STOCK WARNING SYSTEM

- **Objective**: The client in the retail industry needs an automated **Stock Warning System** to predict stock shortages based on current inventory levels, sales velocity, and replenishment schedules. The system will provide daily updates, flagging products with low stock levels within the next 30 days.
- **Technology Stack**: The system will be built using **Azure SQL Database** for data storage, **Azure Data Factory (ADF)** for automation and **Azure Data Studio** for running SQL queries.

SQL QUERY

This query:

- Calculates daily sales rate based on historical sales.
- Projects future stock levels considering replenishments.
- Flags products with low stock (below a specified threshold, e.g., 10 units).

```
CREATE TABLE dbo.Inventory (
    ProductID INT PRIMARY KEY,
    CurrentStock INT NOT NULL
);

CREATE TABLE dbo.Products (
    ProductID INT PRIMARY KEY,
    ProductName NVARCHAR(100),
    Category NVARCHAR(50),
    SupplierID INT,
    ReorderLeadTime INT NOT NULL
);

CREATE TABLE dbo.Sales (
    SaleID INT PRIMARY KEY,
    ProductID INT,
    Quantity INT NOT NULL,
    SaleDate DATE NOT NULL
);

CREATE TABLE dbo.Replenishment (
    ReplenishmentID INT PRIMARY KEY,
    ProductID INT,
```

```
    Quantity INT NOT NULL,
    DeliveryDate DATE NOT NULL
);

INSERT INTO dbo.Inventory (ProductID, CurrentStock)
VALUES
    (1, 50),
    (2, 30),
    (3, 0),
    (4, 20),
    (5, 70);

INSERT INTO dbo.Products (ProductID, ProductName, Category, SupplierID, ReorderLeadTime)
VALUES
    (1, 'Laptop', 'Electronics', 101, 7),
    (2, 'Headphones', 'Electronics', 102, 5),
    (3, 'Monitor', 'Electronics', 103, 10),
    (4, 'Mouse', 'Electronics', 104, 3),
    (5, 'Keyboard', 'Electronics', 105, 4);

INSERT INTO dbo.Sales (SaleID, ProductID, Quantity, SaleDate)
VALUES
    (1, 1, 5, '2024-11-01'),
    (2, 2, 3, '2024-11-02'),
    (3, 3, 2, '2024-11-03'),
    (4, 4, 1, '2024-11-04'),
    (5, 5, 4, '2024-11-05');

INSERT INTO dbo.Replenishment (ReplenishmentID, ProductID, Quantity, DeliveryDate)
VALUES
    (1, 1, 20, '2024-11-10'),
    (2, 2, 10, '2024-11-12'),
    (3, 3, 15, '2024-11-15'),
    (4, 4, 25, '2024-11-18'),
    (5, 5, 30, '2024-11-20');

CREATE PROCEDURE dbo.StockPrediction
    @StartDate DATE = NULL,  -- Default is NULL, can be passed in
    @ProjectionDays INT = 30,  -- Default to 30 days for stock projection
    @LowStockThreshold INT = 10  -- Default threshold for low stock
```

```
AS
BEGIN
    -- Use the passed parameters (or defaults) in the query
    IF @StartDate IS NULL
        SET @StartDate = DATEADD(day, -30, GETDATE());  -- Default to 30 days ago if not provided

    WITH SalesVelocity AS (
        SELECT ProductID,
            AVG(Quantity) OVER(PARTITION BY ProductID ORDER BY SaleDate DESC ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS DailySalesRate
        FROM dbo.Sales
        WHERE SaleDate >= @StartDate  -- Use passed start date
    ),
    StockProjection AS (
        SELECT i.ProductID,
            i.CurrentStock - (sv.DailySalesRate * @ProjectionDays) + COALESCE(SUM(r.Quantity), 0) AS
ProjectedStock
        FROM dbo.Inventory i
        JOIN SalesVelocity sv ON i.ProductID = sv.ProductID
        LEFT JOIN dbo.Replenishment r ON i.ProductID = r.ProductID
            AND r.DeliveryDate BETWEEN GETDATE() AND DATEADD(day, @ProjectionDays, GETDATE())  --
Use dynamic projection days
        GROUP BY i.ProductID, i.CurrentStock, sv.DailySalesRate
    )
    -- Insert the results into the StockWarning table, using the low stock threshold parameter
    SELECT ProductID, ProjectedStock
    INTO dbo.StockWarning
    FROM StockProjection
    WHERE ProjectedStock < @LowStockThreshold;  -- Use passed low stock threshold
END;

EXEC dbo.StockPrediction @StartDate = '2024-11-01', @ProjectionDays = 30, @LowStockThreshold = 10;

SELECT * FROM dbo.StockWarning;
```

This query is designed for daily refreshes and calculates the projected stock based on recent sales and replenishments.

	ProductID	ProjectedStock
1	4	-10
2	3	-60
3	2	-60
4	5	-50
5	1	-100

Figure 1: StockWarning table after running this query

AZURE DATA FACTORY PIPELINE

Pipeline Overview: Your pipeline uses two stored procedure activities:

1. DropStockWarning: Drops the StockWarning table before executing the prediction to avoid conflicts.
2. ExecuteStockForecast: Runs the StockPrediction stored procedure to insert new data into the StockWarning table.

Automation: The pipeline is scheduled to run daily, ensuring daily stock projections are updated.

