**LPS Code Challenge**

**Prepared for:** *Literacy Pro Systems*

**Prepared by: *Gokul Ram Thangamani***

**1) DB query efficiency**

● Provide metrics on this query as is.



vie\_consolidate\_queue where manual\_hash in (**select distinct manual\_hash from vie\_conform\_01\_hashed**). The highlighted part of query doesn’t work as intended. There is no such column called **manual\_hash** in the vie\_conform\_01\_hashed table. It’s the behavior of mysql that considers manual hash from the main table and executes the sql and returns all records from vie\_consolidate\_queue though there is no matching records for manual hash in the vie\_conform\_01\_hashed table.

**2) Query improved for efficiency:**

( select

'incoming' rec\_ind,soft\_hash,hard\_hash,manual\_hash\_from\_cq

From vie\_conform\_01\_hashed

where manual\_hash\_from\_cq is not null )

union all

(select distinct

'existing' rec\_ind,customer\_soft\_hash,customer\_hard\_hash,manual\_hash

From vie\_consolidate\_queue vq

join vie\_conform\_01\_hashed vh

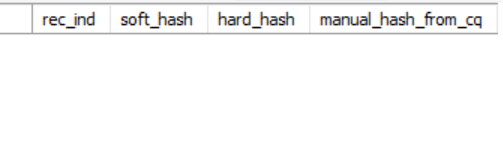
on vq.manual\_hash = vh.manual\_hash\_from\_cq and vh.manual\_hash\_from\_cq is not null )

order by 2,3,1

**3) Explain any considerations you took into account after reviewing the resource constraints on the RDS instance.**

In query is sent to AWS Redshift query is parsed, parser then sends the query to nodes. More the nodes, faster the processing is. Since the hardware requirement says it has only 2 vCPUs , there is going to be a max of 2 nodes. Right now processing with the small data set doesn’t take much time. But when the data grows, it will take longer time. So having a subquery in an union statement will not be efficient because it will split the second part of the query again and the sub query will execute first and the main query cannot be run parallel as it has to wait for the sub query to finish because of the resource constraint. This will slow down the entire process as the first part of the query will be processed faster and it has to wait for the sub query and second part of the query to be processed before doing an union. So considering the resource constraint, I avoided the subquery and used a join on the common key from both table. This way , the parser will parse the query into two ways and each query will execute in each node and it makes the process faster. The keys on which the join is made has to be sorted / added as a sort key in red shift for faster processing. And also the columns soft\_hash, hard\_hash and rec\_ind should be added to sort key as well so it will be faster. I separated the query into two pieces so that the sorting will happen after the union which will help us stay under the memory contingent upon the size of data.

**4) Provide metrics on your updated version of the query.**



This query will not return anything because there is no matching record for manual\_hash in the vie\_conform\_01\_hashed table and it has null values only.

**5) Explain how you analyze poor performing queries on an AWS RDS instance**

**Identifying slow process in the query:**

First step is to identify which part of the query is taking longer time to execute. This can be identified using query plan. This will gives us all the information about a query execution , the order it executes, cost involved . Based on cost involved we can identify which part of the query takes longer to execute.

**Indexing:**

Not just with RDS, but with any database slow running query can be fixed by applying right index in right order. 80 % of the time indexing the tables in right order will speed up the poor performing queries. Coming to RDS, adding the sort keys on the columns that are used in joins, where clauses and order by clauses will speed up the poor performing queries.

**Using Join functions:**

Join function is faster as compared to any other function. So using join operators will also speed up the performance of the query.

Performance is also based on the data structure and also size of the data set. Vaccum command helps to clean and re organize the table

**DB Architecture**

**● Using the given DB backup, identify areas of the database that are poorly architected.**

First of all , without knowing how the data is stored and how it is used for reporting, it is very hard to comment on the architecture of the database. I have had previous experiences where initially looking at the db structure I assumed the DB was poorly architected then after knowing the reason why it was done so , I realized my assumption was wrong. But anyways, I can use foreign key to analyze the connection between tables and comment on what would have done better.

