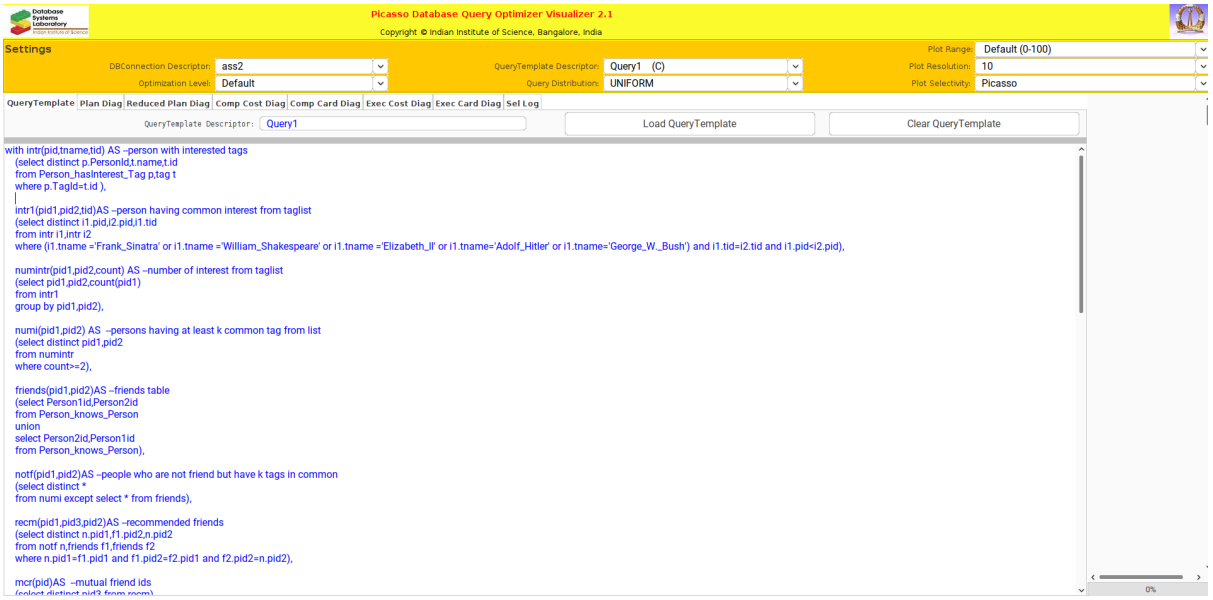
COL362
ASSIGNMENT 2

Sibasish Rout(2020CS10386)
Ankit Raushan (2020CS10324)

1. QUERY1

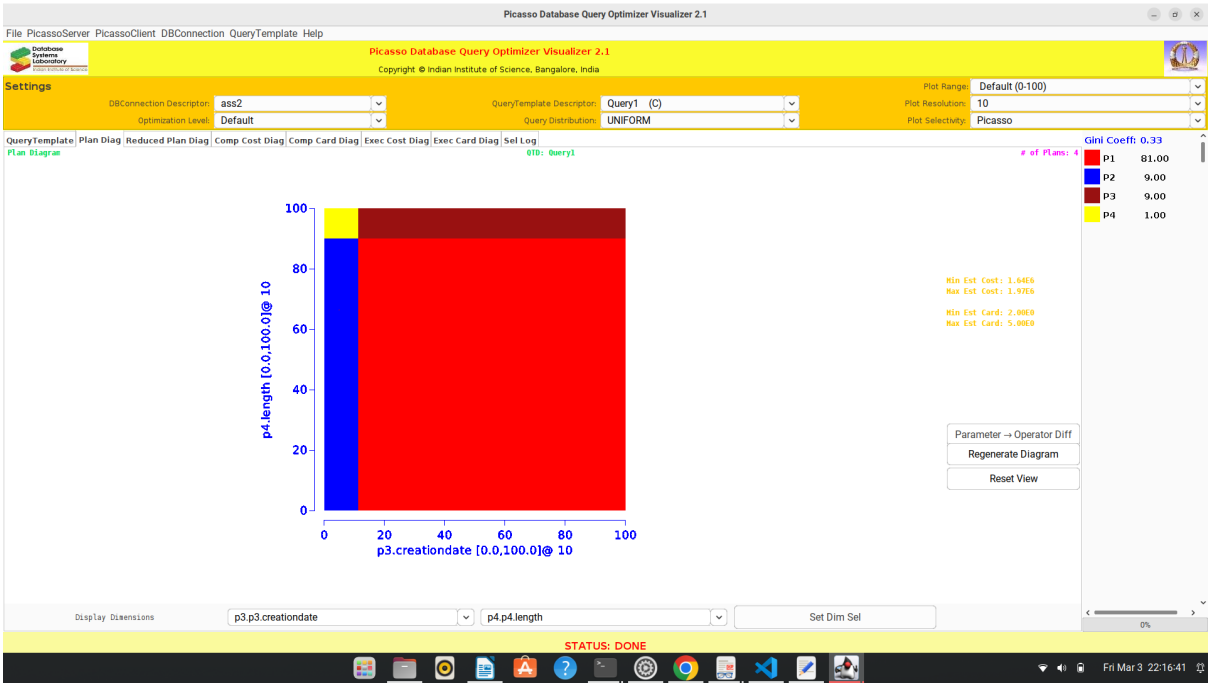
1.1 Query Template



1.2 Execution Plan Diagram

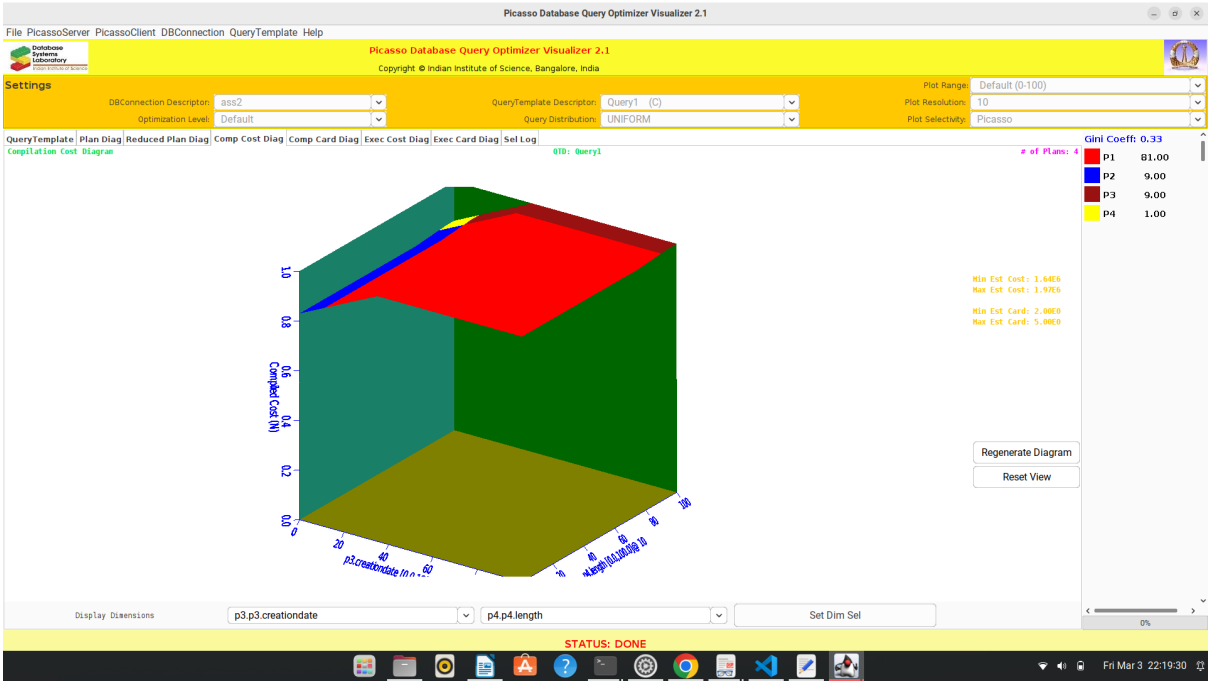
This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 3 plans are generated for different values of range of length and creationdate Which means that if we take 50% of creationdate column and 50% of length column then the Plan P1 is the most optimal as shown by the red value. Similarly we can also deduce the optimal plans for other regions also.The presence of multiple plans implies that we can still Optimise the query for cases when either of the tables has a lower number of values.



1.3 Compilation Cost diagram

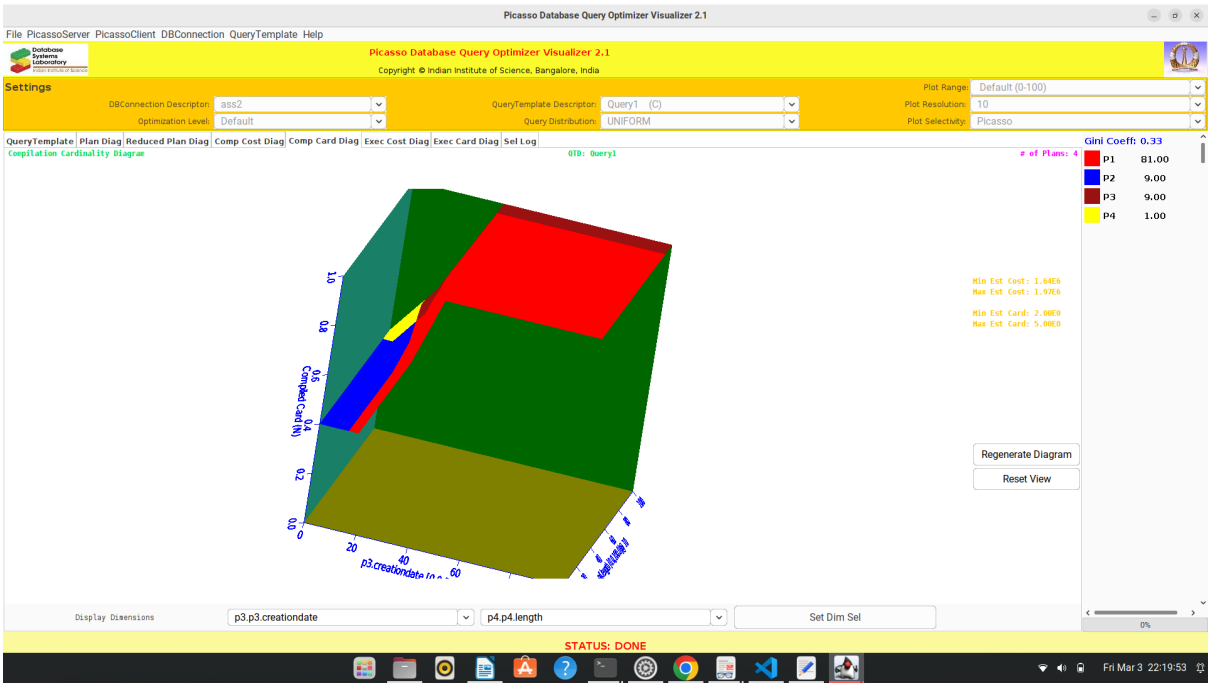
This graph represents the computation time along varying range of length and creationdate along the optimal plan.The value along the z axis in the diagram represents the computation time.The maximum computation time is when we use the whole range for both length and Creationdate. It shows strong correlation with creationdate attribute.



1.4 Compilation Cardinality Diagram

This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result Increases with the range of values taken from creationdate .

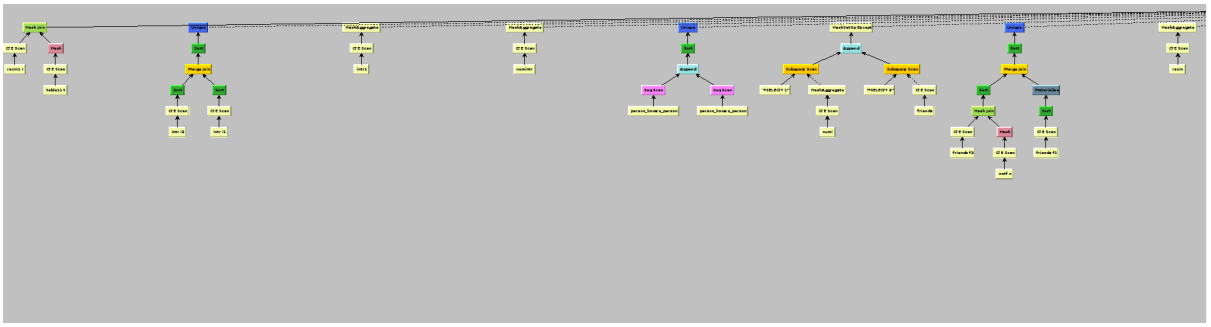


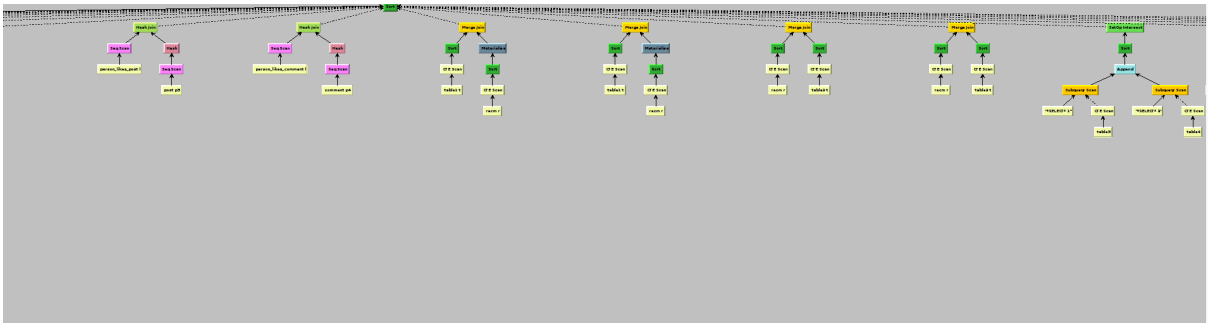
1.5 Plan diagrams

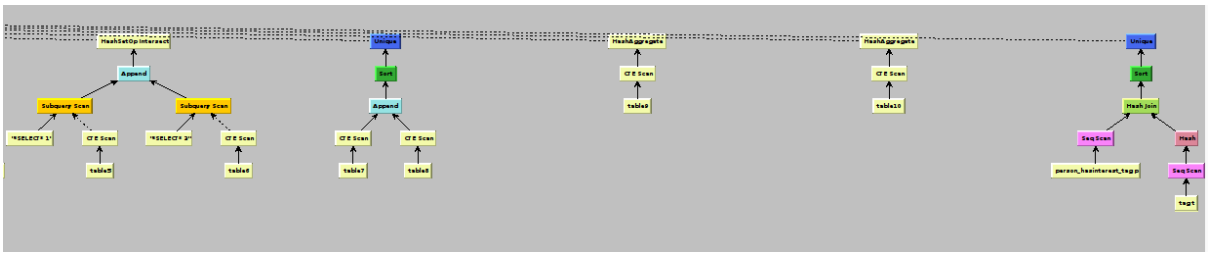
Plan 1

Fig: Total plan 1 diagram

Since plan diagram is not visible I am pasting screenshots dividing it in 3 parts







Plan diagram is overall almost balanced and distributed. From the diagram we can see that Merge join, hash join, sort, append, scan etc. has been used.

