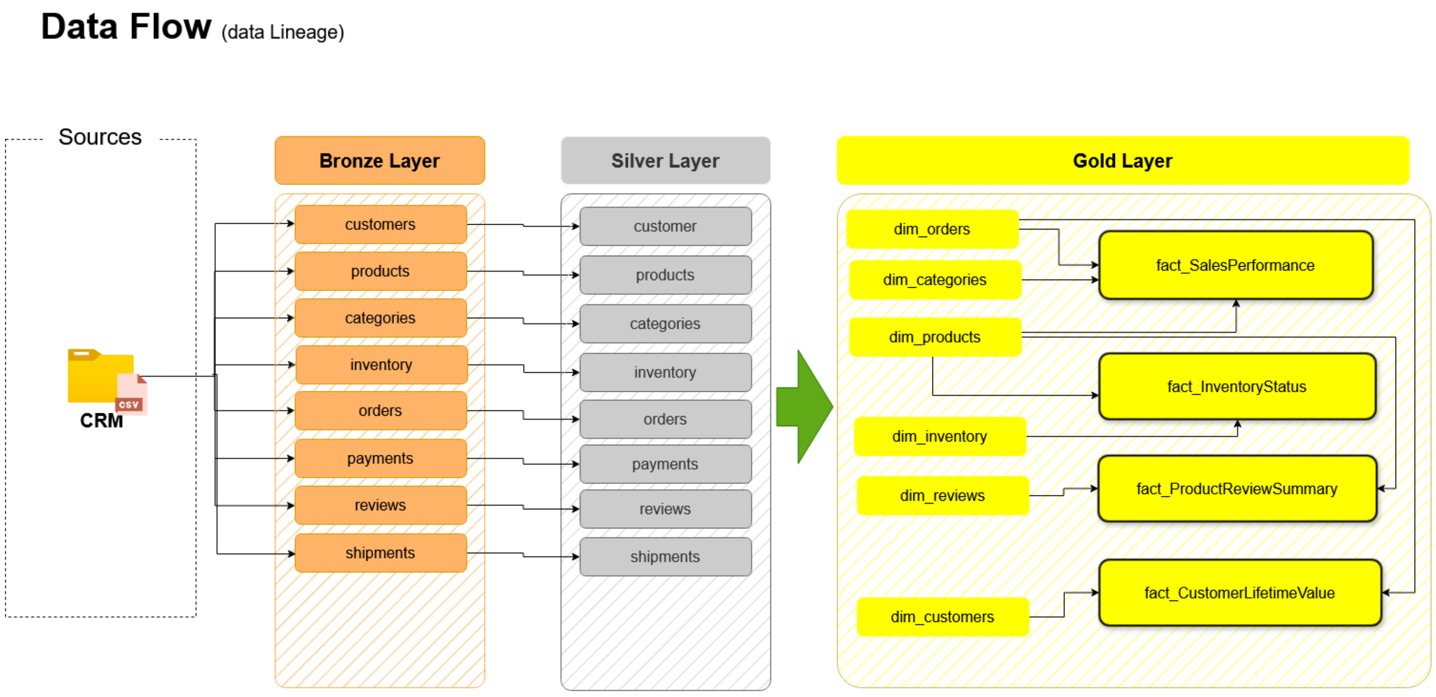
Documentation

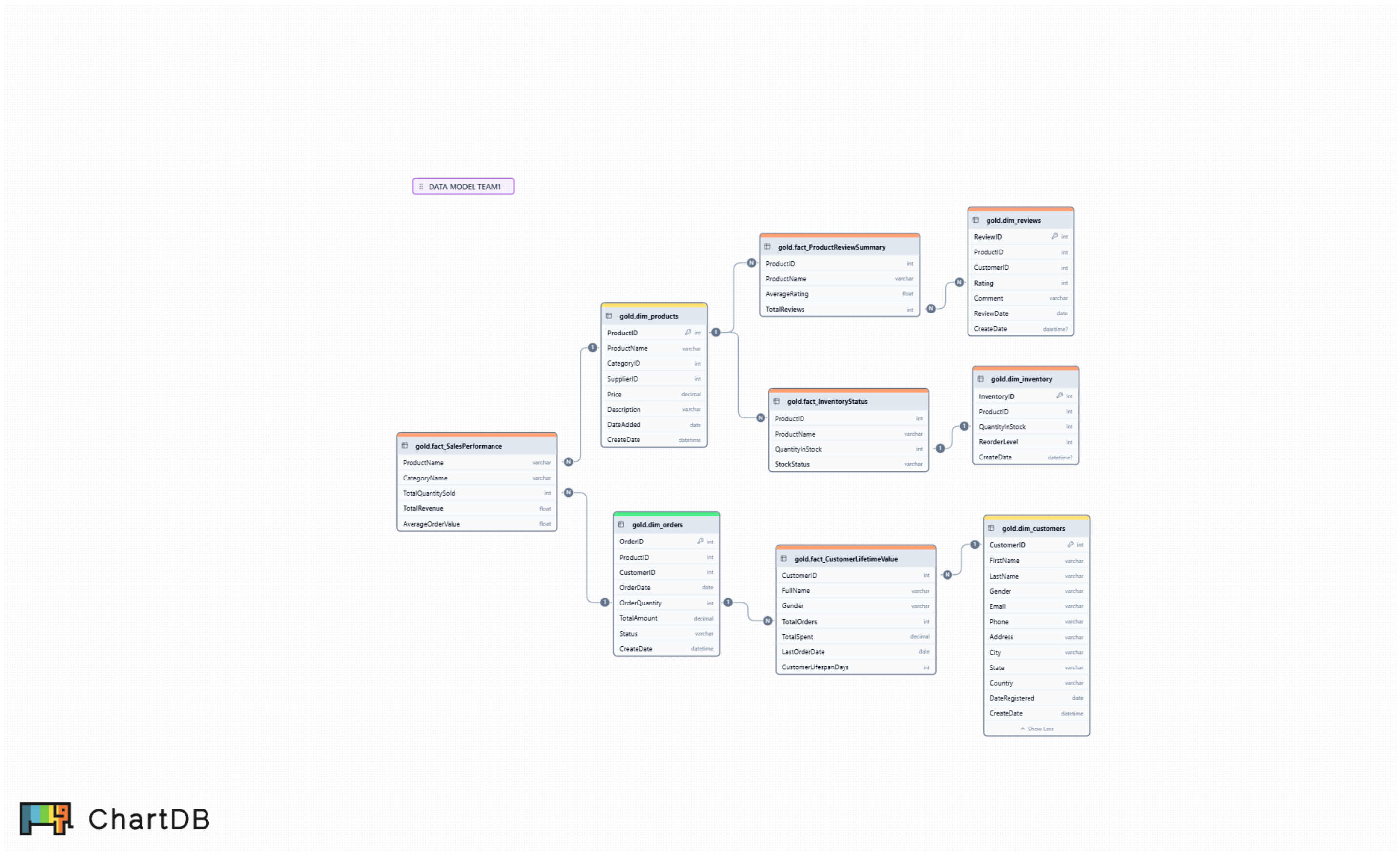
PROJECT: Online_store_DW_project_team1

Online store data warehouse designing project

Data Flow



Data Model



gold.fact_SalesPerformance	
ProductName	varchar
CategoryName	varchar
TotalQuantitySold	int
TotalRevenue	float
AverageOrderValue	float

gold.dim_products	
ProductID	int
ProductName	varchar
CategoryID	int
SupplierID	int
Price	decimal
Description	varchar
DateAdded	date
CreateDate	datetime

gold.fact_ProductReviewSummary	
ProductID	int
ProductName	varchar
AverageRating	float
TotalReviews	int

gold.dim_reviews	
ReviewID	int
ProductID	int
CustomerID	int
Rating	int
Comment	varchar
ReviewDate	date
CreateDate	datetime?

