SQL Documentation for Urban Retail Co. Inventory Management System

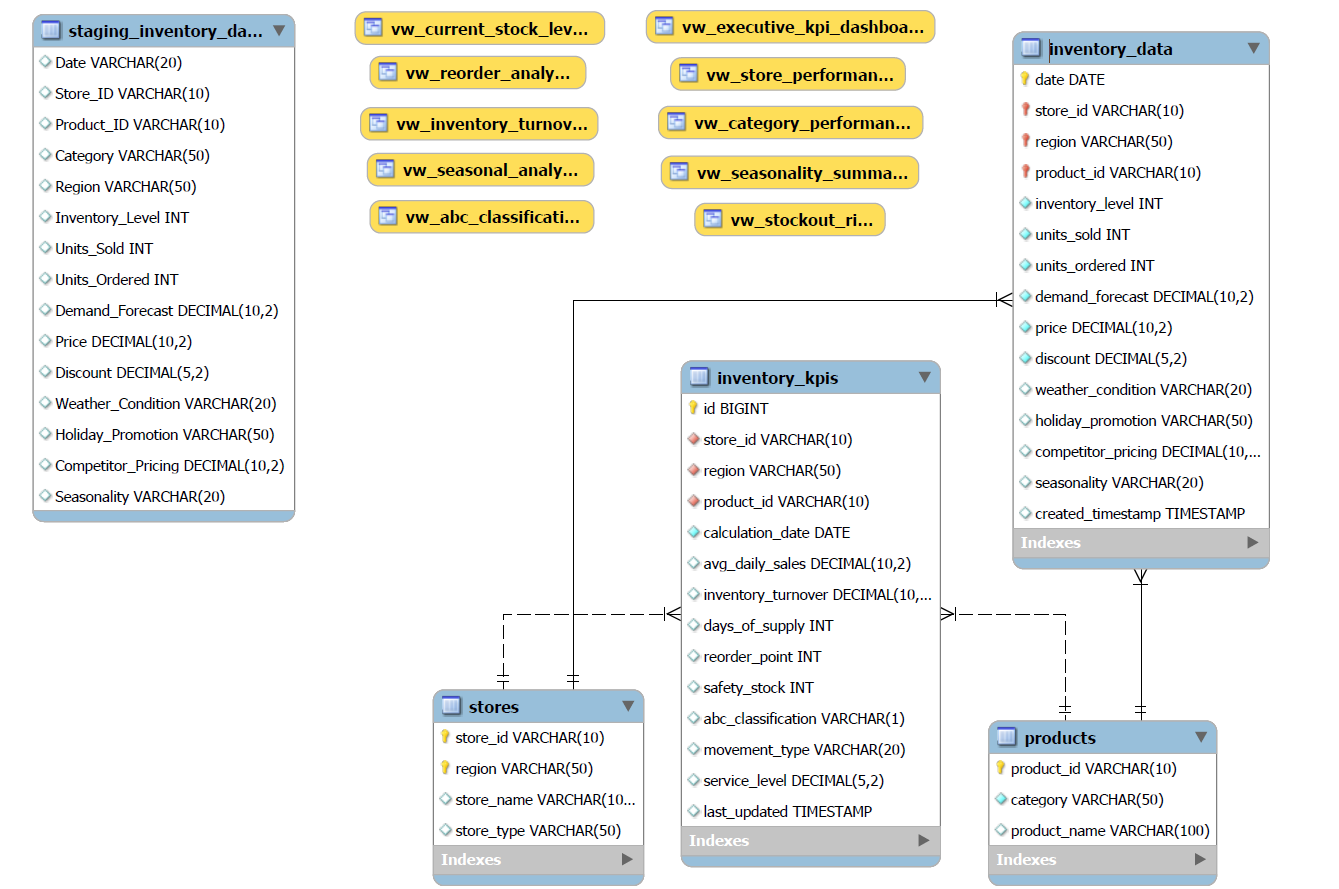
**Table of Contents**

1. SQL Infrastructure Design
2. Database Schema Overview
3. Table Descriptions
4. View Descriptions
5. Query Documentation
6. Data Import Procedures
7. Performance Considerations
8. Maintenance Guidelines

# 1. SQL Infrastructure Design

A modular, SQL-centric pipeline ensures clear separation of concerns:

* **Staging**: Temporary tables mirror raw data, isolating ingestion issues.
* **Core Schema**: Master (stores, products) and fact (inventory\_data) tables store cleaned data.
* **Analytics Layer**: SQL VIEWs encapsulate business logic (reorder points, turnover).
* **Reporting Layer**: Aggregated VIEWs present KPIs for visualization tools or email reports.



# 2. Database Schema Overview

The Urban Retail Co. inventory management database follows a normalized design with composite primary keys to accurately model the retail inventory environment. The schema consists of four main tables:

* **stores**: Master table for store information with (store\_id, region) composite primary key
* **products**: Master table for product catalog data
* **inventory\_data**: Main fact table containing daily inventory transactions
* **inventory\_kpis**: Calculated metrics table for performance analysis

The schema supports relational integrity through foreign key constraints while maintaining optimal query performance through strategic indexing.