Like table1 and table2 are hashed separately and then merge joined,

table5 and table6 is scanned separately and then appended,

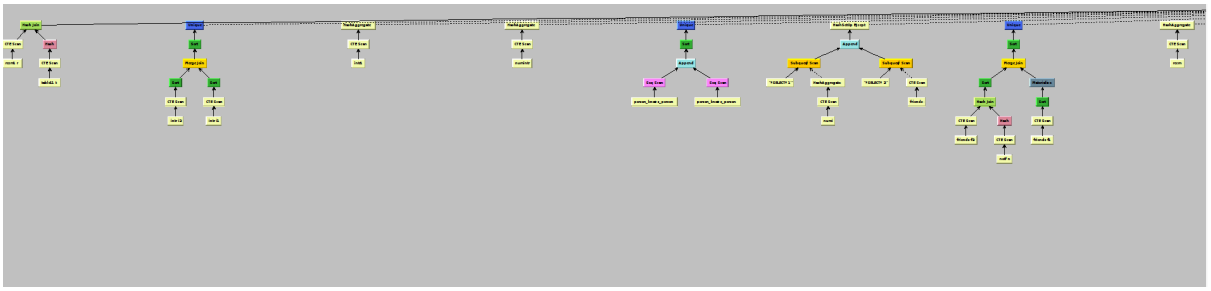
table7 and table 8 have been appended,

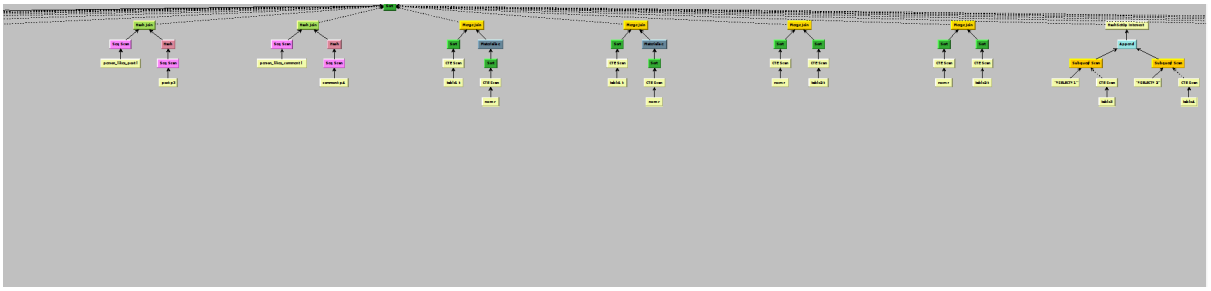
Person_hasinterest_tag is scanned tag table is hashed and both are then hash joined

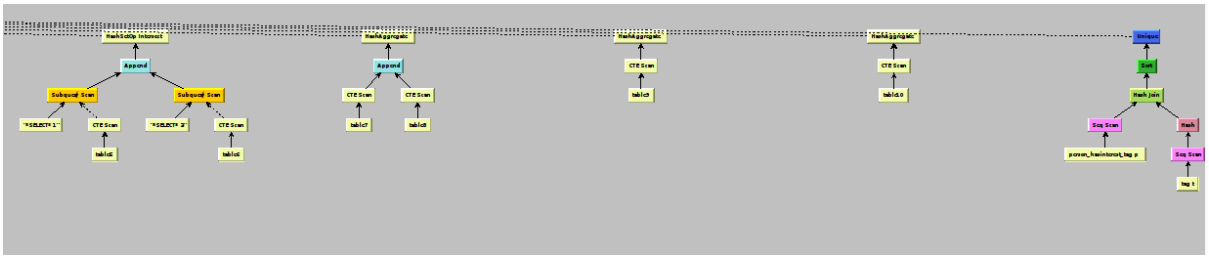
Plan 2

Fig: Total plan 2 diagram

Screenshots of splitted plan 2 diagram







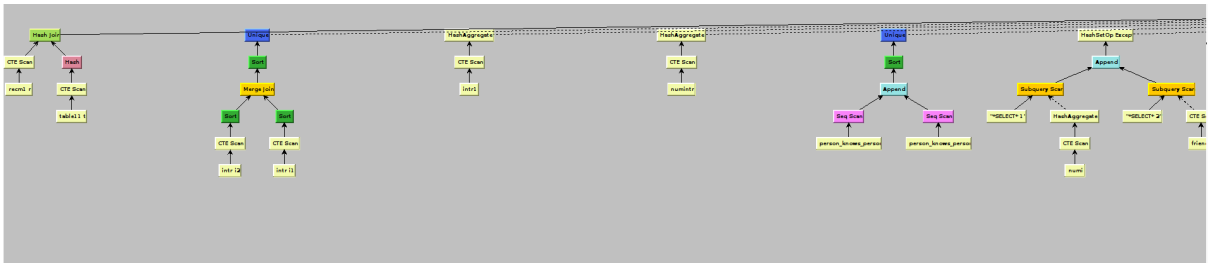
Plan diagram is overall almost balanced and distributed like plan 1

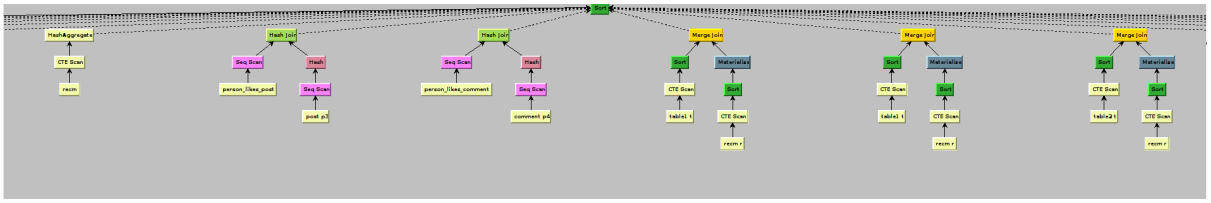
Merge join, hash join, sort, append, scan etc. has been used.

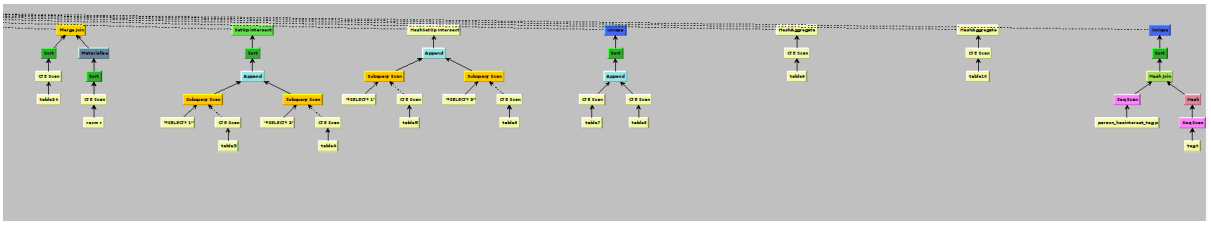
Plan 3

Fig: Total plan 3 diagram

Screenshots of plan 3 after splitting it







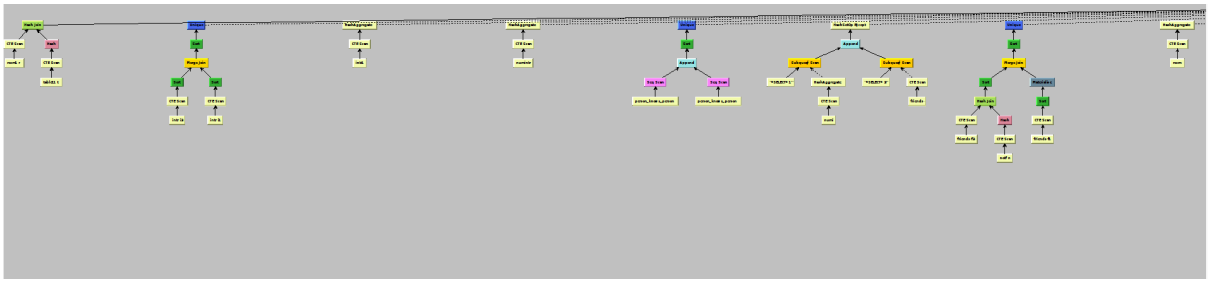
Plan diagram is overall almost balanced and distributed like plan 1 and 2

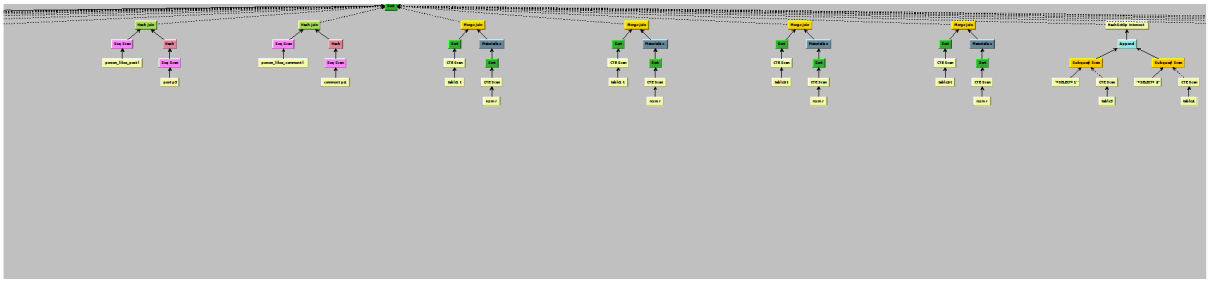
Merge join, hash join, sort, append, scan etc. has been used.

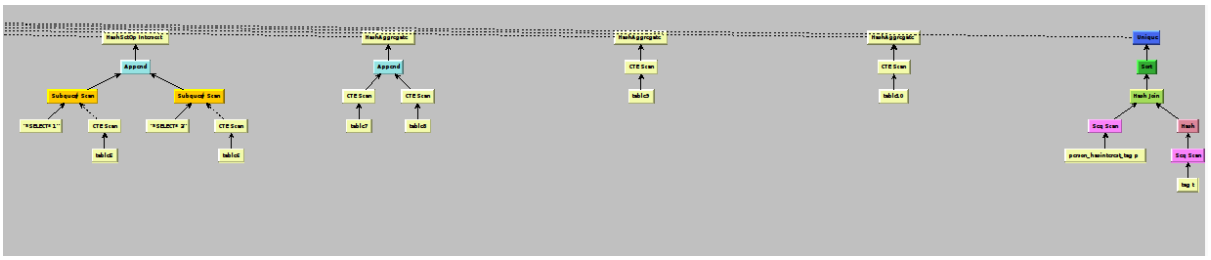
Plan 4

Fig: Total plan 4 diagram

Screenshot of splitted plan 4





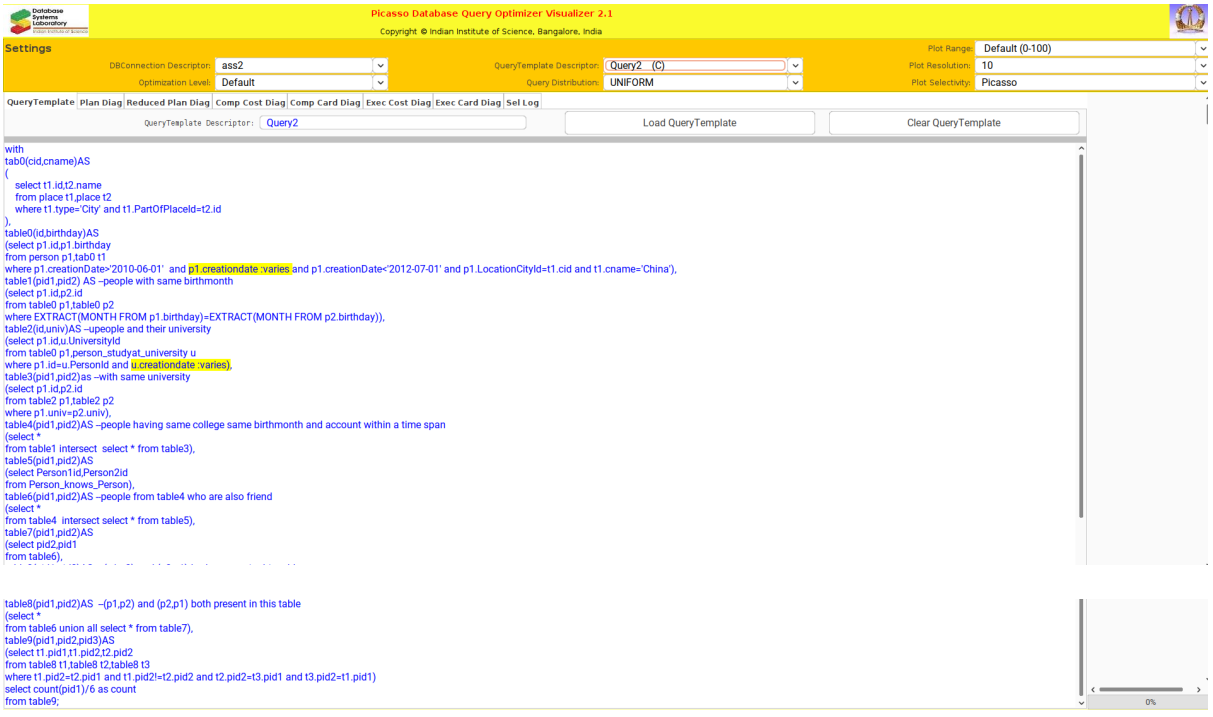


Plan diagram is overall almost balanced and distributed like plan 1,2 and 3

Merge join, hash join, sort, append, scan etc. has been used.

