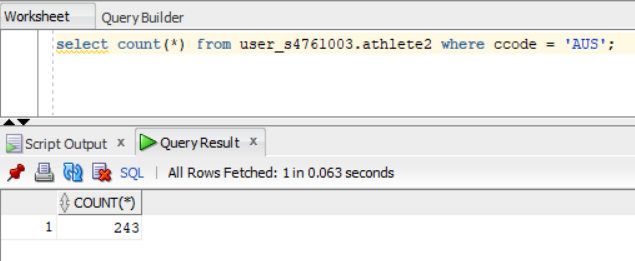
**TASK 1**

* ALTER SESSION SET "\_ORACLE\_SCRIPT"=TRUE;
* CREATE USER USER\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";
* GRANT DBA TO USER\_S4761003;

1. **Count the number of players from Australia (country code=AUS) in Athlete2 table.**

select count(\*) from user\_s4761003.athlete2 where ccode = 'AUS';

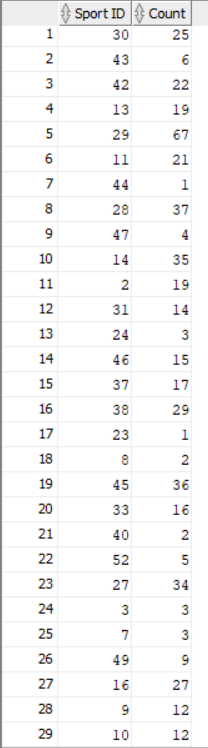
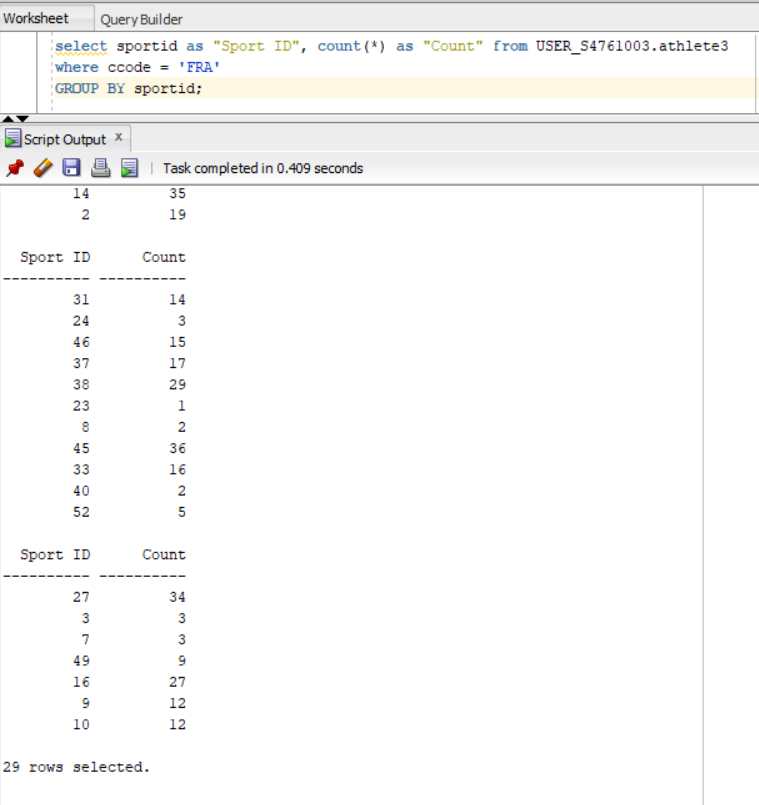


1. **For all French (FRA) players in table Athlete3, count the number of players participating in each sport.**

select sportid as "Sport ID", count(\*) as "Count" from USER\_S4761003.athlete3

where ccode = 'FRA'

GROUP BY sportid;



