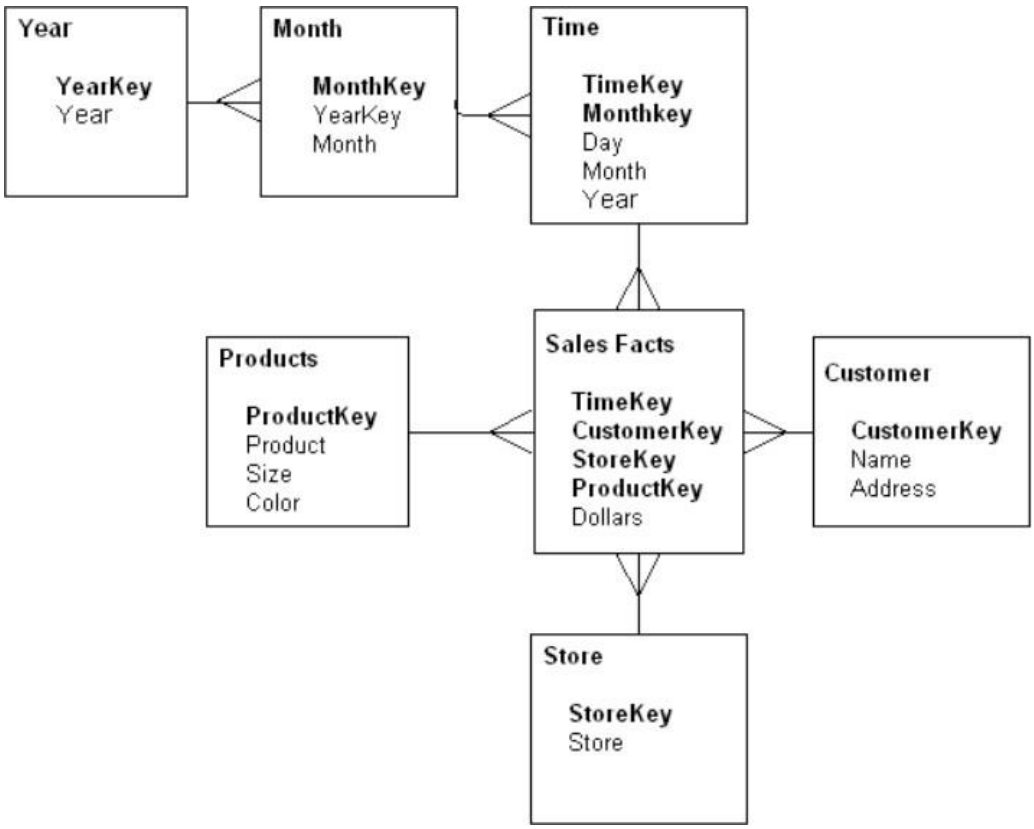
Q1. For the given Dimensional Modelling, please identify the following:



i) How many dimensions and Facts are present?

Ans:-

Dimensions	Facts
Customer	Sales
Products	
Store	
Time	
Month (Normalized Dimension)	
Year (Normalized Dimension)	
Total:- **6**	Total:- **1**

ii) Please identify the cardinality between each table?

Ans:-

Tables	Cardinality	Tables
Customer	**One to Many**	Sales Fact
Products	**One to Many**	Sales Fact
Store	**One to Many**	Sales Fact
Time	**One to Many**	Sales Fact
Month	**One to Many**	Time
Year	**One to Many**	Month

iii) How to create a Sales_Aggr fact using the following structure (SQL Statement):

Ans:- CREATE TABLE Sales_Aggr (YearKey as Year_ID int(10) FOREIGN KEY REFRENCES Year(YearKey),