2. QUERY 2

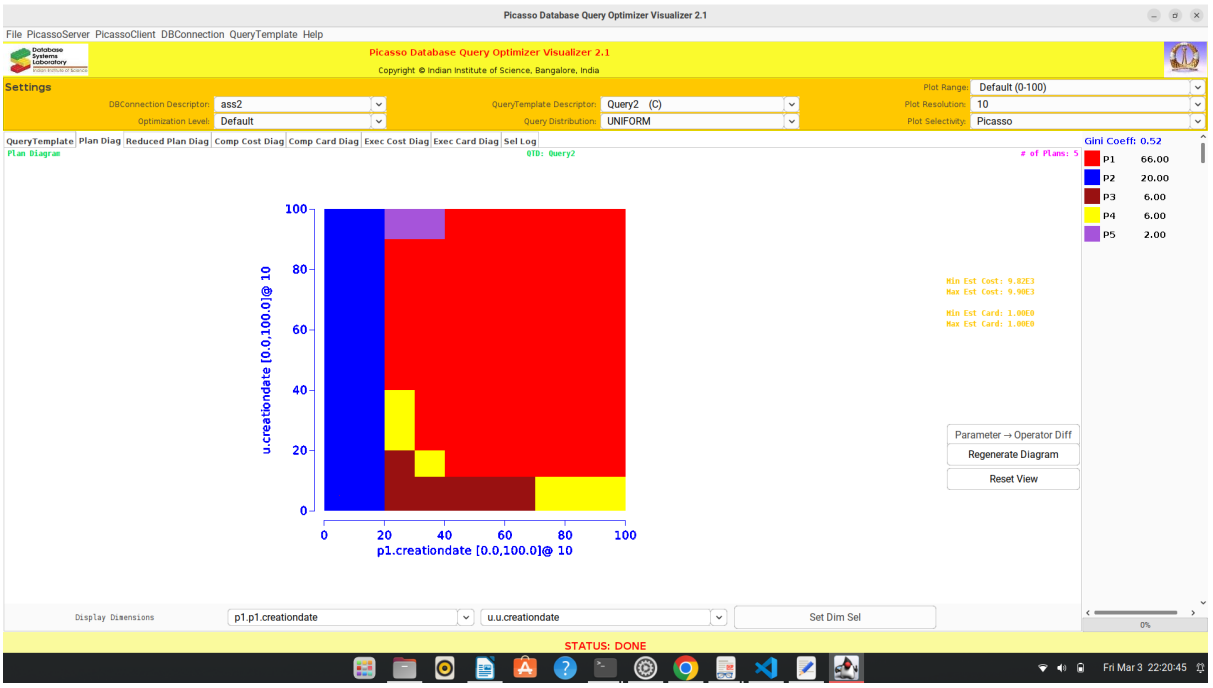
2.1 Query Template



2.2 Execution plan diagram

This diagram represents the plan that is optimal at various percentages of the varying tables For eg here 5 plans are generated for different values of range of length and creationdate Which means that if we take 50% of person creationdate column and 50% of Person_studyat_university creationdate column then the

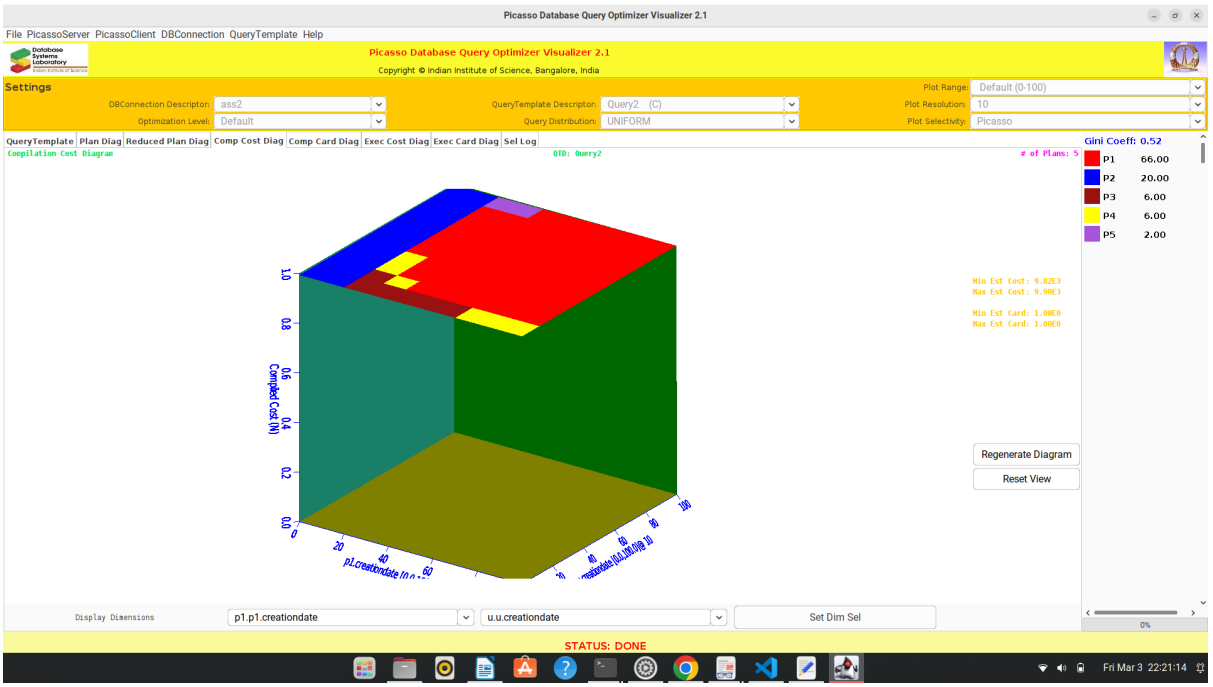
Plan P1 is the most optimal as shown by the red value. Similarly we can also deduce the optimal plans for other regions also.The presence of multiple plans implies that we can still Optimise the query for cases when either of the tables has a lower number of values.



2.3 Compilation cost diagram

This graph represents the computation time along varying range of person creationdate and person_studyat_university creationdate

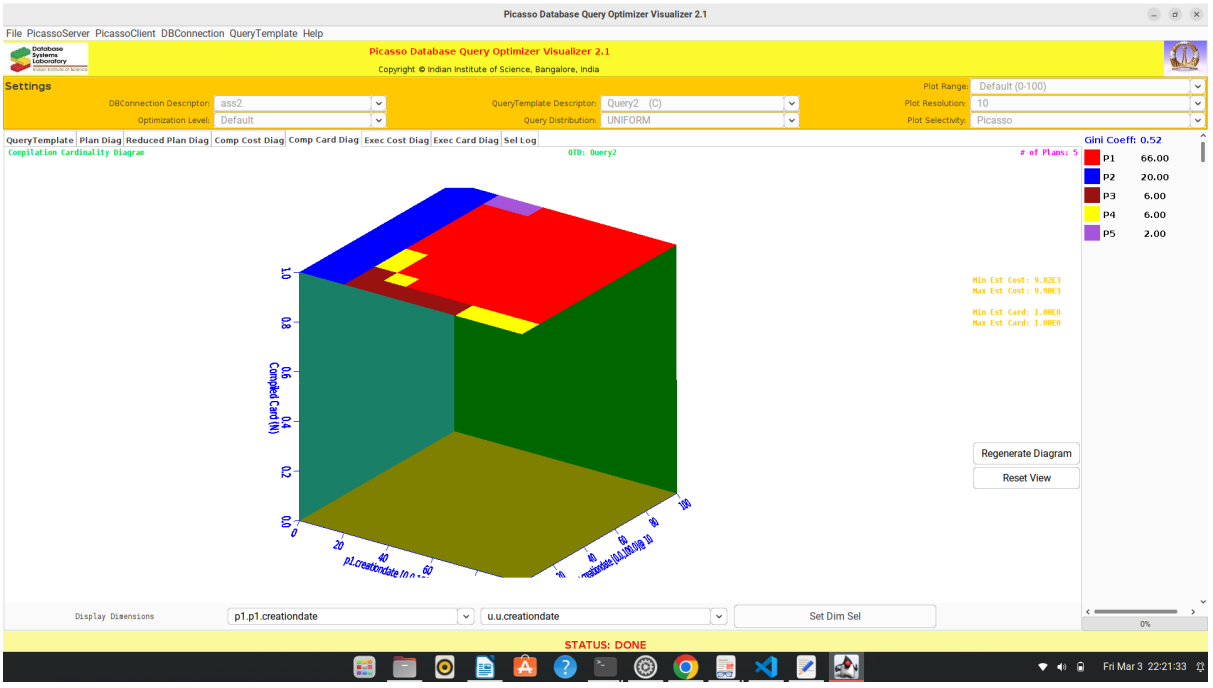
along the optimal plan.The value along the z axis in the diagram represents the computation time.The maximum computation time is when we use the whole range for both Creationdates.The computation time mostly remains same with change in the predicates.



2.4 Compilation cardinality diagram

This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result More or less remains the same with change in predicate values.



2.5 Plan diagrams

Plan 1

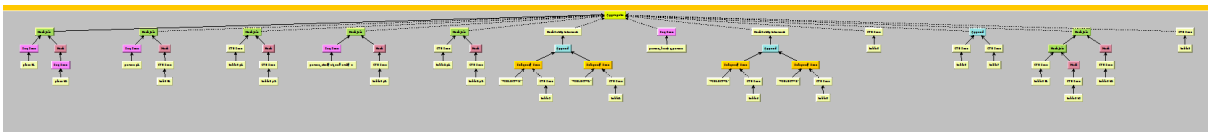
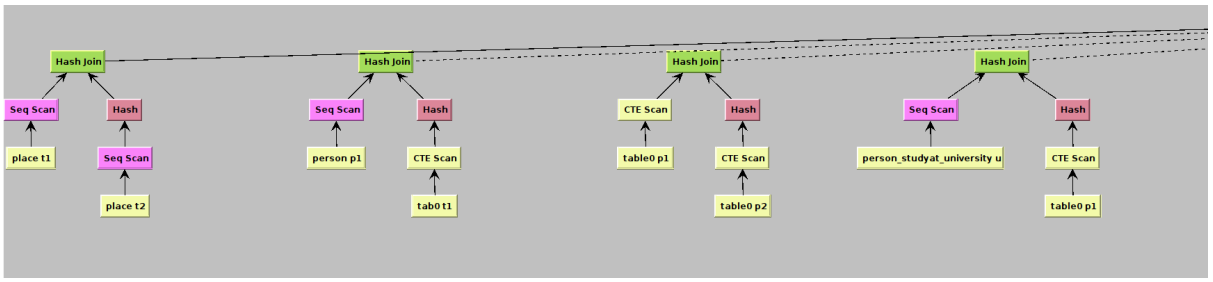
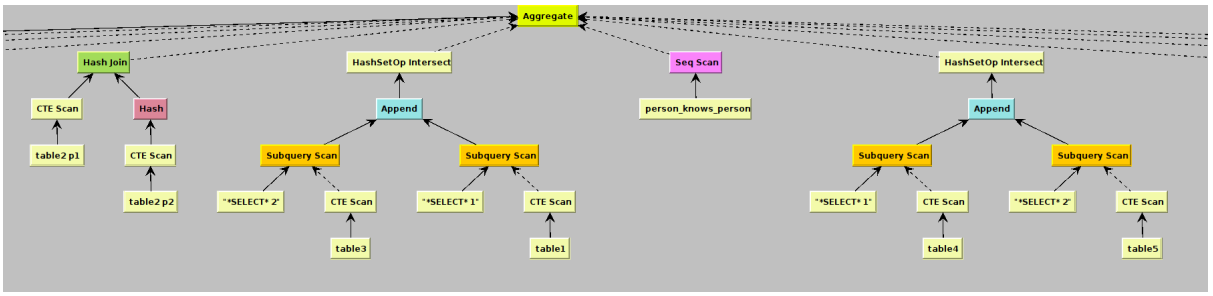
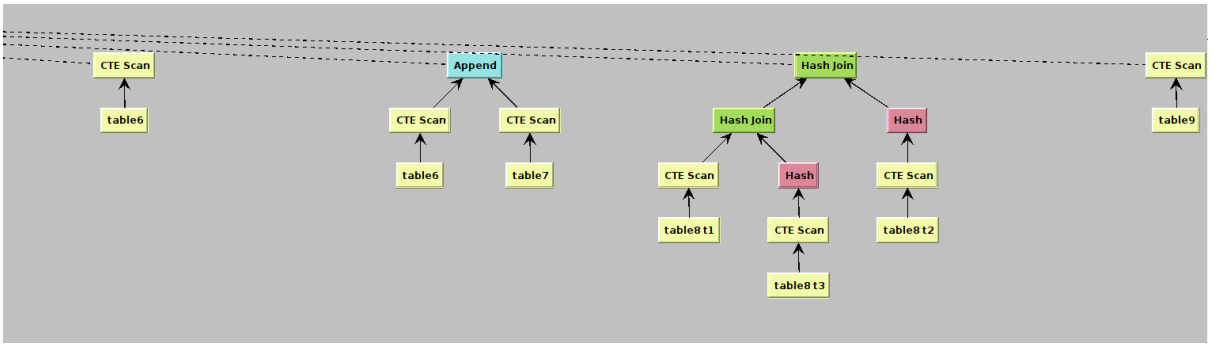


Fig: Total plan 1 diagram

Since it is not clearly visible I am pasting plan diagram by dividing it







From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t3, table8 t2 is hashed and then hash Joined with resultant of t1 and t3