1. **Create a new table named ATHLETE\_FULL which combines all records from tables Athlete1, Athlete2, and Athlete3. Use this table along with the country information from the Country table to count the total number of players from Africa.**

create table user\_s4761003.ATHLETE\_FULL AS

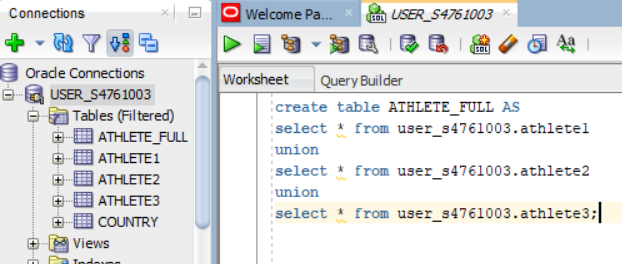
select \* from user\_s4761003.athlete1

union

select \* from user\_s4761003.athlete2

union

select \* from user\_s4761003.athlete3;



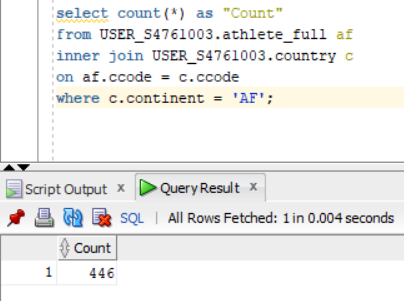
select count(\*) as "Count"

from USER\_S4761003.athlete\_full af

inner join USER\_S4761003.country c

on af.ccode = c.ccode

where c.continent = 'AF';



**TASK 2**

**Horizontal fragmentation**

**Job 1 - Full Replication**

CREATE USER USER1\_HF\_FULL\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

CREATE USER USER2\_HF\_FULL\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

CREATE USER USER3\_HF\_FULL\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

GRANT DBA TO USER1\_HF\_FULL\_S4761003;

GRANT DBA TO USER2\_HF\_FULL\_S4761003;

GRANT DBA TO USER3\_HF\_FULL\_S4761003;

**Full Replication Transaction**

BEGIN

update USER1\_HF\_FULL\_S4761003.athlete1\_replica1 set CCODE = 'AUS' where ATHLETEID = 128;

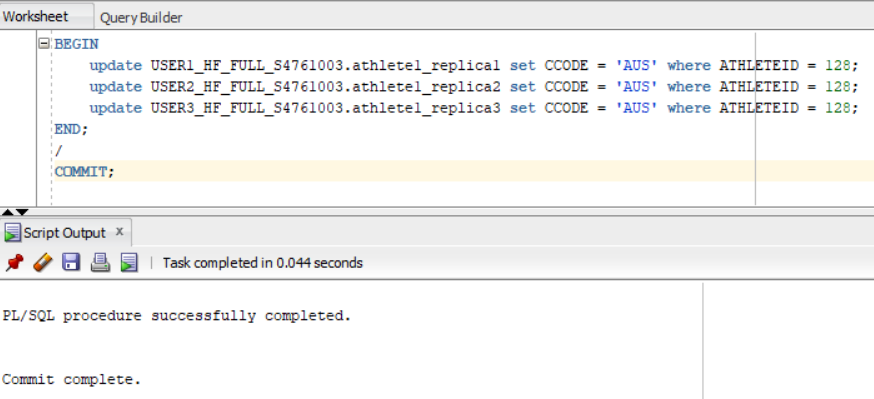
update USER2\_HF\_FULL\_S4761003.athlete1\_replica2 set CCODE = 'AUS' where ATHLETEID = 128;

update USER3\_HF\_FULL\_S4761003.athlete1\_replica3 set CCODE = 'AUS' where ATHLETEID = 128;

