

Prac 1: Distributed Databases (5%)

Semester 2, 2022

Due: Week 5 – Friday August 26th at 4pm

Submission: via Blackboard → [Assessment](#) > [Practicals](#) > [Practical One](#) > [Submission](#)

Introduction

Learning Objectives

- Learn how to use Oracle DBMS through SQL Plus and SQL Developer. Oracle will be used in both Pracs 1 & 2.
- Get familiar with the basic SQL queries and keywords. Write SQL queries for data retrieval.
- Simulate horizontal and vertical fragmentation using a centralized Oracle DB.
- Understand how to update records on a distributed database with data replication.
- Apply the semi-join algorithm to simulate data transmission cost reduction over computer networks. Understand when and why a semi-join is more efficient than alternatives.

Marking Scheme

- **2 marks:** Receive one mark for completing two of the Task 1 questions. Receive two marks for completing all three Task 1 questions.
- **1 mark:** Answer the question of Task 2 correctly. Half marks will be awarded for unnecessary updates or inappropriate/insufficient explanations.
- **1 mark:** Complete Task 3.
- **1 mark:** Complete Task 4.

Submission Format

Screenshot your results for each task and compile them into a document with appropriate explanation. Keep your explanations terse but make sure you contain all necessary information. Make sure your screenshots contain **your student ID** (since your student ID will be included in the name of the users you created) as proof of originality. Export your report into a **PDF document**. Copy your scripts into a subdirectory named `scripts` and pack your scripts and report into a zip file. Your zip file should be named appropriately, eg `JohnnyExample_s1234567890_prac1.zip`. Please format your document and code nicely to assist the tutor's marking process. A poorly formatted document may receive a reduced mark. **Due 4pm on Friday the 26th of August, 2022.**

Late Penalties (from the ECP)

“Where an assessment item is submitted after the deadline, without an approved extension, a late penalty will apply. The late penalty shall be 10% of the maximum possible mark for the assessment item will be deducted per calendar day (or part thereof), up to a maximum of seven (7) days. After seven days, no marks will be awarded for the item. A day is considered to be a 24 hour block from the assessment item due time. Negative marks will not be awarded.”

Part 1: Oracle 12c Enterprise Basics

Prerequisites

You should either be in the lab on a UQ machine, connected to a UQ machine remotely (see the RDP guide), or have installed Oracle DB on your own device (see the installation guide). All guides are in the [Assessment > Practicals > Resources](#) directory on the LMS. You may wish to consult the RDP guide or the other resources in the directory listed above, as well as the ED discussion board, as they contain useful information for troubleshooting.

Start the Oracle software

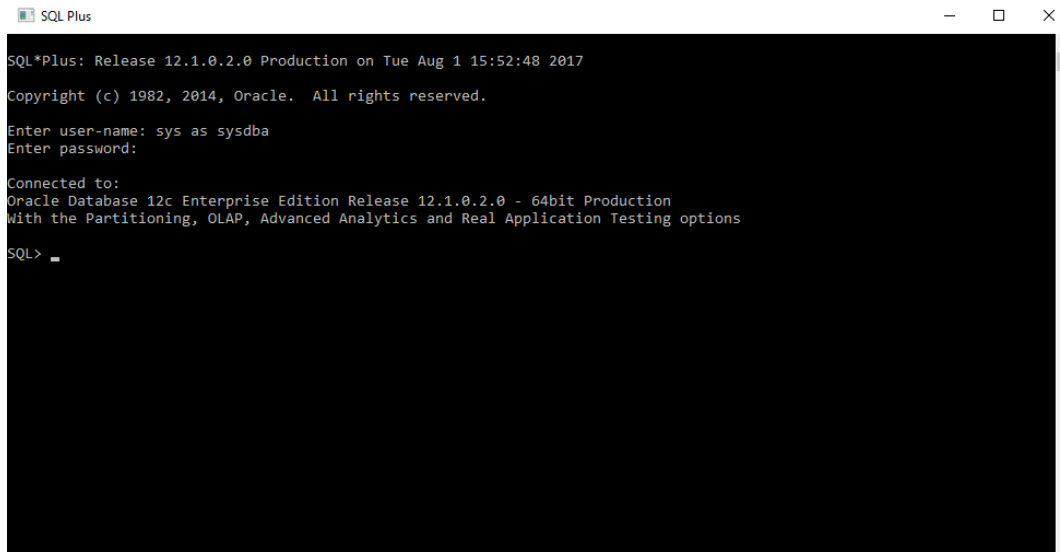
We will use [SQL Plus](#) to create database users, and use [SQL Developer](#) to connect as these users and interact with the database. To start [SQL Plus](#), open the command prompt (search [cmd](#) in the windows start menu) and type [sqlplus](#). You can find [SQL Developer](#) in the start menu.