Plan 2

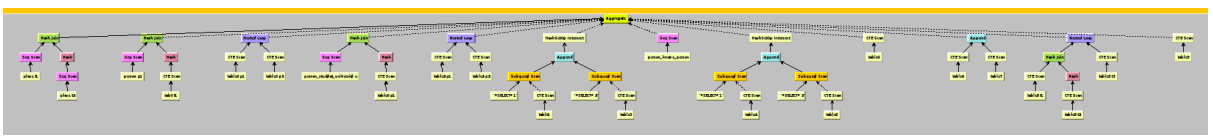
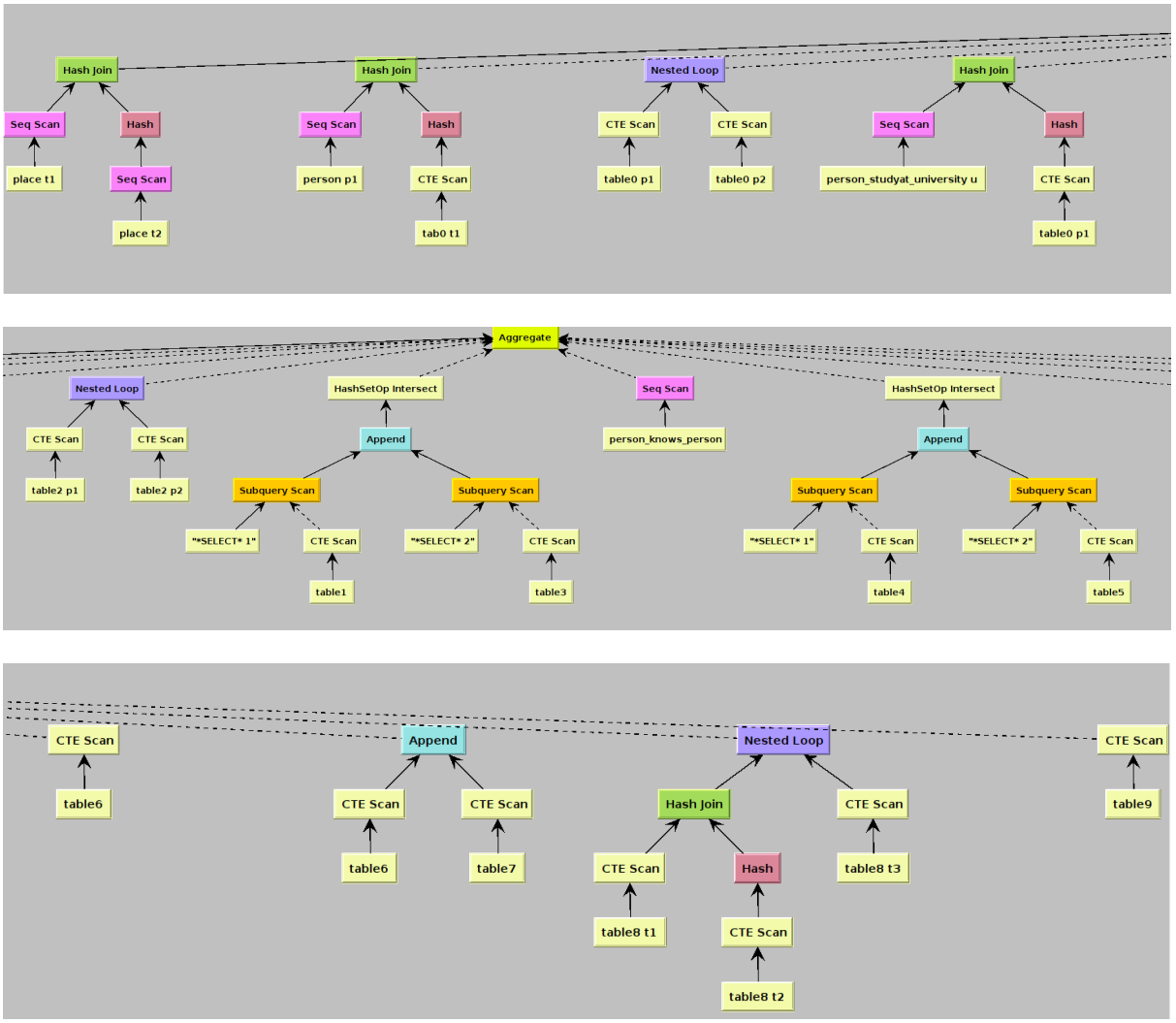


Fig: Total plan 2 diagram



From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table like plan1

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t2, table8 t3 is hashed and then hash Joined with resultant of t1 and t2 unlike plan1

Plan 3

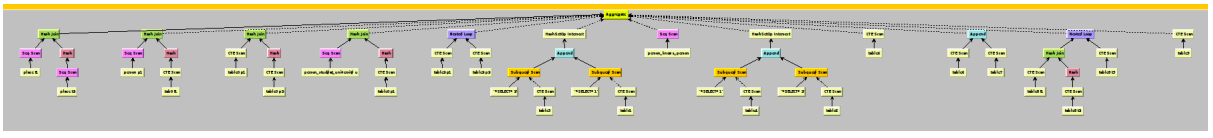
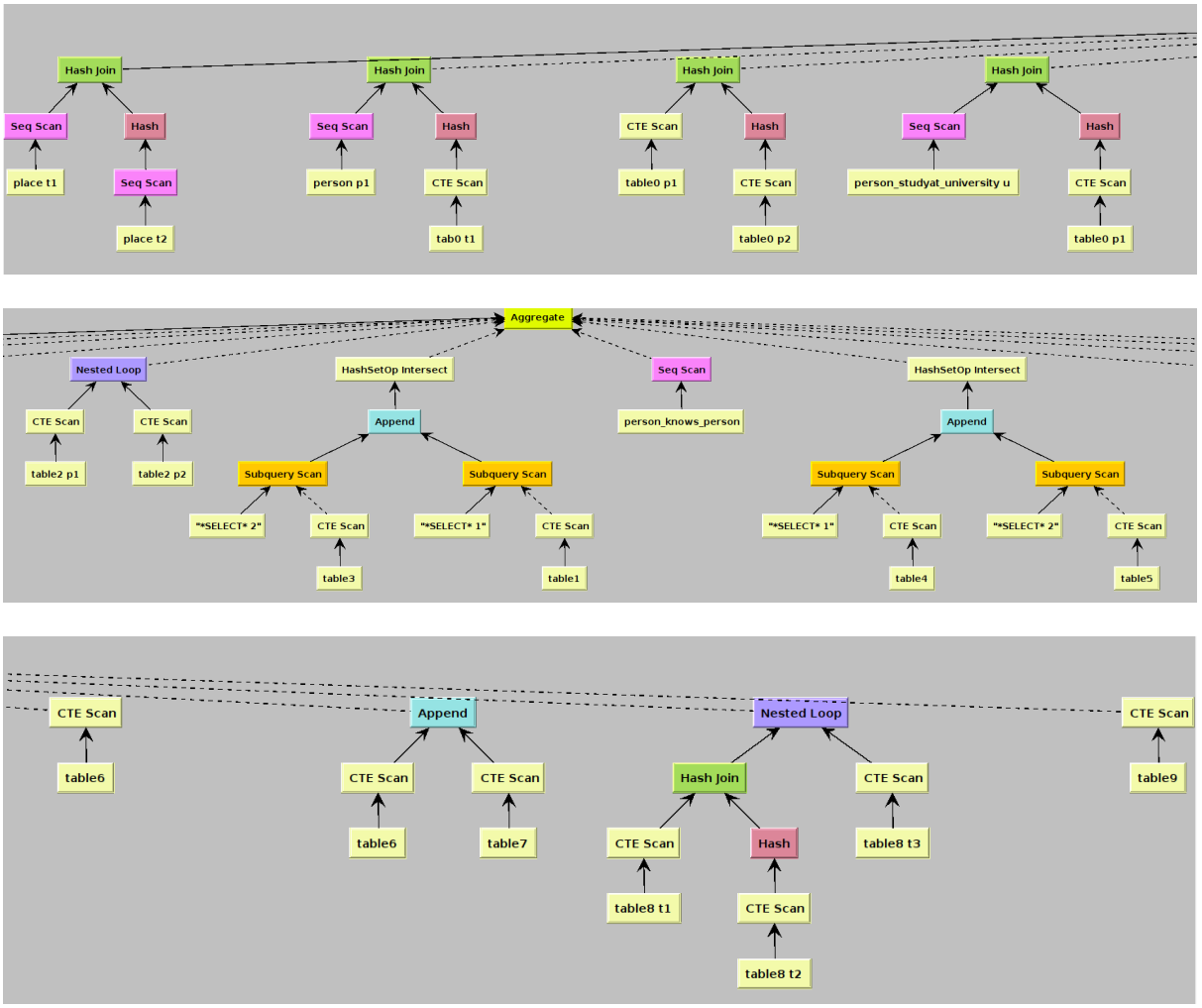


Fig: Total plan 3 diagram



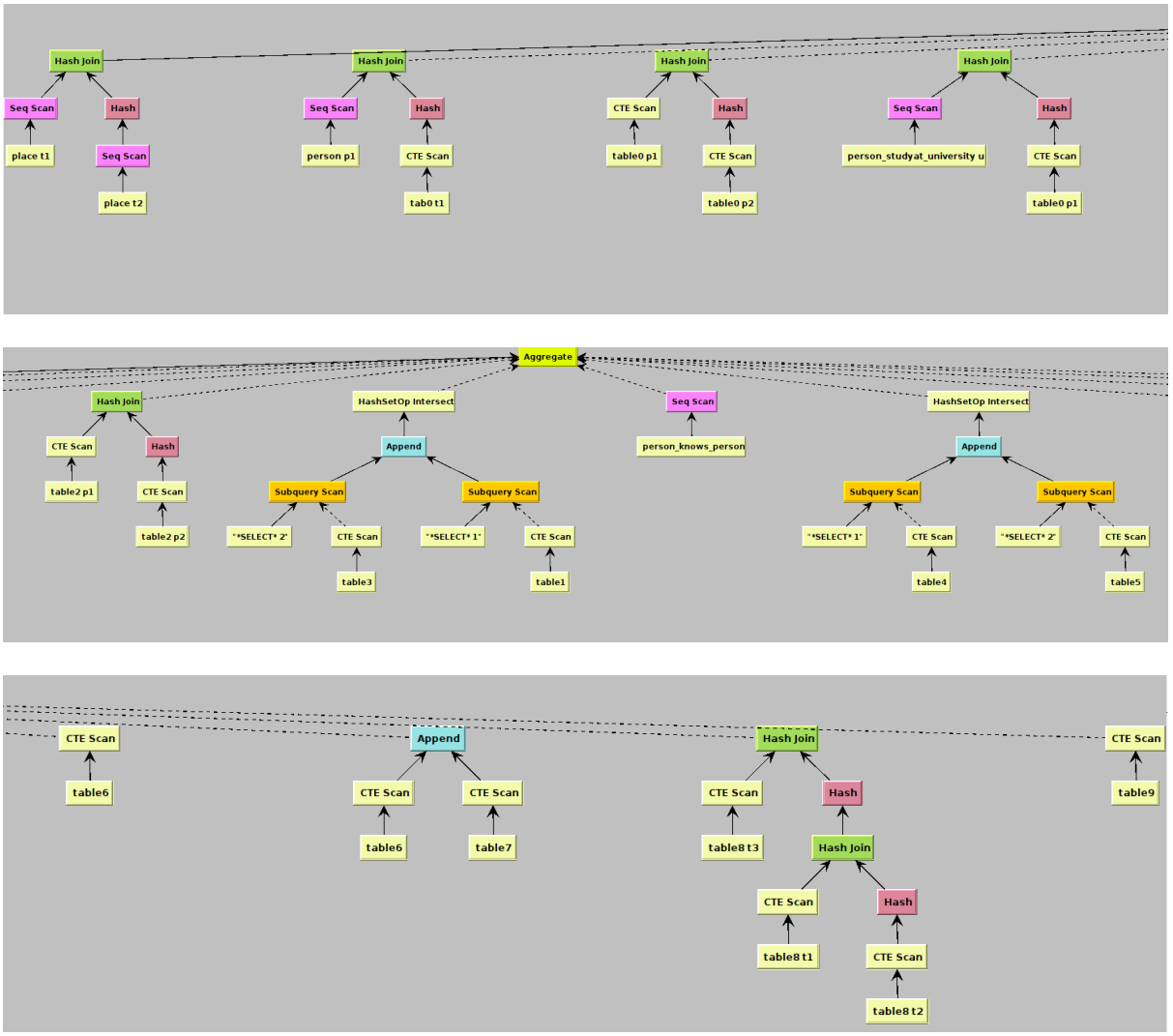
From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t2, table8 t3 is hashed and then hash Joined with resultant of t1 and t2

Plan 4

Fig: Total plan 4 diagram



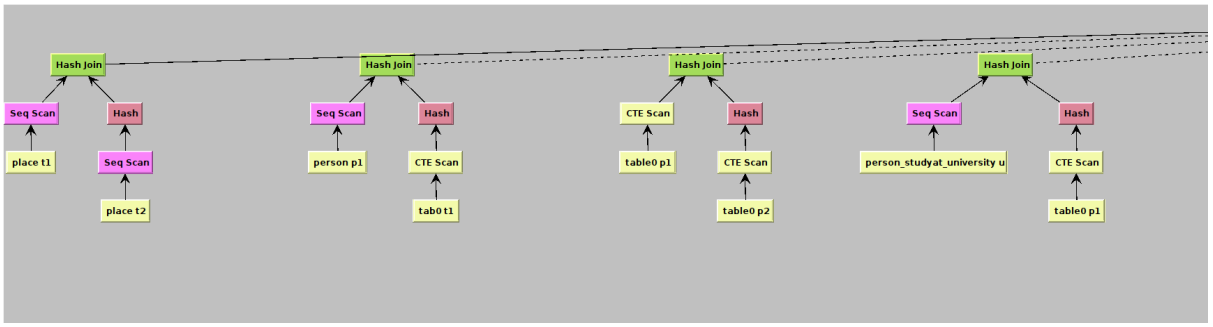
From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

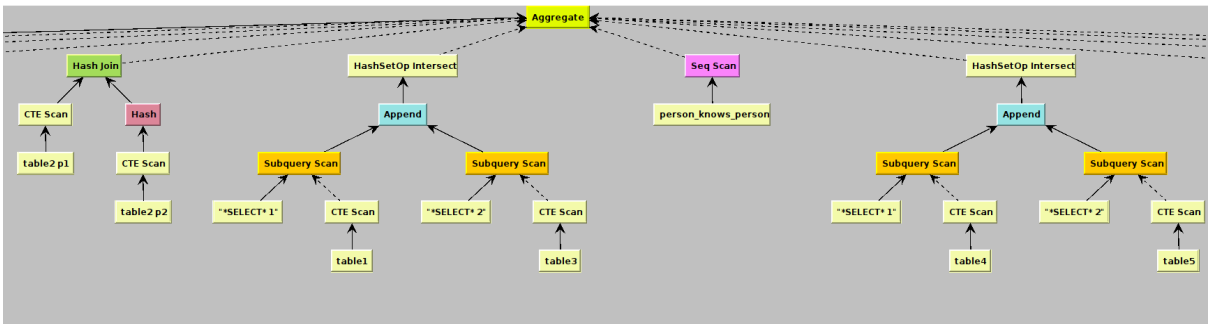
Table 3 and table 1 are scanned and appended separately,

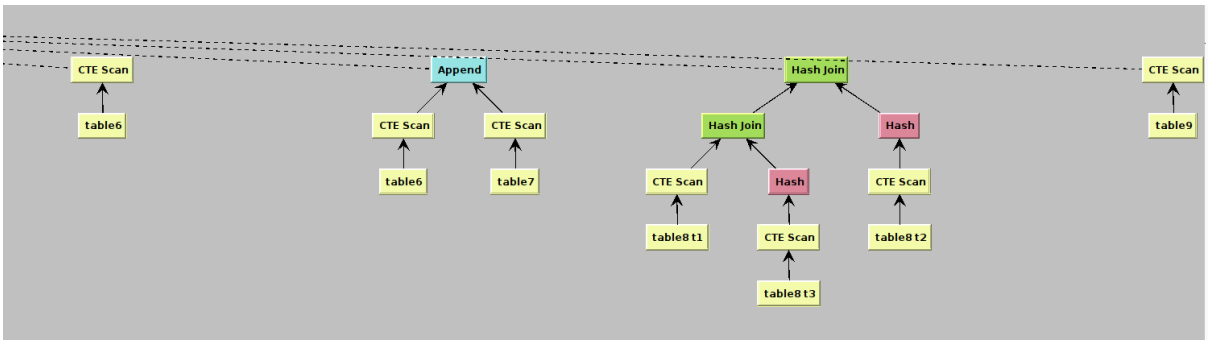
Table8 t3 is scanned and hash joined with table8 t1, table8 t2 is hashed and then hash Joined with resultant of t3 and t1