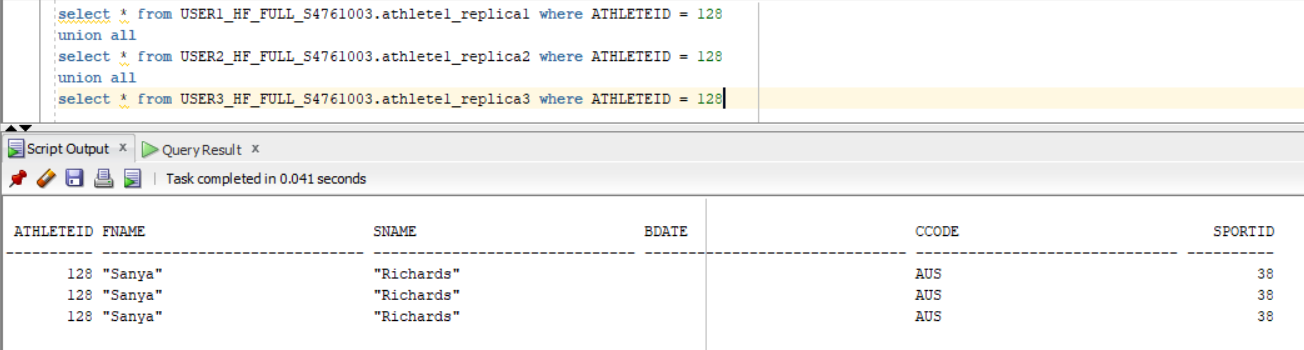
END;

/

COMMIT;



**OUTPUT -**



**Job 2 – Partial Replication**

CREATE USER USER1\_HF\_PA\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

CREATE USER USER2\_HF\_PA\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

CREATE USER USER3\_HF\_PA\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

GRANT DBA TO USER1\_HF\_PA\_S4761003;

GRANT DBA TO USER2\_HF\_PA\_S4761003;

GRANT DBA TO USER3\_HF\_PA\_S4761003;

**Partial Replication Transaction**

BEGIN

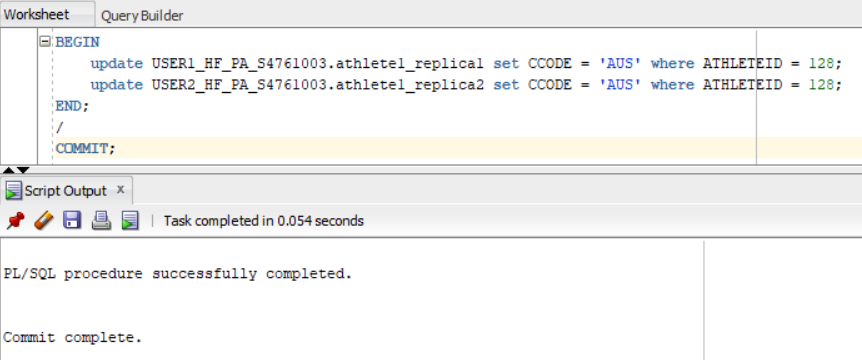
update USER1\_HF\_PA\_S4761003.athlete1\_replica1 set CCODE = 'AUS' where ATHLETEID = 128;

update USER2\_HF\_PA\_S4761003.athlete1\_replica2 set CCODE = 'AUS' where ATHLETEID = 128;

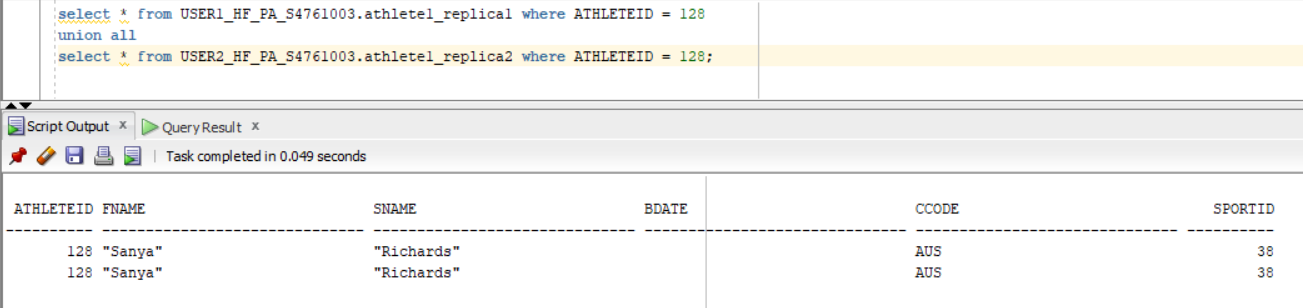
END;

