RETAIL INVENTORY MANAGEMENT | WEEK 1

Team 2

Agenda

Overview

The project is on Retail Inventory Management and Forecasting. It should provide a comprehensive system for managing retail inventory and forecasting future stock requirements.
Effective inventory management would reduce the cost in the retail business while maximizing profits. This project addresses the need for a data-driven inventory control and demand forecasting approach.

01 Scope:

- Develop a SQL database for inventory management
- Implement data warehousing and Python integration
- Create forecasting models using Azure
- Design a user interface for inventory management

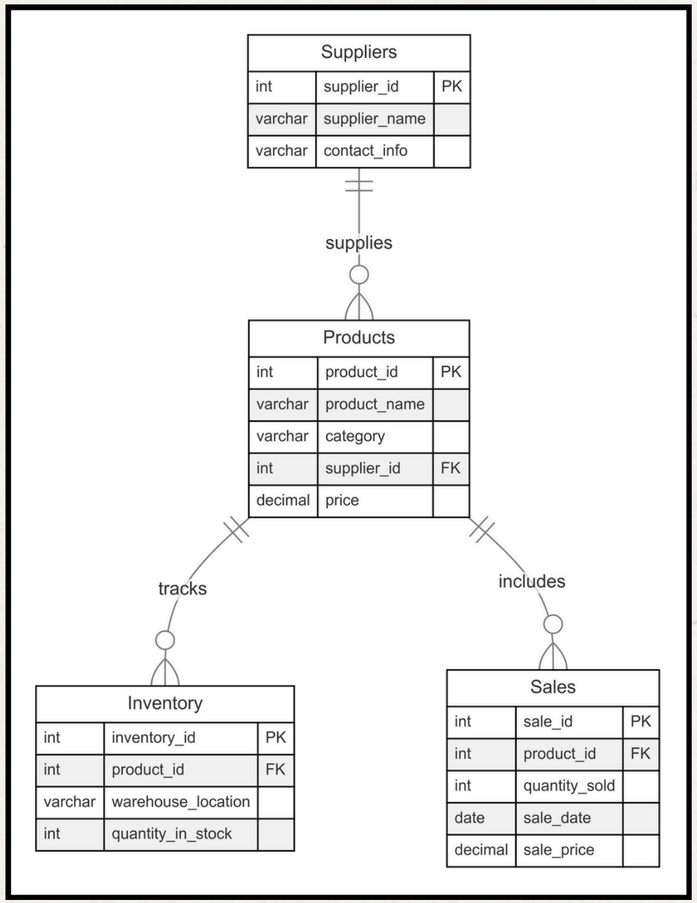
02 Expected Outcomes:

- A fully functional SQL database for inventory tracking
- Efficient data warehousing system
- Accurate forecasting models for inventory demand
- User-friendly interface for inventory management and forecasting

Week 1 Focus: SQL Database Design and Implementation

- **Task:** Design a SQL database schema for managing retail inventory, including tables for products, stock levels, sales, and suppliers.
- **Deliverable:** Implemented SQL database with sample data and documentation