Login to the Database

In the SQL Plus Command Line window, first log in with the following credentials:

- Username: [SYS AS SYSDBA](#)
- Password: [Password1](#)
 - **NOTE:** If the above password does not work, try using [Password1!](#)

A successful login should show the SQL> prompt. Otherwise, see the *troubleshooting* section below.



```
SQL Plus
SQL*Plus: Release 12.1.0.2.0 Production on Tue Aug 1 15:52:48 2017
Copyright (c) 1982, 2014, Oracle. All rights reserved.
Enter user-name: sys as sysdba
Enter password:
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
SQL> _
```

Create Users

Execute the commands below to create users. You can copy them, excluding comments (words in green), and right click your mouse in “SQL Plus” to paste.

NOTE: All “S1234567” mentioned in the document should be replaced by your student ID to distinguish your work from others, case insensitive.

```
/*Enable user creation*/
ALTER SESSION SET "_ORACLE_SCRIPT"=TRUE;

/* Create a user named "USER_S1234567" with password "w" */
CREATE USER USER_S1234567 IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT
TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";

/* Grant DBA privilege to "USER_S1234567" */
GRANT DBA TO USER_S1234567;

/* Check if "USER_S1234567" has been created */
SELECT USERNAME FROM DBA_USERS;
```

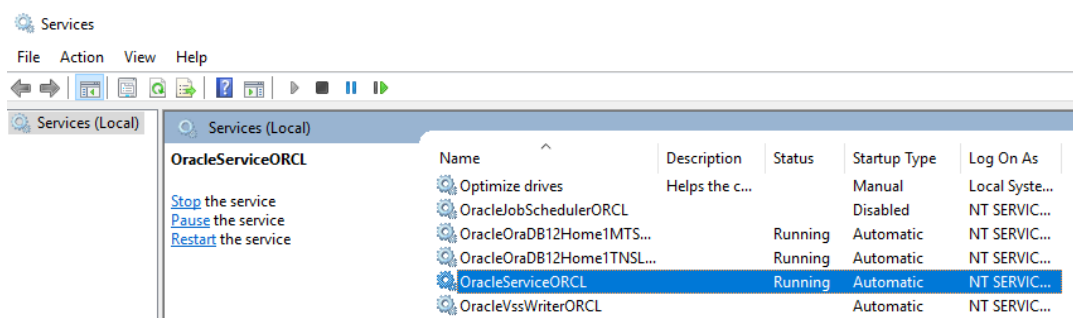
Proceed to step 3 if you complete the above processes successfully. Otherwise, continue below to troubleshoot the issues.

Troubleshooting

Problem: TNS: Protocol adapter error: Oracle services are closed, check the service state:

From the Windows 10 Start menu, search for **services**. In services, check to see if the following services are running. If they are not, then right-click and start them.

- OracleOraDB12Home2MTSRecoveryService
- OracleOraDB12Home2TNSListener
- OracleServiceORCL



Problem: Logon denied: Carefully check the username and password. If they seem correct and it is not working, then reset the user follows:

```
/* WARNING: this will drop everything under that user */  
DROP USER USER_S1234567 CASCADE;
```

Then redo the `CREATE` and `GRANT` commands (above), ensuring that the password is being set correctly as w and the username is correct.

