INFS3200 Advanced Database System

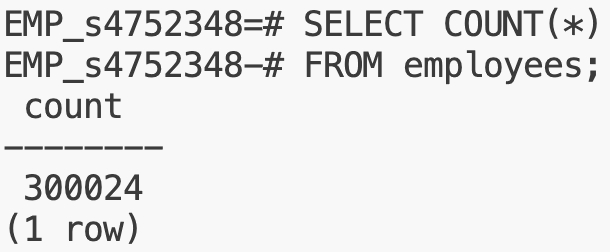
Assignment 1

Student Name: Wei-Ting, Hong

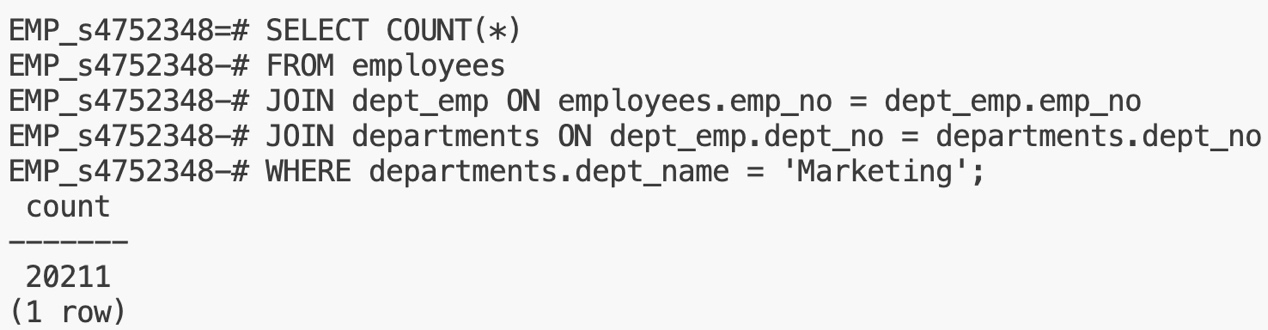
Student Number: 47523483

Task 1.

(1).



(2).