Database Schema



ER Diagram

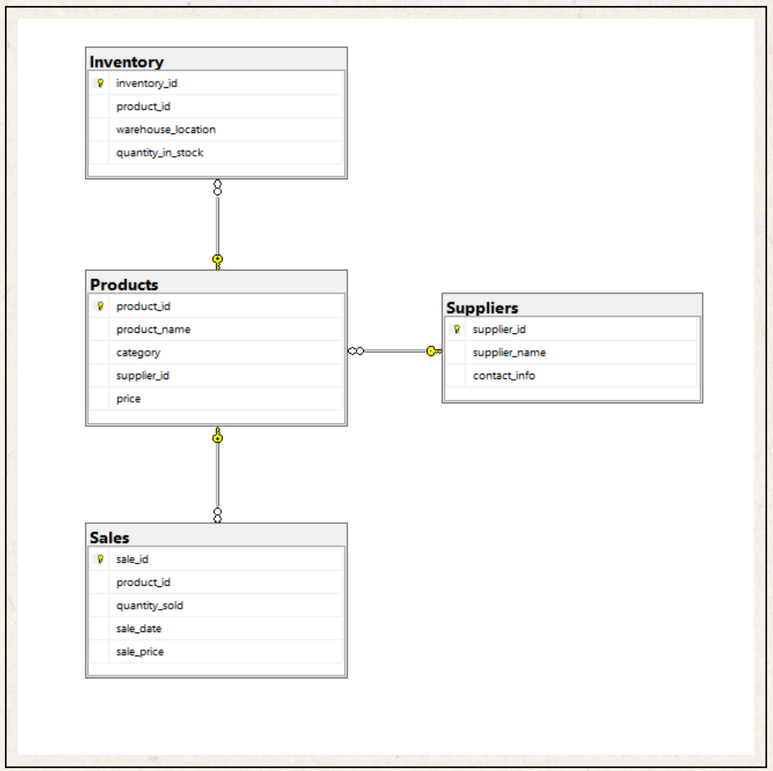


Table Insertion Queries


```
CREATE TABLE [dbo].[Suppliers] (
    [supplier_id] INT PRIMARY KEY,
    [supplier_name] VARCHAR(255) NOT NULL,
    [contact_info] VARCHAR(255) NOT NULL
) ON [PRIMARY];
```

Suppliers' Table

```
CREATE TABLE [dbo].[Products] (
    [product_id] INT PRIMARY KEY,
    [product_name] VARCHAR(255) NOT NULL,
    [category] VARCHAR(100),
    [supplier_id] INT,
    [price] DECIMAL(10, 2),
    FOREIGN KEY (supplier_id) REFERENCES dbo.Suppliers(supplier_id)
) ON [PRIMARY];
```

Products' Table

Table Insertion Queries


```
CREATE TABLE [dbo].[Inventory] (
    [inventory_id] INT PRIMARY KEY,
    [product_id] INT,
    [warehouse_location] VARCHAR(255),
    [quantity_in_stock] INT,
    FOREIGN KEY (product_id) REFERENCES dbo.Products(product_id)
) ON [PRIMARY];
```

Inventory’s Table

```
CREATE TABLE [dbo].[Sales] (
    [sale_id] INT PRIMARY KEY,
    [product_id] INT,
    [quantity_sold] INT,
    [sale_date] DATE,
    [sale_price] DECIMAL(10, 2),
    FOREIGN KEY (product_id) REFERENCES dbo.Products(product_id)
) ON [PRIMARY];
```

Sales’ Table

SQL Queries

1. Products Running Low on Stock:

```sql
SELECT p.product_name, i.warehouse_location, i.quantity_in_stock
FROM Products p
JOIN Inventory i ON p.product_id = i.product_id
WHERE i.quantity_in_stock < 10;
```

2. Total Sales Revenue for Each Product:

```sql
SELECT p.product_name, SUM(s.quantity_sold * s.sale_price) AS total_sales_revenue
FROM Products p
JOIN Sales s ON p.product_id = s.product_id
GROUP BY p.product_name
ORDER BY total_sales_revenue DESC;
```

SQL Queries

3. Daily Sales Summary:

```sql
SELECT s.sale_date, SUM(s.sale_price * s.quantity_sold) AS daily_sales
FROM Sales s
GROUP BY s.sale_date
ORDER BY s.sale_date;
```

4. Supplier-wise Product Distribution:

```sql
SELECT su.supplier_name, COUNT(p.product_id) AS number_of_products
FROM Suppliers su
JOIN Products p ON su.supplier_id = p.supplier_id
GROUP BY su.supplier_name;
```

SQL Queries

5. Products with Highest Sales Volume:

```sql
SELECT p.product_name, SUM(s.quantity_sold) AS total_quantity_sold
FROM Products p
JOIN Sales s ON p.product_id = s.product_id
GROUP BY p.product_name
ORDER BY total_quantity_sold DESC;
```

6. Sales by Warehouse Location:

```sql
SELECT i.warehouse_location, SUM(s.quantity_sold * s.sale_price) AS warehouse_sales
FROM Inventory i
JOIN Sales s ON i.product_id = s.product_id
GROUP BY i.warehouse_location
ORDER BY warehouse_sales DESC;
```

SQL Queries

7. Average Sales Price for Each Product:

```sql
SELECT p.product_name, AVG(s.sale_price) AS avg_sale_price
FROM Products p
JOIN Sales s ON p.product_id = s.product_id
GROUP BY p.product_name
ORDER BY avg_sale_price DESC;
```

