SQL Documentation

**Table of Contents**

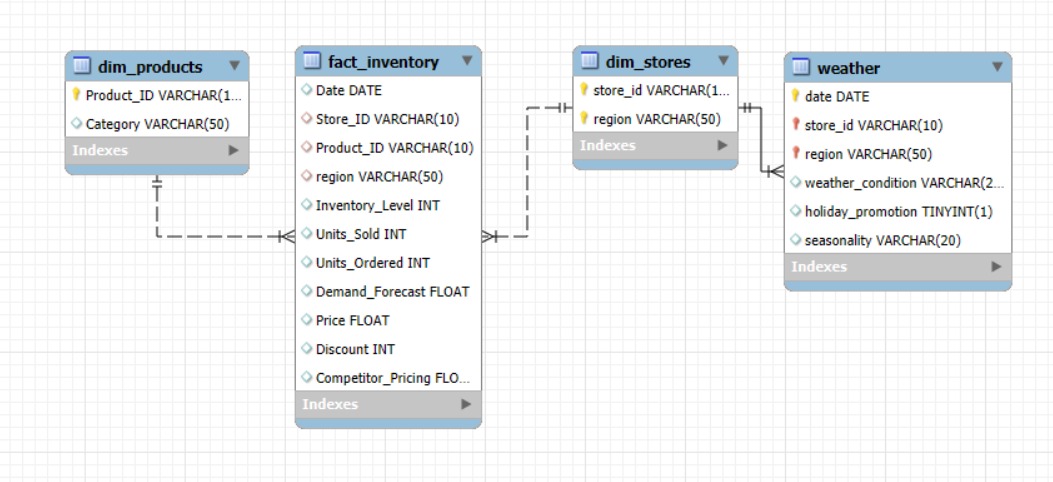
1. SQL Design
2. Database Schema Overview
3. Table Descriptions
4. Query Documentation
5. Data Import Procedures
6. Performance Considerations
7. Maintenance Guidelines

# SQL Design

The SQL design aims to leverage raw data using structured queries and reporting logic to generate actionable insights. This documentation outlines how a relational database schema, supported by SQL-based analytics.

To ensure scalability, accuracy, and performance, the SQL design follows key best practices:

* **Normalized schema**:  
  The original raw dataset (inventory\_forecasting.csv) was transformed into a normalized database structure. This involved breaking down the flat file into separate relational tables:
  + **dim\_Products** for product-level attributes like category
  + **dim\_Stores** for store and regional mapping
  + **fact\_Inventory** for transactional data like sales, inventory, and pricing
  + **Weather** for environmental factors like weather conditions and seasonality  
    This reduces data redundancy, improves data integrity, and enhances query efficiency.
* Use of indexing on frequently queried columns such as store\_id, product\_id, and date to optimize performance
* Clear separation of fact and dimension tables, creating a star-schema style model for better reporting and scalability
* Modular query structure, allowing users to build complex analytical queries from reusable logic.



# 2. Database Schema Overview

The database schema for **Urban Retail Co.** has been designed using a **star schema** approach, which is optimized for analytical querying and reporting. It clearly separates dimension tables (which describe business entities) from the central fact table (which stores transactional-level data).

* **Fact Table: fact\_inventory — stores quantitative metrics related to product inventory, demand forecasting, and sales**
* **Dimension Tables:**
  + **dim\_products — product category and identifiers**
  + **dim\_stores — store-level details and regional info**
  + **weather — contextual information like weather, seasonality, and promotions**

**This design ensures:**

* **Efficient querying through indexed keys**
* **Easy filtering, grouping, and joining for reports and dashboards**

# Table Descriptions

This section provides a concise explanation of each table in the database, including its purpose and key fields.

**dim\_products**

CREATE TABLE dim\_Products (

Product\_ID VARCHAR(10) PRIMARY KEY,

Category VARCHAR(50)

);

**Purpose:  
Stores product-level information, such as unique product identifiers and category classification. This helps in grouping and analysing SKUs across broader product families.**

**Key Columns:**

* **Product\_ID *(VARCHAR)* – Unique identifier for each product (Primary Key)**
* **Category *(VARCHAR)* – Broad classification such as “Groceries”, “Electronics” etc.**

**dim\_stores**

CREATE TABLE dim\_Stores(

store\_id VARCHAR(10),

region VARCHAR(50),

PRIMARY KEY (store\_id, region)

);

**Contains static information about each retail store, including its regional affiliation. This allows aggregation and filtering of sales/inventory data at a store or regional level.**

**Key Columns:**

* **store\_id *(VARCHAR)* – Unique identifier for each store (Primary Key)**
* **region *(VARCHAR)* – Geographic zone or operational region of the store**

**fact\_inventory**

CREATE TABLE fact\_Inventory (

Date DATE,

Store\_ID VARCHAR(10),

Product\_ID VARCHAR(10),

region varchar(50),

Inventory\_Level INT,

Units\_Sold INT,

Units\_Ordered INT,

Demand\_Forecast FLOAT,

Price FLOAT,

Discount INT,

Competitor\_Pricing FLOAT,

FOREIGN KEY (store\_id, region) REFERENCES dim\_Stores(store\_id, region),

FOREIGN KEY (product\_id) REFERENCES dim\_Products(product\_id)

);