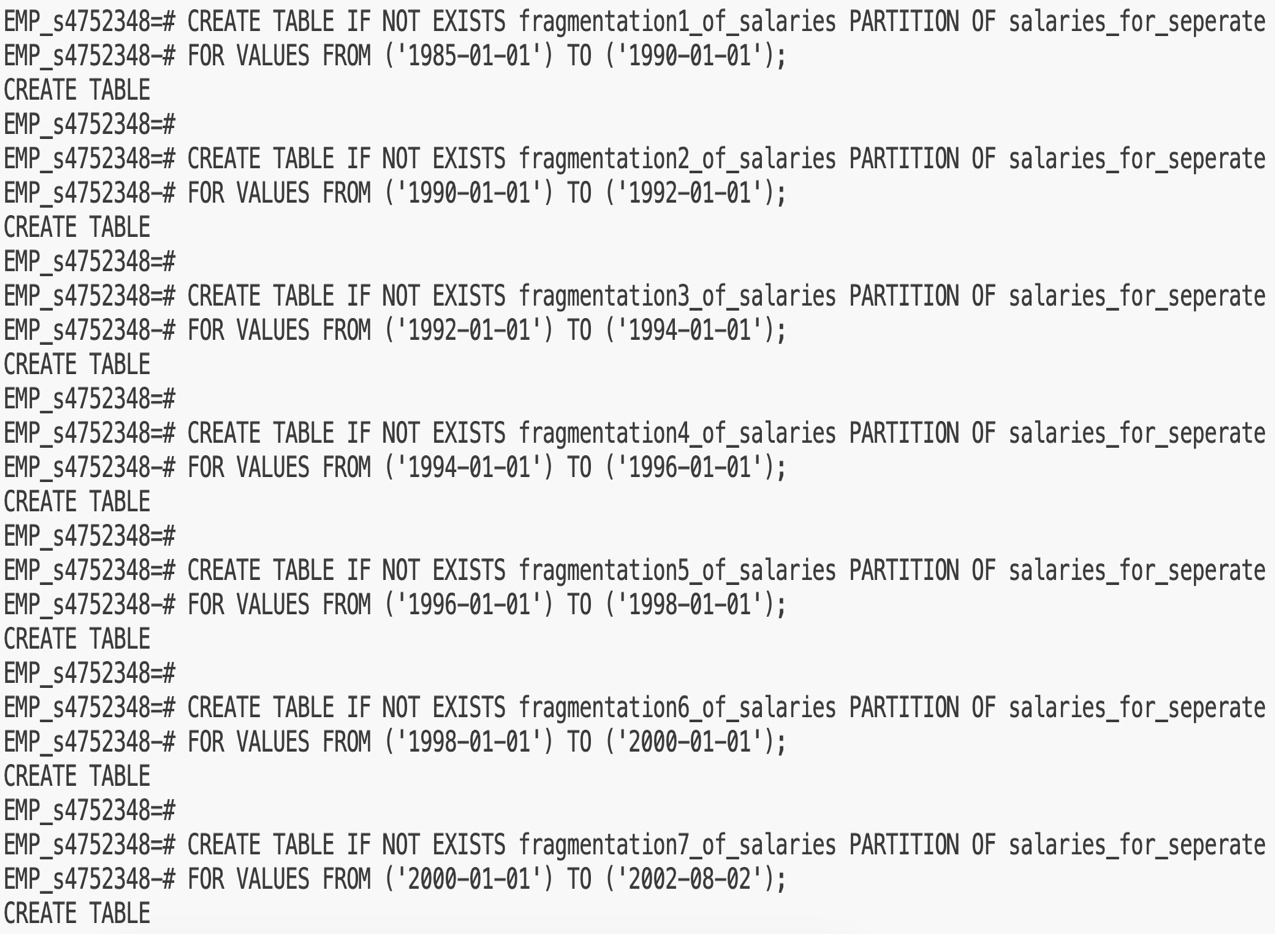
Task 2.

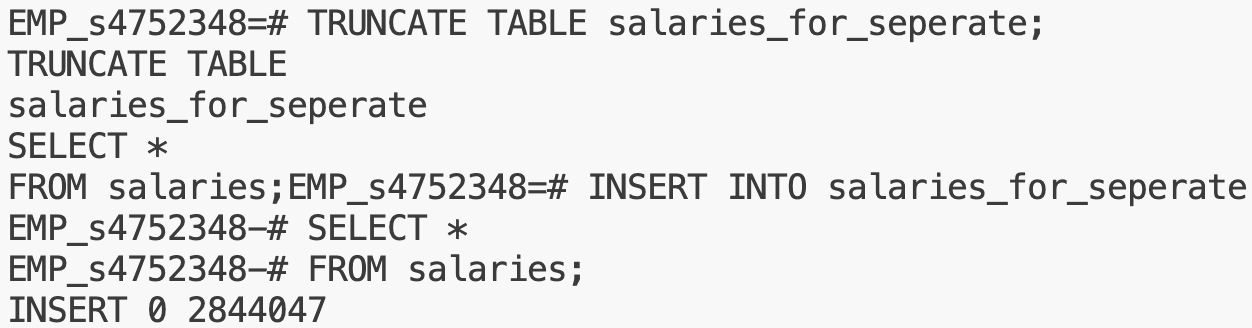
Before answering the questions, we first need to create a new table with clear definition on how to separate the fragmentations, in this case, we use “from\_date” as the separation standard. Therefore, he last sentence of the SQL should be `PARTITION BY RANGE (from\_date);`, then we could start the separation.

