**Part 1 Data Integration and Quality**

**1. Write a query that displays parts that are in common between the red and blue orgs.**

Select r.\*, b.\* from red r inner join blue b on SUBSTRING(b.partdescription, 10 , len(b.partdescription))

= SUBSTRING(r.partdescription, 1 , len(r.partdescription) - 7)

**2. Write a query that displays the parts that are unique to the blue org**

Select b.\* from red r right join blue b on SUBSTRING(b.partdescription, 10 , len(b.partdescription))

= SUBSTRING(r.partdescription, 1 , len(r.partdescription) - 7) where (SUBSTRING(r.partdescription, 1 , len(r.partdescription) - 7)) is null

**3. How might getting a summary report for total units sold be simplified?**

We see that the summary report generated for the unitssold is the sum of unitssold for the partdescriptions which are common in both red and blue table.

As of now there is no common column/referential integrity that is common to both the tables. Hence, we are joining the two tables with the help of the partdescriptions that are common to both the tables. But this may not always help if there are data entry errors

***Solution 1***

The possible solutions that I would recommend getting a simplified summary report for units sold are-

* To perform aggregate functions on a table after joining it is essential that both the tables have a common point of reference
* This can be accomplished by having a common code table that defines the type of parts
* The common code tables will be Code\_colour and Code\_shape which contain specific numbers for colour and shape

|  |  |
| --- | --- |
| **Code\_shape table** | |
| **Colour** | **Reference** |
| round | 1 |
| triangular | 2 |

|  |  |
| --- | --- |
| **Code\_colour table** | |
| **Colour** | **Reference** |
| red | 1 |
| blue | 2 |
| yellow | 3 |

* For example, the part description column can the removed from both blue and red tables and 4 other columns can be added.
* 4 columns – each specifying the color, shape, widget/not widget, length

On changing the red and blue table so that it refers the code table, It will look something like this

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Red table** | | | | | |
| **partnumber** | **unitssold** | **Widget (1) /Not widget (0)** | **Shape** | **Color** | **Length** |
| 1006 | 2 | 1 | 0 | 2 | 0 |
| 1008 | 1 | 1 | 0 | 3 | 0 |

So, the first row is a blue widget and second row is a yellow widget

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Blue table** | | | | | |
| **partnumber** | **Unitssold** | **Widget/Not widget** | **Shape** | **Color** | **Length** |
| 1 | 45 | 1 | 0 | 0 | 5 |
| 9 | 12 | 1 | 1 | 0 | 0 |
| 6 | 90 | 1 | 0 | 2 | 0 |

Here the first row is a widget of length 5cm (Widget-5cm) and second row is widget-round

So basically, we are trying to store the values for different parts in code tables and referencing the values in blue and red tables by creating foreign keys.

Row3 in blue table and row1 table are blue widgets. They can be fetched out by doing multiple joins or by using multiple criteria to join.

**Example- Select sum(unitssold) as unitssold, Shape,Color,Length from red r join blue b on b.Shape=r.Shape and b.Color=r.Color and b.Length=r.Length group by Shape, Color, Length**

|  |  |  |  |
| --- | --- | --- | --- |
| unitssold | Shape | Color | Length |
| 92 | 0 | 2 | 0 |

**Result: 92 units of blue widget**

It is better to create a composite primary key for shape, color and length so that no duplicates will be allowed.

***Solution 2***

For every combination of shape, color and length a code value can be created in the code table and can be later referred to in the color tables

**Example-**

|  |  |
| --- | --- |
| **Code** | **Reference** |
| red | 1 |
| blue | 2 |
| Round | 3 |
| Triangular | 4 |
| Yellow round | 5 |

But in this case as the size of data increases lot of permutations have to be performed for every combination.

Owing to data integrity, joining two tables on numbers rather than on characters is faster and can be error free.

This is because:

* Numbers use less space than characters
* Access time is reduced
* Duplicate logs can be eliminated
* Memory consumption is higher to read characters than to read numbers

**4. Do you think that the way this data in these 2 tables is organized would adversely impact data quality? Provide examples**

Yes, this would affect data quality.

This is because:

* **Missing Primary Key-** Primary key is essential for any table. Neither of the tables red nor blue have primary key which means there is higher chance of duplicate values to be entered. The timeliness and uniqueness are essential to understand the overall quality of data.

**Example-**There is no column that identifies every row**.** An identity column can be created and added to both the tables that will serve as a primary key to both the tables.

* **Higher Access time**-Each time query is run the table has to be scanned which means it has to read characters each time and match them. The access time to do this longer than usual. The **memory consumption** and the time taken to read characters each time is higher than reading integers. Hence the process gets slower over a period of time
* **Lack of Accuracy**- The accuracy becomes less when the joins are based on characters. This happens if there are data entry errors.