CustomerKey as Customer_Key int(10) FOREIGN KEY REFRENCES Customer(CustomerKey),

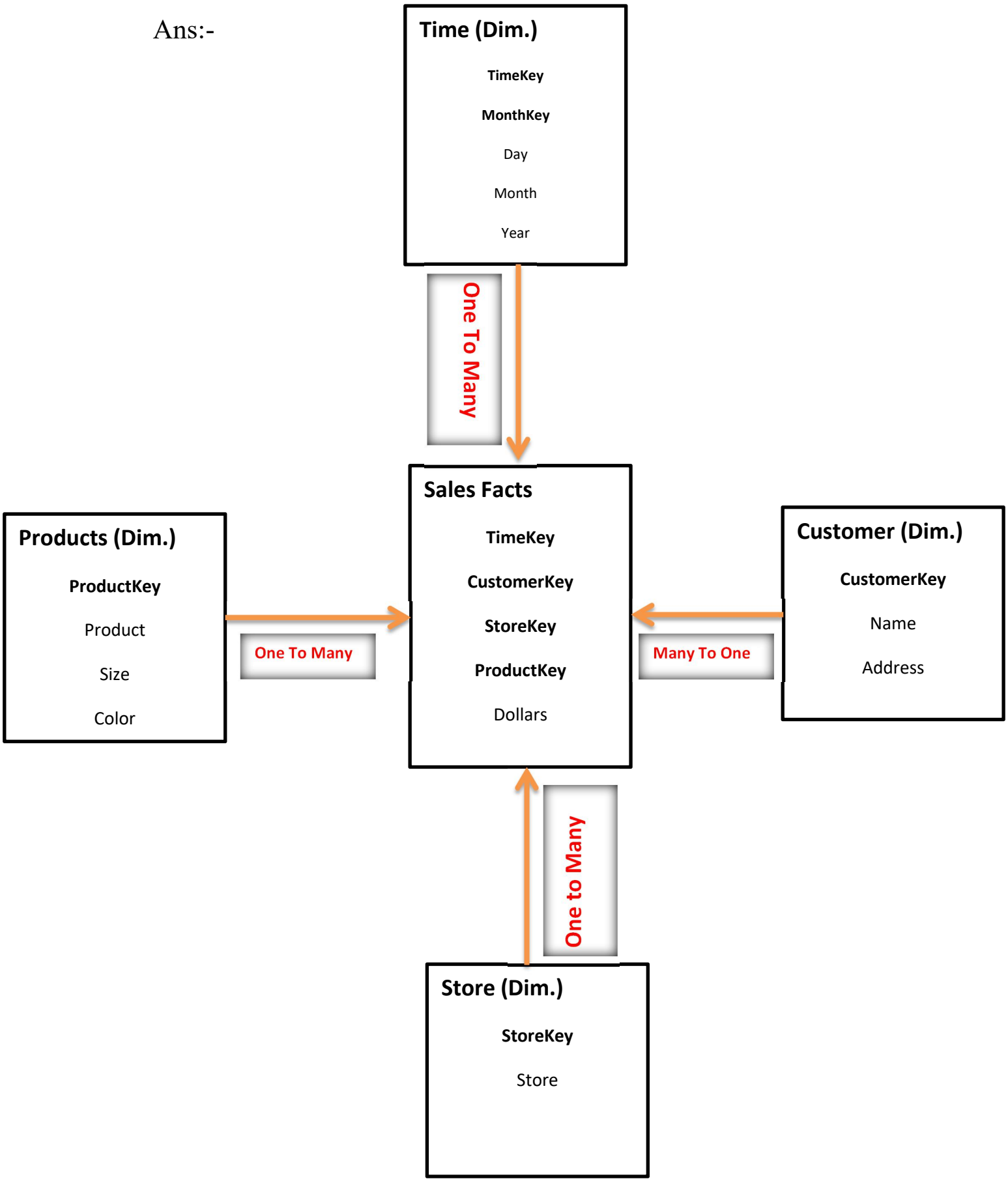
StoreKey as Store_key int(10) FOREIGN KEY REFRENCES Store(StoreKey),

