Memo

**To:** SummitStyle Retail Executive Team  
**From:** Vincent Mugambi Mutungi  
**Subject:** DBA Plan to Support SummitStyle’s Analytics Initiative  
**Date:** April 27, 2025

# 1. Executive Summary

This memo presents a Database Administration (DBA) plan for SummitStyle Retail, aimed at supporting the company's new business analytics initiative using Microsoft SQL Server. The plan establishes best practices for securing data access, automating database maintenance, and ensuring reliable data integration for analytics.

Key strategies include:

* Defining structured user roles with appropriate permissions.
* Implementing automated maintenance to ensure system reliability.
* Designing an ETL pipeline to transform transactional data into analytics-ready formats.
* Documenting transactional data structures for clarity and scalability.

Adopting this plan positions SummitStyle to build a scalable, efficient analytics environment capable of delivering timely business insights.

# 2. User Roles and Access Controls

To safeguard data integrity and support efficient operations, three main database roles will be implemented:

* **Administrator:** Full control over database structure and security settings.
* **Analyst:** Read-only access to transactional and reporting tables.
* **Data Entry Specialist:** Insert and update permissions for operational tables; no deletion or schema modification rights.

Implementation using SQL Server syntax:

-- Create roles

CREATE ROLE Admin\_Role;

CREATE ROLE Analyst\_Role;

CREATE ROLE DataEntry\_Role;

GO

-- Grant permissions

GRANT CONTROL ON DATABASE::SummitStyleDB TO Admin\_Role;

GRANT SELECT ON SCHEMA::Sales TO Analyst\_Role;

GRANT SELECT ON SCHEMA::Production TO Analyst\_Role;

GRANT INSERT, UPDATE ON Sales.Customer TO DataEntry\_Role;

GRANT INSERT, UPDATE ON Sales.SalesOrder TO DataEntry\_Role;

DENY DELETE ON Sales.Customer TO DataEntry\_Role;

DENY DELETE ON Sales.SalesOrder TO DataEntry\_Role;

GO

-- Add users to roles

ALTER ROLE Admin\_Role ADD MEMBER [admin\_user];

ALTER ROLE Analyst\_Role ADD MEMBER [analyst\_user];

ALTER ROLE DataEntry\_Role ADD MEMBER [dataentry\_user];

GO

This RBAC model ensures only authorized access based on job duties, minimizing security risks while supporting business functions.

# 3. Basic Maintenance Plan

To maintain system stability and performance, SummitStyle will implement three critical recurring maintenance tasks:

* **Daily Full Database Backups:**  
  Full backups will run nightly to protect against hardware failures or accidental data loss. Backups will be scheduled using SQL Server Agent.
* **Weekly Index Reorganization and Statistics Update:**  
  Indexes will be reorganized and statistics refreshed weekly to optimize query performance. Automated via a scheduled Maintenance Plan.
* **Monthly Database Integrity Checks (DBCC CHECKDB):**  
  Integrity checks will be conducted monthly to detect and repair corruption early. Alerts will notify DBAs of any failures requiring intervention.

Automating these tasks reduces administrative overhead and ensures the database remains healthy and responsive.

# 4. ETL/Data Warehousing Considerations

The ETL process will extract, transform, and load transactional data into a centralized data warehouse for analytics:

* **Extraction:**  
  Transactional data (e.g., sales orders, customer profiles) will be pulled nightly to minimize impact on operational systems.
* **Transformation:**  
  Data will be cleansed, deduplicated, and structured into a star schema suitable for efficient querying and reporting.
* **Loading:**  
  Fact and dimension tables will be populated using batch loads with error handling.

Identified Tables:

* **Fact Table:** Fact.SalesFact—capturing sales transaction metrics (e.g., revenue, quantity sold).
* **Dimension Tables:**
  + Dim.CustomerDimension—customer attributes for segmentation analysis.
  + Dim.DateDimension—temporal attributes for trend analysis.

This ETL approach ensures clean, timely data is available for SummitStyle’s business intelligence tools.