一張含有 文字, 收據, 白色, 字型 的圖片

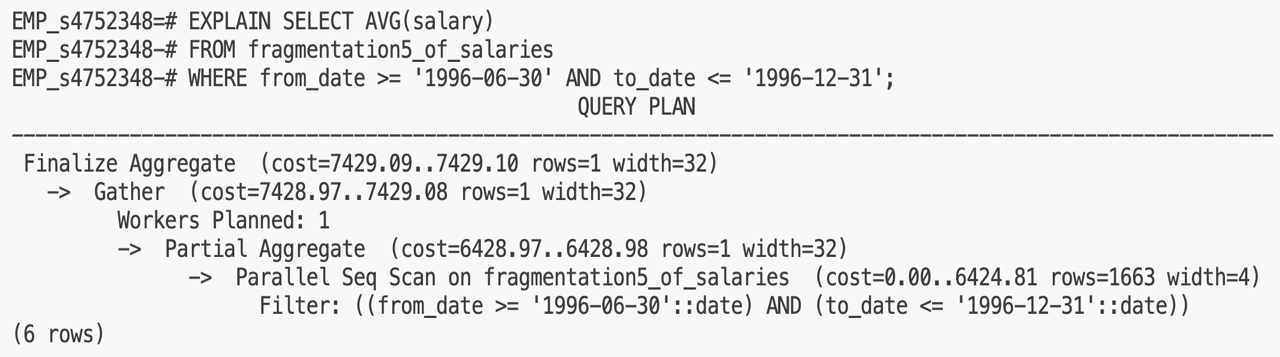
自動產生的描述

1. For fragmentation of 'from\_date' before 1990-01-01, first we could use SELECT clause to obtain the earliest “from\_date” in table “employees”, which was “1985-01-01”. In terms of fragmentation of 'from\_date' no earlier than 2000-01-01, the latest “from\_date” in table “employees” was “2002-08-02”. After confirming the earliest and latest “from\_date”, we could separate all required fragmentations as follow:



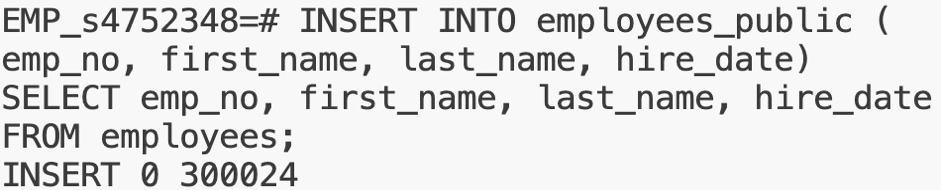
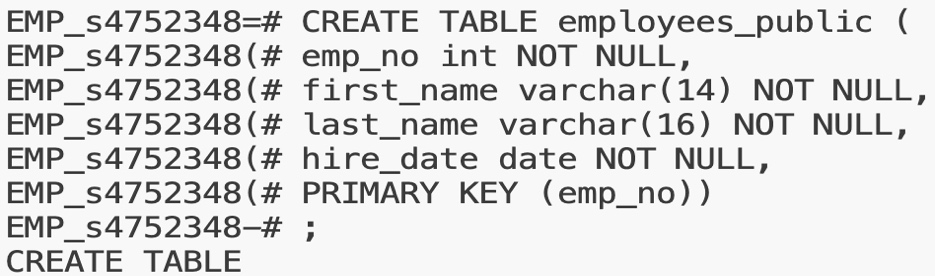