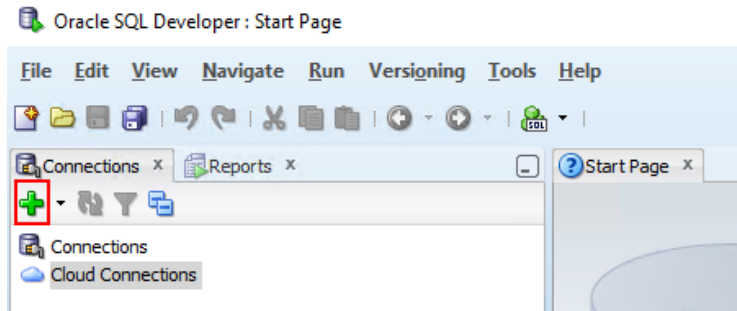
Problem: “USER_S1234567” conflicts with another user: When creating user, drop the existing user by running:

```
/* WARNING: this will drop everything under that user */  
DROP USER USER_S1234567 CASCADE;
```

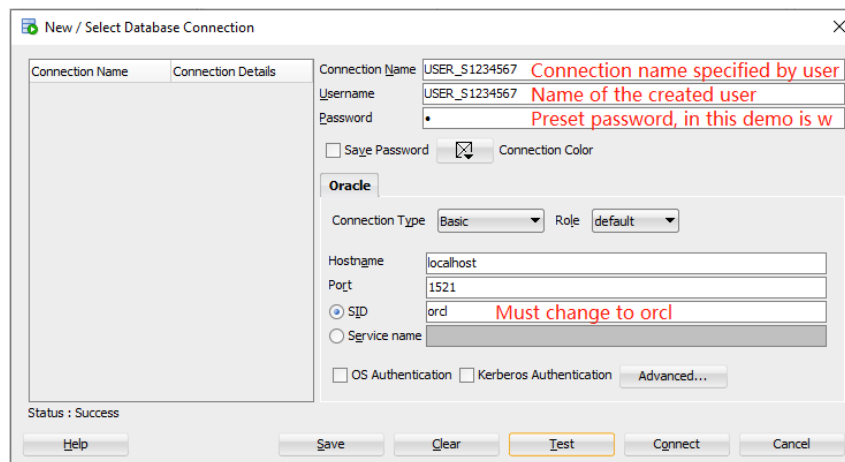
Then redo the `CREATE` and `GRANT` commands.

Start and Configure Oracle SQL Developer

Open [SQL Developer](#) in the start menu. We will use it to connect as the user we just created. Click the green “+” button as shown below.



Fill in the connection information in the prompted dialog window as shown below. The connection name is specified by the user. Username should be a user already existing in the database. In this example, it is the [USER_S1234567](#) we just created. Password is the password for that user, namely [w](#) (yes, the single character “w”) in this case. You should also change SID to [orcl](#). Then press the “Connect” button to connect to the user.

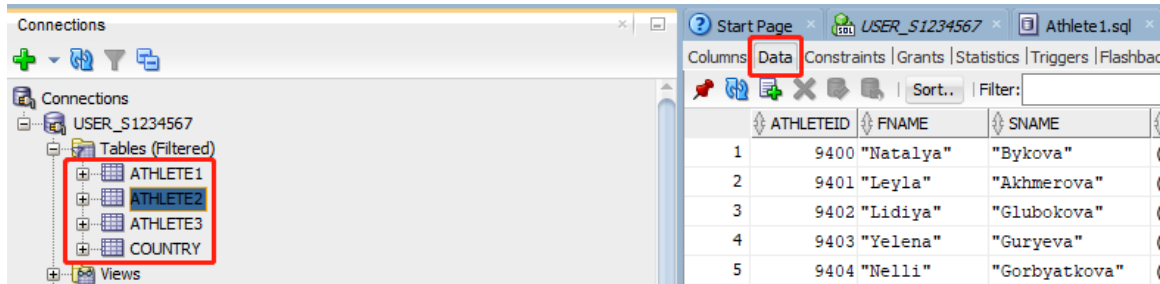


Import data to the database

Download the data files from the LMS to the system (there is a dataset zip file at [Assessment > Practicals > Prac One > Prac_1_Data.zip](#)). We recommend storing the files in the **H:** drive if you are using the UQ systems. Select **File > Open** (in the menu bar) and open the SQL scripts you just downloaded. It will show the script in the window. The script contains a table creation command ([CREATE TABLE](#)) and a list of record insertions ([INSERT INTO](#)). Click the second button (run all the scripts in the current tab) on its menu bar. A dialog will pop up asking for connection details. You should choose which connection you want to run the script. In this step, we choose the [USER_S1234567](#).