8. Inventory Status by Supplier:

```sql
SELECT su.supplier_name, SUM(i.quantity_in_stock) AS total_stock
FROM Suppliers su
JOIN Products p ON su.supplier_id = p.supplier_id
JOIN Inventory i ON p.product_id = i.product_id
GROUP BY su.supplier_name
ORDER BY total_stock DESC;
```

SQL Queries

9. Products Never Sold:

```sql
SELECT p.product_name
FROM Products p
LEFT JOIN Sales s ON p.product_id = s.product_id
WHERE s.sale_id IS NULL;
```

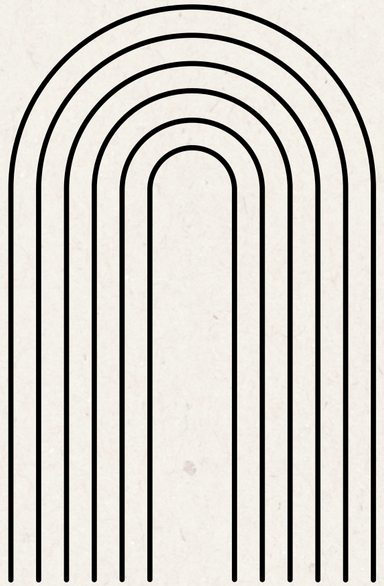
10. Most Popular Categories by Sales Revenue:

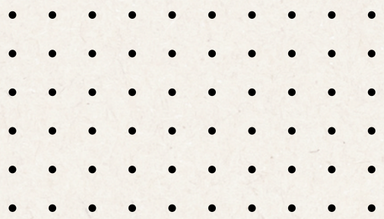
```sql
SELECT p.category, SUM(s.quantity_sold * s.sale_price) AS total_category_sales
FROM Products p
JOIN Sales s ON p.product_id = s.product_id
GROUP BY p.category
ORDER BY total_category_sales DESC;
```

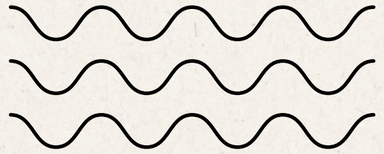
Conclusion

THE SQL DATABASE DESIGN FOR THIS PROJECT ESTABLISHES A ROBUST FOUNDATION FOR RETAIL INVENTORY MANAGEMENT AND FORECASTING. KEY FEATURES INCLUDE:

1. RELATIONAL STRUCTURE: THE SCHEMA CREATES RELATIONSHIPS BETWEEN TABLES USING FOREIGN KEY CONSTRAINTS. FOR EXAMPLE, THE PRODUCTS TABLE LINKS TO SUPPLIERS THROUGH SUPPLIER_ID, WHILE INVENTORY AND SALES CONNECT TO PRODUCTS VIA PRODUCT_ID.
2. ANALYTICAL CAPABILITIES: IMPLEMENTED SQL QUERIES OFFER DIVERSE INSIGHTS INTO INVENTORY LEVELS, SALES PERFORMANCE, SUPPLIER RELATIONSHIPS, AND PRODUCT POPULARITY. THESE ANALYTICAL TOOLS SUPPORT DATA-DRIVEN DECISION-MAKING IN INVENTORY MANAGEMENT.
3. COMPREHENSIVE TESTING: TABLES ARE POPULATED WITH DUMMY DATA TO THOROUGHLY TEST SQL QUERIES, ENSURING SYSTEM RELIABILITY AND PERFORMANCE.







THIS WELL-STRUCTURED DATABASE FORMS A SOLID BASE FOR SUBSEQUENT PROJECT PHASES, INCLUDING DATA WAREHOUSING, FORECASTING MODEL DEVELOPMENT, AND USER INTERFACE DESIGN. IT PROVIDES THE NECESSARY DATA INFRASTRUCTURE TO SUPPORT EFFICIENT INVENTORY MANAGEMENT AND ACCURATE DEMAND FORECASTING IN THE RETAIL SECTOR.

Our team

- Omar Baher
- Yahia Mohamed
- Eyad gamal
- Habiba Sherif
- Mariam Ibrahim
- Eriny Mouner

Thank you