1. The query screenshot and explanation are shown as follow:一張含有 文字, 收據, 字型, 白色 的圖片

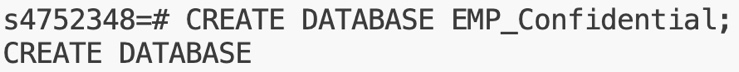
   自動產生的描述

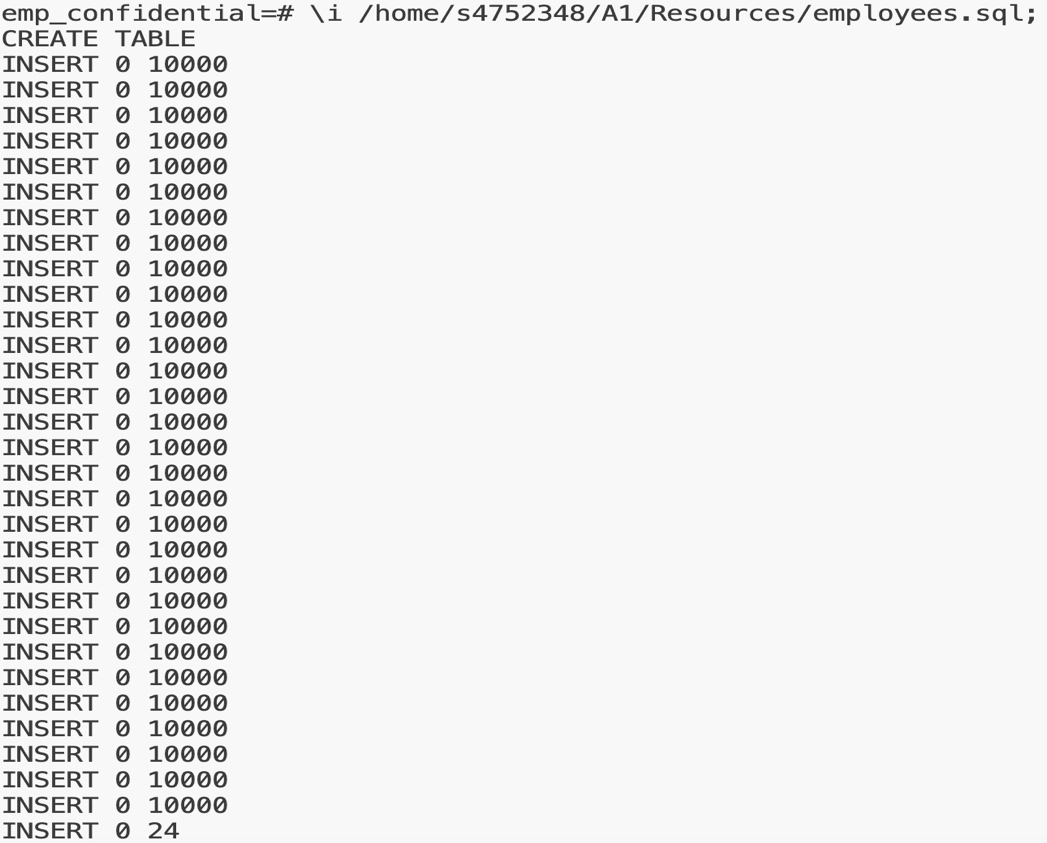
First, the database system scans all records in the target file and leaves only records that match the WHERE clause. Second, the database system will perform partial aggregate. It will divide the records which need to be calculated by the aggregation function into several small pieces, and then perform gather operation after calculating them separately. By collecting the answers from the partial calculations, finally, the result is calculated and return.

(3) The first two screen is to create first vertical table “employees\_public” in original database.

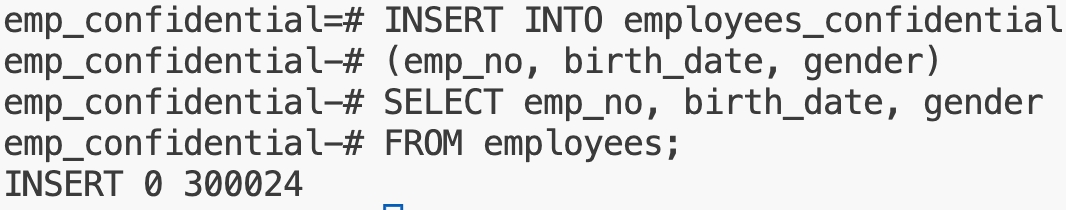
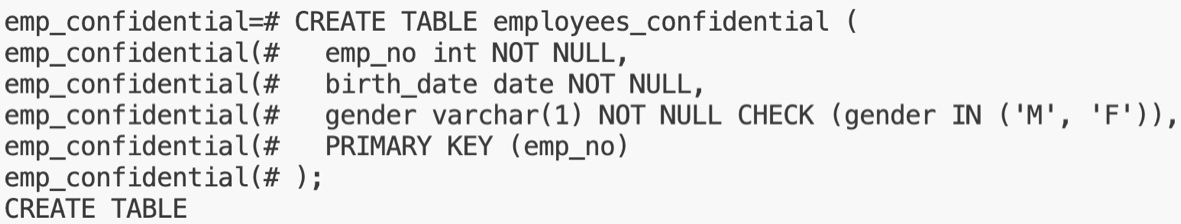


