# Part 3

## Question 1

If Bob wants to see updates thee times per day in data warehouse, Job Agent must be implemented. Details of design can be seen in Figure 1. At the bottom of the figure data warehouse – Product Dimension is updated. Implementation of this incremental load responsible for handling these changes can be found in *Appendix A – Incremental Load Source Code*.

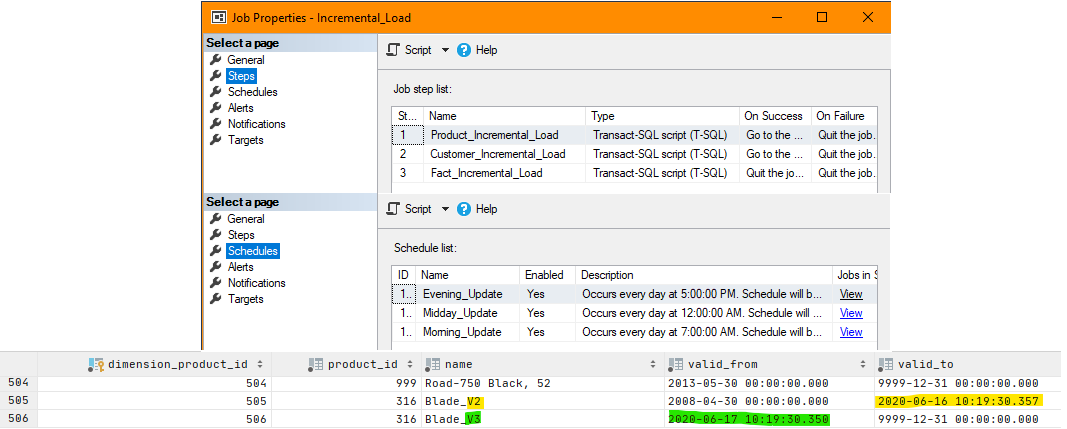


Figure Job agent and result

## Question 2

To see demand based on small changes in a price, a Bridge Table containing price range should be implemented. Reason is to avoid overloading data warehouse with big amount of data. Product Dimension will hold reference to Price Range bridge entity (Figure 2), and whenever the new price is out of this range, Dimension is updated accordingly (*valid\_to* set to yesterday’s date, new entry created with new reference).

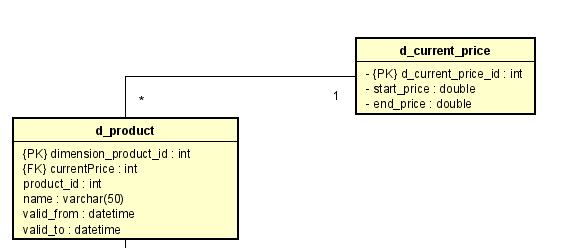


Figure Bridge table Current Price

The table will be pre-populated with data, containing various ranges between minimal and maximum price in relational database. Later, relevant tuple will be looked up based on *currentPrice* and foreign key assigned.

CREATE TABLE StagingDatabase.staging.d\_current\_price  
(  
 d\_current\_price\_id INT IDENTITY,  
 start\_price decimal(10,2),  
 end\_price decimal(10,2),  
 PRIMARY KEY (d\_current\_price\_id)  
);  
DECLARE @StartPrice decimal(10,2) = 0.00;  
DECLARE @EndPrice decimal(10,2) = 199.99;  
DECLARE @Increment decimal(10,2) = 200.00;  
  
DECLARE @MaxPrice decimal(10,2) = (SELECT *MAX*(ListPrice)  
 FROM AdventureWorks2017.Production.Product);  
  
WHILE @EndPrice <= @MaxPrice  
 BEGIN  
 INSERT INTO StagingDatabase.staging.d\_current\_price (start\_price, end\_price) VALUES (  
 @StartPrice,  
 @EndPrice  
 );  
 SET @StartPrice = @StartPrice + @Increment;  
 SET @EndPrice = @EndPrice + @Increment;  
 END

# Appendices

#### Appendix A – Incremental Load Source Code