0.0.1 Question 2dii

1. **Explain your answer** to the previous part (Question 2di) based on your understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results - but bear in mind that sometimes EXPLAIN ANALYZE results may be unpredictable/hard to explain (because of contention, caching, noise/randomness).
2. If there were any options you did NOT select, **choose any one of them and explain why you did not select it.**

Please answer in at most 5 sentences and **explicitly state which answer option(s) you chose or didn't choose** in addition to your explanations. For example, you could write, "I chose (A) because…" or "I did NOT choose (B) because…"

I chose (D) and (J) because EXPLAIN ANALYZE results showed that materialized views significantly reduce both execution time and cost compared to standard views and direct queries. This aligns with the theoretical understading that materialized views store precomputed results, eliminating redundant comutation during query execution. However, materialized views take more time to create, as they execute and store results upon creation, which is why I selected (J). I did not choose (A) because views do not inherently reduce query execution time or cost-each time a query references a view, it still needs to be executed as if it were an inline query. This was confirmed by the fact that the execution cost and time of the view-based query were almost identical to running the full query directly.

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 understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results. Your response should be no longer than 3 sentences.

Note: Your answer should be formatted as follows if you think option A is true: `(A) because ...`

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

(B) because adding a filter reduced the number of rows processed early in the query execution, allowing the optimizer to apply predicate pushdown. The query cost decreased from 528.30 to 209.85, and execution time improved from 4.57ms to 0.757 ms. That demonstrates that filtering earlier in the execution plan reduces unnecesary computations, especially in joins and aggregations, leading to better query performance.

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 understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results. Your response should be no longer than 3 sentences.

Note: Your answer should be formatted as follows if you think option A is true: `(A) because ...`

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

(B) Again because adding a filter reduced the execution time from 4.57 ms to to 0.757 ms. The filter on year_id > 2010 allowed the optimizer to eliminate rows early in the query execution reducing the number of rows processed in subsequent operations. This demonstrates predicate pushdown, where applying filters early in the query plan minimizes unnecessary computations and improves performance.

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? How does table size affect which join is used?

Note: - **Explain your answer** based on your understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results - but bear in mind that sometimes EXPLAIN ANALYZE results may be unpredictable/hard to explain (because of contention, caching, noise/randomness). - Restate your answer for each of the subparts. You should write no more than 5 sentences. Your answer should be formatted as follows:

```
Q4a: (A) because ...
Q4b: (A) because ...
Q4c: (A) because ...
```

Q4a:(C) because the optimizer chose a Hash Join due to its efficiency in handling large tables with an equality join condition, avoiding the higher computational cost of Nested Loop Join.

Q4b: (B) because the optimizer selected Merge Join since the query included ORDER BY, and sorting allows efficient processing when inputs are ordered.

Q4c: (A) because the inequality condition ('<>') prevents the use of a Hash Join, requiring the optimizer to use a Nested Loop Join, which is better suited for nonequijoin conditions.

0.3.1 Question 5di Justification

1. **Explain your answer to Question 5d** above based on your understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results - but bear in mind that sometimes EXPLAIN ANALYZE results may be unpredictable/hard to explain (because of contention, caching, noise/randomness).
2. If there were any options you did NOT select, **choose any one of them and explain why you did not select it.**

Your answer should be **no longer than 3 sentences. Explicitly state which answer option(s) you chose or didn't choose** in addition to your explanations. For example, you could write, "I chose (A) because…" or "I did NOT choose (B) because…"

I chose (D) because adding the appearances_g_batting_idx index significantly improved both the query cost and execution time, as seen in the reduced values 3567.06 to 2308.42 (cost) and 15.727 to 7.984 ms (exceution time). I also chose (F) because adding the salary_idx index improved execution (7.984 ms to 7.124 ms) but had no meaningful impact on cost. I did not chose (B) because, while appearance_g_batting_idx improved execution time, it also significantly reduced cost, make (D) a more accurate statement.