For the second vertical fragmentation, it should be stored in new DB “EMP\_Confidential”, so the first step is to create the new DB.



Then we import the master table “employees” into the new DB.

Next step is to define the vertical fragmentation and import the qualified data into it.



Task (3).

• Full replication method stores every fragmentation in every site, i.e., each site would have ten fragmentations. The advantages of this method would be the high accessibility, every site has whole data, we could access every data we need in any site. Even though some of the sites is not working, this issue would not influence the operation at all. The disadvantage is that the storage cost, we store entire table data in every site, we may need to prepare bigger site capacity. In addition, update processes may be significantly expensive as well. The reason is every site should always store identical data, so if we update one of the fragmentations in single site, the other sites should be updated as well, it would require considerable cost.

Partial replication method makes every site store some of the fragmentations but not all. And the overlapping store may occur. The advantage of this method is it could save the storage cost, each site only needs to store the distributed data pieces. This condition could also increase loading efficiency, we just only need to take involved fragmentation into account while querying. The disadvantage would be the complicated query processes and combination of query results. Scattered data would require extra actions to combine for obtaining the final result.

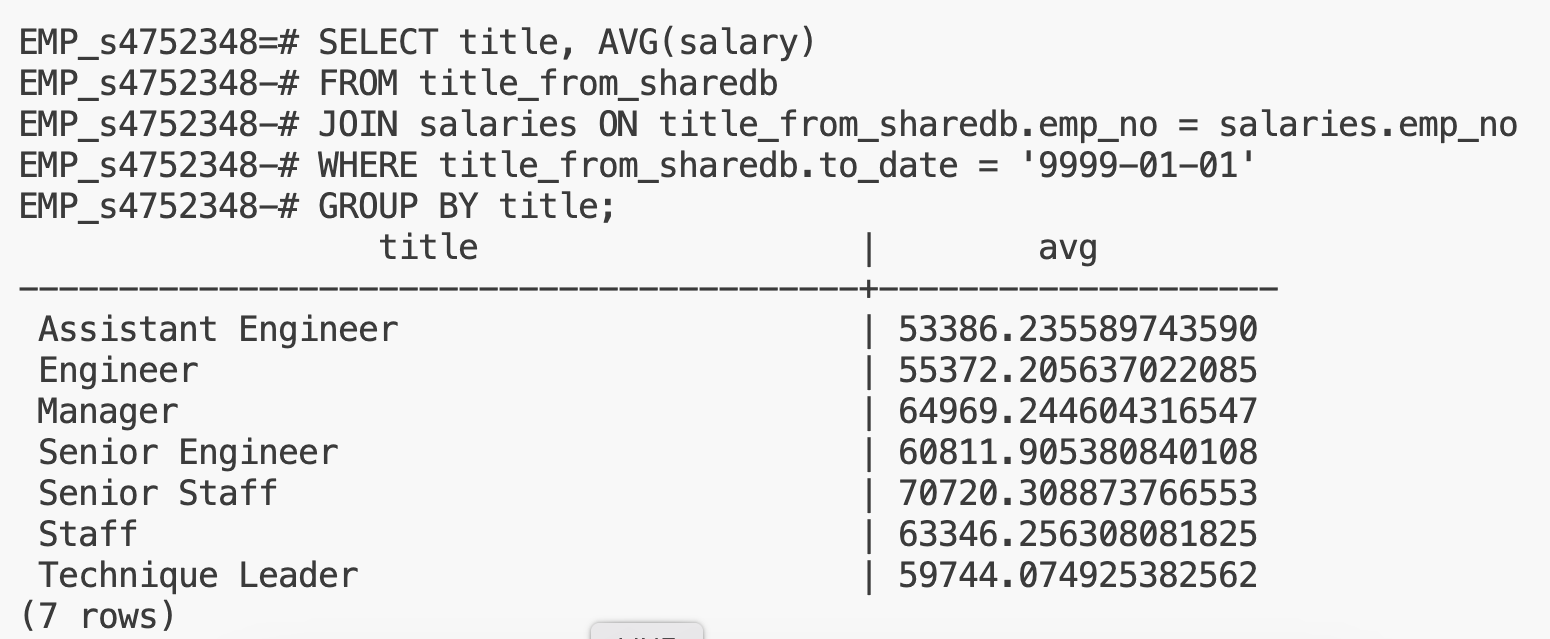
No replication method also makes every site to store some fragmentations, however, the overlapping would not occur. The pros and cons of this method basically are more extreme than partial replication method. Moreover, if one of the site is not working, than the query would not be approved to execute because no overlapping is allowed in this method.