Plan 5

Fig: Total plan 5 diagram







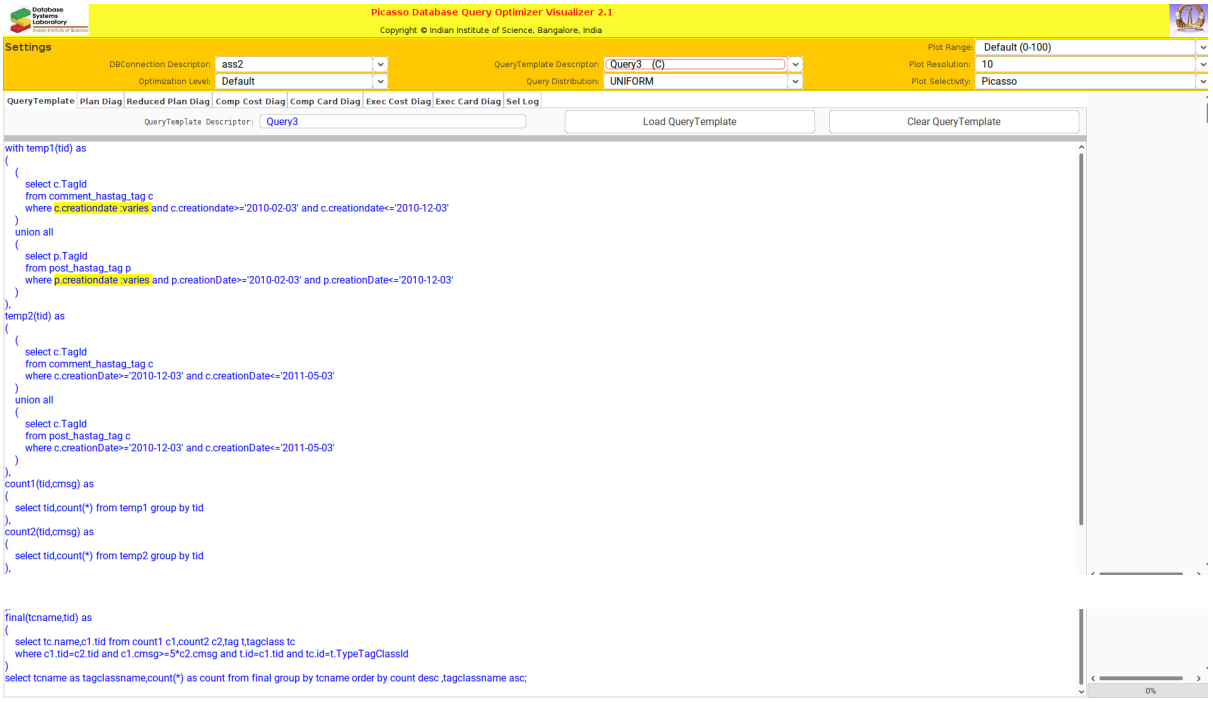
From diagram we can see that place t1 table is scanned and then hash joined with hashed Place t2 table,

Table 3 and table 1 are scanned and appended separately,

Table8 t1 is scanned and hash joined with table8 t3, table8 t2 is hashed and then hash Joined with resultant of t1 and t3

3. QUERY 3

3.1 Query Template



3.2 Execution Plan Diagram

This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 1 plan is generated for different values of range of length and creationdate

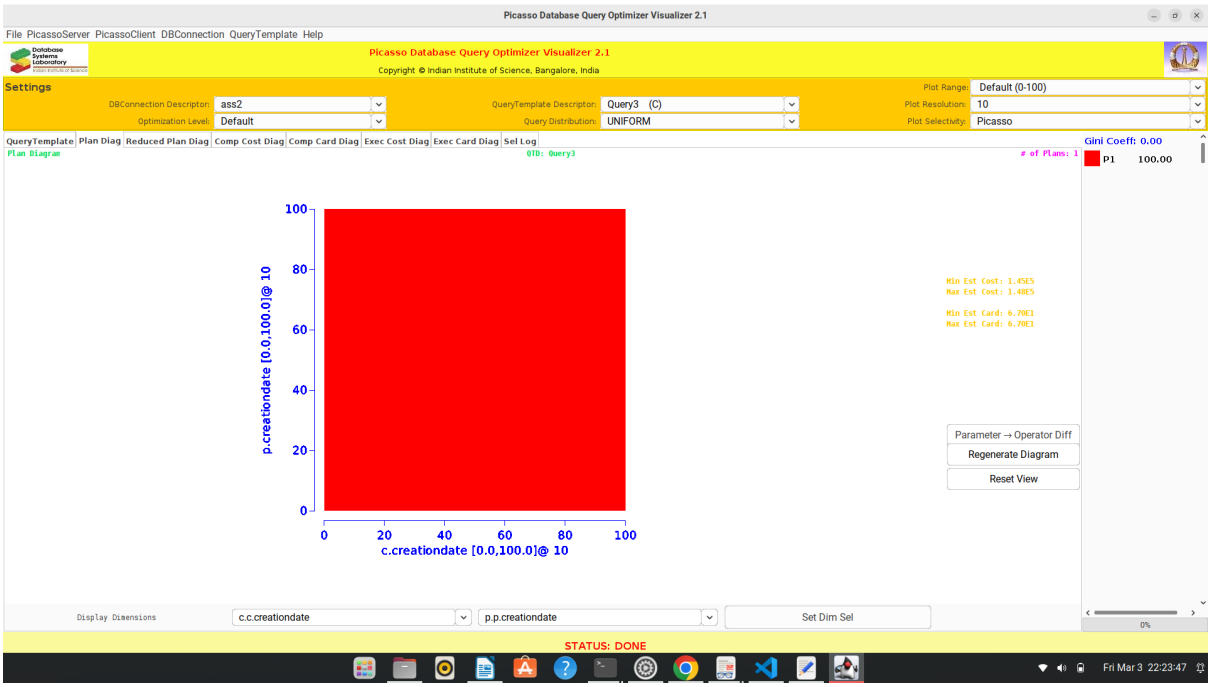
Which means that for any value of range of post creationdate and comment creationdate

We only have one plan given in red.

.Similarly we can also deduce the optimal plans for other regions also.

The presence of single plan implies that we have a

Optimal query plan for all ranges of predicate values which means that our query is quite efficient.

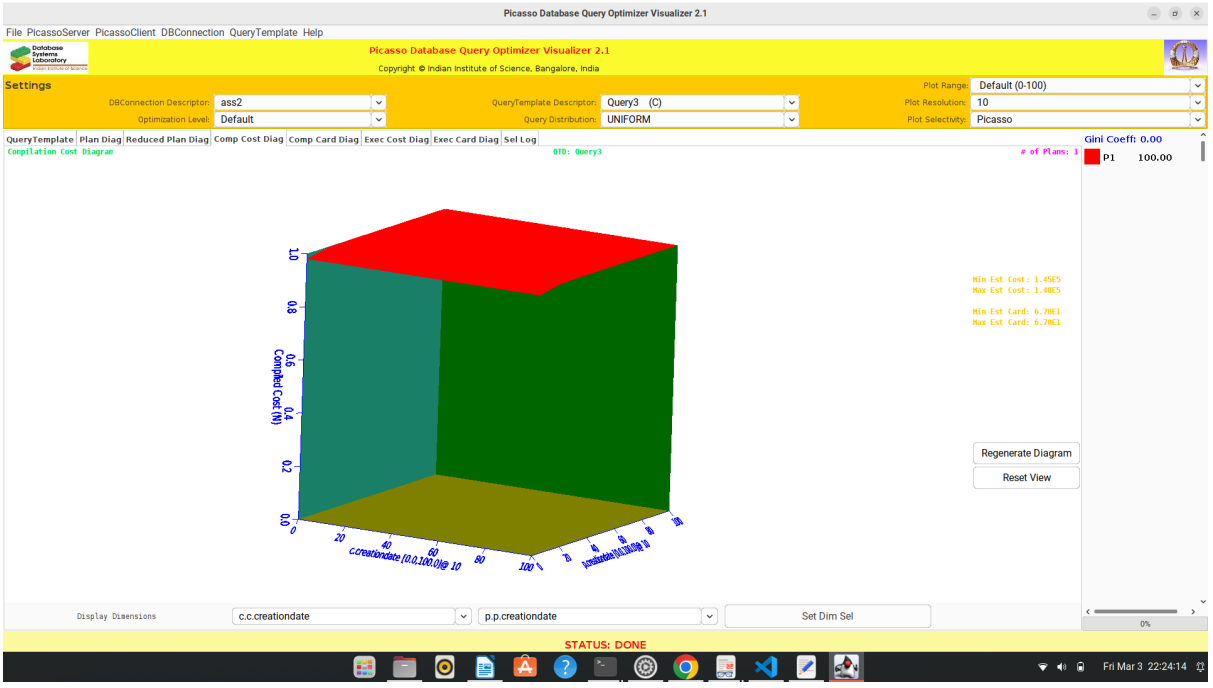


3.3 Compilation Cost diagram

This graph represents the computation time along varying range of post creationdate and comment creationdate

along the optimal plan.The value along the z axis in the diagram represents the computation time.The maximum computation time is when we use the whole range for creationdate Value of both the predicates

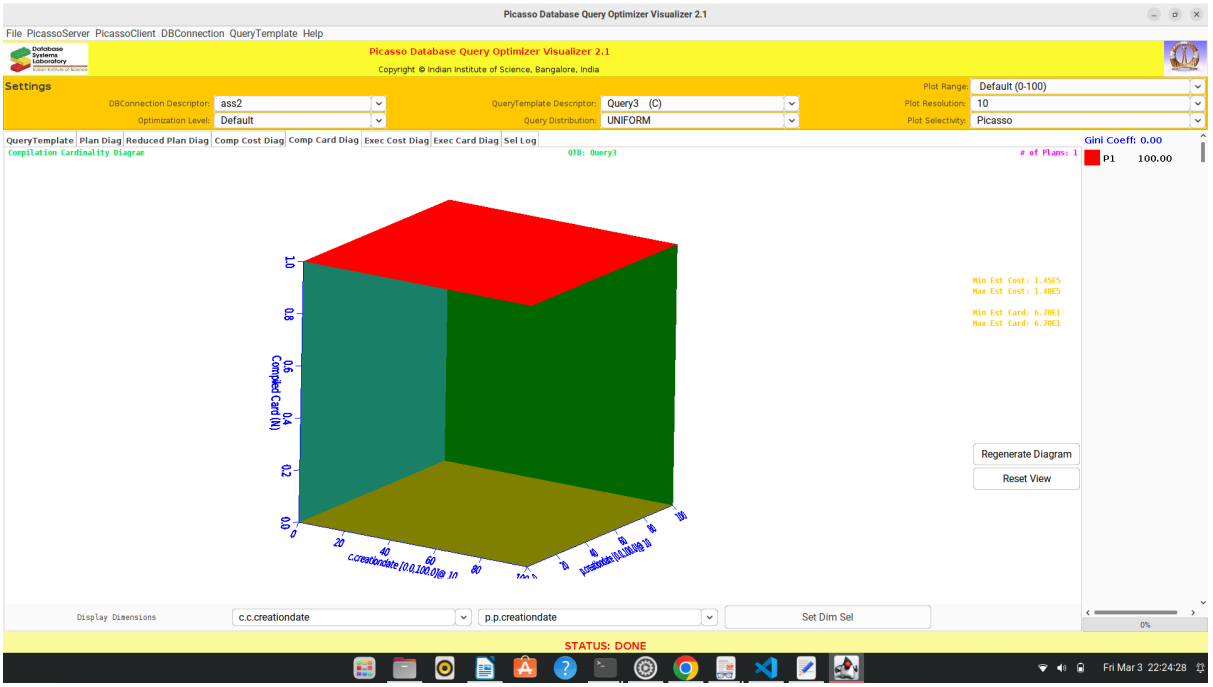
.The computation time mostly remains same with change in the predicates.



3.4 Compilation Cardinality Diagram

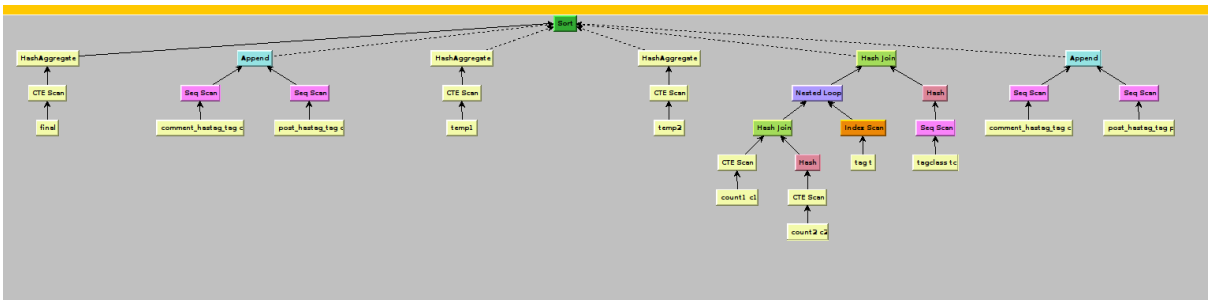
This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result More or less remains the same with change in predicate values.