/

COMMIT;



**OUTPUT -**



**Job 3 – No Replication**

CREATE USER USER1\_HF\_NO\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

CREATE USER USER2\_HF\_NO\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

CREATE USER USER3\_HF\_NO\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

GRANT DBA TO USER1\_HF\_NO\_S4761003;

GRANT DBA TO USER2\_HF\_NO\_S4761003;

GRANT DBA TO USER3\_HF\_NO\_S4761003;

**No Replication Transaction**

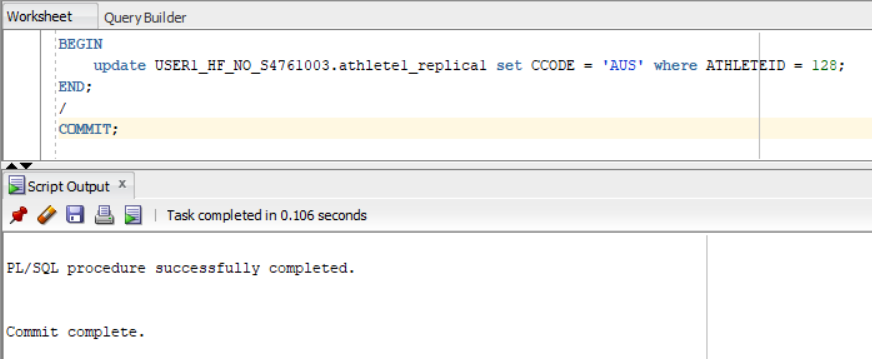
BEGIN

update USER1\_HF\_NO\_S4761003.athlete1\_replica1 set CCODE = 'AUS' where ATHLETEID = 128;

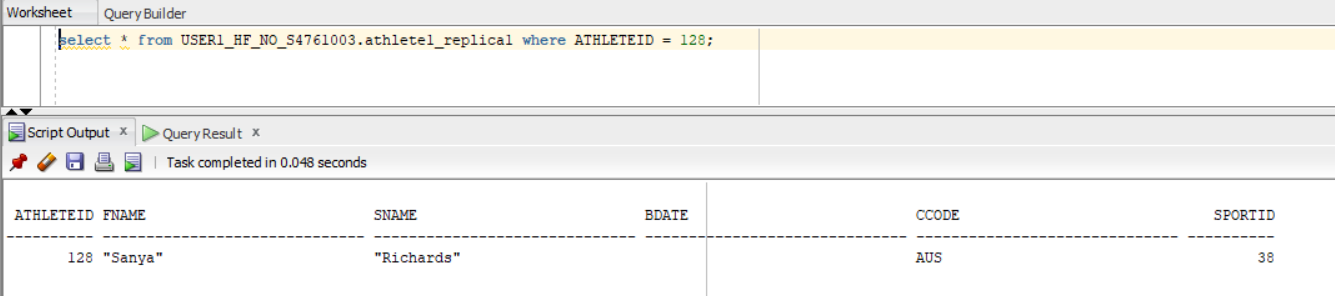
END;

/

COMMIT;



**OUTPUT -**



In FULL Replication, the database tables are present in every user on the system. This will allow for high availability of data for the users but the updates will be slow to trickle down the system so as to maintain consistency between different sites and will be expensive as the tables at all the different nodes will be needed to be updated.

In NO Replication, the tables or data is stored at only one site/node. If multiple users start accessing this node to make updates, it will take more time to execute the updates as multiple users are accessing the same node for the data.