gold.fact_InventoryStatus	
ProductID	int
ProductName	varchar
QuantityInStock	int
StockStatus	varchar

gold.dim_inventory	
InventoryID	int
ProductID	int
QuantityInStock	int
ReorderLevel	int
CreateDate	datetime?

gold.dim_orders	
OrderID	int
ProductID	int
CustomerID	int
OrderDate	date
OrderQuantity	int
TotalAmount	decimal
Status	varchar
CreateDate	datetime

gold.fact_CustomerLifetimeValue	
CustomerID	int
FullName	varchar
Gender	varchar
TotalOrders	int
TotalSpent	decimal
LastOrderDate	date
CustomerLifespanDays	int

gold.dim_customers	
CustomerID	int
FirstName	varchar
LastName	varchar
Gender	varchar
Email	varchar
Phone	varchar
Address	varchar
City	varchar
State	varchar
Country	varchar
DateRegistered	date
CreateDate	datetime

Data Warehouse Project Documentation

Table of Contents

1. Project Overview

Purpose:

The purpose of this Data Warehouse project is to provide a structured and reliable framework for analyzing data from an online store. The system supports advanced analytics by organizing data into three distinct layers: **Bronze**, **Silver**, and **Gold**.

Target Audience:

- **Technical Users**: Data Engineers, Database Administrators, and Developers.
- **Business Users**: Data Analysts, Business Intelligence Teams, and Decision-Makers.

Key Objectives:

1. **Bronze Layer**: Load raw data from source CSV files with minimal transformations.
2. **Silver Layer**: Perform ETL (Extract, Transform, Load) processes to clean, validate, and enrich the data.
3. **Gold Layer**: Create analytical views for business intelligence purposes, such as sales performance, customer lifetime value, inventory status, and product review summaries.

---

2. Database Initialization Script

Purpose:

This script initializes the database and sets up the necessary schemas (`bronze`, `silver`, `gold`) for the Data Warehouse.

Code Snippet:

```
USE master;
GO

-- Drop and recreate the 'Team1' database
IF EXISTS (SELECT 1 FROM sys.databases WHERE name = 'Team1')
BEGIN
    ALTER DATABASE Team1 SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE Team1;
END;
GO

CREATE DATABASE Team1;
GO

USE Team1;
GO

-- Create Schemas
```

```
CREATE SCHEMA bronze;
GO

CREATE SCHEMA silver;
GO

CREATE SCHEMA gold;
GO
```

Warning:

Running this script will permanently delete all data in the `Team1` database. Ensure proper backups before execution.

---

3. Bronze Layer

Purpose:

The Bronze layer serves as the landing zone for raw data. It contains exact copies of the source tables without any transformations.

Table Definitions:

DDL Script:

- Creates tables for `Customers` , `Products` , `Categories` , `Orders` , `Shipments` , `Inventories` , and `Reviews` .
- Each table mirrors the structure of the corresponding source CSV file.

Example Table Definition:

```
CREATE TABLE bronze.orders (
    OrderID       INT,
    ProductID     INT,
    CustomerID    INT,
    OrderDate     DATE,
    OrderQuantity INT,
    TotalAmount   DECIMAL(10, 2),
    Status        NVARCHAR(20) NOT NULL CHECK (Status IN ('Cancelled', 'Sh
);
```

Bulk Insert Process:

1. **Script Name:** `proc_load_bronze.sql`
2. **Purpose:** Bulk inserts data from CSV files into the corresponding Bronze tables.
3. **Key Features:**
    - Automates the ingestion of raw data.
    - Ensures data integrity by matching column structures.

---

4. Silver Layer

Purpose:

The Silver layer performs ETL operations to transform and enhance the data from the Bronze layer. This ensures high-quality, standardized data for downstream analysis.

