**Lab 5 Submittal**

**Step 1: Launch Oracle SQL Developer or equivalent SQL application IDE.**

A screenshot of a computer

Description automatically generated

**Step 2: Create and Populate the Table**

**Create Table:**

****

Output of the create table query.

**Populating the table:**

A screenshot of a computer

Description automatically generated

Output after adding 6 new entries to the table

**Step 3: Create and Run Various PL / SQL Scripts with Your Data Table**

**Script 1: Determine the Count of the Records**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Output of the script where I entered v\_name = Harshal

**Script 2: Determine the Count of the Records for Some Criteria**

**A screenshot of a computer program

Description automatically generated**

Output of the script where I made the changes where needed.

**Script 3: Determine the Average of the xValues and yValues.**

**A number on a white background

Description automatically generated**

Normal Output

A screenshot of a computer

Description automatically generated

Output after using ROUND function with 2 decimal places.

**Script 4: Determine the Greater Average of two Columns**

A screenshot of a computer

Description automatically generated

Output

**Script 5: Determine the Greater Average of two Columns**

A screenshot of a computer program

Description automatically generated

Output

**Script 6: Determine the Weighted Average**

**A screenshot of a computer program

Description automatically generated**

Output

**Script 7: Using Substitution Variables**

A screenshot of a computer

Description automatically generated

Script 8: Create another Table.

A screenshot of a computer

Description automatically generated

Output

A screenshot of a computer

Description automatically generated

Adding the data

**Script 9: Compare the tables.**

A screenshot of a computer program

Description automatically generated

The script

A screenshot of a computer

Description automatically generated

The Output

**Script 10: Using Substitution Values.**

A screenshot of a computer

Description automatically generated

Accepting name of the analyst at runtime

A screenshot of a computer

Description automatically generated

Output

**Step 4: PL/SQL Tuning**

**1)**

**A white box with black text

Description automatically generated**

**2)**

**A screenshot of a computer program

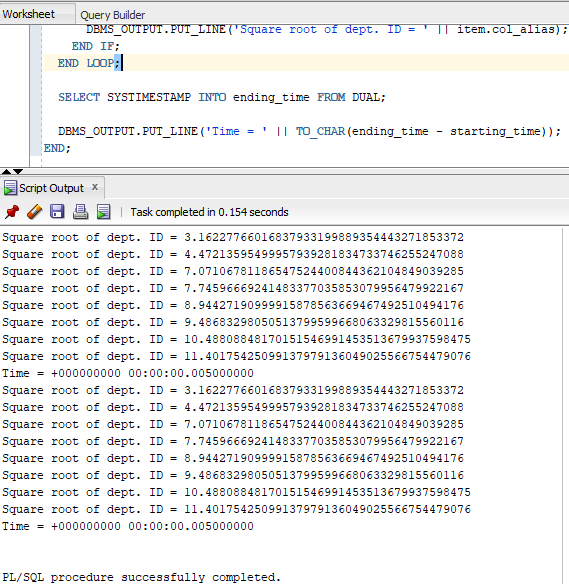
Description automatically generated**

**3)**

A screenshot of a computer error

Description automatically generated

**4)**



Because it applies the SQRT function to the workers table directly and doesn't require an additional subquery, the first portion is more efficient. In general, this direct square root computation for every individual department is more efficient. On the other hand, the second component adds cost and becomes somewhat less efficient by requiring an extra subquery to get unique department IDs. The first portion is typically more efficient because it minimizes needless subqueries, even if the two sections' timing differences are negligible (both reporting 0.005 seconds).

But if you run the query again it gives different output:

A screenshot of a computer code

Description automatically generated

But Efficiency in database queries is often about minimizing the amount of work the database needs to do. The second query is more efficient because it reduces the number of calculations performed by applying the function to a smaller subset of data.

**Step 5: Questions and Reflections Concerning this Database Project**

1. When and where should EXCEPTION statements be used in a PL - SQL block

statement?

* In PL/SQL, EXCEPTION statements are used to deal with mistakes or special circumstances. They can be used in SQL statements to handle SQL exceptions or in exception handlers to capture particular problems (like {WHEN NO\_DATA\_FOUND}). The `RAISE` statement can also be used to re-raise exceptions or construct custom exceptions. You can handle errors at higher levels of your code thanks to exception propagation. Furthermore, EXCEPTION statements in database triggers manage failures that arise during trigger execution. Robust PL/SQL applications must have proper exception management, which may also provide error handling and response that is gracious.

1. When using PL - SQL , differentiate between a function, a procedure and a Package. Point when each of these entities may be used.

* Functions, procedures, and packages are separate database objects in PL/SQL, each with a particular function:

**Function**:

A PL/SQL software unit that returns a single value is called a function.

Usually, a computation is carried out and the caller receives the result.

Functions are frequently used in SQL expressions to obtain and compute data. Examples of these computations include finding the square root of an integer and converting units.

**Procedure**:

A procedure is a unit of PL/SQL code that executes one or more operations.

Unlike functions, it doesn't return a result; instead, it's utilized for its side effects, which include managing errors, creating reports, and altering database information.

Procedures are frequently used to generate reusable code, encapsulate business logic, and manipulate data.

**Package**:

A package serves as a container to hold variables, processes, and associated functions together into a single entity.

It facilitates improved code management and offers modular organization.

Code reusability and maintainability are improved through the usage of packages, which are used to distribute and maintain code among other applications or modules.

1. Distinguish between Oracle date types RRRR and YYYY.

* When displaying the year in Oracle date formats, {YYYY} presents it as a four-digit number without any modifications. It indicates the date's actual year. Nevertheless, `RRRR} accounts for two-digit year ambiguity while formatting the year as four digits as well. Based on the current date and system settings, Oracle reads two-digit years (e.g., '00' to '49') as 2000-2049 and (e.g., '50' to '99') as 1950-1999 when using `RRRR}. When working with dates where the century is ambiguous, {RRRR~ is frequently recommended to avoid misinterpreting dates close to the turn of the century.

1. Can substitution variables be used in a function definition? Support your answer.

* No, you cannot utilize substitution variables directly in a PL/SQL function declaration. Substitution variables are not a component of PL/SQL syntax; instead, they are a feature of SQL\*Plus or SQL Developer tools, used for scripting or interactive input.

In PL/SQL, function definitions need a well-defined structure and certain data types for arguments. A function's arguments can be defined, but they are typed explicitly and cannot be changed using substitution variables.

When calling a function, substitution variables can be used to supply input values; however, these values must be supplied as arguments to the function and cannot be used directly in the function declaration.

1. When should for loops be used as opposed to using while loops? Support your answer with examples.

* **For loops**:

When you know ahead of time the precise amount of iterations, you should use for loops. They are perfect when you want to run a piece of code a certain number of times or within a specific range of values. For loops are more suitable, for instance, for iterating through an array's items or carrying out a certain action a predetermined number of times.

Example:

**FOR i IN 1..10 LOOP**

**DBMS\_OUTPUT.PUT\_LINE(i);**

**END LOOP**;

**While loops:**

Conversely, while loops are employed when the precise number of iterations is unknown beforehand or when you need to iterate depending on a condition that could change while the loop is being executed. While loops keep running as long as the given condition is still true. When specific conditions are met, they can be used for activities like processing items in a list or reading data till the end of a file.

Example:

**DECLARE**

**countdown NUMBER := 10;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Countdown to liftoff:');**

**WHILE countdown >= 1 LOOP**

**DBMS\_OUTPUT.PUT\_LINE(countdown);**

**countdown := countdown - 1;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Liftoff!');**

**END;**