# 2. Table Descriptions

**stores**

CREATE TABLE stores (  
 store\_id VARCHAR(10) NOT NULL COMMENT 'Store identifier',  
 region VARCHAR(50) NOT NULL COMMENT 'Store region (West, East, South, etc.)',  
 store\_name VARCHAR(100) COMMENT 'Generated store name',  
 store\_type VARCHAR(50) DEFAULT 'Retail' COMMENT 'Type of store',  
   
 PRIMARY KEY (store\_id, region),  
 INDEX idx\_region (region),  
 INDEX idx\_store\_id (store\_id)  
) ENGINE=InnoDB COMMENT='Master table for store information with composite PK';

* **Purpose**: Maintains store master data with composite primary key to support multiple regions per store ID
* **Key Fields**: store\_id, region (composite PK)
* **Indexes**: Optimized for region-based and store-based lookups

**products**

CREATE TABLE products (  
 product\_id VARCHAR(10) NOT NULL PRIMARY KEY COMMENT 'Unique product identifier',  
 category VARCHAR(50) NOT NULL COMMENT 'Product category (Electronics, Toys, etc.)',  
 product\_name VARCHAR(100) COMMENT 'Product name (auto-generated)',  
   
 INDEX idx\_category (category)  
) ENGINE=InnoDB COMMENT='Master table for product information';

* **Purpose**: Stores product catalog information
* **Key Fields**: product\_id (PK), category
* **Indexes**: Category-based lookups for category analysis

**inventory\_data**

CREATE TABLE inventory\_data (  
 date DATE NOT NULL COMMENT 'Transaction date',  
 store\_id VARCHAR(10) NOT NULL COMMENT 'Store identifier',  
 region VARCHAR(50) NOT NULL COMMENT 'Store region',  
 product\_id VARCHAR(10) NOT NULL COMMENT 'Product identifier',  
 inventory\_level INT NOT NULL DEFAULT 0 COMMENT 'Inventory level on this date',  
 units\_sold INT NOT NULL DEFAULT 0 COMMENT 'Units sold on this date',  
 units\_ordered INT NOT NULL DEFAULT 0 COMMENT 'Units ordered on this date',  
 demand\_forecast DECIMAL(10,2) NOT NULL DEFAULT 0 COMMENT 'Demand forecast',  
 price DECIMAL(10,2) NOT NULL DEFAULT 0 COMMENT 'Product price',  
 discount DECIMAL(5,2) NOT NULL DEFAULT 0 COMMENT 'Discount percentage',  
 weather\_condition VARCHAR(20) COMMENT 'Weather condition on this date',  
 holiday\_promotion VARCHAR(50) COMMENT 'Holiday/promotion information',  
 competitor\_pricing DECIMAL(10,2) DEFAULT 0 COMMENT 'Competitor pricing',  
 seasonality VARCHAR(20) COMMENT 'Season on this date',  
 created\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP COMMENT 'Record creation timestamp',  
   
 PRIMARY KEY (date, store\_id, region, product\_id),  
 FOREIGN KEY (store\_id, region) REFERENCES stores(store\_id, region),  
 FOREIGN KEY (product\_id) REFERENCES products(product\_id),  
   
 INDEX idx\_date\_store\_region (date, store\_id, region),  
 INDEX idx\_store\_region\_product (store\_id, region, product\_id),  
 INDEX idx\_date\_range (date)  
) ENGINE=InnoDB COMMENT='Main fact table for daily inventory data';

* **Purpose**: Main fact table containing daily inventory transactions
* **Key Fields**: date, store\_id, region, product\_id (composite PK)
* **Foreign Keys**: References stores and products tables
* **Indexes**: Optimized for date-range queries and store/product analysis

**inventory\_kpis**

CREATE TABLE inventory\_kpis (  
 id BIGINT AUTO\_INCREMENT PRIMARY KEY COMMENT 'Auto-generated ID',  
 store\_id VARCHAR(10) NOT NULL COMMENT 'Store identifier',  
 region VARCHAR(50) NOT NULL COMMENT 'Store region',  
 product\_id VARCHAR(10) NOT NULL COMMENT 'Product identifier',  
 calculation\_date DATE NOT NULL COMMENT 'KPI calculation date',  
 avg\_daily\_sales DECIMAL(10,2) DEFAULT 0 COMMENT 'Average daily sales (90-day)',  
 inventory\_turnover DECIMAL(10,2) DEFAULT 0 COMMENT 'Annualized inventory turnover',  
 days\_of\_supply INT DEFAULT 0 COMMENT 'Days of supply at current sales rate',  
 reorder\_point INT DEFAULT 0 COMMENT 'Calculated reorder point',  
 safety\_stock INT DEFAULT 0 COMMENT 'Calculated safety stock',  
 abc\_classification VARCHAR(1) DEFAULT 'C' COMMENT 'ABC classification (A/B/C)',  
 movement\_type VARCHAR(20) DEFAULT 'SLOW\_MOVER' COMMENT 'Movement classification',  
 service\_level DECIMAL(5,2) DEFAULT 95.00 COMMENT 'Target service level percentage',  
 last\_updated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
   