**Example-**When data is inserted into the table, if by any chance there is extra space in the newly inserted row, joins won’t work fine thereafter, and this is highly undesirable

* **Lack of Consistency and conformity-** The result obtained by establishing relationships based on characters cannot be assured of consistency.

**Example-** This makes the data lose its conformity as it is not reliable across all the environments

* **Accessibility not easier-** The data should be easily accessible regardless of the form in which they are stored**.** The data here is not easily accessible as they don’t have common standard and unique identifier**.**

**Example-** According to the current scenario the data is accessed using

Select \* from red where partdescription like ‘%blue%’

This gives all the parts which contain blue and not the exact part that we want

* **Lack of Completeness**- In near future there is a chance that many different types of widgets can be inserted,

**Example**- Blue round shaped widget. In the current scenario this type of parts cannot be addressed.

As the two tables are entirely different and do not have a relationship between them, it is difficult to perform aggregate functions as they are not recognized

**5. How does this sql exercise relate to the concept of data governance**

Data governance is more about proactively managing the data assets. Data quality is like a subset of data governance. More appropriately data quality and data governance are mutually dependent on each other. Hence both are equally important in order to successfully manage and improvise the quality of data.

The benefits of data governance can be fully exercised only when the data quality of the dataset is at its best else data governance has no meaning. In our current scenario there is a lot to be improved on the data quality part and then apply data governance to it. A set of data quality rules have to be set that any data set should adhere to. This is widely referred to as Data Quality Dimension which is a measure of data quality that is helpful in comparing the data quality across different levels.

The standard data quality dimensions include-

**Accuracy, completeness, uniqueness, timeliness, consistency, validity**

**Hence the appropriate data quality standards should be set to govern the data in the best possible way.**

**Capture right data-** It is very important the right data should be created/collected. Right data here refers to the data that data consumers are looking for. Timeliness of data also plays a vital role -that is giving the right data at the right time for data consumers to work with it.

The data set given has to be improvised in many disciplines like accuracy, completeness, uniqueness, accessibility etc. so that it can adhere to the data integrity which is in turn a major part of data governance.

None of the rows in the table have a unique identifier nor there is a primary key that can be used to establish relationships. This makes the data less accessible. And also, there are no security features added like views that gives the right data to the appropriate users. Nor there are features like indexing that can improve the overall performance and execution time.

These are some of the pitfalls of the given data set from the perspective of data governance which indirectly calls for the need of data governance.

On a broader view data governance is management of data quality, meta data, master data and architecture of data.

Hence, the first step towards data governance with the current data set is to improvise its quality. Once the data quality and table architecture is improved further betterment can be made like the views and indexes that will improve the query performance to finally achieve data governance.

**Part 2 Data Quality and Synthesis**

**1.Data Quality**

**a. Run these scripts and write a sql query to display the name of the person(s) who speak the most language**

Declare @count1 int=0

Declare @count2 int=0

Select a.First\_Name from (Select

case

when Lang1 <>'' and Lang2<>'' and Lang3<>'' then @count2+3

when Lang1 ='' and Lang2='' and Lang3='' then @count2+0

when Lang1 <>'' and Lang2='' and Lang3='' then @count2+1

when Lang1 ='' and Lang2<>'' and Lang3='' then @count2+1

when Lang1 ='' and Lang2='' and Lang3<>'' then @count2+1

when Lang1 <>'' and Lang2<>'' and Lang3='' then @count2+2

when Lang1 ='' and Lang2<>'' and Lang3<>'' then @count2+2

when Lang1 <>'' and Lang2='' and Lang3<>'' then @count2+2

end

As NumOfLanguagesSpoken,First\_Name

from Person) a where a.NumOfLanguagesSpoken=(Select max(p.NumOfLanguagesSpoken)from(Select

case

when Lang1 <>'' and Lang2<>'' and Lang3<>'' then @count1+3

when Lang1 ='' and Lang2='' and Lang3='' then @count1+0

when Lang1 <>'' and Lang2='' and Lang3='' then @count1+1

when Lang1 ='' and Lang2<>'' and Lang3='' then @count1+1

when Lang1 ='' and Lang2='' and Lang3<>'' then @count1+1

when Lang1 <>'' and Lang2<>'' and Lang3='' then @count1+2

when Lang1 ='' and Lang2<>'' and Lang3<>'' then @count1+2

when Lang1 <>'' and Lang2='' and Lang3<>'' then @count1+2

end

As NumOfLanguagesSpoken,First\_Name

from Person)p)