3.5 Plan Diagram

Plan

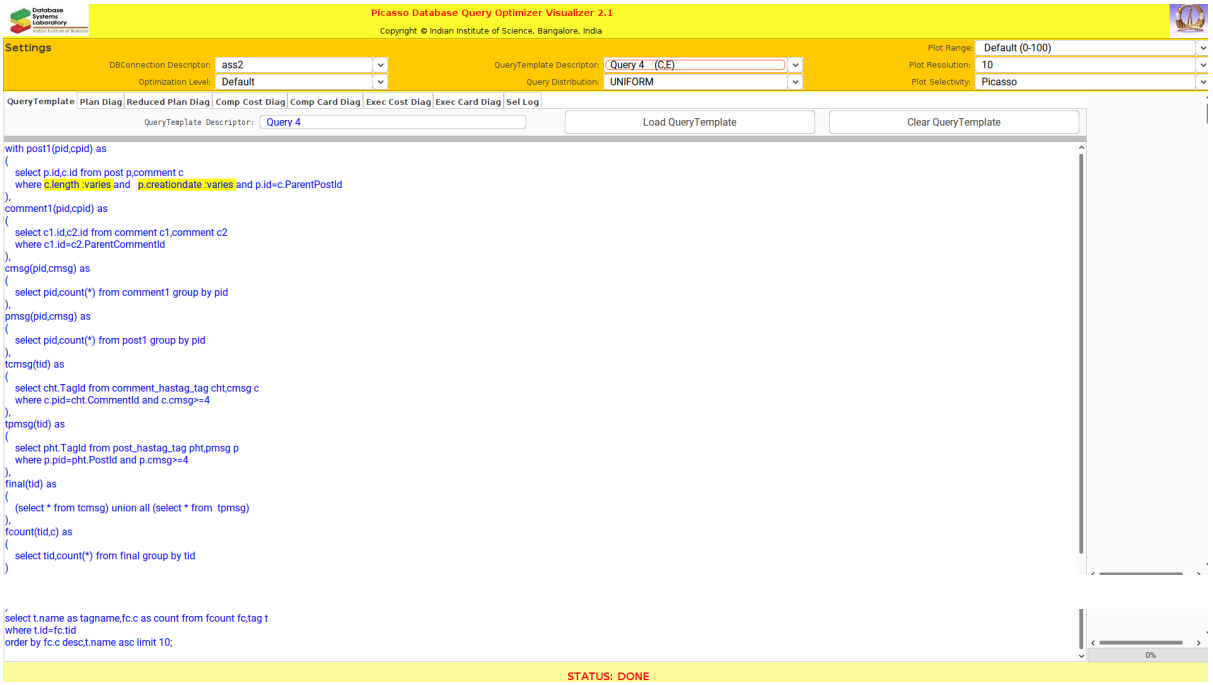


The plan diagram is quite balanced except along one subtree where it becomes a little Skewed as the height is relatively same across the different

Subtrees. The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan in one subtree .

4. QUERY 4

4.1 Query Template



4.2 Execution Plan Diagram

This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 1 plan is generated for different values of range of length and creationdate

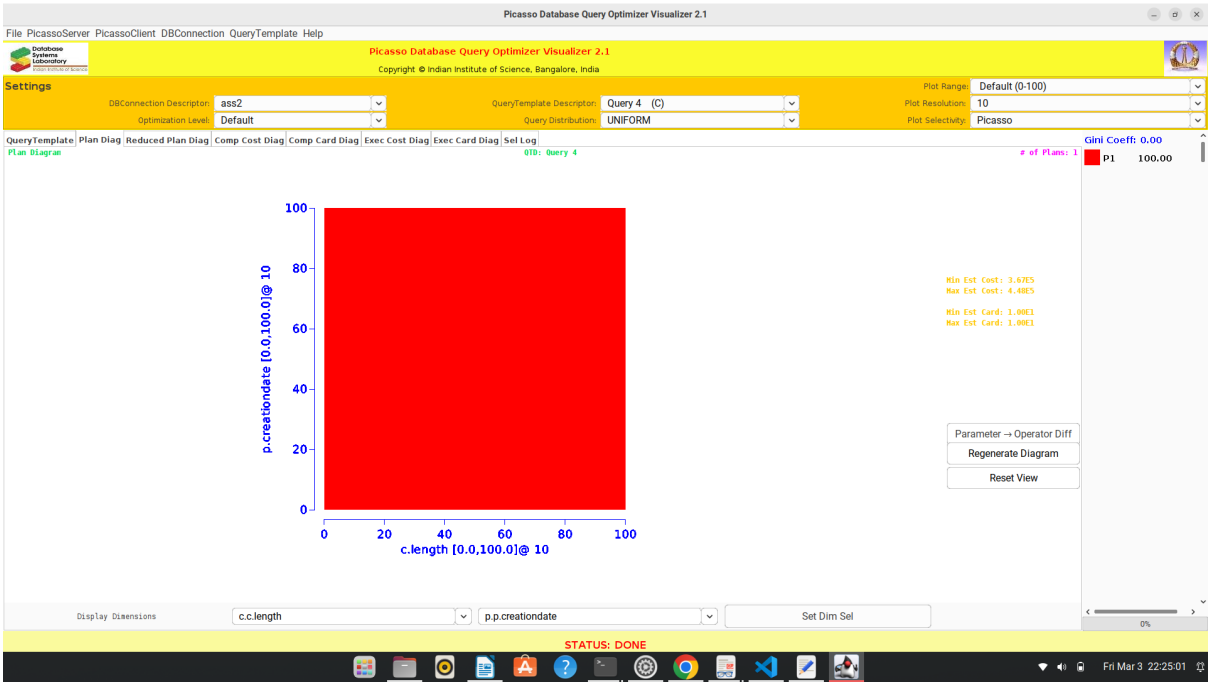
Which means that for any value of range of post creationdate and comment creationdate

We only have one plan given in red.

.Similarly we can also deduce the optimal plans for other regions also.

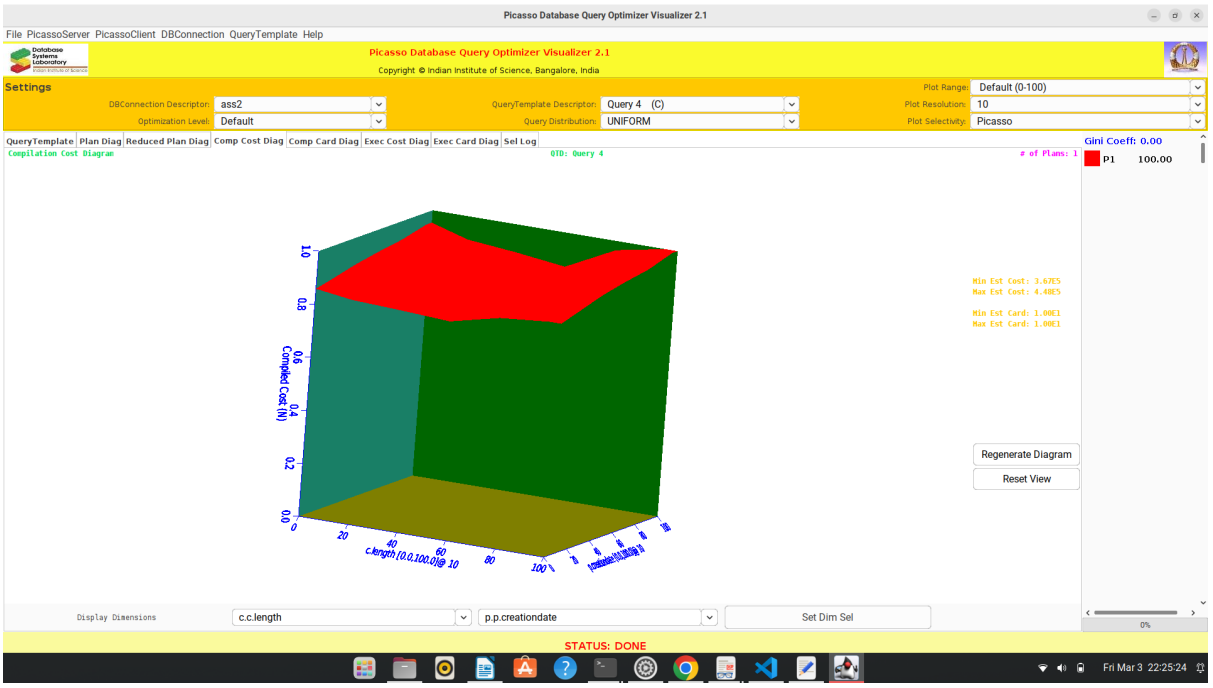
The presence of single plan implies that we have a

Optimal query plan for all ranges of predicate values which means that our query is quite efficient.



4.3 Compilation Cost diagram

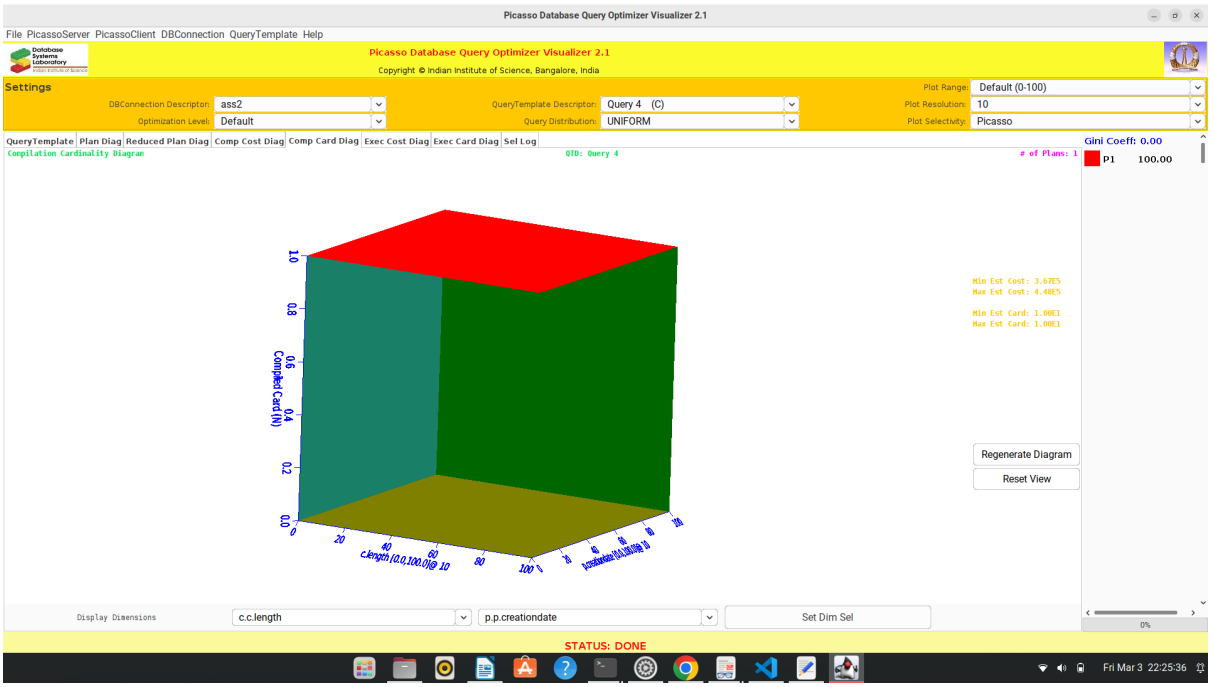
This graph represents the computation time along varying range of length and creationdate along the optimal plan.The value along the z axis in the diagram represents the computation time.The maximum computation time is when we use the whole range for both length and Creationdate. It showsdependence on both length and creationdate.



4.4 Compilation Cardinality Diagram

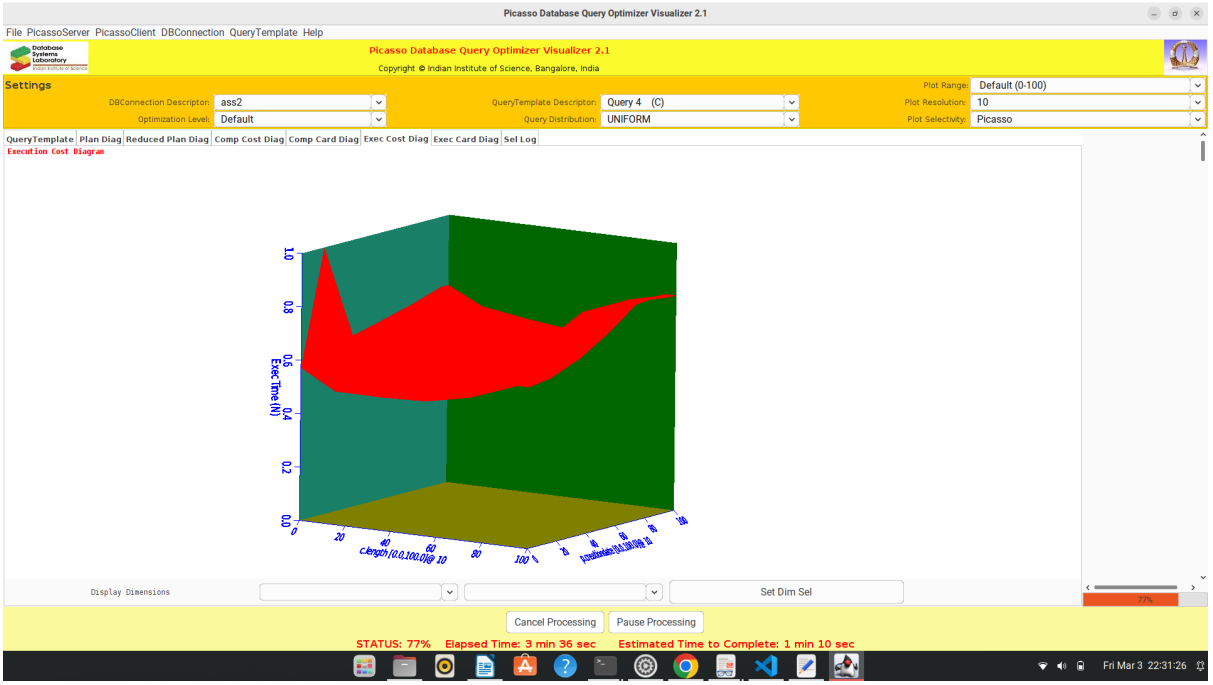
This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result More or less remains the same with change in predicate values.



4.5 Execution Cost diagram

This diagram represents the runtime query response times for different portions of the tables Used. From the diagram it appears that when we use small portion of both the tables the Response time is quite high. When we use almost 50% values of both the the tables the Response time for the query is minimum.. Also when we use large portion of both tables We get thigh response time.