View the imported data

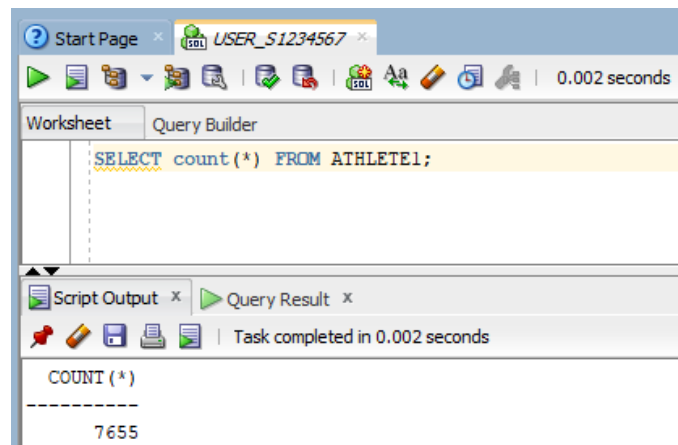
In the left panel, expand your connection and find your tables under the `Tables` directory. The basic information will be shown in the right window. Click the second (`Data`) tab in the right window to find the data you just imported, as shown below.



Note: Some of the data may be null. This is expected.

Interact with the database using SQL

You can write any SQL query in the corresponding connection window and run it by clicking either the first button (run a single query under current cursor) or the second one (run every script in the current tab). For example, the query below tries to find out how many records we have imported to table `Athlete1`. We will ask you a few similar questions in task 1 which requires you understand the meaning of the table attributes and help you review the basic SQL keywords (`SELECT`, `WHERE`, `GROUP BY`, `JOIN`, etc.) which are necessary for the rest of the course.



Assessment Task 1

Write SQL queries to answer the following questions.

Your queries and screenshots of the results should be included in your submission.

- (1) Count the number of players from Australia (country code=AUS) in *Athlete2* table.
- (2) For all French (FRA) players in table *Athlete3*, count the number of players participating in each sport. The result should be a list containing records like:

	Sport ID	Count
1		
2		

- (3) 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.

Part 2: Distributed DB Design

In part 2, we aim to simulate a distributed database using a standalone computer. In essence, we are acting as a global site, which possesses the global conceptual schema and data replication info, and deals with all the incoming queries. To simulate distributed databases, we create multiple user accounts in Oracle 12c (refer back to *Part1: Oracle 12c Enterprise Basics* for a refresher on how to create a user account), each of which represents a distributed database site. Then, the data transferred among relations belonging to different user accounts corresponds to the data transferred over computer network among different sites.

We provide you with an `Athlete` table which is defined as follows:

```
Athlete [AthleteID, FName, LName, DOB, CountryCode, SportID]
```

There are 24,591 records with `AthleteID` ranging from 1 to 24,591, inclusive.

1. Horizontal fragmentation

There are three types of replication strategies: *Full Replication*, *Partial Replication* and *No Replication*. In this task, you are asked to simulate these three strategies on a distributed database which contains three local sites (`USER1`, `USER2`, and `USER3`). Each strategy then requires 3 unique users for simulation – you need to **create 9 users** in total for Task 2 (this will be shown below). After creating the users, you are asked to run the scripts in the respective folders to perform different data replications.

Job 1 - Full Replication

The data is split into three fragments:

- `Athlete1: 1 <= AthleteID < 7656`
- `Athlete2: 7657 <= AthleteID < 17318`
- `Athlete3: 17319 <= AthleteID <= 24591`

Each fragment will be a relation located on every site in the computer network (i.e. **each site has a full copy of each fragment**). You should create three sites to simulate the full replication in SQL Plus command line:

- `USER1_HF_FULL_S1234567`
- `USER2_HF_FULL_S1234567`
- `USER3_HF_FULL_S1234567`

To load fragments into site `USER1_HF_FULL_S1234567`, connect to user `USER1_HF_FULL_S1234567` in SQL Developer and run all script files in folder `...\P1\Part 2\HF\HF-Full\USER1_HF_FULL\` – Repeat the same process for other sites.