In Partial Replication, it is a mix of both Full and No Replication. Here the data is available on more than one node/sites of the network. So, depending on the type of data it may be replicated to one or more nodes. As, the number of replication sites will be less than full replication, it will be faster and less expensive to update the data at multiple sites. In comparison to NO replication, as the data is available at more than one site, it will give users an ease of access and take less time to update the data in its local site whose change will get trickled down the system.

**TASK 3**

CREATE USER USER1\_VF\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

CREATE USER USER2\_VF\_S4761003 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

GRANT DBA TO USER1\_VF\_S4761003;

GRANT DBA TO USER2\_VF\_S4761003;

1. **Write a SQL query to retrieve the full name and date of birth (DOB) of all the athletes satisfying the following criteria: 445 <= AthleteID <= 450**

SELECT replace(v1.fname, '"', '') || ' ' || replace(v1.sname, '"', '') as "full name",

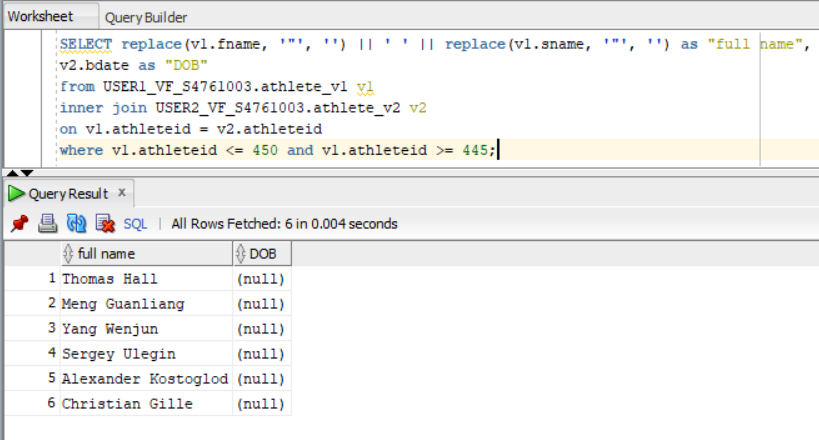
v2.bdate as "DOB"

from USER1\_VF\_S4761003.athlete\_v1 v1

inner join USER2\_VF\_S4761003.athlete\_v2 v2

on v1.athleteid = v2.athleteid

where v1.athleteid <= 450 and v1.athleteid >= 445;



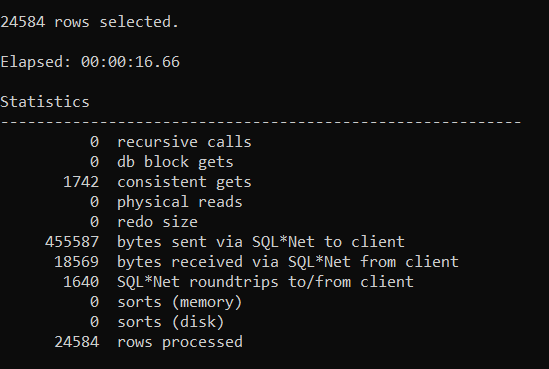
**TASK 4**

SET AUTOTRACE ON STATISTICS;

SET TIMING ON;

**SEMI JOIN -**

select v1.athleteid from user1\_vf\_s4761003.athlete\_v1 v1;



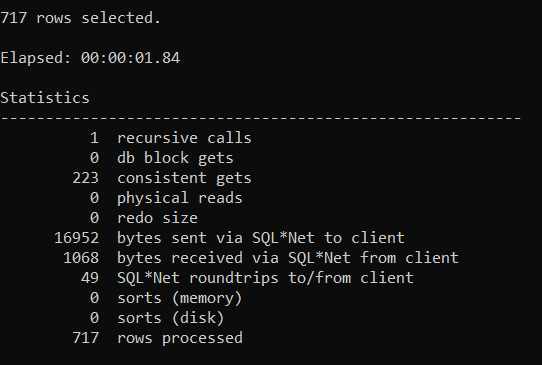
select v2.athleteid, v2.bdate, v2.ccode, v2.sportid

