We have not talked about commit yet. Any time you make changes to the database (DML) you must commit the changes to make them permanent. A commit is similar to saving a file. If you end a file and you close the file without saving, you lose the changes.

SQL/Plus defaults to committing all changes on exit. Do not rely on SQL/Plus. End your pop script with COMMIT.

Having a private instance variable to store the connection is correct, but the result set and statement should be local to the function. A function may execute multiple queries and combine results. That would require two statements and two result sets.

All of your code looks good. You may shorten the code for each test case using the Java conditional operator (link below). There's nothing wrong with the IF/ELSE that you used, but if you're creating hundreds of test cases, the conditional operator is compact and less to type.

<https://www.w3schools.in/java/operators/conditional>

There's no need for the driver to exit. Let the program end naturally. To be safe, the driver should call DataSource.close() to confirm that the database connection is closed. The close method should check the connection. If the connection variable is NOT NULL, the method should close the connection.

There's no need for the Ctrl class to have any instance variables. Instance variables represent properties of a class. DataSource, TextIO, and query results are not properties. Use local variables.

Names are extremely important in software design. The software developer's top priority is readability. The compiler has no difficulty understanding code. Code is written for people. Use meaningful names.

Many software companies do not allow any comments in source code files. If developers choose variables names, method names, etc. wisely, there is no need for comments.

Figure out what your doStuff method does and give the method a name that clearly states what the method does.

DataSource.getInstance should not have any parameters. You may need to create an open() method in the DataSource class to initialize the username and password.

The way you parsed the commands for this assignment is fine. For the project, you will not display the commands. Just like the Unix or Windows command line, users are expected to know the commands. You will provide a help command that lists all commands. This will be clearer in the project assignment.

The first input is always the command followed by a number of parameters. Hence, the code should get the command and use a switch statement to process the command. Each command knows how many parameters are expected, so let each command process the parameters. For the project, each command will have its own method. You will see how this is much easier to read, extend, and maintain.

Again, use meaningful variable names: command, param1, param2, etc.

My first job out of college was working for a major telecommunications company developing software for the world's first all-digital telecommunications switch. The code base was 16M LOC (lines of code). Imagine if I had to read 16M LOC written with methods named doStuff and variables named one and two. Code is written for people to read.

empList method  
Methods and functions should be short (6 - 8 lines of code) and they should do one thing ([Single Responsibility](https://stackify.com/solid-design-principles/)).  
  
I like that you created optional parameters to specify the results to be returned. The command parameters should be processed in the Controller (Ctrl), not the data source. This also applies to the Single Responsibility pattern. The data source interacts with the database. That is the ONLY responsibility the data source has.

The Controller (Ctrl) controls what the application does. Hence, the controller processes commands.

One option could have the controller build query strings and pass a query string to a single method, executeQuery, in the data source. That design would violate single responsibility. The data source is the only class that builds queries. Why?

Encapsulation is crucial to quality software design. No data source client should know how the data source is storing data. The data source could be use a database, a flat file, a JSON datafile, or a NoSQL database. Why? Because the system must support change. The data source might be implemented initially with a flat CSV datafile and eventually the system needs a database, so the data source is changed to use a database. If clients of the data source are passing queries to the data source, every client must be changed. That's a nightmare.

The data source should have one method for each command (e.g., listEmployeeNames, listAllEmployeeDetails, listEmployees). The controller processes the command and calls the appropriate method in the data source. This design creates lots of small methods that have a single responsibility. Easy to implement, test, maintain, and extend.

Excellent demo. Excellent work on this exercise.

**Design Decisions**  
This exercise was pretty specific, but you still made several decisions, but no major decisions. Your discussion of having one method or multiple methods in the data source was valuable.

You could have an overloaded empList method in the data source, but you would only have two methods: with and without an optional parameter. Hence, the with param method would have to determine which parameter was passed in. Better design to create one method that returns a specific dataset: list of names, list of ID and names, all employee info.

A Simple Question to Validate Your Design

Unfortunately, following the single responsibility principle sounds a lot easier than it often is.

If you build your software over a longer period and if you need to adapt it to changing requirements, it might seem like the easiest and fastest approach is adding a method or functionality to your existing code instead of writing a new class or component. But that often results in classes with more than responsibility and makes it more and more difficult to maintain the software.

You can avoid these problems by asking a simple question before you make any changes:

*What is the responsibility of your class/component/ microservice?*