
        },
        subtotal: { bsonType: "number", minimum: 0 },
        taxAmount: { bsonType: "number", minimum: 0, "default": 0 },
        total: { bsonType: "number", minimum: 0 },
        expectedDeliveryDate: { bsonType: "date" },
        actualDeliveryDate: { bsonType: "date" },
        status: {
          bsonType: "string",
```

```
          enum: ["draft", "sent", "confirmed", "partial_received",
"received", "cancelled"]
        },
        notes: { bsonType: "string" },
        createdAt: { bsonType: "date" },
        updatedAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
      }
    }
  }
});

// Purchase Orders Indexes
db.purchase_orders.createIndex({ "poNumber": 1 }, { unique: true });
db.purchase_orders.createIndex({ "storeId": 1 });
db.purchase_orders.createIndex({ "status": 1 });
db.purchase_orders.createIndex({ "createdAt": -1 });

// ================================
// 11. ACTIVITY_LOGS COLLECTION
// ================================
db.createCollection("activity_logs", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["userId", "action", "entity", "timestamp"],
      properties: {
        _id: { bsonType: "objectId" },
        userId: { bsonType: "objectId" },
        userName: { bsonType: "string" },
        action: {
          bsonType: "string",
          enum: ["create", "update", "delete", "login", "logout", "view",
"export"]
        },
        entity: {
          bsonType: "string",
          enum: ["user", "store", "product", "customer", "invoice",
"payment", "inventory", "purchase_order"]
        },
        entityId: { bsonType: "objectId" },
        storeId: { bsonType: "objectId" },
        details: { bsonType: "object" }, // Additional context
        ipAddress: { bsonType: "string" },
        userAgent: { bsonType: "string" },
        timestamp: { bsonType: "date" }
      }
```

```
    }
  }
});

// Activity Logs Indexes
db.activity_logs.createIndex({ "userId": 1, "timestamp": -1 });
db.activity_logs.createIndex({ "entity": 1, "entityId": 1 });
db.activity_logs.createIndex({ "storeId": 1, "timestamp": -1 });
db.activity_logs.createIndex({ "timestamp": -1 });

// ================================
// 12. REPORTS COLLECTION
// ================================
db.createCollection("reports", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "type", "createdBy", "createdAt"],
      properties: {
        _id: { bsonType: "objectId" },
        name: { bsonType: "string", minLength: 1 },
        type: {
          bsonType: "string",
          enum: ["sales", "inventory", "financial", "customer", "custom"]
        },
        description: { bsonType: "string" },
        parameters: { bsonType: "object" }, // Report filters and
parameters
        schedule: {
          bsonType: "object",
          properties: {
            frequency: {
              bsonType: "string",
              enum: ["daily", "weekly", "monthly", "quarterly", "yearly",
"custom"]
            },
            recipients: {
              bsonType: "array",
              items: { bsonType: "string" } // Email addresses
            },
            isActive: { bsonType: "bool", "default": false }
          }
        },
        lastGenerated: { bsonType: "date" },
        createdAt: { bsonType: "date" },
        createdBy: { bsonType: "objectId" }
      }
```

```
    }
  }
});

// Reports Indexes
db.reports.createIndex({ "type": 1 });
db.reports.createIndex({ "createdBy": 1 });
db.reports.createIndex({ "createdAt": -1 });

// ================================
// 13. SYSTEM_SETTINGS COLLECTION
// ================================
db.createCollection("system_settings", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["key", "value", "updatedAt"],
      properties: {
        _id: { bsonType: "objectId" },
        key: { bsonType: "string", minLength: 1 },
        value: {}, // Mixed type - can be string, number, object, array
        description: { bsonType: "string" },
        category: { bsonType: "string" },
        isSystem: { bsonType: "bool", "default": false },
        updatedAt: { bsonType: "date" },
        updatedBy: { bsonType: "objectId" }
      }
    }
  }
});
```