4.6 Plan Diagram

Plan

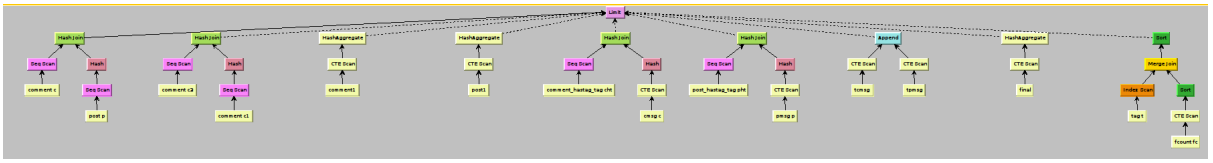
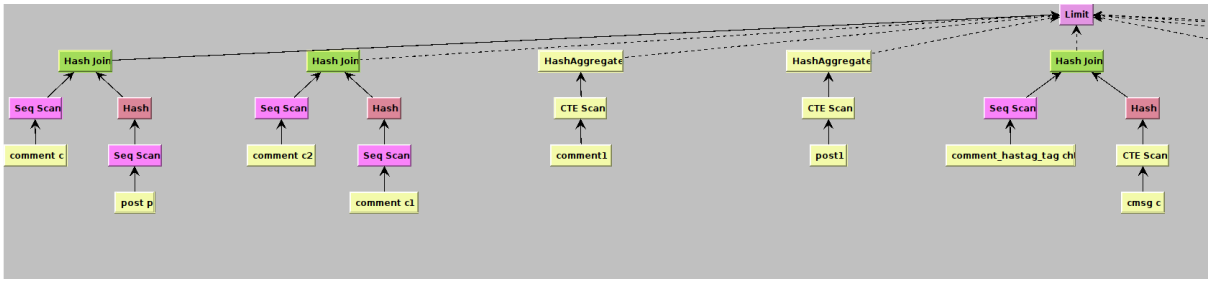
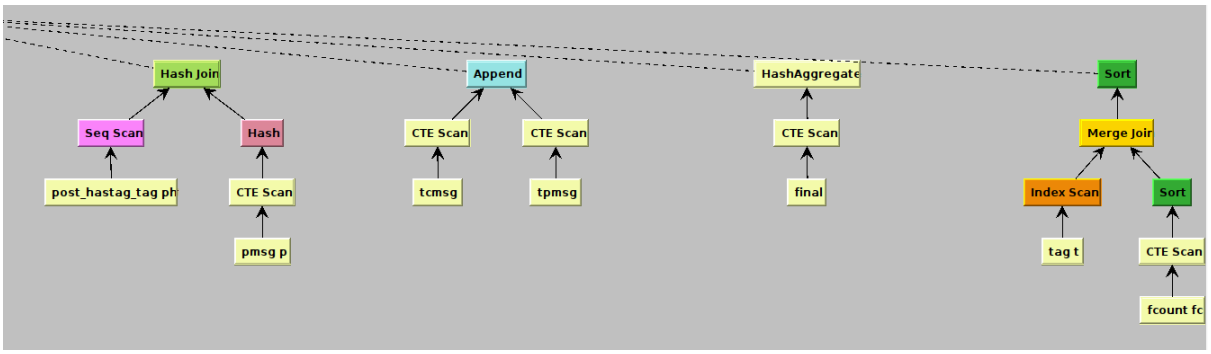


Fig: Total plan diagram





The plan diagram is quite balanced as the height is relatively same across the different Subtrees. The common steps used are hash joins, CTE scans and sequential scan of tables Along with index scan in one subtree which uses sorting so here merge join is a better option.

5. QUERY 5

5.1 Query template

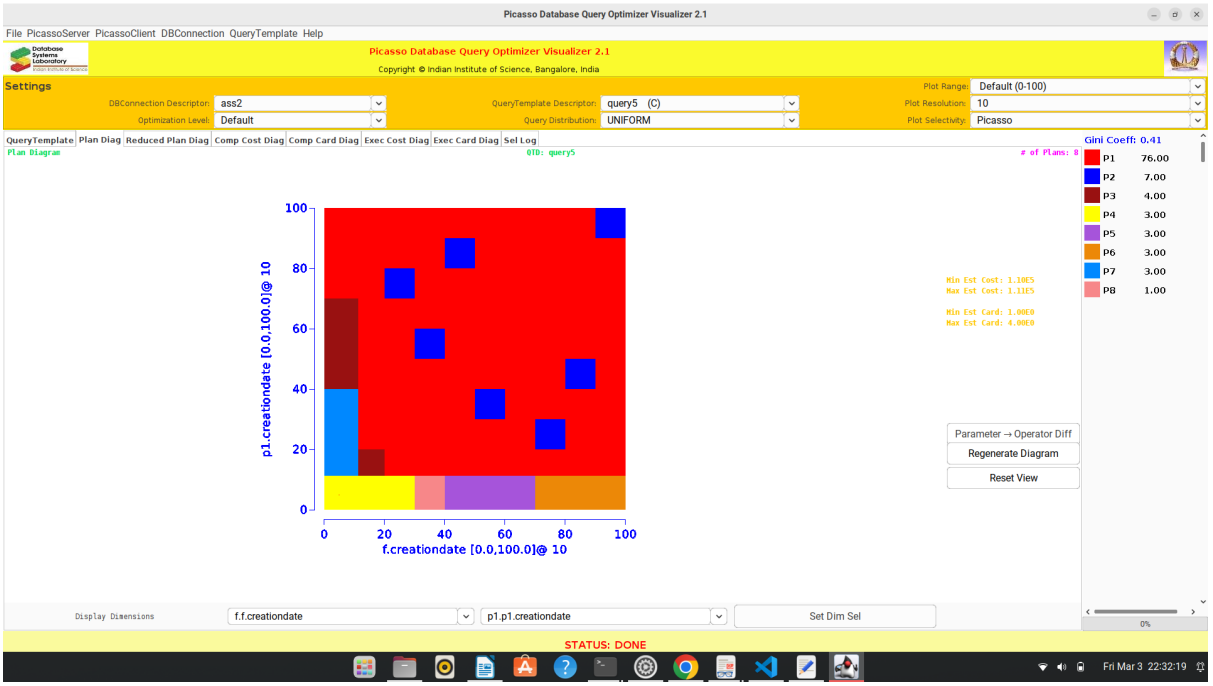
5.2 Execution Plan Diagram

This diagram represents the plan that is optimal at various percentages of the varying tables

For eg here 8 plans are generated for different values of range of length and creationdate

Which means that if we take 50% of person creationdate column and 50% of forum creationdate column then the

Plan P1 is the most optimal as shown by the red value. Similarly we can also deduce the optimal plans for other regions also.The presence of multiple plans implies that we can still Optimise the query for cases when either of the tables has a lower number of values.

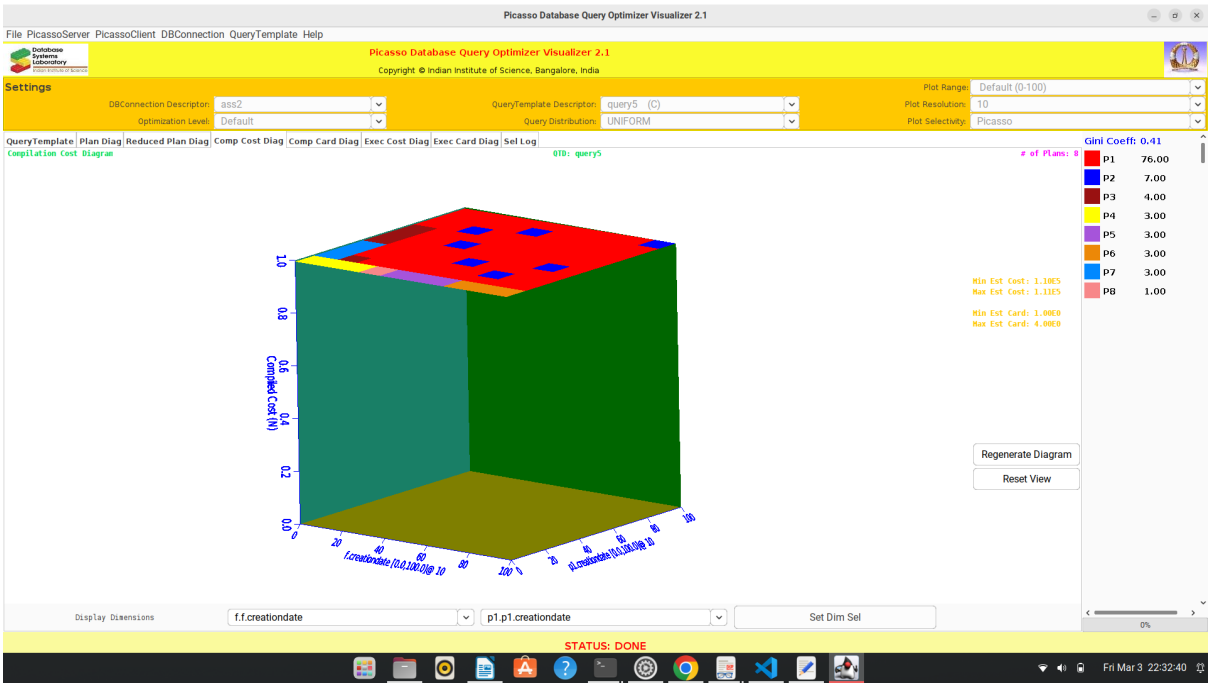


5.3 Compilation Cost diagram

This graph represents the computation time along varying range of post creationdate and comment creationdate

along the optimal plan.The value along the z axis in the diagram represents the computation time.The maximum computation time is when we use the whole range for creationdate Values of both the predicates

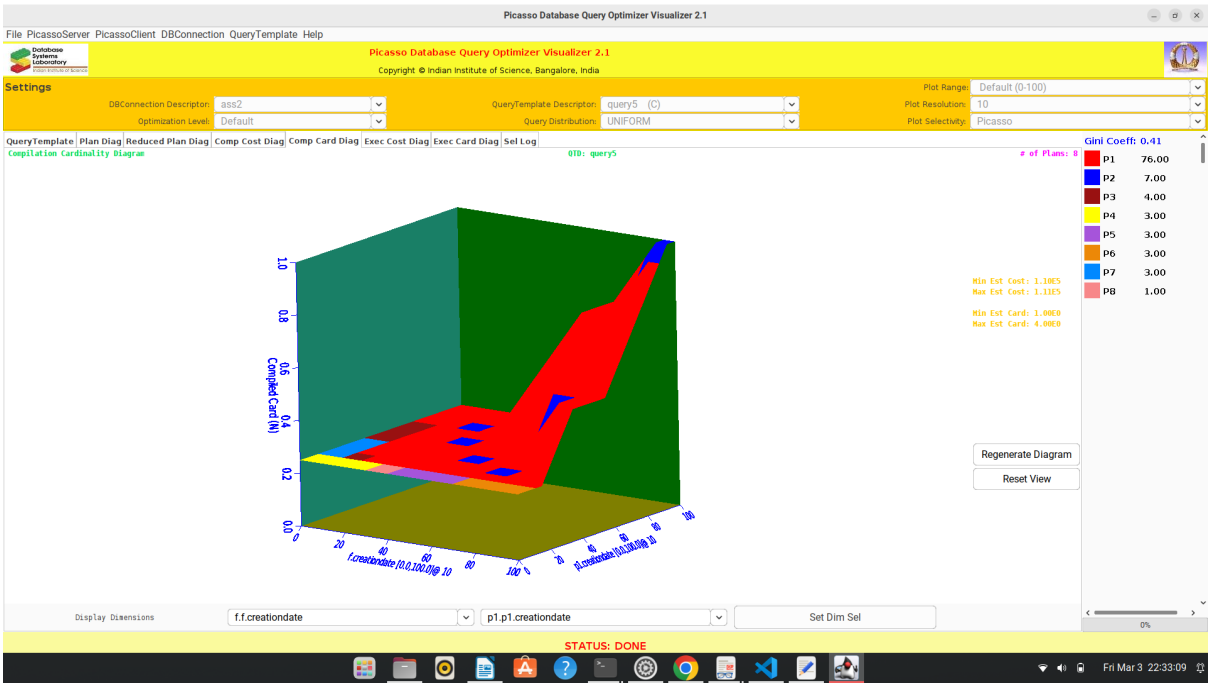
.The computation time mostly remains same with change in the predicates.



5.4 Compilation Cardinality Diagram

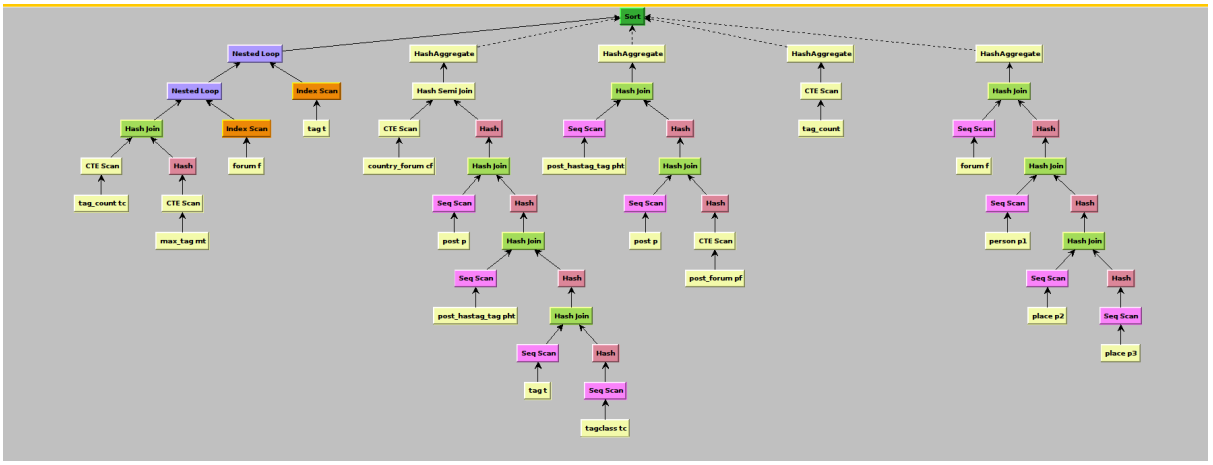
This represents the cardinality of the query values in the results . By cardinality we mean the Number of distinct values in the result represented in a range from 0 to 1.

From the above diagram we can see that the number of distinct values in the query result The number of distinct values in the result increases with increase in portions of the relation Predicate tables used.