• As industry convention, I will choose partial replication to manage the distributed database. For updating a specific record, first I will check which fragmentations contain the record is going to update by using primary key. Next, I will start to execute queries on these selected sites, to ensure the consistency, I will use 2PC or even 3PC methods. After the queries are successfully executed, the final result should return.

Task (4).

(1)(2).一張含有 文字, 螢幕擷取畫面, 字型 的圖片

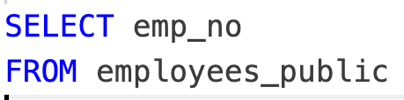
自動產生的描述



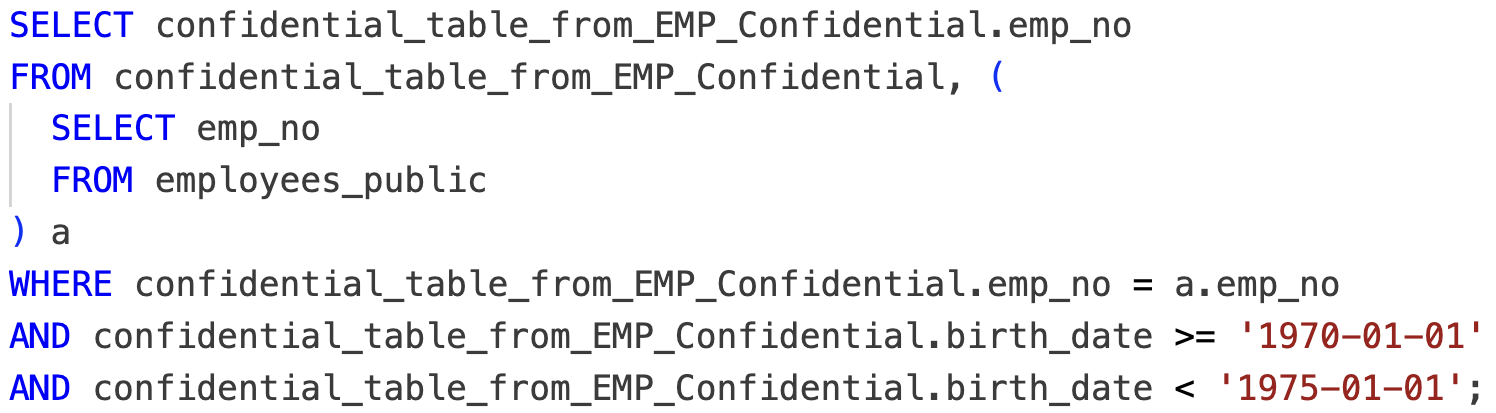
1. Basically, we do the same thing as (1) for building a new foreign table, with several updates (name of source table, store table, server name, change host to localhost because both of source table and store table are in local database warehouse) of SQL to make the correct FDW. 一張含有 文字, 螢幕擷取畫面, 字型 的圖片

   自動產生的描述

For implement the semi join, first we need to SELECT join key for preparation from local table. Like screen below.



