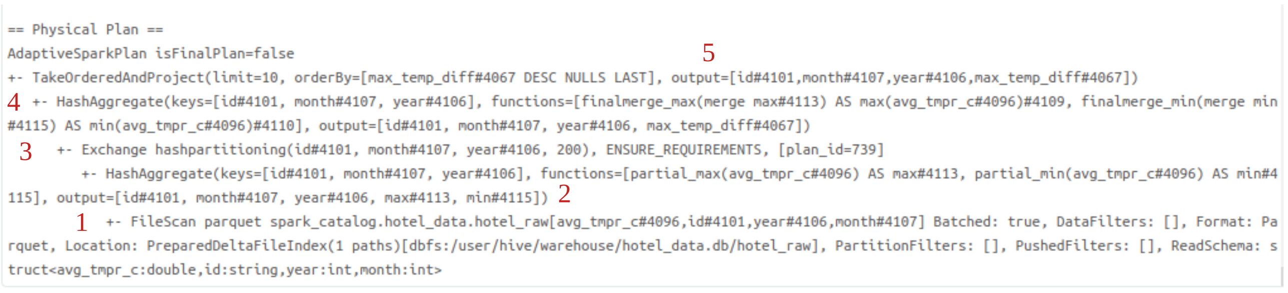
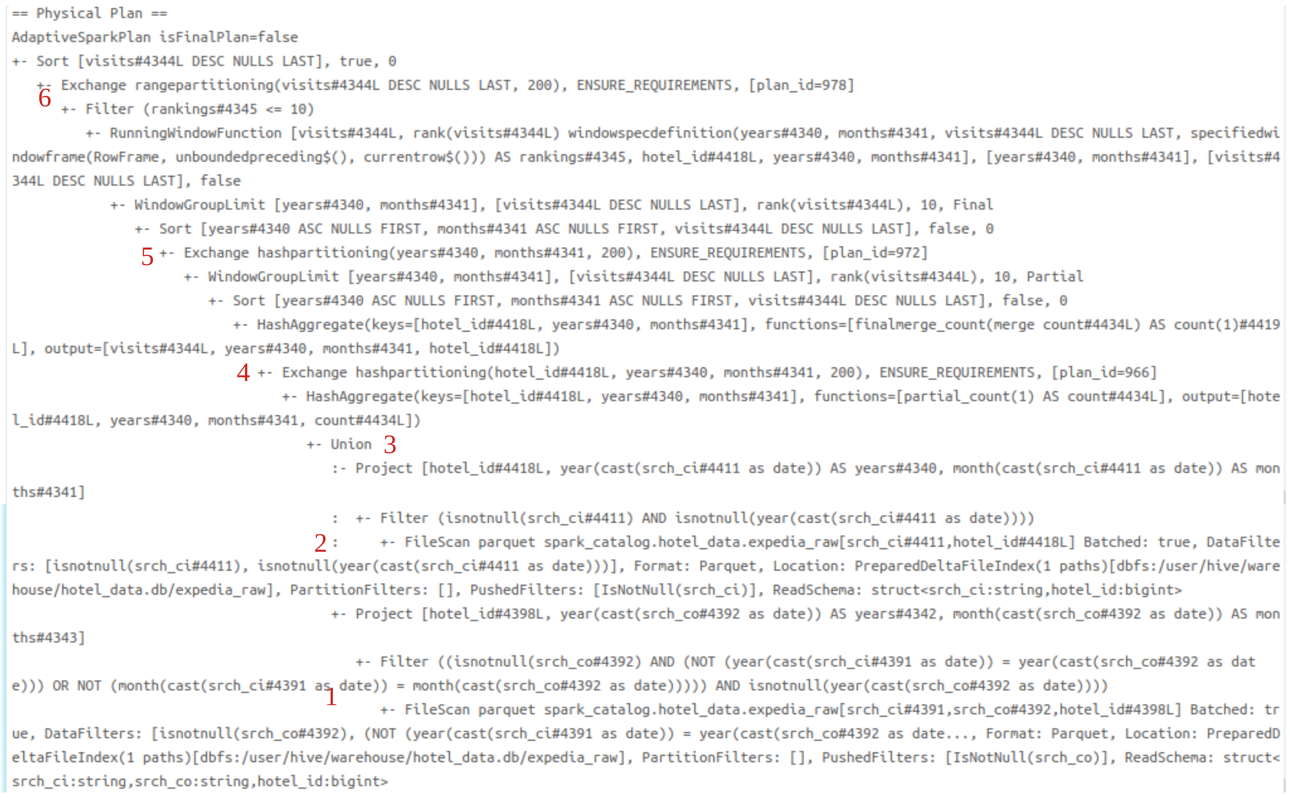
1. TASK



```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false                                                        5
+- TakeOrderedAndProject(limit=10, orderBy=[max_temp_diff#4067 DESC NULLS LAST], output=[id#4101,month#4107,year#4106,max_temp_diff#4067])
4 +- HashAggregate(keys=[id#4101, month#4107, year#4106], functions=[finalmerge_max(merge max#4113) AS max(avg_tmpr_c#4096)#4109, finalmerge_min(merge min
#4115) AS min(avg_tmpr_c#4096)#4110], output=[id#4101, month#4107, year#4106, max_temp_diff#4067])
 3   +- Exchange hashpartitioning(id#4101, month#4107, year#4106, 200), ENSURE_REQUIREMENTS, [plan_id=739]
        +- HashAggregate(keys=[id#4101, month#4107, year#4106], functions=[partial_max(avg_tmpr_c#4096) AS max#4113, partial_min(avg_tmpr_c#4096) AS min#4
115], output=[id#4101, month#4107, year#4106, max#4113, min#4115]) 2
       1  +- FileScan parquet spark_catalog.hotel_data.hotel_raw[avg_tmpr_c#4096,id#4101,year#4106,month#4107] Batched: true, DataFilters: [], Format: Pa
rquet, Location: PreparedDeltaFileIndex(1 paths)[dbfs:/user/hive/warehouse/hotel_data.db/hotel_raw], PartitionFilters: [], PushedFilters: [], ReadSchema: s
truct<avg_tmpr_c:double,id:string,year:int,month:int>
```