* **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

**weather**

CREATE TABLE Weather(

date DATE,

store\_id VARCHAR(10),

region VARCHAR(50),

weather\_condition VARCHAR(20),

holiday\_promotion BOOLEAN,

seasonality VARCHAR(20),

PRIMARY KEY (date, store\_id, region),

FOREIGN KEY (store\_id, region) REFERENCES dim\_Stores(store\_id, region)

);

**Purpose:  
Captures contextual, non-transactional information that could influence consumer behavior or sales patterns — such as weather, promotions, and seasonal tags.**

**Key Columns:**

* date *(DATE)* – Matches fact\_inventory.date
* store\_id *(VARCHAR)* – Matches fact\_inventory.Store\_ID
* region *(VARCHAR)* – Matches dim\_stores.region
* weather\_condition *(VARCHAR)* – e.g., Sunny, Rainy, Cloudy
* holiday\_promotion *(TINYINT)* – Binary flag (1 = promotion active, 0 = none)
* seasonality *(VARCHAR)* – Winter, Summer, Spring, Autum

# 4. Query Documentation

**Fast/Slow Moving Classification**

**Objective:**  
Classify products based on their average daily sales to determine which items require faster replenishment and which are overstocked or underperforming.

**Key Analysis Points:**

* Calculates average Units\_Sold per product across a defined time window
* Flags products as:
  + **Fast-Moving**: High daily turnover
  + **Slow-Moving**: Low or irregular sales
* Helps optimize **shelf space**, **restocking frequency**, and **working capital allocation**

**ABC Product Analysis**

**Objective:  
Prioritize products based on their contribution to overall revenue, applying the classic Pareto Principle (80/20 Rule).**

**Key Analysis Points:**

* **Computes cumulative revenue per product**
* **Classifies SKUs into:**
  + **A-Class: High-value items (top 70–80% revenue)**
  + **B-Class: Mid-range items (15–20% of revenue)**
  + **C-Class: Low-value items with high volume**
* **Supports strategic pricing, inventory priority, and supplier negotiation**

**Rolling Reorder Point Alerting**

Proactively trigger reorder alerts by comparing current inventory against a rolling average of sales.

**Key Analysis Points:**

* Calculates 7-day or 30-day **moving average** of Units\_Sold
* Compares Inventory\_Level with dynamic thresholds
* Flags low-stock items **before stockout occurs**
* Enables **automated replenishment** planning

**Reorder Point Forecasting**

**Objective:**  
Forecast future inventory requirements using trends in demand over time.

**Key Analysis Points:**

* Uses recent Demand\_Forecast and actual sales data
* Predicts **when** and **how much** to reorder
* Accounts for **lead time**, **seasonality**, and **historical consumption**
* Reduces overstocking and understocking risks

**Category Performance Overview**

**Objective:**  
Analyze sales and profitability trends across different product categories and regions.

**Key Analysis Points:**

* Aggregates metrics like Units\_Sold, Revenue, and Gross Margin by Category and Region
* Identifies top-performing and underperforming segments
* Informs **assortment planning** and **category-specific promotions**

Competitive Pricing Strategy

**Objective:**  
Assess pricing competitiveness by comparing internal pricing with market benchmarks.

**Key Analysis Points:**

* Compares Price vs. Competitor\_Pricing
* Identifies SKUs where prices are:
  + **Above market** (risk of lost sales)
  + **Below market** (opportunity to improve margins)
* Helps design **price-matching**, **discounting**, and **premium pricing** strategies

Stockout & Risk Classification

**Objective:**  
Detect stockout events and classify high-risk items to prevent future disruptions.

**Key Analysis Points:**

* Detects where Inventory\_Level = 0 and Units\_Sold > 0
* Highlights products and stores frequently at risk
* Supports **supply chain risk mitigation** and **supplier performance audits**

Forecast Accuracy & Planning Evaluation

**Objective:**  
Evaluate how closely the Demand\_Forecast aligns with actual Units\_Sold.

**Key Analysis Points:**

* Computes **forecast accuracy metrics** (e.g., MAPE or absolute error)
* Highlights over-forecasted and under-forecasted SKUs
* Enables **continuous improvement** of planning models

# 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 dim\_stores(store\_id, region)

SELECT DISTINCT store\_id, region FROM inventory\_forecasting;

-- Insert products

INSERT IGNORE INTO dim\_Products(product\_id, category)

SELECT DISTINCT product\_id, category FROM inventory\_forecasting;

-- Insert inventory

INSERT INTO fact\_Inventory(date, store\_id, product\_id, region, inventory\_level, units\_sold, units\_ordered, demand\_forecast, Price, Discount, Competitor\_Pricing)

SELECT date, store\_id, product\_id, region, inventory\_level, units\_sold, units\_ordered, demand\_forecast, Price, Discount, Competitor\_Pricing

FROM Inventory\_forecasting;

-- Insert Weather

INSERT IGNORE INTO weather (date, store\_id, region, weather\_condition, holiday\_promotion, seasonality)

SELECT DISTINCT date, store\_id, region, weather\_condition, holiday\_promotion, seasonality

FROM Inventory\_forecasting;

* 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

**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-based solution equips **Urban Retail Co.** with a robust data analytics foundation to improve inventory management and decision-making. By leveraging structured queries, a normalized schema, and optimized performance practices, the company can:

* Identify fast/slow-moving products
* Forecast reorder needs accurately
* Monitor category and pricing performance
* Reduce stockouts and overstocking

With continuous data updates and well-maintained processes, the system enables smarter, data-driven operations across all retail channels.

1. [↑](#footnote-ref-1)