from USER2\_VF\_S4761003.athlete\_v2 v2

inner join

(select v1.athleteid from user1\_vf\_s4761003.athlete\_v1 v1) v1

on v1.athleteid = v2.athleteid where v2.ccode = 'AUS';



select v1.athleteid, v1.fname, v1.sname, v2.bdate, v2.ccode, v2.sportid

from user1\_vf\_s4761003.athlete\_v1 v1

inner join

(

select v2.athleteid, v2.bdate, v2.ccode, v2.sportid

from USER2\_VF\_S4761003.athlete\_v2 v2

inner join

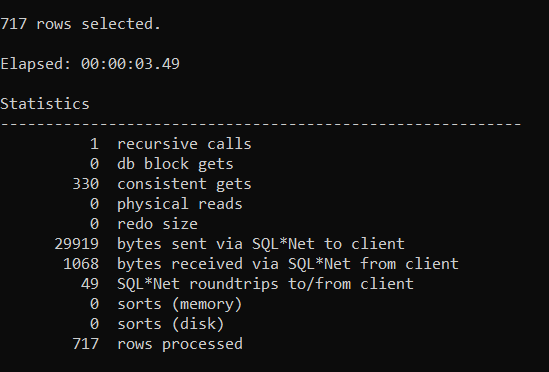
(select v1.athleteid from user1\_vf\_s4761003.athlete\_v1 v1) v1

on v1.athleteid = v2.athleteid

where v2.ccode = 'AUS'

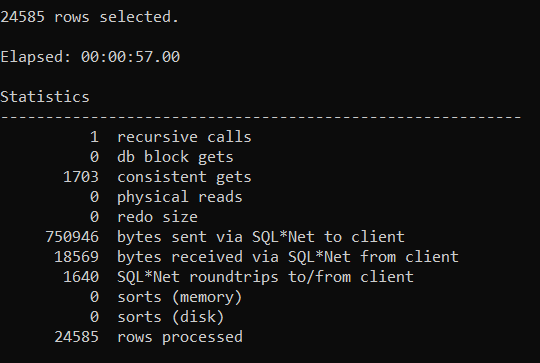
) v2

on v1.athleteid = v2.athleteid;



**INNER JOIN**

select \* from USER2\_VF\_S4761003.athlete\_v2;

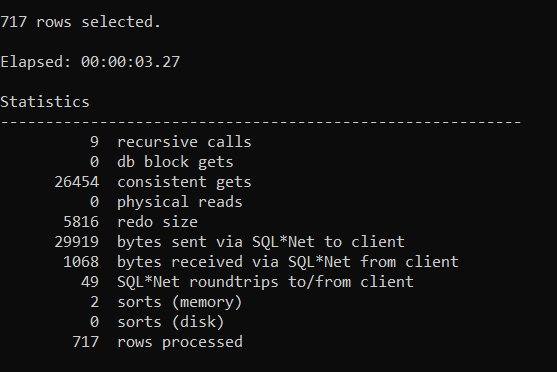


select v1.athleteid, v1.fname, v1.sname, v2.bdate, v2.ccode, v2.sportid from user1\_vf\_s4761003.athlete\_v1 v1

inner join (select \* from USER2\_VF\_S4761003.athlete\_v2) v2

on v1.athleteid = v2.athleteid

where v2.ccode = 'AUS';



The total transmission cost for the inner-join plan is **750946 bytes** (query 1)

Finally, the number of bytes sent is 29919 bytes for the inner join query.

The total transmission cost for the semi-join plan is 455587 bytes (query 1) + 16952 bytes (query 2) = **472539 bytes**

Finally, the number of bytes sent is 29919 bytes for the semi join query.

Here we can see that the final query result is of the same size.

Semi join will be ideal for our case as we are doing a join only once to get the data and the number of bytes transmitted is less.