**b. Run these scripts and write a q sql query to display the name of the person(s) who speak the most languages**

Select First\_name from PersonLangAssoc pa join Person\_NF pf on pa.Person\_Id=pf.Person\_Id

group by First\_Name having count(\*)=(select max(a) from(Select count(\*) a,Person\_Id from PersonLangAssoc group by Person\_Id)p)

**c. Review your solutions and describe the conceptual complexities for each sql statement**

Complexities of querying data set before normalisation:

* Performing aggregation functions is very difficult as the data is scattered in different columns. The whole query has to be re written for every aggregation function.
* Performing joins is also difficult as the data set does not have a unique identifier like primary key/referential integrity
* It contains data that is redundant. This means the same type of data is present as different columns (Poor design). Therefore, the same data can be entered multiple times thus allowing duplicate values.
* Physical difficulty in typing long codes

Complexities of querying after Normalisation(1NF):

* As compared to querying done before normalisation, querying after 1NF is much simpler. This is because the data that is redundant in split into two tables. One which has languages and the other one which has the names associated with them. Both the tables have a common reference person\_id based on which they can be joined when required
* But still there is no unique identifier like a primary key which allows us to have unique one-way relationship such that for any unique value of one attribute there is only one unique value in the other table. Not having a primary key may result in redundant or even null values

**2. Data Synthesis: Pivot Operator**

**b. Using the starting sql, write the sql to produce this output**

Select \*from (Select [WHERE],[HOW],[REVENUE] from exped)p

PIVOT

(

sum(REVENUE)

FOR [HOW]

IN ([Catalog],[Store],[Web])

)As PivotTable Order by [WHERE]

**c. Perform some research on the pivot operator and provide 4 facts regarding the usage of this operator**.

Usages of Pivot operator:

* Pivot operator is used to transpose the data from rows to columns. Unique values in a row can be converted to individual columns which helps the user to isolate, sum and group the particular data in real time. It is used to store the summary of a data set in a more concise manner.
* It takes the user defined fields from the original tables and allows the user to manipulate in a way that is easier to read. This way pivot tables helps us to create customized data sets from large data sets which in turn helps us in deciphering the data patterns.
* As this involves working with only the essential data it is easy for us to analyse larger data sets. This helps in quick decision making.
* Also, as this gives the summary of the data set like sum average by grouping attributes which helps us in easier comparison between different fields.

This gives efficient reports without much strenuous efforts

There is no need of using a separate reporting tool.

* Using pivot operator is a flexible option than using multiple aggregate functions over a simple SQL query. This saves time and also for each query the whole large data set does not have to be scanned every time

**Part 3 Discussion**

**a. What possible technologies (Hadoop, type of NoSql, relational data store) could help facilitate fulfilling this company’s analytic needs? Pick one of the top business questions and explain how your data store solution would help.**

In the current scenario the data -the customer and order data including the survey and reviews is structured. This does not call for the need of technologies that can accommodate unstructured data. When we use relational data store it is easier to integrate with the data of the acquisition company no matter which type of technology they are using (Hadoop, NoSQL or relational data store) as any technology can support the highly structured data from relational databases. This allows us to maintain data quality and integrity.

**Business question**- Should any products be discontinued based on sales and poor ratings?

For the below listed reasons I would go for **relational data store** to resolve analytical needs of the company-

* Data is structured as of the current situation. In case when there is large amount of unstructured data in future (which is not mentioned) we may have to adopt other technologies like Hadoop
* The company wants the best analysis of their data to find the loopholes and to improvise their marketing strategies. And it does not mention how fast the data has to be analysed. If there is a concern for processing time and performance issues, we can use other technologies like Hadoop as it can cater to the needs of higher processing time. But the above case study does not mention about the needs of quick processing time
* Owing to the increasing size of data every year relational data stores are scalable to the extent that they can meet the current requirements of the case study.

<https://www.itprotoday.com/sql-server/essential-guide-sql-server-2014-series-scalability>

For example – In the above link it is mentioned that enterprise edition of relational databases (SQL server in this case) can be up to 524 petabytes which is approximately 5.24 \* 10 8 gigabytes. This is extensively huge amount of data space available for the current data set.

Addressing the business question:

It is easier to retrieve the data from a relational data store for the products which have lower sales and poor ratings. This can be done using a simple SQL query even if the job runs over night (as there is no concern about processing time), the next day we will have a report which is more concise, accurate and consistent for analysis as it has been retrieved from a relational database. The IT tier of the company can generate an accurate and consistent report so that the business tier of the company can effectively analyse on if any products should be stopped producing/which products should be stopped producing based the sales and customer ratings.