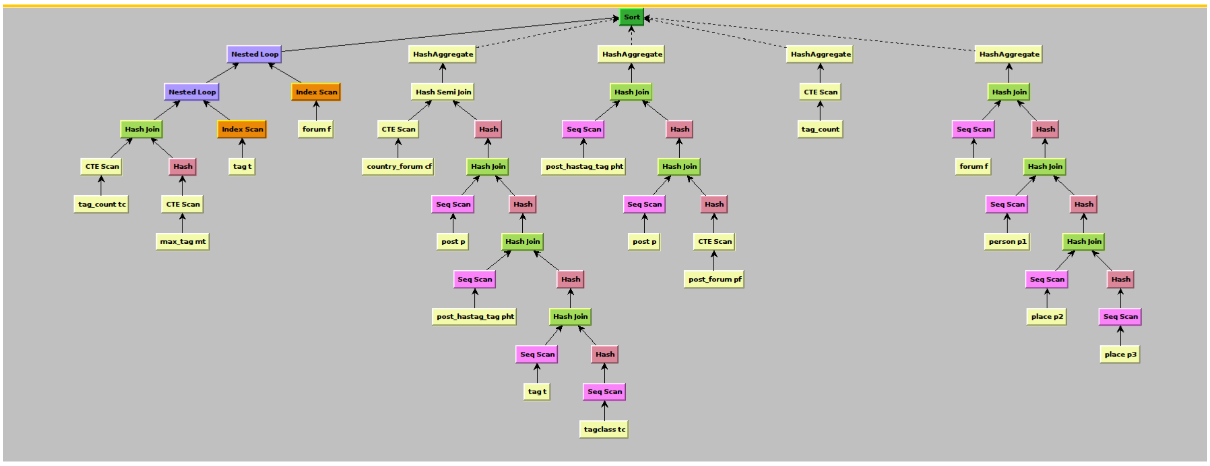
5.5 Plan Diagrams

Plan 1



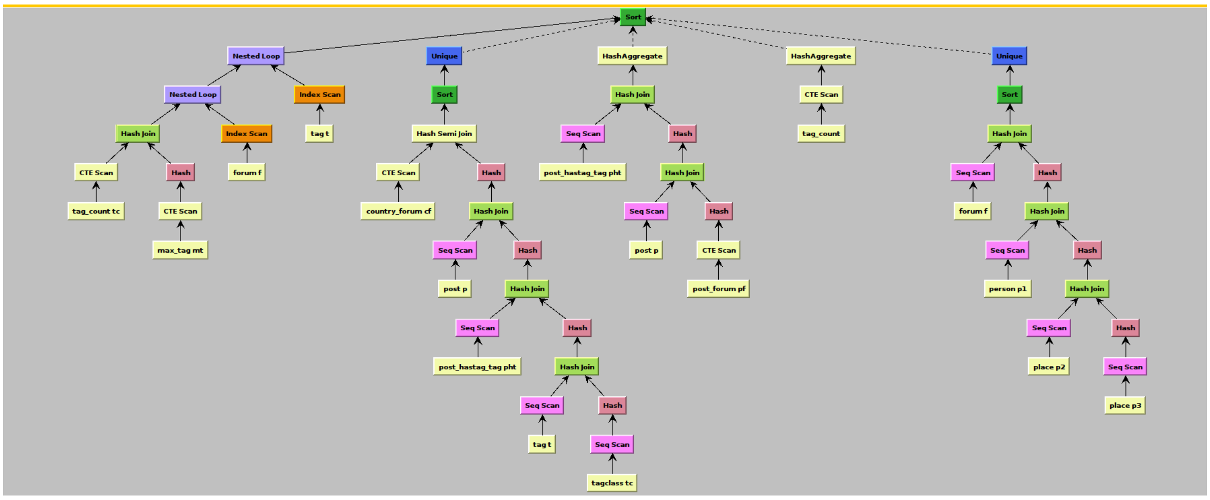
The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 2



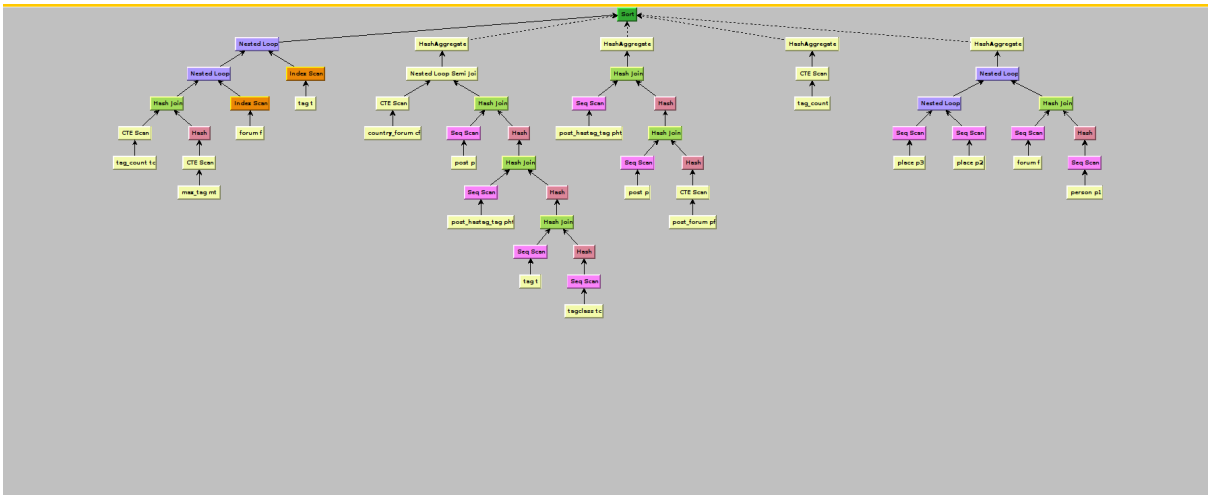
The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 3



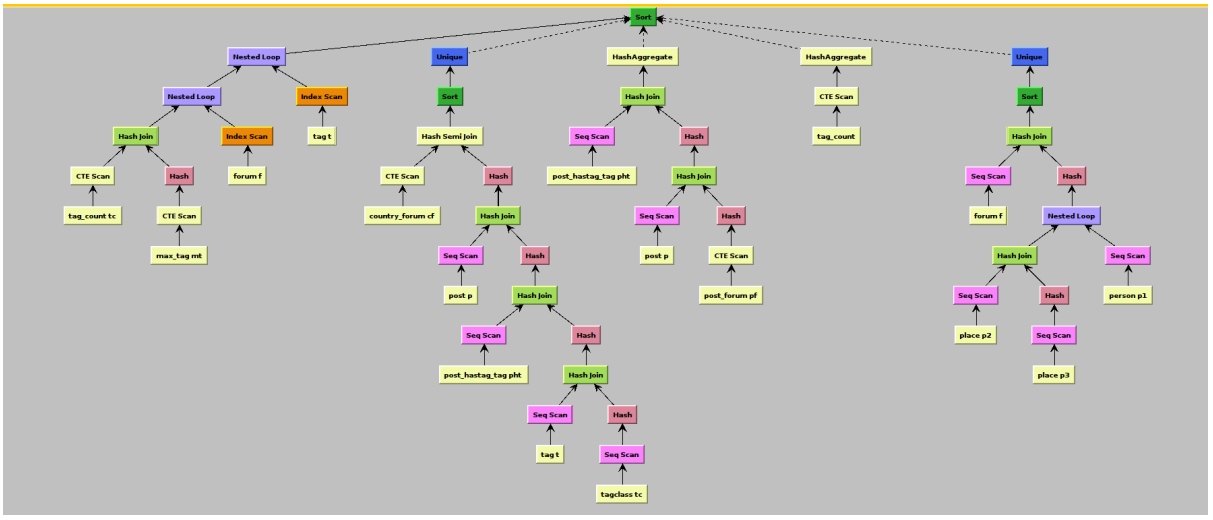
The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 4



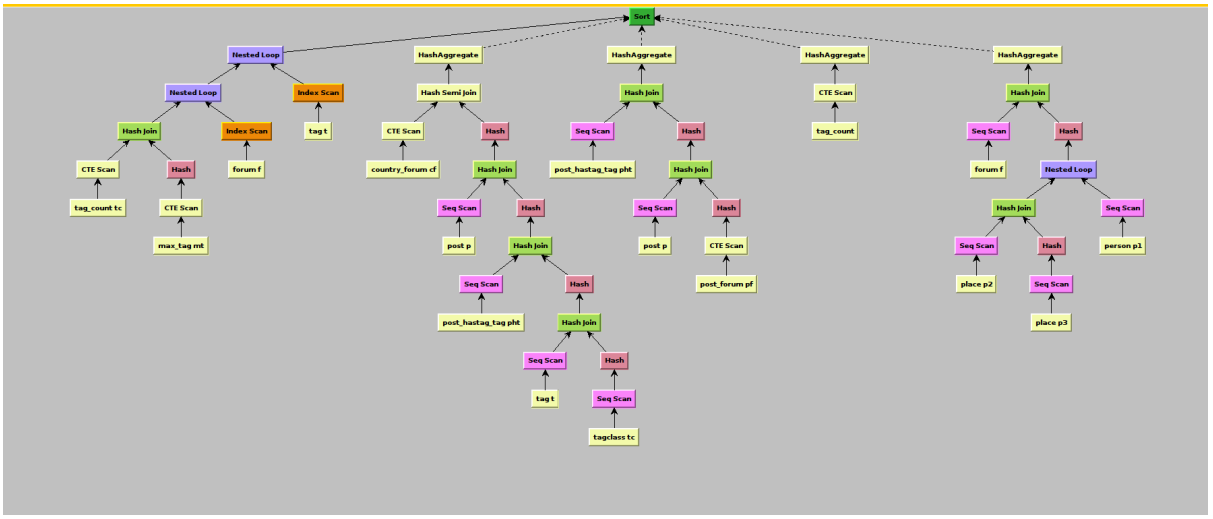
The plan diagram is quite balanced except along one subtree where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .The subtree which was previously unbalanced Is now somewhat balanced.

Plan 5



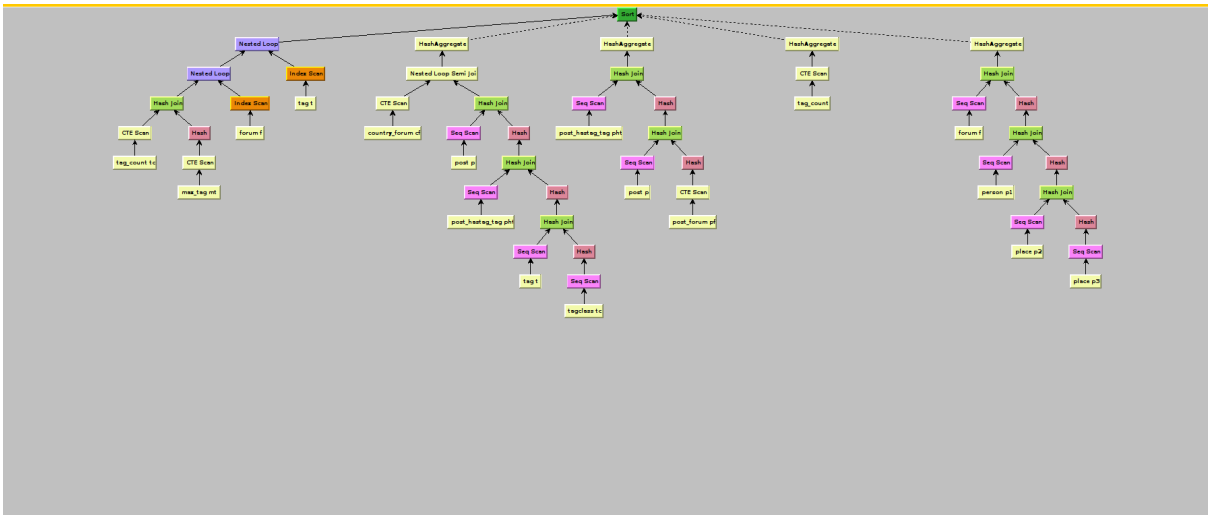
The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 6



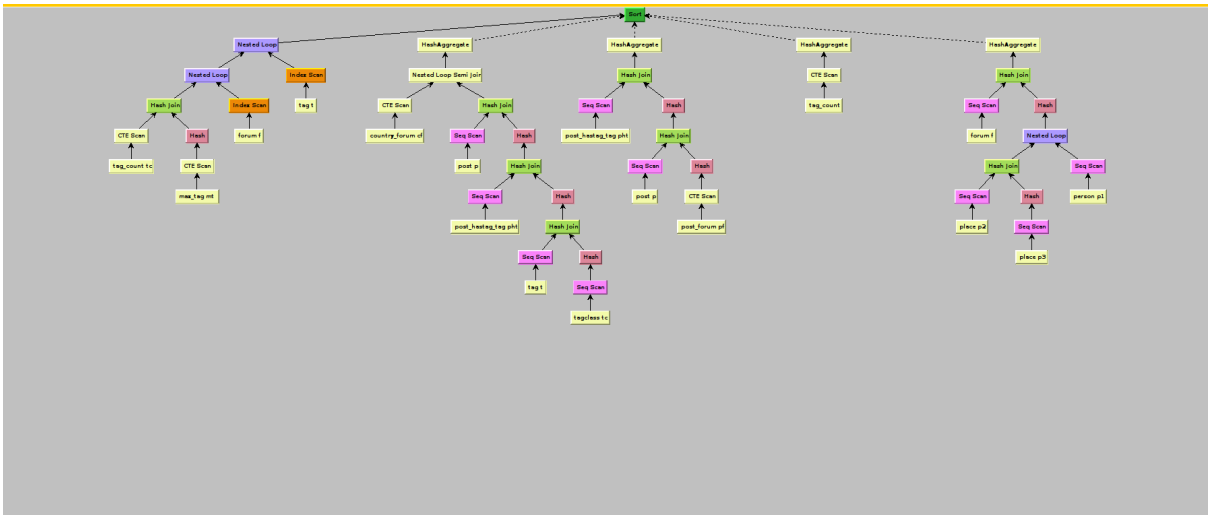
The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 7



The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

Plan 8



The plan diagram is quite balanced except along two subtrees where it becomes a little Skewed as the height is relatively same across the different subtrees except few . The common steps used are hash joins, CTE scans and sequential scan of tables along with index scan as well as nested loop .

QUERY OPTIMISING

Query 1

Since we have to output in the format person1sid<person2sid hence in the table Where we have find people having at least K tag in common we can limit the table From preventing it to grow by symmetric tuples by imposing a less than relation. It reduces the time to 50%. Indexing doesn't helped much.

Query 2

Here joining person_knows_person directly with person1 and person2 to find friends Having common birthmonth and same university instead of finding them separately Reduced time from 180 ms to 20 ms. Here I also tried indexing but it didn't help much.

Query 3

Here I have used indexing on 3 attributes from post_hastag_tag, comment_hasttag_tag and tag tables as these attributes are quite extensively used in comparisons and joins which take up a lot of time if not indexed . By using indexing we could reduce the time by 50%.

Query 4

Here I have used indexing on tables post,comment ,comment_hasttag_tag and post_hast tag_tag and tag tables as these tables have been widely used in joins and also the tables are of large size which may affect efficiency . I have also tried to improve efficiency by removing unnecessary distincts and select * as they take up large time.and union all instead Of union operator which takes large time because of removing duplicates.

Query 5

Here the query written was much optimal in beginning itself and using index on tables like Place person and forum tables gave slight improvement in time. Further improvements can be obtained by using using unnecessary distincts and slight modification inlogic.

.