0.3.2 Question 6ei Justification

1. **Explain your answer to `Question 6e`** above based on your understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results - but bear in mind that sometimes EXPLAIN ANALYZE results may be unpredictable/hard to explain (because of contention, caching, noise/randomness).
2. If there were any options you did NOT select, **choose any one of them and explain why you did not select it.**

Your answer should be **no longer than 3 sentences** and **explicitly state which answer option(s) you chose or didn't choose** in addition to your explanations. For example, you could write, "I chose (A) because…" or "I did NOT choose (B) because…"

(B) is correct because AND predicates benefit from indexing by reducing both cost and execution time. (F) is correct because multi-column indexes work well with OR predicates, reducing both cost and execution time. (A), (C), and (E) are incorrect because they either underestimate or misrepresent the impact of indexing on execution time.

0.4 Question 7c

Given your findings from `Question 7`, which of the following statements is true? **Select one.**

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.

State and justify your answer.

1. **Explain your answer** based on your understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results - but bear in mind that sometimes EXPLAIN ANALYZE results may be unpredictable/hard to explain (because of contention, caching, noise/randomness).
2. **Of the answer options you did not select, choose any one of them and explain why that option is wrong.**

Your response should be no longer than 3 sentences.

Note: Your answer should be formatted as follows: `(A) because ...` and `Not (A) because ...`

(D) because MIN/MAX queries benefit from indexed columns by using index scans, while AVG/COUNT require full table scans, making indexes ineffective. Not (A) because indexes do not always improve query performance, as seen with AVG(salary), which still required a full scan.

0.4.1 Question 8di Justification

1. **Explain your answer to `Question 8d`** above based on your understanding both of the **theoretical aspects of query planning and optimization discussed in class**, as well as EXPLAIN ANALYZE results - but bear in mind that sometimes EXPLAIN ANALYZE results may be unpredictable/hard to explain (because of contention, caching, noise/randomness).
2. If there were any options you did NOT select, **choose any one of them and explain why you did not select it.**

Your answer should be **no longer than 3 sentences** and **explicitly state which answer option(s) you chose or didn't choose** in addition to your explanations. For example, you could write, "I chose (A) because…" or "I did NOT choose (B) because…"

I chose (A) and (D) because clustering on ab_idx significantly reduced both query and execution time, as shown in EXPLAIN ANALYZE results. This improvement is expected since indexing on ab allows the database to use bitmap index scan instead of a sequential scan, making lookups more effficient. I also chose (E) and (G) because clustering on batting_pkey slightly decreased cost but increased execution time, likely due to suboptimal data locality for this specific query.

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.

When the index on salary was added, the wall time for inserting 300,000 rows increased from 2.82s (no index) to 3.85s (with index). This happened because PostgreSQL had to update the salary_idx index every time a new row was inserted, adding overhead to the operation. The increase in time demonstrates the trade-off between faster queries with an index and slower insert performance due to index maintenance.

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 non-trivial, **distinctly different** methods to optimize query performance that you learned in this project. Some places to start looking are some of the question topics. 2. For each method, summarize how and why it can optimize query performance. Feel free to discuss any drawbacks, if applicable.

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

1. Indexing- creating indexes or frequently queried columns improves search performance by enabling indexed lookups instead of full table scans. However, indexes increase the time required for insert, update, and delete operations since the database must maintain the index structure.
2. Clustering- clustering a table based on an index physically reorders the rows to match the index order, reducing disk I/O and improving query performance for range scans. A drawback is that clustering does not persist when new data is added, requiring manual reclustering.
3. Query Excecution Analysis (EXPLAIN ANALYZE)- Using EXPLAIN ANALYZE helps identify inefficiencies in query execution, such as unnecessary sequential scans or missing indexes. It also allows developers to fine-tune queries and indexes but can be effected by caching and runtime variations, making repeated analysis necessary.