Hence for the above reasons I would recommend using a **relational data store** to address the business question **Should any products be discontinued based on sales and poor ratings?**

**b. List any unknown items requiring clarification that would affect your recommendation**

There are certain parts of the company’s analytical needs that cannot be addressed using relational data store alone.

The unknown items are-

1. How soon will the acquisition take place?
2. If it happens what is the size of data in the new company/What will be the total data size of both the companies after merging?
3. What type of data (Structured, semi-structure, unstructured or only structured) is being used in the new company?
4. Is the current company focussed only on improvising the marketing strategy or are they concerned about processing time and performance also?

During below scenarios, usage of relational data store can affect the business needs of the company-

* **Concern for Processing Time**-This is not explicitly mentioned. Assuming that the company’s products were a massive hit and the company size has grown immensely. In this case there will be a large volume of data to handle and it is difficult to maintain such huge amount of data using relational data stores. When such large amount of data has to be analysed and when there is a concern for processing time and performance, retrieving data from relational databases can be really slow causing performance and processing time issues.
* **Marketing strategy that acquisition company will use-**If in futurein order to improvise the marketing strategies if it was decided that the users can upload pictures of the torn invoices or broken package or even to display the bad effects of any of the products on their skin, relational database will not cater to the storage needs in such cases. When data like images or videos are stored, the data size increases exponentially, and the data may not be structured. In such cases it is advisable to go for other technologies like Hadoop which is highly scalable and supports unstructured data.
* **Type of data in future**-Moreover, Hadoop uses schema on read technique which means it can take in both structured and unstructured data and change a particular data only when they are read or used. But that is not the case n relational databases. They use the technique schema on write which means the data should be structured before feeding into the database. This is not really helpful while dealing with excess amount of data
* Relational databases have to be highly maintained. If there are mirror databases for the original databases, they have to be maintained manually. But that is not the case with

Hadoop as they have copies of data that can be maintained within themselves and has better **fault tolerance**.

**c. Is this a big data use case? Why or why not?**

[**http://www.webopedia.com/TERM/B/big\_data.html**](http://www.webopedia.com/TERM/B/big_data.html)

According to the above link a data qualifies to be big data if it is in the ranges of petabytes and exabytes and if it contains structured, unstructured and semi-structured data. But in our case study currently only structured data is being used.

For example-

* Facebook produces 500+ terabytes of data everyday
* A single jet engine generates 10+ terabytes of data everyday
* New York stock exchange produces 1 terabyte of trade data every day.

As compared to the data being produced in the above cases, the company in the case study has 30 gigs of data and it increases 2 gigs a year which is very small.

1 terabyte =1000 gigabytes

At such a rate it will take approximately more than 200 years for the data size of the company to reach 1000 gigabytes if the acquisition does not happen.

Even if the acquisition takes place according to the scalability capabilities (

<https://www.itprotoday.com/sql-server/essential-guide-sql-server-2014-series-scalability>

) of SQL server enterprise edition is capable of scaling up to 524 petabytes which is 5.24\*108 gigabytes.

With the current data size and keeping in mind the current rate of growth, even in the next 30 years the data would take up 128 gigs of space which is not even close to 5.24\*108 gigabytes.

This size of this data is very much comfortable for accommodating data of the current company and also the data of the acquisition company. The only ambiguity is that will there be only structured data for a long term even after acquisition?

Most of the business questions in the case study can be answered using relational database systems as all the data they have is structured. It is easier to generate accurate and consistent reports from relational databases and also it is easier to analyse the data.

The case study about the company does not satisfy the 4 V’s of big data for it to qualify as big data before acquisition

**As the above website says that big data might be in petabytes or exabytes**

**Volume-** Volume is in gigs and increases by 2 gigs a year.

Will take years to reach 1 terabyte

**Velocity**- The speed at which data increases is not even exponential

**Variety**- The company currently holds only structured data whereas variety means it should contain all types of data (Structured, semi-structure and unstructured)

**Veracity-** This criterion is satisfied as the company uses only structured data stored in relational database system.

Moreover, the data in the company is stored in relational database systems which has got very **low fault tolerance.** In case if the data is lost due to corruption or network issue it takes more time, cost and resource to get back the lost data. This is highly undesirable in case of big data. Other technologies like Hadoop have multiple copies of data stored on different servers so that the data can be retrieved easily. But Hadoop is designed in such a way that it can handle only large amount of data.

Hence for the above reasons the data in the company as of now does not classify as big data and thus, this is **not a big data use case**. Once the acquisition and data integration are done if the size, nature of data and speed in which data is being generated is known then we will know if the data qualifies to be big data or not.