* No database diagram available to understand the structure of the database
* All information are condensed in one place like both transactional and analytical data at same place.
* Storing information like social security number of a customer without any encryption.
* Database is very well de normalized in few areas for better performance but the naming conventions doesn’t show the difference between a metric table and dimension table.
* I don’t see any constraints added to the database may be everything is handled as business constraints in reporting

**● What are the top three areas you would focus upon to fix / improve those areas?**

* I would separate the databases and keep transactional db separate and analytical/ reporting db separate. This will help in improving the performance of the report and we can refrain from locking up the tables while reporting. This would give clear picture about the metrics and dimension that are used in reporting.
* Improve the security of the data by storing it in more secured manner. We can do that by encrypting the data before it gets into the database and store the encryption key securely.
* I would add the business constraints at the database level itself, that way we can be more proactive than to be reactive. (Making this comment purely based on cases I have faced and It’s hard for me to make this comment without understanding the business need)

**3rd Party Source Systems**

**● How can we work with 3rd party source systems to more easily get data from them?**

We can build a transformation with switch cases for multiple transformations in it to go and get the required information from any server that we have access to. For example, we can build one single job with multiple transformations in it, where each transformation reading from various sources using certain conditions. To read from API which is always JSON format, we can use steps like REST API in tools like Pentaho (open source) and write them back to our DB. Parsing JSON can also be done just by writing sql with JSON\_value function. We can also build control tables to control if we have dependency of one process over other. It makes it easier to block the process A before process B completes to proceed to Process C which has dependency from A and B. Reading from CSV is easier when we FTP / SFTP all of the files from the third party system in one place. Sometime we get access to their systems or sometimes they SFTP it over to us. So instead of directly connecting to their server and reading the file, it’s always better to get them and store it locally. Once we process it we can move the files to completed folder and we know what is processed so far. And in case of any errors , we can go back look at the copy we have in our end first and fix it fi it’s a minor issue instead of reaching out to sources. Also , it’s easier for us to go back and reload the file if in case something went wrong and we have to reload the file because the source system may not always carry the history

**● Describe any experiences you've had with this and/or ideas that would facilitate the acquisition of data from 3rd party source systems.**

**APIS:**

I have worked on reading data from multiple API sources. But the most interesting job I had was reading data of Power Probes from the API. The real time readings of each device attached to the probes were available in the API. My process reads the data from API and stores the data every 20 secs. And this is used for real time reporting of status of the probes in a given Comcast facility. The facility manager will login and know how much energy being used and what they can do to prevent going beyond the threshold of the probes. Reading from APIs are always easier if we know what data points to collect and the structure of the JSON doesn’t change.

**CSV:**

I worked with CSV, flat files, excel files , text files e.t.c. When it comes to CSV files or any flat file for that matter , it’s easier to load the files when the file size are relatively smaller. I worked with flat files that has more than billion records and it was taking longer time to process. In cases like that , I always prefer parallel processing running the job in multi cluster environment by splitting the file based certain on condition and loading it into the DB. This makes the job to run faster. In one of my project , by exercising this process in Pentaho , I brought down the loading time of 2 billion records from 7 hrs to less than 2 hrs with the same available processing units.

**Understanding Data:**

How would you integrate a master data management system to help track and understand metadata for hundreds/thousands of data points from multiple source systems that we do not control? For example we collect a data point called "Highest education level upon entry" from various systems but it carries a slightly different meaning depending on which system it comes from. How can we best track our understanding of that data?

Any system / Any format from which the data comes from the meta data header can be used to identify the right data points. If the meta data of a field that we are looking for is not named or labeled exactly in the order that we are looking for, then we would definitely need data dictionary from the source system to map it to our process. In previous jobs, I have created control table which for each source system with their meta data mapped to destination column header. This can be done with the help of source system developers. For example, in your case let’s say **highest level education level** upon entry comes from ten different system. For the sources that has metadata label with the exact or similar names can be found using regex or fuzzy match function but for the sources that doesn’t explicitly provide it, then we create a control table that has all data points of that system and which one relates to the particular destination field. Here let’s say source system one stores **highest level education level** as Data Point 1 , then in the control table, we have the source system name and data point name with ID. We map the data point with the required data point with the ID. We can also put constraints in database to satisy the requirement. For example, there can be a maximum of 7-8 level of education any one can enter. We can make sure that database constraints are set to take values that match the criteria. This way we can keep track of meta data and make sure we are storing the right values in the database.

**Reporting Structure:**

**What are three changes you would make to the structure of the DB to improve the ease of creating reports?**

Once again, I’m not sure about the complete reporting process that is done using this DB to comment on the DB structure. But based on the report, I created

* I would create a reporting/ analytics DB separately for all sort of reporting. This will help us to render the report faster without having any interference with the transactional database.
* I will maintain metrics separately and the dimensions separately to make it easier for reporting purposes. Depending on the severity and complexity of the data I would build a multi-dimensional database model for the faster aggregate calculation and high performance of reports.
* I would create views to do major aggregate calculation and retain the business constraints in db level than writing complex sql in reporting layer as it will help to reduce latency since the query will be processed at the db level. This will also help us in increasing the performance.