ProductKey as Product_key int(10) FOREIGN KEY REFRENCES Product(ProductKey),

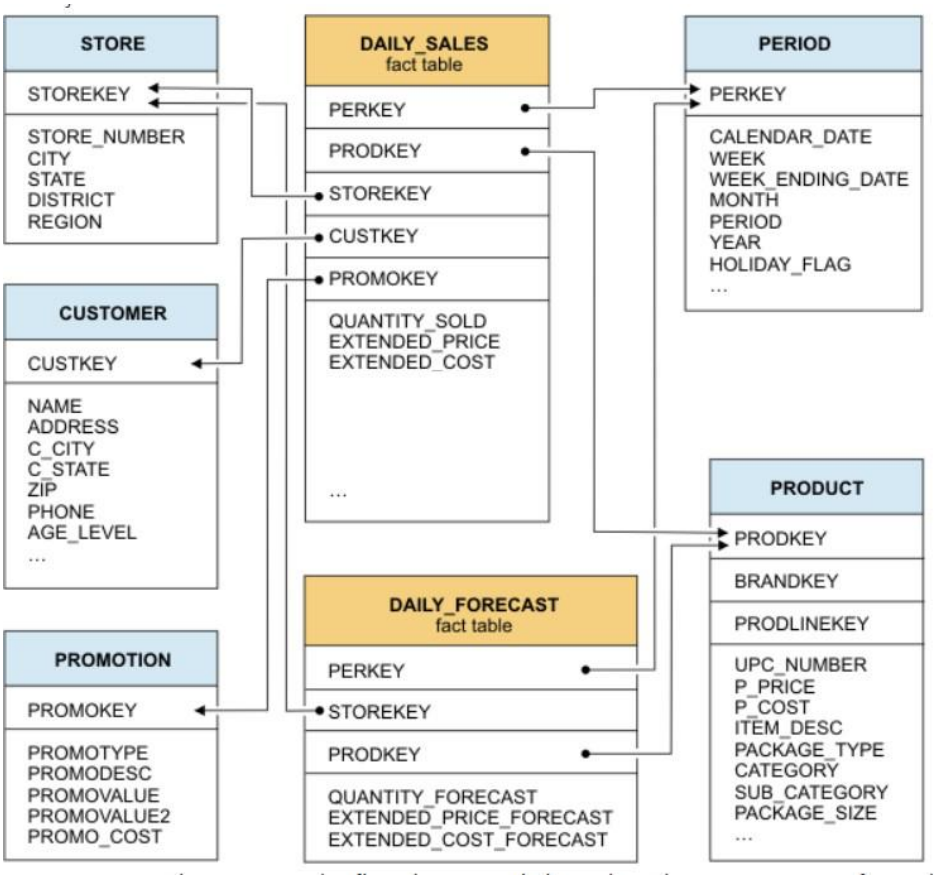
Dollar double);

iv) Can you Please Modify the above snowflake schema to Star schema and draw the dimension model, showing all the cardinality?

Ans:-



Q2. For the given Dimension Model, can you please generate a sql to get the total divergence between Quantity sold and Quantity Forecast for the current month for all the stores:



Ans:- select sum(ds1.QUANTITY_SOLD) - sum(df1.QUANTITY_FORECAST)

AS DIVERGENCES

from DAILY_SALES ds1, DAILY_FORECAST df1, PERIOD pd1

where pd1.PERKEY = ds1.PERKEY And pd1.PERKEY = df1.PERKEY

And MONTH(pd1.MONTH) = MONTH(CURRENT_DATE())

And YEAR(pd1.YEAR) = YEAR(CURRENT_DATE());

Q3. For the above-mentioned dimension model, please identify the conformed and non- conformed dimensions. Additionally, identify the measure types?

Ans:-

Conformed Dimensions	Non-Conformed Dimensions
Product	Customer
Store	Promotion
Period	

In above-mentioned dimension model, there are two types of measures present in fact tables:-

1. Additive Measures
   - ◆ QUANTITY_SOLD
   - ◆ QUANTITY_FORECAST

2. Semi-Additive Measures
   - ◆ EXTENED_PRICE
   - ◆ EXTENED_COST
   - ◆ EXTENED_PRICE_FORECAST
   - ◆ EXTENED_COST_FORECAST

Q4. Make a list of differences between DW and OLTP based on Size, Usage, Processing and Data Models.

Ans:-

Basis	OLTP	DW(OLAP)
Size	The size of the data is relatively small as the historical data is archived. For ex MB, GB	Large amount of data is stored typically in TB, PB
Usage	To control and run fundamental business tasks	To help with planning, problem solving, and decision support
Processing	Very Fast as the queries operate on 5% of the data.	Relatively slow as the amount of data involved is large. Queries may take hours.
Data Models	Highly Normalized with many tables(3NF)	Typically de-normalized with fewer tables, by using star and/or snowflake schemas

Q5. What is a semi-additive measure? Give an example.

Ans:- **SEMI – ADDITIVE MEASURES**

- Semi Additive measures are values that you can summarize across any related dimension except time.
- These measures cannot be summarized. But some analytical functions can be used to get these.
- For example, Sales and costs are fully additive; if you sell 100 yesterday and 50 today then you've sold 150 in total. You can add them up over time.
- One of the common example is the account balance.(Balance_Amount can only be added for individual customer having different for getting total balance amount in its all accounts, but Balance_Amount can't be added for all customers)

Transaction_ID	Customer_ID	Date	Account_No	Transaction Type	Balance_Amount
12654	727598456	3/1/2015	0005437675423	Credit	20000
12655	727598456	3/1/2015	0005437675423	Debit	18000
12656	727598456	3/5/2015	0005437675423	Credit	21000
12657	727598456	3/5/2015	0005437675423	Debit	15000
12658	727598456	3/5/2015	0005437675423	Credit	32000
12659	727598456	3/5/2015	0005437675423	Debit	10000

Q6. What is surrogate key? Why it is required?