Job 2 – Partial Replication

The data is split into three fragments in the same way as in Job 1.

Each fragment will be a relation located on some of the sites in the computer network (i.e., **more than one site may have a copy of this fragment, but not all of them. You should read through the scripts to understand how fragments are replicated and allocated here**). You should create three sites:

- USER1_HF_PA_S1234567
- USER2_HF_PA_S1234567
- USER3_HF_PA_S1234567

In order to load the above fragments into site USER1_HF_PA_S1234567, connect to user USER1_HF_PA_S1234567 in SQL Developer and run all script files in folder ...\\P1\\Part 2\\HF\\HF-Partial\\USER1_HF_PA\\ – Repeat the same process for other sites.

Job 3 – No Replication

The data is split into three fragments in the same way as in Job 1.

Each fragment will be a relation located on **only one site** in the computer network. You should create three sites:

- USER1_HF_NO_S1234567
- USER2_HF_NO_S1234567
- USER3_HF_NO_S1234567

In order to load the above fragments into site USER1_HF_NO_S1234567, connect to user USER1_HF_NO_S1234567 in SQL Developer and run all script files in folder ...\\P1\\Part 2\\HF\\HF-No\\USER1_HF_NO\\ – Repeat the same process for other sites.

Assessment Task 2

Given the update query below, write a set of SQL queries (or preferably, one transaction) which applies this update to the system under each replication strategy, respectively. (**Hint:** Three sets of SQL queries (or three transactions) will be required for the three different strategies. Each of your update transactions should guarantee consistency between copies and should not perform updates to sites which do not have the record).

Update Query: Change the country code of the player whose ID is 128 to AUS

Don't forget to put your SQL queries/transactions, your result screenshots, and the explanation of the differences of update operations between three replication strategies in your submission.

Vertical Fragmentation

In our simulation of vertical fragmentation, we will fragment as follows:

- AthleteV1[AthleteID, FName, LName]
- AthleteV2[AthleteID, DOB, CountryCode, SportID]

Create two users to simulate two sites, and load the data from folder ...\\P1\\Part 2\\VF\\

- USER1_VF_S1234567
- USER2_VF_S1234567

Assessment Task 3

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
```

Include your query and the screenshot of the result in your submission.

Part 3: Distributed Query Processing

In part 3, we still simulate a distributed database using this centralised Oracle database. However, the distributed sites are now organised in a peer-to-peer architecture, which means the queries can be issued at any of the local sites and receive an answer from it. In this part, you are asked to perform inner join queries in our simulated distributed environment efficiently. In particular, you should find the *optimal execution plan* for the join queries, as mentioned in *Lecture 3*. The optimal execution plan refers to the one with minimum data transmission cost. To help you trace the data transmission cost, we provide the following statistical tool:

Open **SQL Plus** and login as **SYS AS SYSDBA** (also applicable to other users you created). Type in the following commands to turn on query statistics:

```
SQL> SET TIMING ON; /* enable timing */
SQL> SET AUTOTRACE ON STATISTICS; /* enable statistics */
```

We use a simple query as an example. Here is the statistical result of the following query:

```
SELECT a.AthleteID, a.CCODE, b.CONTINENT
FROM "USER2_VF_S1234567"."ATHLETE_V2" a,
"USER_S1234567"."COUNTRY" b
WHERE a.CCODE=b.CCODE;
```

```

24585 rows selected.

