Question 2dii: Explain your answer to the previous part (Question 2di) based on your knowledge from lectures and details from the query plans. Your explanation should also include why you didn't choose certain options. Please answer in maximum 5 sentences.

I chose D in this case since materialized views are temporary and thus they take less time to execute and are thus cheaper. Furthermore, for J, materialized views often take more time to create because they involve precomputing and storing data, which regular views don't.

0.1 Question 3c:

Given your findings from inspecting the query plans of queries from Questions 3a and 3b, fill in the blank and **justify your answer**. Explain your answer based on your knowledge from lectures, and details from the query plans (your explanation should include why you didn't choose other options). Your response should be no longer than 3 sentences.

Note: Your answer should be formatted as follows: `A because ...`

Adding a filter _____ the cost. A. increased B. decreased C. did not change

B decreased the cost since it reduces the amount of data that needs to be processed and returned by the query. This in turn minimizes cost and causes our execution to be quicker.

0.2 Question 3d:

Given your findings from inspecting the query plans of queries from Questions 3a and 3b, fill in the blank and **justify your answer**. Explain your answer based on your knowledge from lectures, and details from the query plans (your explanation should include why you didn't choose other options). Your response should be no longer than 3 sentences.

Note: Your answer should be formatted as follows: `A because ...`

Adding a filter _____ the execution time. A. increased B. decreased C. did not change

B because similar to the response above, we cut down on execution time the amount of rows of data is cut down with a filter.

0.3 Question 4d

Given your findings above, why did the query optimizer ultimately choose the specific join approach you found in each of the above three scenarios in Questions 4a, 4b, and 4c? Feel free to discuss the pros and cons of each join approach as well.

If you feel stuck, here are some things to consider: Does a non-equijoin constrain us to certain join approaches? What's an added benefit in regards to the output of merge join?

Note: Your answer should be formatted as follows: `Q4a: A because ... Q4b: A because ...` You should write no more than 5 sentences.

Q4a: C because we are essentially putting different rows into buckets and then merging on the bucket we choose, we can desrcibe it as: reading all tuples hashed to b1 from both people and collegeplaying at a time, perform join, repeat. Q4b: B because we do an effective job of sorting by the conditon we want with the Where in addition to merging which is what a merge join is effective at, the Where conditioning is best addressed by a merge join. q4c: A becuase the dataset is checking itself to see which rows meet the conditon, also the datatset is smaller so a loop works well here.

0.3.1 Question 5di Justification

Explain your answer to `Question 5d` above based on your knowledge from lectures, and details from inspecting the query plans (your explanation should include why you didn't choose certain options). Your answer should be no longer than 3 sentences.

Based on the resulting values, I noticed that the g_batting index was more effective since it was used in finding the rows we would want in a large table and has distainct values, cardinality is low. For salary, the cost did not change suggesting that the index here was not very effective due to cardinality or a complex query cuasing cost to remain relatively the same.

0.3.2 Question 6ei Justification

Explain your answer to `Question 6e` above based on your knowledge from lectures, and details from inspecting the query plans (your explanation should include why you didn't choose certain options). Your answer should be no longer than 3 sentences.

For 6e, adding an index to an AND predicate helped to further filter and thus reduce time and cost. For 6e, for the case of adding an index to an OR predicate was helpful in some sense but did not help the cost, it remained the same. Lastly for the multicolumn index is helpful for sorting and filtering which goes a long way in reducing cost and time.

0.4 Question 7c

Given your findings from `Question 7`, which of the following statements is true? A. An index on the column being aggregated in a query will always provide a performance enhancement. B. A query finding the MIN(salary) will always benefit from an index on salary, but a query finding MAX(salary) will not. C. A query finding the COUNT(salary) will always benefit from an index on salary, but a query finding AVG(salary) will not. D. Queries finding the MIN(salary) or MAX(salary) will always benefit from an index on salary, but queries finding AVG(salary) or COUNT(salary) will not.

Justify your answer. Explain your answer based on your knowledge from lectures, and details of the query plans (your explanation should include why you didn't choose certain options). Your response should be no longer than 3 sentences.

Note: Your answer should be formatted as follows: "A because …"

D because the indexes will play a role in distinguishing the minimum value which is not the case finding the count since you would have to go about counting up the values and then taking the mean if that is what you were looking for, which makes indexes redundant.

0.5 Question 9c:

What difference did you notice when you added an index into the salaries table and re-timed the update? Why do you think it happened? Your answer should be no longer than 3 sentences.

I noticed that the timing resulted in a higher time, which I fund particularly odd, but I think this is the case because the database now has one more thing to keep track of which may be better or worse depending on the size of the database, in this case it was to our detriment.

1 Question 10: Project Takeaways

In this project, we explored how the database system optimizes query execution and how users can futher tune the performance of their queries.

Familiarizing yourself with these optimization and tuning methods will make you a better data engineer. In this question, we'll ask you to recall and summarize these concepts. Who knows? Maybe one day it will help you during an interview or on a project.

In the following answer cell, 1. Name 3 methods you learned in this project. The method can be either the optimization done by the database system, or the fine tuning done by the user. 2. For each method, summarize how and why it can optimize query performance. Feel free to discuss any drawbacks, if applicable.

Your answer should be no longer than ten sentences. Each method identification/discussion is 2 points.

The first method I would like to discuss is the insert index method, which creates an index for a column and helps to optimize query performance by helping to sort and filter when it comes to finding specific rows or entries in a dataset. Beyond that, indexing proves to be useful for joining tables which is often an expensive operation. Secondly, Clustering is another method that is effective at optimizing query operations since it allows for the clustering of similar columns which proves to be effective when it comes to using Where filters. This makes sorting and filtering far more rapid than without clustering. Lastly, I think multi-attribute indexing is a very useful practice in some scenarios, since this allows you to create indexes which would allow you to filter by multiple things and thus work with more features of the data but this comes at a great cost. Performing a large multi-attribute index may be very computationally expensive and thus may hinder one from using it since in large datasets it is not worth running. I also found the Nested Loop Join to be very useful since it allows for an effective way of joining smaller tables, very useful when it comes to joining smaller tables to themselves, and doing so avoids dealing with buckets or hashing which in turn reduces cost and execution time thus optimizing our query performance.