TRIAL 1

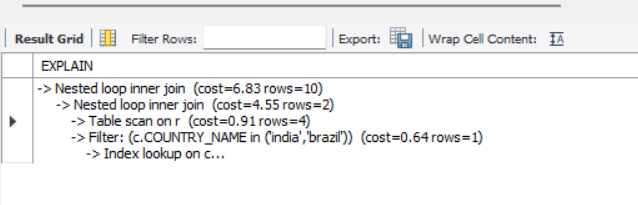
explain format =tree

select l.city, r.region\_name, c.country\_name, l.location\_id

from locations l inner join countries c on l.country\_id = c.country\_id

inner join regions r on r.region\_id = c.region\_id

where c.country\_name in ('india','brazil');



TRIAL 2 – REPLACING IN WITH = REDUCED COST

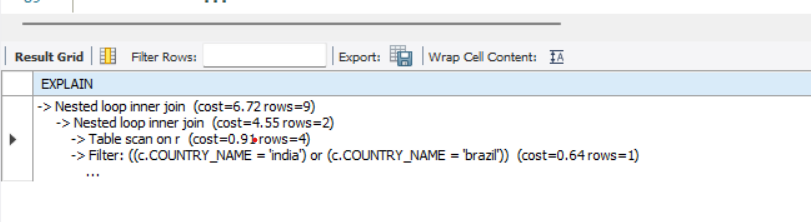
explain format=tree

select l.city, r.region\_name, c.country\_name, l.location\_id

from locations l inner join countries c on l.country\_id = c.country\_id

inner join regions r on r.region\_id = c.region\_id

where c.country\_name = 'india' or c.country\_name = 'brazil';



TRIAL 3 – ON FORCING REGION PRIMARY INDEX COST IS INCREASED HENCE OPTIMISER IGNORED IT

explain format=tree

select l.city, r.region\_name, c.country\_name, l.location\_id

from locations l inner join countries c on l.country\_id = c.country\_id

inner join regions r force index(primary) on r.region\_id = c.region\_id

where c.country\_name = 'india' or c.country\_name = 'brazil';

-- PARTITION TABLE based on salary

create table range\_part\_1sal (employee\_id int ,

first\_name varchar(20),

last\_name varchar(20),

salary int

)

partition by range(salary)

(

partition p\_5k values less than (5000),

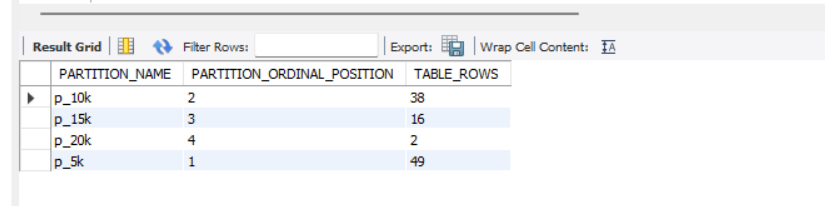
partition p\_10k values less than (10000),

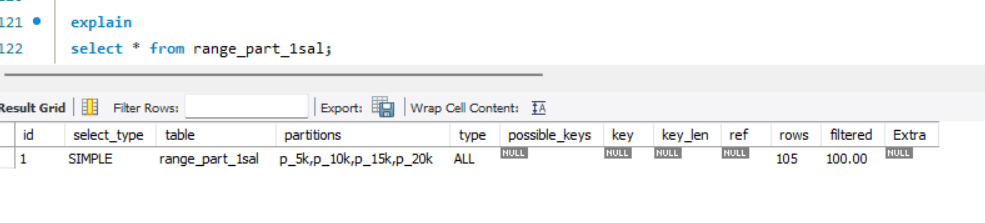
partition p\_15k values less than (15000),

partition p\_20k values less than (20000)

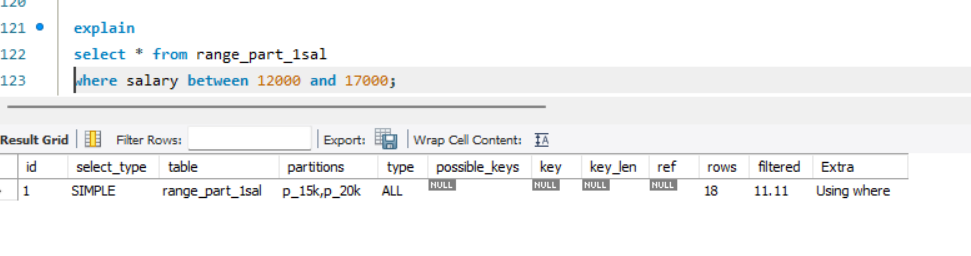
);

insert into range\_part\_1sal select employee\_id, first\_name, last\_name, salary from employees;