 FOREIGN KEY (store\_id, region) REFERENCES stores(store\_id, region),  
 FOREIGN KEY (product\_id) REFERENCES products(product\_id),  
   
 UNIQUE KEY uk\_store\_region\_product\_date (store\_id, region, product\_id, calculation\_date)  
) ENGINE=InnoDB COMMENT='Calculated KPIs and metrics for inventory management';

* **Purpose**: Stores calculated inventory metrics for performance tracking
* **Key Fields**: id (PK), store\_id, region, product\_id, calculation\_date
* **Foreign Keys**: References stores and products tables
* **Unique Constraint**: Prevents duplicate KPI records

# 3. View Descriptions

The system implements several analytical views to support business intelligence and decision-making:

**vw\_current\_stock\_levels**

* **Purpose**: Provides current inventory levels and status for all products
* **Key Features**: Uses window functions to get latest inventory position
* **Business Use**: Stock level monitoring and inventory status reporting

**vw\_reorder\_analysis**

* **Purpose**: Identifies products requiring reorder with calculated reorder points
* **Key Features**: Implements statistical safety stock calculations
* **Business Use**: Procurement planning and stockout prevention

**vw\_inventory\_turnover**

* **Purpose**: Calculates inventory efficiency metrics
* **Key Features**: Annualized turnover calculations and movement classification
* **Business Use**: Inventory efficiency monitoring and optimization

**vw\_abc\_classification**

* **Purpose**: Implements Pareto (ABC) analysis for product classification
* **Key Features**: Cumulative percentage calculations using window functions
* **Business Use**: Focused inventory management based on value contribution

**vw\_seasonal\_analysis**

* **Purpose**: Analyzes seasonal impact on sales patterns
* **Key Features**: Seasonal index calculations versus baseline
* **Business Use**: Seasonal planning and demand forecasting

# 4. Query Documentation

**Stock Level Monitoring**

* Identifies low or critical stock conditions
* Output: product ID, stock level, store ID, region

**Reorder Point Analysis**

* Compares inventory to reorder points to detect replenishment needs

**Inventory Turnover Analysis**

* Outputs turnover ratio and classification
* View supports filtering by category or product

**ABC Classification Analysis**

* Aggregates product count and revenue contribution by ABC class

SELECT store\_id, region, abc\_classification, COUNT(\*), SUM(...) FROM vw\_abc\_classification GROUP BY ...;

**Seasonal Impact Analysis**

* Analyzes seasonality trends for planning
* Example category and season values provided for demonstration

# 5. Data Import Procedures

**Staging Import**

* CSV load into staging\_inventory\_data via LOAD DATA INFILE
* Fields parsed and cleansed for date formats and nulls

**Data Population**

-- Insert stores

INSERT IGNORE INTO stores (...) SELECT DISTINCT ... FROM staging\_inventory\_data;

-- Insert products

INSERT IGNORE INTO products (...) SELECT DISTINCT ... FROM staging\_inventory\_data;

-- Insert inventory

INSERT IGNORE INTO inventory\_data (...) SELECT ... FROM staging\_inventory\_data;

* Ensures deduplication and correct referential links

# 6. Performance Considerations

**Index Strategy**

The database uses a strategic indexing approach:

* **Composite Primary Keys**: For enforcing uniqueness and referential integrity
* **Secondary Indexes**: For common query patterns on date ranges, categories, etc.
* **Covering Indexes**: Where possible to avoid table lookups

**Query Optimization**

* **Window Functions**: Used for efficient ranking and row selection
* **Common Table Expressions (CTEs)**: For readability and query organization
* **Join Order**: Optimized to start with the most restrictive conditions
* **Selective Indexing**: Balances performance with storage requirements

**Large Dataset Handling**

* **Partitioning**: Consider partitioning inventory\_data by date for large datasets
* **Summary Tables**: Pre-calculate metrics for faster dashboard performance
* **Data Retention**: Implement archiving strategy for historical data

# 7. Maintenance Guidelines

**Regular Maintenance Tasks**

1. **Update Table Statistics**: Run ANALYZE TABLE periodically
2. **Index Optimization**: Review and optimize indexes based on query patterns
3. **KPI Recalculation**: Refresh inventory\_kpis table daily **\***

**Monitoring Recommendations \*[[1]](#footnote-1)**

1. **Query Performance**: Monitor slow queries log
2. **Storage Growth**: Track database size and growth rate
3. **Data Quality**: Implement checks for data completeness and accuracy

# Conclusion

This SQL documentation provides a comprehensive reference for the Urban Retail Co. Inventory Management System. The schema design with composite primary keys, strategic indexing, and analytical views enables efficient inventory analysis and decision support. Regular maintenance and optimization will ensure continued performance as the system scales with business growth.

1. **\*** To be implemented in actual working environment [↑](#footnote-ref-1)