Ans:- Surrogate Key (SK) is sequentially generated meaningless unique number attached with each and every record in a table in any Data Warehouse (DW).

- It is *UNIQUE* since it is sequentially generated integer for each record being inserted in the table.
- It is *MEANINGLESS* since it does not carry any business meaning regarding the record it is attached to in any table.
- It is *SEQUENTIAL* since it is assigned in sequential order as and when new records are created in the table, starting with one and going up to the highest number that is needed.

It is required because:-

Basically it's an artificial key that is used as a substitute for a Natural Key (NK). We should have defined NK in our tables as per the business requirement and that might be able to uniquely identify any record. But, SK is just an Integer attached to a record for the purpose of joining different tables in a Star or Snowflake schema based DW. SK is much needed when we have very long NK or the data-type of the NK is not suitable for Indexing.

We can take example of SCD type 2 for surrogate key:-

Cust_No	Cust_Name	Job	DBMD
111	Shivaji	Soft. Devp.	5-1-2015
112	Prithviraj	Elect. Engg.	21-10-2018

After Change in data:-

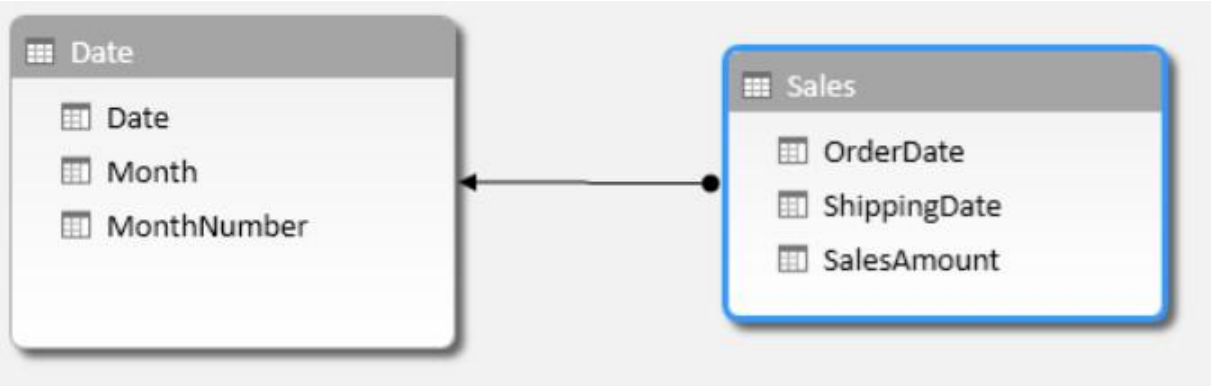
Srg_Key	Cust_No	Cust_Name	Job	Start_Date	End_Date
1	111	Shivaji	Soft. Devp.	5-1-2015	25-12-2019
2	112	Prithviraj	Elect. Engg.	21-10-2018	Null
3	111	Shivaji	Architect	25-12-2019	Null

In the above example we have used **Srg_Key** as **Surrogate Key.**

It is useful here because, let take a scenario i.e. if we want the **current job** done by the customer named **Shivaji**, then if we get the data without use of surrogate key then we will get the both jobs -> (Soft. Devp. and Architect), which is wrong, as one person can't do two jobs at the same time.

So we will use the **Surrogate Key** for distinguishing between **Shivaji's** last job and recent job as it will act as **Unique Key** for both records of **Shivaji.**

Q7. For the following Dimension Model can you please give an example of Circular Join and how to avoid it:



Ans:- A circular join is where there are 2 routes to get from one table to another. Here also in above dimension model, we can see that **OrderDate** and **ShippingDate** of **Sales Dimension** both points to **Date** of **Date Dimension** which generates a **Circular Join.**

So in order to remove it or avoid the circular join, we have to make two alias of **Date Dimension** , one, **Order_Date** as **alias** of **Date Dimension** and other, **Shipping_Date** as **alias** of **Date Dimension,** so **Date attribute** of both aliases which will points to **OrderDate** and **ShippingDate attributes** of **Sales Dimension** respectively.