Scripts:

1. Silver DDL:

- Defines transformed tables with additional columns (e.g., `Created_At`).
- Includes constraints, indexes, and relationships.

2. Silver Procedure:

- Pulls data from the Bronze layer.
- Applies transformations such as data cleaning, deduplication, and enrichment.

Example Transformation Logic:

```
INSERT INTO silver.orders (
    OrderID,
    ProductID,
    CustomerID,
    OrderDate,
    OrderQuantity,
    TotalAmount,
    Status,
    CreatedAt
)
```

```
SELECT
    OrderID,
    ProductID,
    CustomerID,
    OrderDate,
    OrderQuantity,
    TotalAmount,
    Status,
    GETDATE() AS CreatedAt
FROM bronze.orders
WHERE Status IN ('Pending', 'Shipped', 'Delivered');
```

Stored Procedures:

1. **PlaceOrder**:
    - Inserts new orders into the `silver.orders` table.
    - Automatically generates unique `OrderID` values using an `IDENTITY` column or sequence.
2. **UpdateInventory**:
    - Updates inventory levels based on order and shipment data.
    - Ensures consistency between `silver.inventory` and `silver.orders`.

---

5. Gold Layer

Purpose:

The Gold layer provides analytical views for business intelligence. These views aggregate and summarize data to support specific use cases.

Views:

1. Dimension Tables:

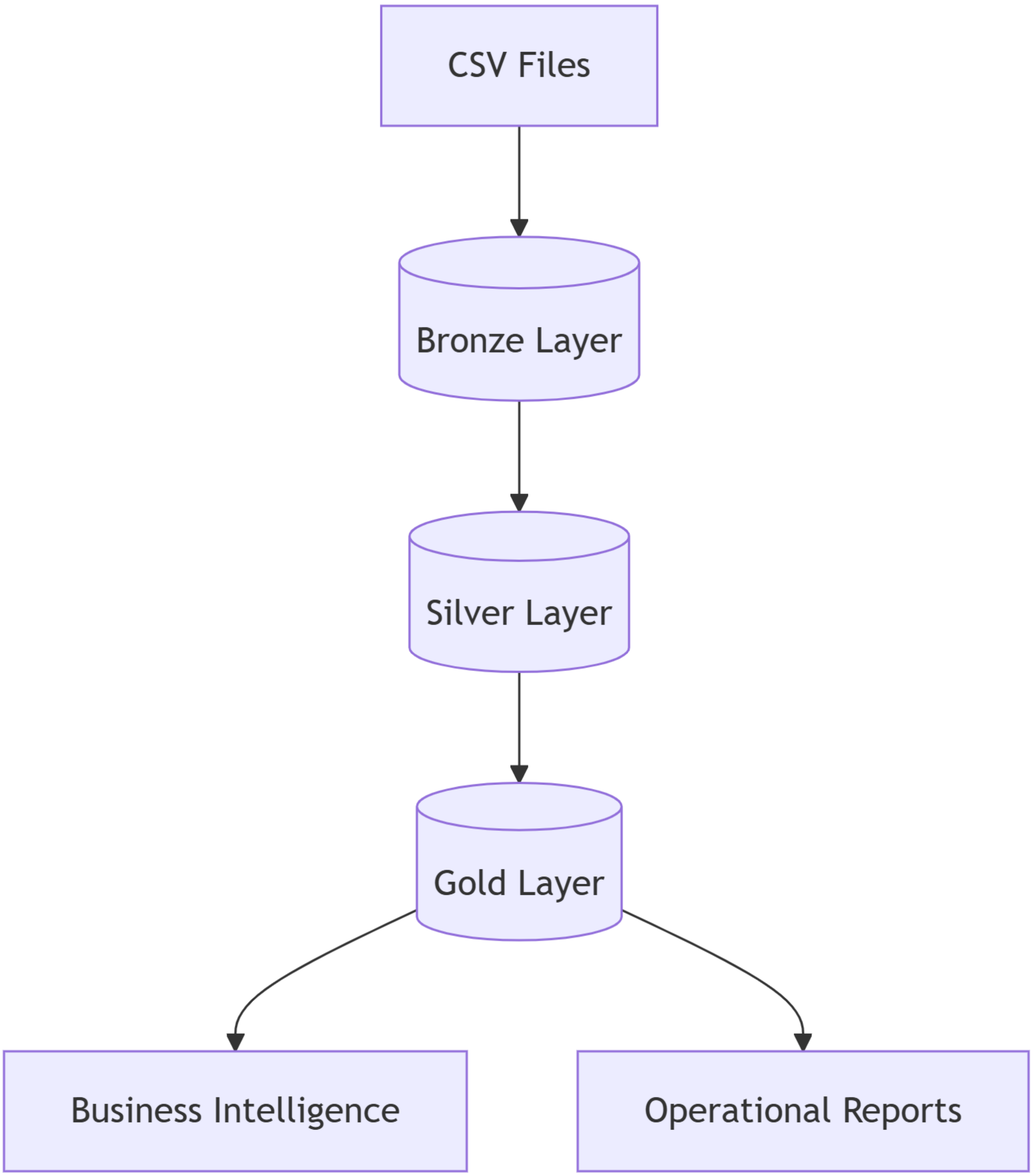
- `DimCustomers` : Customer-related attributes.
- `DimProducts` : Product-related attributes.
- `DimCategories` : Category-related attributes.