# 5. Data Dictionary (Transactional Database)

| **Table Name** | **Column Name** | **Data Type** | **Description** |
| --- | --- | --- | --- |
| **Sales.SalesOrder** | SalesOrderID | INT (PK) | Unique identifier for each sales order. |
|  | CustomerID | INT (FK) | References Sales.Customer; customer placing order. |
|  | OrderDate | DATETIME | Date and time when order was placed. |
|  | StoreID | INT (FK) | References Sales.Store; store where sale occurred. |
|  | TotalAmount | DECIMAL(10,2) | Total dollar amount of the order. |
|  | OrderStatus | VARCHAR(20) | Status of the order (Pending, Completed, etc.). |
| **Sales.SalesOrderLine** | SalesOrderLineID | INT (PK) | Unique identifier for each line item. |
|  | SalesOrderID | INT (FK) | References Sales.SalesOrder. |
|  | ProductID | INT (FK) | References Production.Product. |
|  | Quantity | INT | Number of units sold. |
|  | UnitPrice | DECIMAL(10,2) | Price per unit sold. |
|  | LineTotal | DECIMAL(10,2) | Total price for the line item. |
| **Production.Product** | ProductID | INT (PK) | Unique identifier for each product. |
|  | ProductName | VARCHAR(100) | Name of the product. |
|  | CategoryID | INT | Category identifier (e.g., clothing, electronics). |
|  | UnitPrice | DECIMAL(10,2) | Standard price of the product. |
|  | ActiveFlag | BIT | Indicates if the product is active (1) or inactive (0). |
| **Sales.Customer** | CustomerID | INT (PK) | Unique identifier for each customer. |
|  | FirstName | VARCHAR(50) | Customer's first name. |
|  | LastName | VARCHAR(50) | Customer's last name. |
|  | Email | VARCHAR(100) | Customer email address. |
|  | JoinDate | DATE | Date customer joined loyalty program. |
|  | LoyaltyStatus | VARCHAR(20) | Customer loyalty level (Gold, Silver, Regular). |
| **HR.Employee** | EmployeeID | INT (PK) | Unique identifier for each employee. |
|  | FirstName | VARCHAR(50) | Employee's first name. |
|  | LastName | VARCHAR(50) | Employee's last name. |
|  | Position | VARCHAR(50) | Employee’s job title. |
|  | StoreID | INT (FK) | Store where employee works. |
| **Sales.Store** | StoreID | INT (PK) | Unique identifier for each store. |
|  | StoreName | VARCHAR(100) | Name of the store location. |
|  | Region | VARCHAR(50) | Geographic region (e.g., Northeast, Midwest). |
|  | OpenDate | DATE | Date the store opened. |
|  | ManagerID | INT (FK) | Manager assigned to the store. |

# Appendix

## Appendix A: Example Queries

The following SQL queries illustrate how to interact with the SummitStyle Retail database:

**Query 1: Top 5 Selling Products (Transactional Database)**

SELECT p.ProductID, p.ProductName, SUM(sol.Quantity) AS TotalQuantitySold, SUM(sol.LineTotal) AS TotalRevenue

FROM Sales.SalesOrderLine sol

JOIN Production.Product p ON sol.ProductID = p.ProductID

GROUP BY p.ProductID, p.ProductName

ORDER BY TotalRevenue DESC

TOP 5;

**Description**: This query identifies the top 5 products by total revenue in the transactional database. It helps SummitStyle Retail prioritize inventory and marketing for high-performing products.

**Query 2: Recent Orders by Store (Transactional Database)**

SELECT s.StoreID, s.StoreName, so.SalesOrderID, so.OrderDate, so.TotalAmount

FROM Sales.SalesOrder so

JOIN Sales.Store s ON so.StoreID = s.StoreID

WHERE so.OrderDate >= DATEADD(DAY, -7, GETDATE())

ORDER BY so.OrderDate DESC;

**Description**: This query retrieves sales orders from the past week, joining Sales.SalesOrder and Sales.Store in the transactional database. It supports store managers in monitoring recent sales activity.

**Query 3: Sales by Customer Loyalty Status (Data Warehouse)**

SELECT c.LoyaltyStatus, COUNT(f.SalesFactID) AS NumberOfSales, SUM(f.SalesAmount) AS TotalSalesAmount

FROM Fact.SalesFact f

JOIN Dim.CustomerDimension c ON f.CustomerKey = c.CustomerKey

GROUP BY c.LoyaltyStatus

ORDER BY TotalSalesAmount DESC;

**Description**: This query analyzes sales by customer loyalty status (Gold, Silver, Regular). It helps identify which loyalty tiers drive the most revenue.

**Query 4: Monthly Sales Trends by Region (Data Warehouse)**

SELECT d.Year, d.MonthName, s.Region, SUM(f.SalesAmount) AS TotalSalesAmount, SUM(f.Quantity) AS TotalQuantitySold

FROM Fact.SalesFact f

JOIN Dim.DateDimension d ON f.DateKey = d.DateKey

JOIN Dim.StoreDimension s ON f.StoreKey = s.StoreKey

GROUP BY d.Year, d.MonthName, s.Region

ORDER BY d.Year, d.Month, s.Region;

**Description**: This query aggregates sales by month and store region. It enables SummitStyle to track regional sales trends over time.

**Query 5: Top Product Categories by Sales (Data Warehouse)**

SELECT p.CategoryID, COUNT(f.SalesFactID) AS NumberOfSales,

SUM(f.SalesAmount) AS TotalSalesAmount

FROM Fact.SalesFact f

JOIN Dim.ProductDimension p ON f.ProductKey = p.ProductKey

GROUP BY p.CategoryID

ORDER BY TotalSalesAmount DESC;

**Description**: This query ranks product categories by total sales revenue. It helps SummitStyle optimize product category strategies.

## Appendix B: Schema Diagram Placeholder

### SummitStyle Transactional Database Diagram

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1: Transactional Database Schema

### SummitStyle Database Warehouse Diagram

A screenshot of a computer

AI-generated content may be incorrect.

Figure 2: Database Warehouse Schema