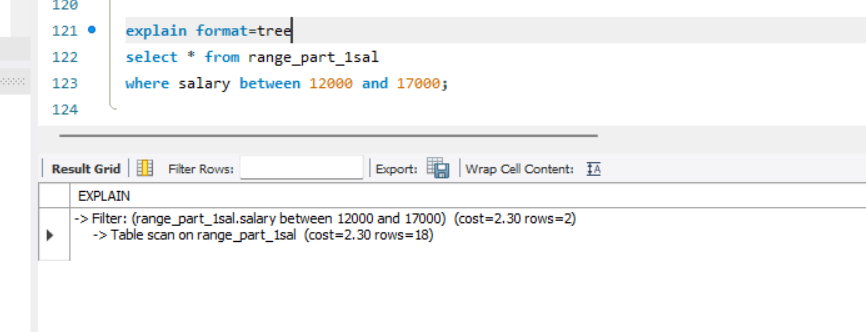


**d**



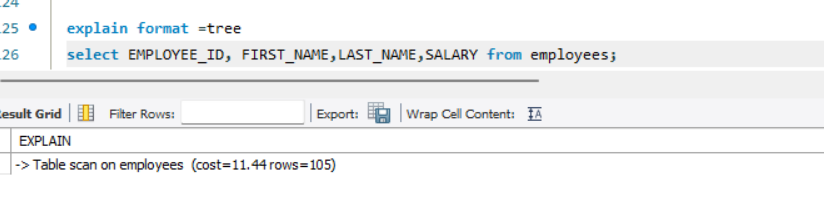
Partitions used are p\_15k and p\_20k as the values between 12000 and 17000 falls in that partitions

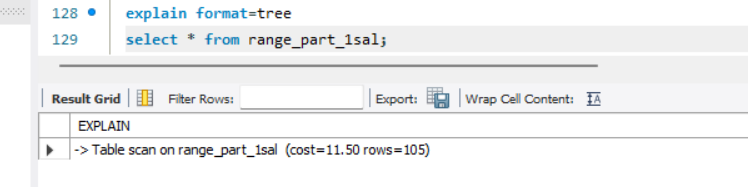
**e**



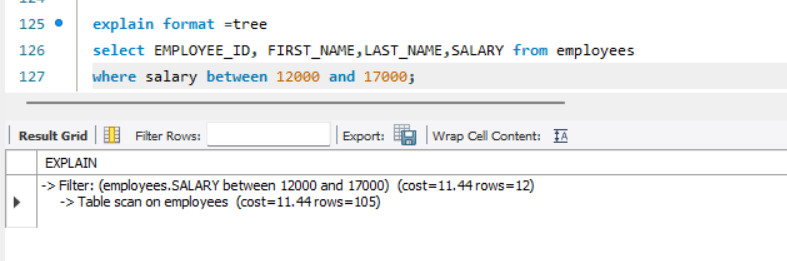
Here table scan is done on 18 rows only as there are 16 rows in p\_15k and 2 rows in p\_20k

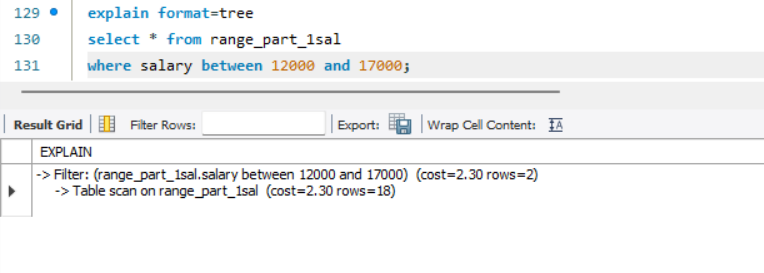
**F**





Here there is not much cost difference in above two. But we are fetching rows based the range of salary, cost will differ drastically.





Here we can see that in employees table cost is 11.44 where as in range\_part\_1sal(partition table) it is 2.30. so partition table reduced the cost.

**G**

**In question C, we created a partition table range\_part\_1sal which contains 4 partitions (p\_5k, p\_10k, p\_15k, p\_20k) based on the range of salary specified. So rows will be partitioned into these partitions based on partition the salary belongs to.**

**In question D, we have fetched the rows with salary between 12000 and 17000 . on explaining the query we can see that table scan is done only on 2 partitions which contains the specified range of salary which is p\_15k an dp\_20k. therefore the total number of rows scanned is also reduced to 18 and reduced cost as well. Without using partition table this will require a table scan on whole table which is highly expensive. So partition table improves the efficiency.**