Next, we perform a query to SELECT the foreign/primary key from foreign table. The table inside FROM clause is foreign table, and another one is all the primary keys from local one. This operation is like JOIN logic in semi join, however, it didn’t join all of the attributes to local table. We only need it to return the keys that satisfied the WHEHE clause.

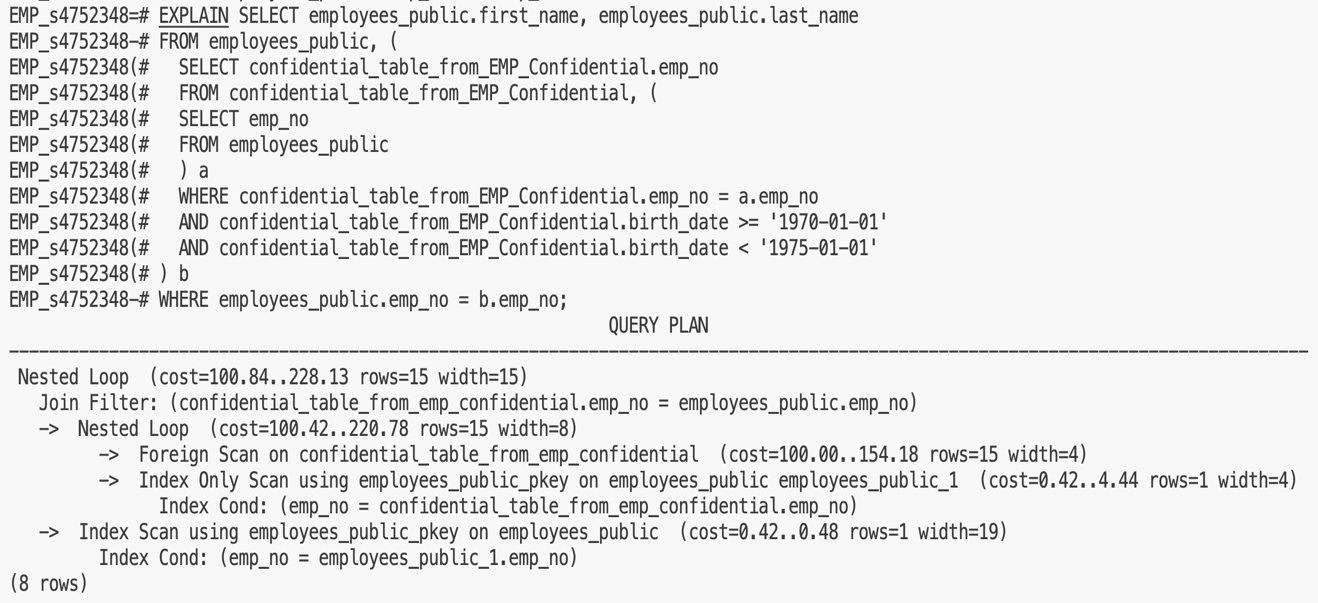


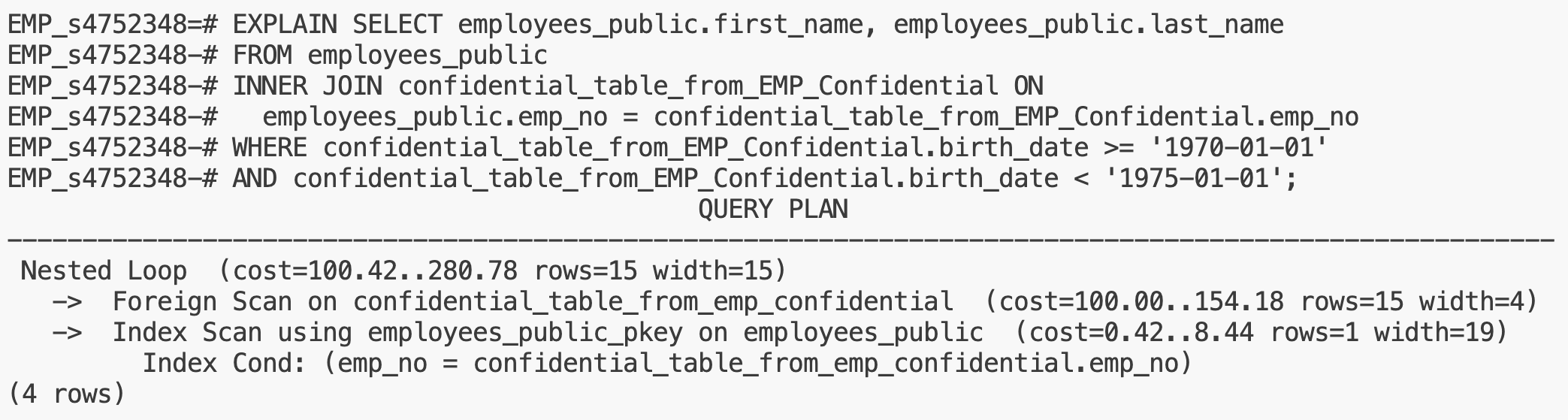
Now, we obtain the emp\_no that birth date is no earlier than '1970-01-01'and before '1975-01-01'. The final step is that SELECT the final attributes needed from local table and those records’ keys which match the WHERE clause (which is check for birth date) in step two. WHERE clause is another logic like JOIN in semi join.

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

1. The following two screenshots are explanations of using semi and inner join:

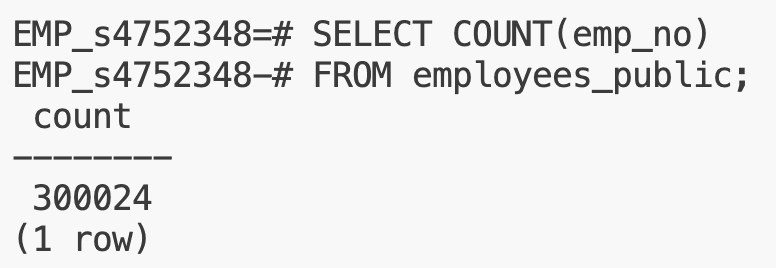




Transmission cost:

* Semi Join:

First, we could calculate the transmission cost that transfer all primary keys from local table to foreign table by following:



We could use COUNT without unique because emp\_no is primary key (no duplication).

The remaining part of transmission cost is from transferring records from foreign to local site, however, in this case, there is no tuple to be return because no record is satisfied the birth date in WHERE clause. Therefore, the transmission cost is 300,024 in semi join.

一張含有 文字, 收據, 字型, 白色 的圖片

自動產生的描述

* Inner join:

We know that there are also 300,024 tuples that should be transfer from foreign table to local table, however, in inner join it would transfer all of the attributes in foreign table. So, the transmission cost would be 300024 \* 3 (which are emp\_no, birth\_date, and gender, all of them are constrained by NOT NULL) = 900,072.

To sum up, in this case, the cost of semi join is cheaper than inner join, the main reason is that the semi join only send necessary record back to local site after receiving and checking the records from local site. On the other hand, the inner join would directly send all of the records from foreign to local site. In terms of the cost calculated by psql. It also indicates that the cost of semi join is averagely cheaper than inner join.