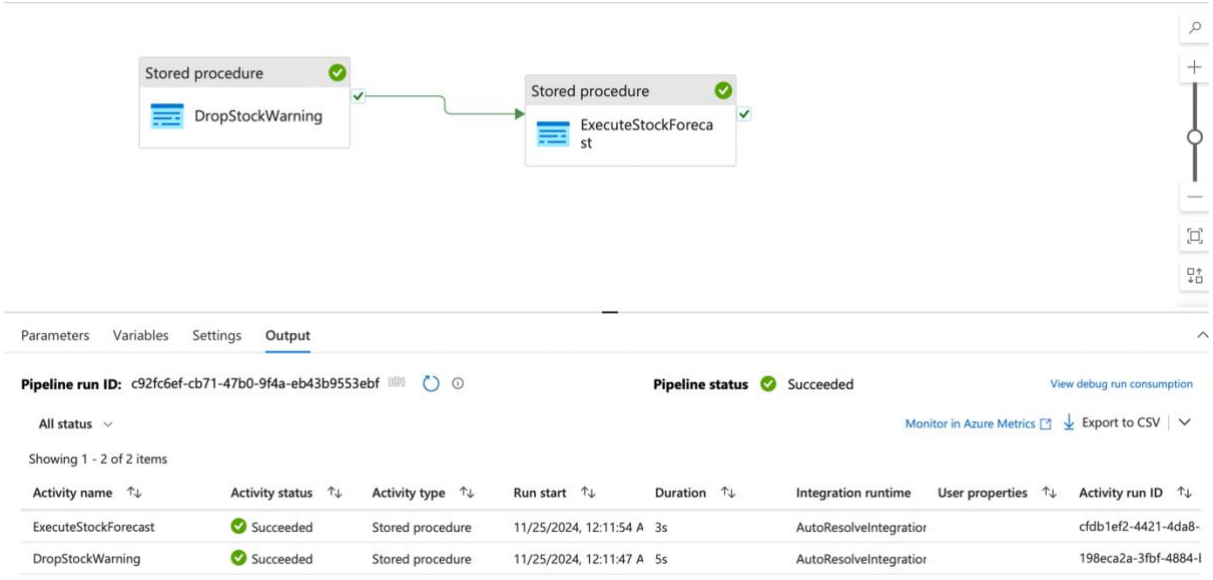

Figure 2: The ADF pipeline design

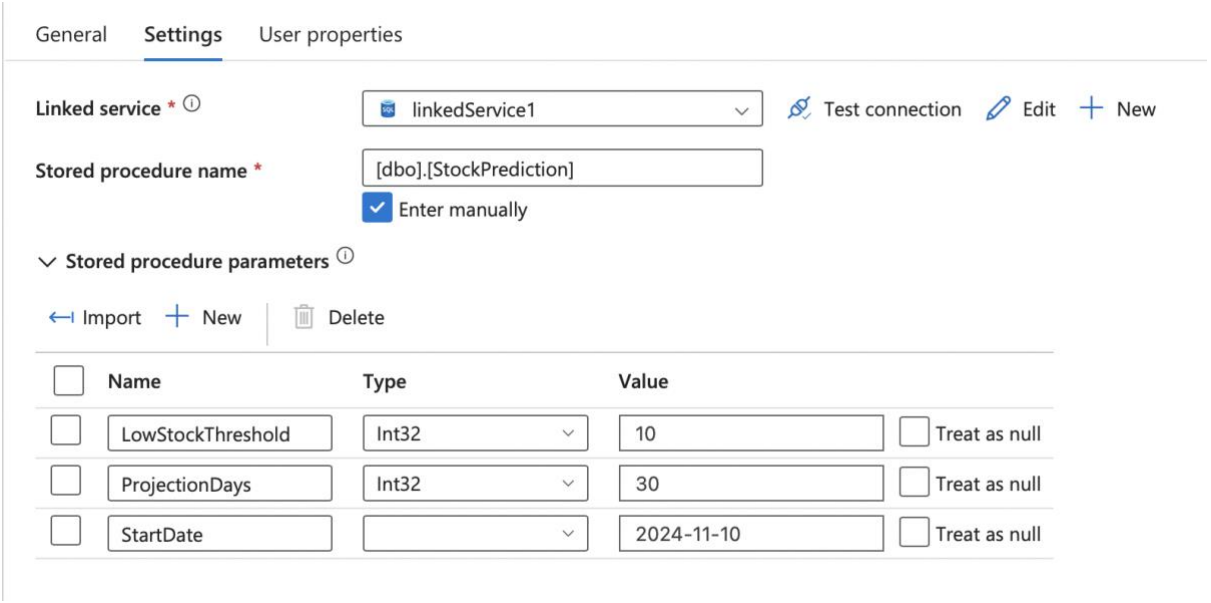

Figure 3: Stored Procedure Activity settings

Parameters: The pipeline uses three parameters (StartDate, ProjectionDays, LowStockThreshold) that are passed to the stored procedure. These allow dynamic control over the start date for sales data, the number of days for stock projection, and the threshold for low stock detection.

CONCLUSION

Challenges:

- Handling the "table already exists" error: The challenge was ensuring that the StockWarning table is cleared or dropped before the stored procedure runs. This was addressed by adding a DropStockWarning activity in the pipeline.
- Parameter mapping in ADF: Ensuring that the correct parameters were passed from ADF to the stored procedure was another challenge, which was resolved by correctly mapping pipeline parameters to stored procedure parameters.

Impact:

- The **automated solution** allows the client to monitor stock levels proactively, avoiding stockouts.
- **Metrics**: The system can be evaluated by monitoring the number of **low-stock alerts** generated, the **accuracy** of stock projections, and the client's **inventory restocking efficiency** based on the predictions.