Question 1ev: Explain your answer to the previous part based on your knowledge from lectures, and details from the query plans (your explanation should include why you didn't choose certain options).

materialized views are stored as "tables" in the system. So it is faster and cost less when trying to use materialized wiew because there is no need to re-compute. However, there is a higher matainance cost to it. In contarst, Views are computed each time that you need to use the view which cost more. SO thats why D I are correct.

Question 2d: Explain your answer based on your knowledge from lectures, and details from the query plans (your explanation should include why you didn't choose certain options).

Filtering can lower the cost becuase it removes rows and reduce the size of the data. Thus when we do other table operations such as join and group the cost and time is lower. That's why I choode A E.

0.0.1 Question 3d

Please discuss your findings in the previous parts. In particular, we are interested in hearing why you think the query optimizer chooses the ultimate join approach in each of the above three scenarios. 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?

When the table is not sorted, the system perform Hash Join, which hash R and S into 2 different buckets and read all tuples hashed to the bucket from both R and S at a time and perform join. It is fast because sequence doesn't matter for this approach.

When the table is sorted, the system perform Merge Join, which sorted two tables first and then merge and match tuples across runs by walking down the runs in sorted order. The euqaljoin is faster but the table must be in sorted order in the first place in order to reduce cost.

When trying to find every copmbination of tuples pair, the system uses nested loop join that Match tuples across pairs of differente blocks. Since it scan through each block.

Question 4dii: Explain your answer based on your knowledge from lectures, and details from inspecting the query plans (your explanation should include why you didn't choose certain options).

I choose D E because index in gbatting index significantly reduce the cost and runtime but index on salary does not. I think the resason is that we group the salary therefore adding salary index doens't help reduce the cost and time.

Bbatting index reduces the data pages the systme need to scan through, so in the gbatting column, it reduce the data pages and narrow down the range we scan through using a B plus tree when joing two table.

Question 4evii Explain your answer to the previous part based on your knowledge from lectures, and details of the query plans (your explanation should include why you didn't choose certain options).

I choose "C I" because adding gbating index reduce cost and time for the AND case, and adding multicolumn index for OR case reduce cost and time.

For C because the system perform a nested loop join, addding a index on a column reduce the data pages the systme need to scan through when joining two table.

For "I" we are not joing to table becase the condition is OR. So we need to look for both columns in the two table and a multicolumn index help narrorw down range when we searching for value.

```
In [120]: %%sql
          drop index g_batting_g_all_idx
```

```
 * postgresql://jovyan@127.0.0.1:5432/baseball
   postgresql://jovyan@127.0.0.1:5432/postgres
Done.
```

```
Out[120]: []
```

0.0.2 Question 5d:

Explain your answer to the previous part based on your knowledge from lectures, and your inspection of the query plans.

When adding a index to find min the cost and time siginificantly reduce so I choose D.

When we need to find a MIN or MAX without index, we need to scan through every number in the column. Adding an index to the column can reduce the pages systme need to scan thorugh and narrow down the range of search, and thus reduce the cost and time.

0.0.3 Question 6e:

Explain your answer to the previous part based on your knowledge from lectures, and your inspection of the query plans (your explanation should include why you didn't choose certain options).

I choose "A", "D" because cluster on index reduce time and cost.

Clustering on the primary key will not always reduce cost and execution time because if the query is filtered on another column that's primary key, the execution time could stay the same or worsen due to traversing the index.

Clustering on index means that the rows in the data pages are in the same order as the rows in the index. It reduces the data pages the system need to scan through and narrow down the rage due to the indexes, and thus cost less and spend less time. For exmaple, if we knew that all values of A = a are together (as opposed to randomly shuffled), we can simply read one block as opposed to 100 blocks

0.0.4 Question 7c:

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?

It took more time when adding the index because index is not very good on tables that are modified more often than queried. If an index is not maintained, it may lead to inaccurate query results, and therefore if a new tuple is inserted but it is not reflected in the index, it won't be returned as part of queries and it cost more to scan. In addition, a constennly updating table may have low cardinality which index may not perform well.

0.1 Question 8: Takeaway

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.

Method 1: Create a table with materialized views cost less and spend less time because since materialized views are stored as "tables" in the system and there is no need to re-compute. However, there is a higher matainance cost to it. In contarst, Views are computed each time that you need to use the view which cost more.

Method 2: If tables are in sorted order, using a Merge Join, which sorted two tables first and then merge and match tuples across runs by walking down the runs in sorted order, could be more effcienct. The drawback is the table must be in sorted order in the first place in order to reduce cost.

Method3: When joining 2 tables, we can index one column because the system perform a nested loop join, addding a index on a column reduce the data pages the systme need to scan through when joining two table.