1. reading data
2. partially aggregate data to calculate max and min of avg_tmpr_c
3. shuffling data for grouping by id, month, year into 200 partitions
4. completed aggregation by calculating the difference between min and max
5. Order it and limit the results to the top 10 rows

2.TASK



```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- Sort [visits#4344L DESC NULLS LAST], true, 0
   +- Exchange rangepartitioning(visits#4344L DESC NULLS LAST, 200), ENSURE_REQUIREMENTS, [plan_id=978]
  6   +- Filter (rankings#4345 <= 10)
         +- RunningWindowFunction [visits#4344L, rank(visits#4344L) windowspecdefinition(years#4340, months#4341, visits#4344L DESC NULLS LAST, specifiedwi
ndowframe(RowFrame, unboundedpreceding$(), currentrow$())) AS rankings#4345, hotel_id#4418L, years#4340, months#4341], [years#4340, months#4341], [visits#4
344L DESC NULLS LAST], false
            +- WindowGroupLimit [years#4340, months#4341], [visits#4344L DESC NULLS LAST], rank(visits#4344L), 10, Final
               +- Sort [years#4340 ASC NULLS FIRST, months#4341 ASC NULLS FIRST, visits#4344L DESC NULLS LAST], false, 0
              5   +- Exchange hashpartitioning(years#4340, months#4341, 200), ENSURE_REQUIREMENTS, [plan_id=972]
                     +- WindowGroupLimit [years#4340, months#4341], [visits#4344L DESC NULLS LAST], rank(visits#4344L), 10, Partial
                        +- Sort [years#4340 ASC NULLS FIRST, months#4341 ASC NULLS FIRST, visits#4344L DESC NULLS LAST], false, 0
                           +- HashAggregate(keys=[hotel_id#4418L, years#4340, months#4341], functions=[finalmerge_count(merge count#4434L) AS count(1)#4419
L], output=[visits#4344L, years#4340, months#4341, hotel_id#4418L])
                         4    +- Exchange hashpartitioning(hotel_id#4418L, years#4340, months#4341, 200), ENSURE_REQUIREMENTS, [plan_id=966]
                                 +- HashAggregate(keys=[hotel_id#4418L, years#4340, months#4341], functions=[partial_count(1) AS count#4434L], output=[hote
l_id#4418L, years#4340, months#4341, count#4434L])
                                    +- Union 3
                                       :- Project [hotel_id#4418L, year(cast(srch_ci#4411 as date)) AS years#4340, month(cast(srch_ci#4411 as date)) AS mon
ths#4341]
                                       :  +- Filter (isnotnull(srch_ci#4411) AND isnotnull(year(cast(srch_ci#4411 as date))))
                                     2 :     +- FileScan parquet spark_catalog.hotel_data.expedia_raw[srch_ci#4411,hotel_id#4418L] Batched: true, DataFilte
rs: [isnotnull(srch_ci#4411), isnotnull(year(cast(srch_ci#4411 as date)))], Format: Parquet, Location: PreparedDeltaFileIndex(1 paths)[dbfs:/user/hive/ware
house/hotel_data.db/expedia_raw], PartitionFilters: [], PushedFilters: [IsNotNull(srch_ci)], ReadSchema: struct<srch_ci:string,hotel_id:bigint>
                                       +- Project [hotel_id#4398L, year(cast(srch_co#4392 as date)) AS years#4342, month(cast(srch_co#4392 as date)) AS mon
ths#4343]
                                          +- Filter ((isnotnull(srch_co#4392) AND (NOT (year(cast(srch_ci#4391 as date)) = year(cast(srch_co#4392 as dat
e))) OR NOT (month(cast(srch_ci#4391 as date)) = month(cast(srch_co#4392 as date))))) AND isnotnull(year(cast(srch_co#4392 as date))))
                                      1      +- FileScan parquet spark_catalog.hotel_data.expedia_raw[srch_ci#4391,srch_co#4392,hotel_id#4398L] Batched: tr
ue, DataFilters: [isnotnull(srch_co#4392), (NOT (year(cast(srch_ci#4391 as date)) = year(cast(srch_co#4392 as date..., Format: Parquet, Location: PreparedD
eltaFileIndex(1 paths)[dbfs:/user/hive/warehouse/hotel_data.db/expedia_raw], PartitionFilters: [], PushedFilters: [IsNotNull(srch_co)], ReadSchema: struct<
srch_ci:string,srch_co:string,hotel_id:bigint>
```

1. reading expedia data and filtering it where check-in-checkout months are different and not null
2. creating an other CTE by reading expedia data again and filtering for not null values
3. appending the two tables together with unified years and months column names
4. hash-aggregate to group values by id, years, months and count rows and shuffling data
5. to run the rank windows function, set group limit to top 10 rows based on descending order of visits
6. sort the data shuffle it again for finishing the window function.