Use of AI to create Pipeline and Stored Procedure in SQL Server

Let's go step by step, I think it's best, I'll tell you the rules and you'll create separate procedures for me, for each table we analyze. We check if the records do not exist in the destination table, then are inserted, if they are already there, nothing of course is inserted.

In the case of FacTransactions we change the logic, we check the TransactionDate field, that is, the maximum date of the movements, if there are movements above that date, they are added to the FacTransactions table, otherwise not.

Please use the source and destination database names in stored procedures: Source: GymSourceDataOLTP, Destination: GITWorldGym

There are several Dimensions tables in the Destination, their names are DimMember, DimProduct, FacTransactions.

*Procedure for DimMember:*

The DimMember table is fill out of the Member table. Exact tables

*Procedure for DimProduct:*

The DimProduct table is made up of data from different tables including OneDaypassCategory, Merchandised, SpecialEvents and the MemberShip table.

The DimProduct table has the following structure five columns ProductID, ProductName, ProductPrice, ProductType and NameLocation. We are going to make a link of the columns and tables for DimProduct.

The ProductType column is created as a category from the Table it comes from. For example, if the data comes from MemberShip, the ProductType field would have the category "Membership", if it comes from OneDayPassCategory the value of ProductType is "OneDayPass", if it comes from the SpecialEvents Table the ProductType is "Event" and if it comes from the Merchandise table the ProductType is "Merchandise".

The NameLocation column is almost empty "", except when comes from SpecialEvents table

Let's now analyze what the Field Link for each table would be like.

*Merchandise table*

The columns in DimProduct would look like:

ProductID = MrchID

ProductName=MrchName

ProductPrice=MrchPrice

ProductType="Merchandise”; which already comes from that table

NameLocation= "“; empty

*SpecialEvents table*

The columns in DimProduct would look like:

ProductID = CCID

ProductName=EventType

ProductPrice=AmountCharged

ProductType="Event”; which already comes from that SpecialEvents table

NameLocation= CCNameLocation

*MemberShip Table*

The columns in DimProduct would look like:

ProductID = MshpID

ProductName=MshpName

ProductPrice=MshpPrice

ProductType="Membership”; which already comes from that table

NameLocation= "“; empty

*OneDayPassCategory table*

The columns in DimProduct would look like:

ProductID = PassCatID

ProductName=CatName

ProductPrice=Price

ProductType="OneDayPass”; which already comes from that table

NameLocation= "“; empty

*Procedure for FacTransactions:*

We finish with the Dimension Tables. Remember In the case of FacTransactions we change the logic, we check the TransactionDate field, that is, the maximum date of the movements, if there are movements above that date, they are added to the FacTransactions table, otherwise not.

Let's now see the Link of the transaction table (FacTransactions) with the other source tables.

TransactionID column: it is an autoincremental column and unique key

*Member table relationship*

TransactionsID checks if there are new members in Member, if there are then there were membership sales

TransactionDate= MemberShipDate

MemberID=MemberID

ProductID= CategoryID

Quantity=1; always 1, meaning only one member per row

Amount= Quantity\*MshpPrice ; from the MemberShip table

*OneDayGuestPass table relationship*

TransactionsID checks if there are new registrations in OneDayGuestPass, if there are then there were OneDayPass sales

TransactionDate=PassDate

MemberID=MembID

ProductID= PassCatID

Quantity= 1, always 1, that is, only one per row (Guest)

Amount= Quantity\*Price; from the OneDayPassCategory table

*Merchandise table relationship*

TransactionsID checks if there are new members on SoldVia, if there are then there were Merchandise sales.

TransactionDate= StrDate; which comes from the SalesTransactions table. Since there is a link between the SoldVia table and SalesTransactions based on the StrID, that is, there is a join through that field, from there the date is taken.

MemberID=MembID; which is in the SalesTransactions table applying the same logic. Since there is a link between the SoldVia table and SalesTransactions based on the StrID, that is, there is a join through that field, from there the MembID is taken.

ProductID= MrchID; from the SoldVia table

Quantity= Quantity; from the SoldVia table

Amount= Quantity\*MrchPrice; of the Merchandise table by the link that exists between Soldvia and Merchandise using the MrchID column

*SpecialEvents table relationship*

TransactionsID checks if there are new records in SpecialEvents, if there are then there were sales of "Events"

TransactionDate= EventDate; from the SpecialEvents table

MemberID= EventTypeCode; from the SpecialEvents table

ProductID= CCID; from the SpecialEvents table

Quantity= 1, always 1, that is, only one per row.

Amount= Quantity\*AmountCharged; from the OneDayPassCategory table