Elapsed: 00:00:07.82

Statistics
-----
      5 recursive calls
      0 db block gets
    1714 consistent gets
      0 physical reads
      0 redo size
    604782 bytes sent via SQL*Net to client
    18626 bytes received via SQL*Net from client
     1640 SQL*Net roundtrips to/from client
      0 sorts (memory)
      0 sorts (disk)
    24585 rows processed

```

Note that the statistics may vary among multiple runs except those marked in red, which are our main focus. The descriptions of the statistics are shown in the appendix. In particular, the “*bytes sent via SQL*Net to client*” refers to the size of the query results sent from the database to the user, which is irrelevant to the data transmission cost. However, this statistical tool can be used to estimate the data transmission cost indirectly. For example, if we issue this query at site USER_S1234567 and run it with the semi-join execution plan, the whole query can be divided into three steps:

- (1) **Send** the projection of “CCODE” from site USER to USER2
- (2) Perform semi-join on site USER2 then **send** the result back to USER
- (3) Perform the final join locally and send the final results to user.

Therefore, the total transmission cost should be the size of step (1) projection results + step (2) results (final result is not included as you have to return the same result to the user regardless of which execution plan is used; the results should be invariant). Specifically, to calculate the cost, you have to perform the following queries to obtain the size of the intermediate results:

Step One – Get the projection of “CCODE” from USER.COUNTRY

```
SELECT DISTINCT(CCODE) FROM "USER_S1234567"."COUNTRY";
```

```

136 rows selected.

Elapsed: 00:00:00.03

Statistics
-----
      1 recursive calls
      0 db block gets
      7 consistent gets
      0 physical reads
      0 redo size
     3017 bytes sent via SQL*Net to client
     707 bytes received via SQL*Net from client
      11 SQL*Net roundtrips to/from client
      0 sorts (memory)
      0 sorts (disk)
     136 rows processed

```

Step Two – Perform the semi join on USER2

```
SELECT a.AthleteID, a.CCODE FROM "USER2_VF_S1234567"."ATHLETE_V2" a WHERE  
a.CCODE IN (SELECT DISTINCT(CCODE) FROM "USER_S1234567"."COUNTRY");
```

```
24585 rows selected.  
Elapsed: 00:00:04.95  
Statistics  
-----  
      1 recursive calls  
      0 db block gets  
    1711 consistent gets  
      0 physical reads  
      0 redo size  
  589666 bytes sent via SQL*Net to client  
   18626 bytes received via SQL*Net from client  
    1640 SQL*Net roundtrips to/from client  
      0 sorts (memory)  
      0 sorts (disk)  
   24585 rows processed
```

Step Three – Perform the final join on USER

```
SELECT a.AthleteID, a.CCode, b.CONTINENT  
FROM "USER_S1234567"."COUNTRY" b,  
(SELECT a.AthleteID, a.CCODE FROM "USER2_VF_S1234567"."ATHLETE_V2" a WHERE  
a.CCODE IN (SELECT DISTINCT(CCODE) FROM "USER_S1234567"."COUNTRY")) a  
WHERE a.CCODE=b.CCODE;
```

