Assignment 1 Submission Guideline Supplementary

- Please read assignment guidelines in the assignment and this supplementary before posting questions on ED.
- Submission: submission for assignment 1 should be one single pdf file, including all the queries, screenshots of execution results, and explanations. The submission expectation for each question will be detailed in the following section. There is no template for the assignment submission, but the answers should be in order and organized.
- Queries and outputs: Queries executions on either PGAdmin or Terminal are accepted. Your queries must be included, and you cannot only provide result, in which case 0 mark will be applied. If query results are very long, you just need to provide the first and last several rows of the outputs (including the last few rows because we need to see the number of rows returned).
- Explanations: you also need to provide some explanations for certain questions, which is discussed below.
- If you are still not sure what should be submitted, provide all the work you have done and your assumptions for the questions. More explanations won't reduce your marks.

Expectation for Each Question

Database setup

You can provide some screenshots to list the database and tables, after creating all necessary tables and database.

Task 1

Only queries and screenshots for each question are required. You don't need to provide explanations for sql queries.

Task 2

2.1: queries and screenshots for creating horizontal fragmentation are required.

2.2: queries and screenshots are required. You also need to show the screenshot of the query plan and give descriptions about how psql database system executes and optimizes the queries.

2.3: Show the queries and results for creating the vertical fragmentation tables, as well as the steps for creating the new database. Give a brief description. The hint in the task sheet is a suggestion that you can dump the table to a sql file, but many ways can also be used to create the 'employees_condifential' table, such as using "\copy" to export data or directly creating a new table in the new database. We do not focus on how the vertical fragmentation is created in the new database, as long as there finally is 'employees_confidential' table in the database called 'EMP_Confidential'.

Task 2.3 clarification: "calculate salary between 1996-06-30 and 1996-12-31" is ambiguous so you can just use "from_date" in the query (we will give marks if you already use "to_date").

Task 3

Given that the task does not involve specific operations, queries are not required. Instead, you can offer clear and thorough explanations. If you feel clarification may be challenging, you can give some simple examples to support the explanations.

Task 4

4.1: please provide the queries/commands that establish the FDW.

4.2: you should query from the foreign table, and provide the queries and screenshots of the outputs.

4.3: you should create another foreign table mapping to the necessary table and access the vertical fragmentation table through FDW. Queries, screenshots and brief descriptions are required.

4.4: you should respectively show the steps for semi-join and inner-join with queries and **transmission cost** (not the join cost), which will be covered in the lecture and tutorial. Queries, screenshots of the query plans, and explanations about how the joins are performed and why one join strategy has more transmission cost than the other should be included in the submission.

Task 4.3 and 4.4 clarification

Databases to be used: In these questions, we assume the database that contains "employees_public" is the local site, and the database "EMP_Confidential" that is created in task 2 and contains "employees_confidential" is the remote site. In this case, you don't have to use "sharedb", so you don't need to modify **sharedb** or access it. If you want to access **employees_confidential** table from the **employee_pulic** database, how can you access it? (Hint: use FDW)

Semi-join analysis: we are simulating distributed databases within a centrialize database system using psql, and we don't actually need to execute semi-join. Instead,

we aim to simulate its behavior. Below is a simple example outlining the steps involved in semi-join emulation, and let's consider the tutorial question:

"Assume R is at site 1 and S is at site 2, and a query R ⋈ S is issued at site 2. List the steps for a query processing strategy using semi-join, and check if the semi-join is a beneficial option in this case (ignore local processing cost).

Relation R has schema R(A, B), and relation S has schema S(B,C,D), where attribute B is the foreign key."

In this case the local site is the site where the query is issued, which is site 2, and the remote site is site 1.

Step 1: The first step is to transmit foreign keys/primary keys from local site to remote site to prepare semi join:

```
SELECT B FROM S;
```

Step 2: Once the foreign keys are transmitted to the remote site 1, semi-join is performed at remote site:

```
SELECT R.A, R.B
FROM R, (SELECT B FROM S) FS
WHERE R.B = FS.B;
```

Step 3: The semi-join result from the previous step will be sent back to the local site, and perform final join:

```
SELECT FR.A, FR.B, SL.C, SL.D -- Should be the final attributes returned to
user
FROM S SL, (SELECT R.A, R.B FROM R, (SELECT B FROM S) FS WHERE R.B = FS.B) FR
WHERE SL.B = FR.B;
```

You should perform similar steps in task 4.3, but note that you may add some filtering conditions to optimize the query and reduce transmission cost at remote and local sites.

The procedure of inner-join in task 4.4 is directly sending all necessary attributes from remote site to the local site. However, you should be aware of which step(s) have transmission cost for semi-join and inner-join?