2. Fact Views:

- `SalesPerformance` : Analyzes revenue, order quantities, and trends over time.
- `CustomerLifetimeValue` : Calculates the total value contributed by each customer.
- `InventoryStatus` : Tracks stock levels and identifies low-stock items.
- `ProductReviewSummary` : Aggregates product ratings and feedback.

Example View:

```
CREATE VIEW gold.SalesPerformance AS
SELECT
    YEAR(OrderDate) AS Year,
    MONTH(OrderDate) AS Month,
    SUM(TotalAmount) AS TotalRevenue,
    COUNT(OrderID) AS TotalOrders
FROM silver.orders
GROUP BY YEAR(OrderDate), MONTH(OrderDate);
```



---

6. Automation with SQL Agent

Purpose:

Automate the entire ETL process to ensure timely and consistent data updates.

Steps:

1. **Schedule Jobs**:
   - Use SQL Server Agent to schedule recurring jobs for:
     - Bulk inserting data into the Bronze layer.
     - Running ETL processes in the Silver layer.
     - Refreshing views in the Gold layer.
2. **Job Configuration**:
   - Define job steps for each layer.
   - Include error handling and logging to track job execution.

Example Job:

```sql
EXEC msdb.dbo.sp_add_job @job_name = 'Bronze_Data_Ingestion';
EXEC msdb.dbo.sp_add_jobstep @job_name = 'Bronze_Data_Ingestion',
    @step_name = 'Bulk_Insert',
    @command = 'EXEC proc_load_bronze;';
EXEC msdb.dbo.sp_add_schedule @schedule_name = 'Daily_8AM',
    @freq_type = 4, -- Daily
    @active_start_time = 080000;
EXEC msdb.dbo.sp_attach_schedule @job_name = 'Bronze_Data_Ingestion',
    @schedule_name = 'Daily_8AM';
```

---

7. Deployment and Maintenance

1. Deployment Checklist:

- Verify that all scripts are tested and error-free.
- Ensure proper permissions for SQL Server Agent jobs.
- Schedule regular backups of the `Team1` database.

2. Maintenance Tasks:

- Monitor job execution logs for errors.
- Periodically review and optimize queries for performance.
- Update transformations and views as business requirements evolve.

---

8. Conclusion

This documentation outlines the architecture, scripts, and processes for the Data Warehouse project. By following this structure, you can ensure a robust, scalable, and maintainable system for analyzing online store data.

For further details, refer to the individual scripts and files provided in the repository.

Let me know if you need further adjustments!