```
24585 rows selected.  
Elapsed: 00:00:05.09  
Statistics  
-----  
      4 recursive calls  
      0 db block gets  
    1714 consistent gets  
      0 physical reads  
      0 redo size  
 604782 bytes sent via SQL*Net to client  
   18626 bytes received via SQL*Net from client  
    1640 SQL*Net roundtrips to/from client  
      0 sorts (memory)  
      0 sorts (disk)  
   24585 rows processed
```

Therefore, the transmission cost of the semi-join plan is $3017(\text{step 1}) + 589666(\text{step 2}) = 592683$.

Note that the final join in step three does not do any data transmission, so we do not count the byte values (it's just joining the data locally). The example shows that by decomposing the query into a step-by-step plan, we can track the data transmission cost by aggregating the sizes of their intermediate results.

Additionally, we provide you with several tips for this task:

Tip 1: For the same query, the size of the final results should always be identical regardless of what execution plan you take. Like in the above example, the final result size is always 604782.

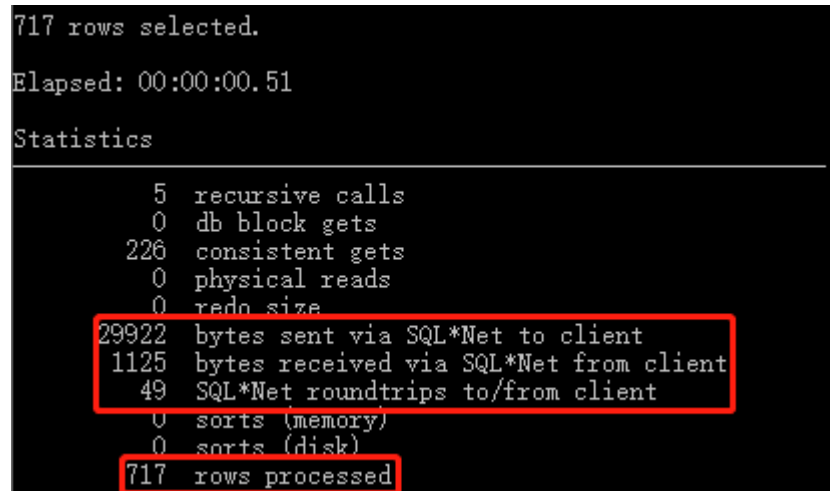
Tip 2: Despite different execution plans, the running time of the query should always be similar. This is because we are using a centralised database to simulate a distributed environment. Oracle will optimise every query and, no matter if it is written as an inner join or semi-join, it will be executed in the same plan as long as those queries are semantically equivalent. In practice, bad execution plans can be very expensive!

Tip 3: Although we regard each user account as a distributed site, you will not see any difference if you run the same query using different users as they are all local users. However, in a real distributed database, queries issued at different sites will result in completely different execution plans. For example, if your query is issued at site 1, it is better to allocate your final join at site 1 so that you can directly return the result to the client (user) once it's completed.

Assessment Task 4

Suppose that we want to retrieve all the information for Australian athletes from the vertical fragments created in Task 2, which can be achieved by the following join query:

```
SELECT b.AthleteID, b.FName, b.SName, c.BDate, c.CCode, c.SportID
FROM "USER1_VF_S1234567"."ATHLETE_V1" b, "USER2_VF_S1234567"."ATHLETE_V2"
c
WHERE b.AthleteID= c.AthleteID and c.CCODE='AUS';
```



```
717 rows selected.
Elapsed: 00:00:00.51
Statistics
-----
      5  recursive calls
      0  db block gets
     226  consistent gets
      0  physical reads
      0  redo size
    29922  bytes sent via SQL*Net to client
     1125  bytes received via SQL*Net from client
      49  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
     717  rows processed
```

If the join query is issued at site USER1, write a semi-join and an inner-join execution plan, respectively, remembering that the database is vertically fragmented. Follow the above example to calculate their respective data transmission cost and decide which plan is better.

Be sure to include step-by-step queries, cost calculations, and your choice of the best execution plan in your submission. Make sure you support your answer with screenshots of the query statistics.

```

4642 rows selected.

Elapsed: 00:00:02.85

Statistics
-----
      12  recursive calls
       0  db block gets
    26691  consistent gets
       0  physical reads
       0  redo size
   192749  bytes sent via SQL*Net to client
    4007  bytes received via SQL*Net from client
     311  SQL*Net roundtrips to/from client
       1  sorts (memory)
       0  sorts (disk)
    4642  rows processed

```

Appendix: Statistics Description

Database Statistic Name	Description
recursive calls	Number of recursive calls generated at both the user and system level. Oracle maintains tables used for internal processing. When Oracle needs to make a change to these tables, it internally generates an internal SQL statement, which in turn generates a recursive call.
db block gets	Number of times a CURRENT block was requested.
consistent gets	Number of times a consistent read was requested for a block
physical reads	Total number of data blocks read from disk. This number equals the value of "physical reads direct" plus all reads into buffer cache.
redo size	Total amount of redo generated in bytes
bytes sent via SQL*Net to client	Total number of bytes sent from Oracle database to the client (user) through network, which includes the message initiation cost and the cost of sending query results.
bytes received via SQL*Net from client	Total number of bytes received from the client over Oracle Net.
SQL*Net roundtrips to/from client	Total number of Oracle Net messages sent to and received from the client
sorts (memory)	Number of sort operations that were performed completely in memory and did not require any disk writes
sorts (disk)	Number of sort operations that required at least one disk write
rows processed	Number of rows processed during the operation

Changelog

V1: Original version

V2: Minor clarifications about data download location, troubleshooting, and null values.