Shown Below:

Circular Join present

SELECT SUM(SalesAmount) from Sales s1, Date d1

WHERE s1.OrderDate = d1.Date AND s1.ShippingDate = d1.Date;

Sales Dimension:

OrderDate	ShippingDate	SalesAmount

Date Dimension:

Date	Month	MonthNumber

Circular Join not present

SELECT SUM(SalesAmount) from Sales s1, Date Order_Date, Date Shipping_Date

WHERE s1.OrderDate = Order_Date.Date

And s1.ShippingDate = Shipping_Date.Date

Sales Dimension:

OrderDate	ShippingDate	SalesAmount

Order_Date Dimension:

Date
Month
MonthNumber

Shipping_Date Dimension:

Date
Month
MonthNumber

Q8. Category of a product may change over a period of time. Historical category information (current category as well as all old categories) has to be stored. Which SCD type will be suitable to implement this requirement? What kind of structure changes are required in a dimension table to implement SCD type 2 and type 3.

Ans:- Slowly Changing Dimensions (SCD) are the most commonly used advanced dimensional technique used in dimensional data warehouses. Slowly changing dimensions are used when you wish to capture the changing data within the dimension over time. There are three methodologies for slowly changing dimensions.

For this scenario we can only use **SCD2.**

SCD2 - This is the most commonly used type of slowly changing dimension. For this type of slowly changing dimension, add a new record encompassing the change and mark the old record as inactive. This allows the fact table to continue to use the old version of the data for historical reporting purposes leaving the changed data in the new record to only impact the fact data from that point forward. Several columns should be added to the dimension table (active record start/end dates and a current active record flag) to provide historical change management and ensure optimal use of the active record. In this example the change in the district will cause the updating of the current active dimension record's active record end data and active record flag denoting this record is no longer actively in use. This will also spawn the creation of a new active record with a new dimension key. This new dimension key will be used in the generation of the fact table moving forward. This allows the fact table to still use the data stored under the old dimension key for historical reporting. This will ensure that the data remains the same and a historical report for the same time-frame run before the update was made will continue to display the exact same data as before the change was made.

Original Record

Key	ID	Name	Region	ACTV RCRD	ACTV START	ACTV END
123	VA-13	ACME Products	Northeast	1	20140328	99999999
234	PA-07	Ace Products	Northeast	1	20140508	99999999

Inserted / Updated Records

Key	ID	Name	Region	ACTV RCRD	ACTV START	ACTV END
123	VA-13	ACME Products	Northeast	0	20140328	20160728
234	PA-07	Ace Products	Northeast	1	20140508	99999999
784	VA-13	ACME Products	Mid-Atlantic	1	20160729	99999999

SCD3 - This is a seldom used type of slowly changing dimension. In this type of slowly changing dimension you add a second column to store the most recent past value of the column(s) you wish to be able to report on. When the data is updated the existing value is "moved" to the column defined to store the previous past value and the new value is placed into the reportable column. This allows you the ability to look back at what the value of the data was previously. This can be a challenge when loading/updating the data.

Original Record

Key	ID	Name	Region	Previous Region
123	VA-13	Ace Hardware	Northeast	
234	PA-07	Ace Products	Northeast	

Updated Record

Key	ID	Name	Region	Previous Region
123	VA-13	Ace Hardware	Mid-Atlantic	Northeast
234	PA-07	Ace Products	Northeast	

Q9. Stores are grouped in to multiple clusters. A store can be part of one or more clusters. Design tables to store this store-cluster mapping information.

Ans:-

Store Table

Store_ID	Store_Name
101	Raam Store
102	Ajay Store
103	Sanjay Store
104	Kamal Store

Category Cluster Table

Category_ID	Category_Name
1	Vegetable Shop
2	Clothes Shop
3	Stationary Shop
4	Shoes Shop
5	Fruit Shop

Location Cluster Table

Location_ID	Location_Name
1001	New Delhi
1002	Mumbai
1003	Bangalore
1004	Chandigarh

Mapped Table:- Mapping both Store Table and Category Table

Mapped_ID	Store_ID	Category_ID	Location_ID
111	**101**	**2**	**1002**
112	**101**	**4**	**1003**
113	**102**	**1**	**1001**
114	**103**	**3**	**1001**
115	**102**	**5**	**1004**
116	**104**	**2**	**1003**
117	**104**	**3**	**1001**
118	**104**	**4**	**1002**
119	**104**	**1**	**1004**
120	**101**	**2**	**1004**

Here in above example, one store has multiple category, has branches present at multiple location**(**one store is in one or more cluster**(here are category and location))** and it is shown through the **Mapped Table,** like, **Raam store** is in the category 2 at location 1002 and 1003 and in category 4 at location 1003 i.e. Raam Store is a **Clothes Shop** at Mumbai and Chandigarh and also as **Shoes Shop** at Bangalore.

We can get this mapping through applying SQL query on Mapped Table.