Apache Derby Database Installation in Windows

Author: Sri Adilakshmi M

Table of Contents

This document outlines the key features of **Apache Derby** database, it's installation on Windows operating system and how it can be accessed.

1. Overview:

Apache Derby is an open source relational database written in Java. Apache Derby is used in the JDK and is called Java DB *(Apache Derby and Java DB are same)*. Apache Derby is the reference implementation for JDBC4.0 and compatible for ANSI-SQL.

Key Features:

Some of the key features include:

- Derby is easy to install, deploy and use.
- Derby has a small footprint which is about 3.5 MB of base engine and JDBC driver.
- Derby is based on Java, JDBC and SQL standards.
- Derby provides an embedded JDBC driver that allows us to embed Derby in any Java-based application.
- Derby supports both client (embedded) and server mode which is possible with Derby Network Client JDBC server (embedded JDBC driver) and Derby Network server respectively.

2. Installation:

Apache Derby database requires Java Runtime Environment (JRE) to run and it works on all Java releases starting from 1.3 to 21.

2.1. Install JDK:

To run Apache Derby, we must have either **JRE** or fully packed **JDK** (Java Development Kit) that includes JRE, JVM and other tools.

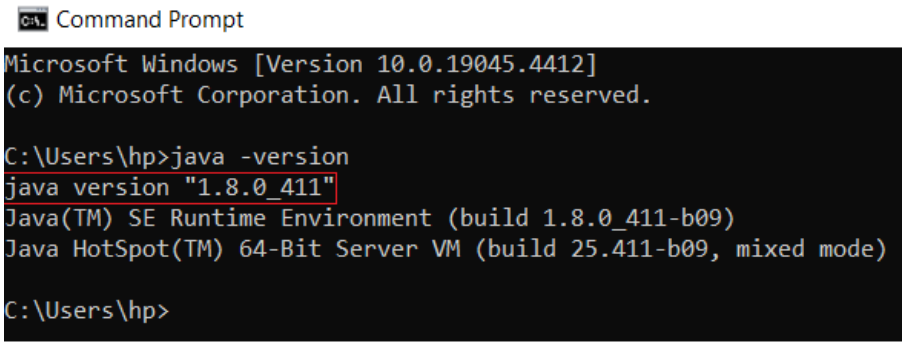
If JDK or JRE is not currently available in the machine, then install JDK from the Oracle Java Downloads website and configure JAVA_HOME and PATH environment variables. For the complete JDK installation steps, refer here.

If you are unable to download Oracle JDK, then install JRE 8 for Windows from the official Java Download website.

Here, I am using Oracle JDK 8 release and installed it in the location `D:\ProgramFiles\Java\jdk-1.8` in my Windows machine.

After JDK is installed and configured, run the following command on **Windows Command Prompt** or **Windows PowerShell** to verify the installed Java version.

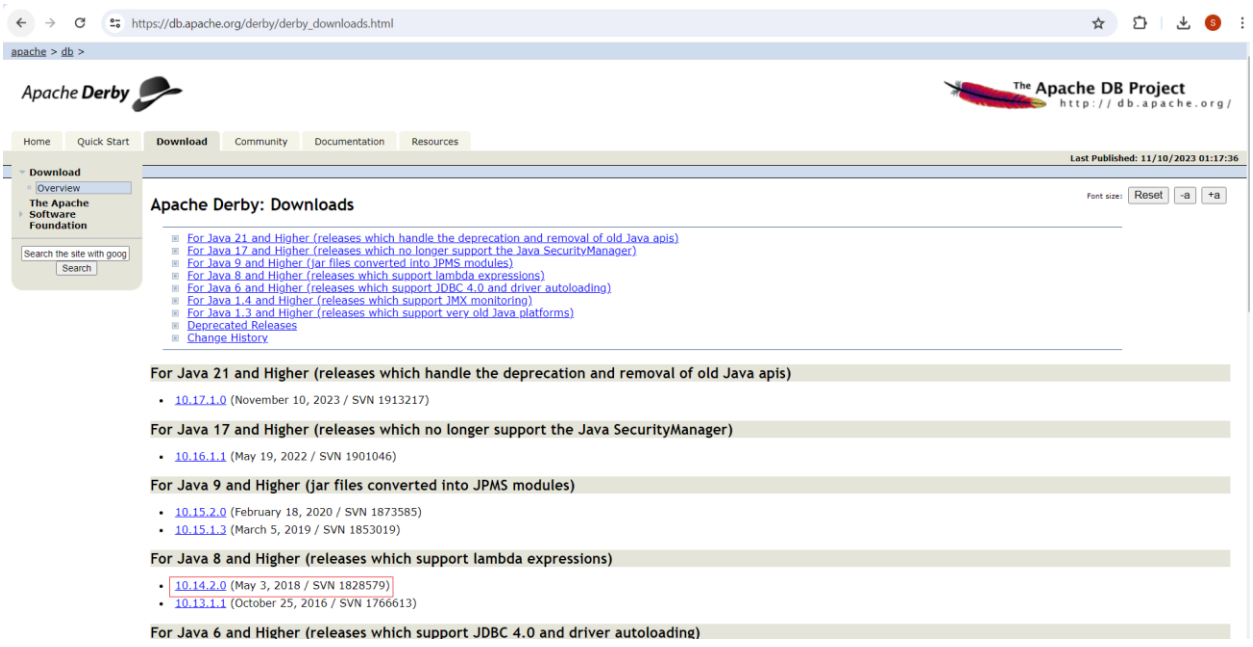
```
java -version
```



2.2. Install Derby:

After JDK is installed, we need to install Apache Derby from the [Apache Derby Downloads](https://db.apache.org/derby/derby_downloads.html) website. Make sure to choose the Derby version for the corresponding Java release running in your machine and download it.

For example, I am choosing **Apache Derby for Java 8** since I have JDK 8 release in my machine. The latest available Derby version for Java 8 is **10.14.2.0.**

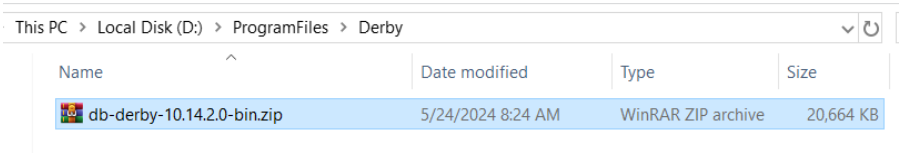


Click on **10.14.2.0** link which takes to Apache Derby 10.14.2.0 Release page.

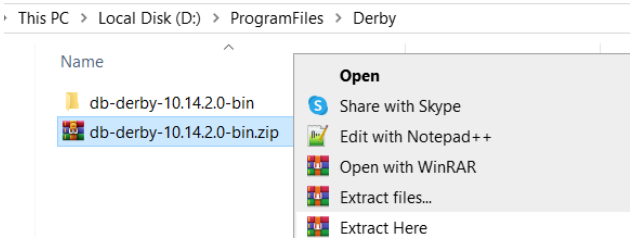
Download **`db-derby-10.14.2.0-bin.zip`** file for Windows installation and it gets downloaded to your default **Downloads** folder.

After the Derby file is downloaded, choose the installation directory in your machine and copy `db-derby-10.14.2.0-bin.zip` file to that directory.

For example, I am choosing my Derby installation directory as `D:\ProgramFiles\Derby`



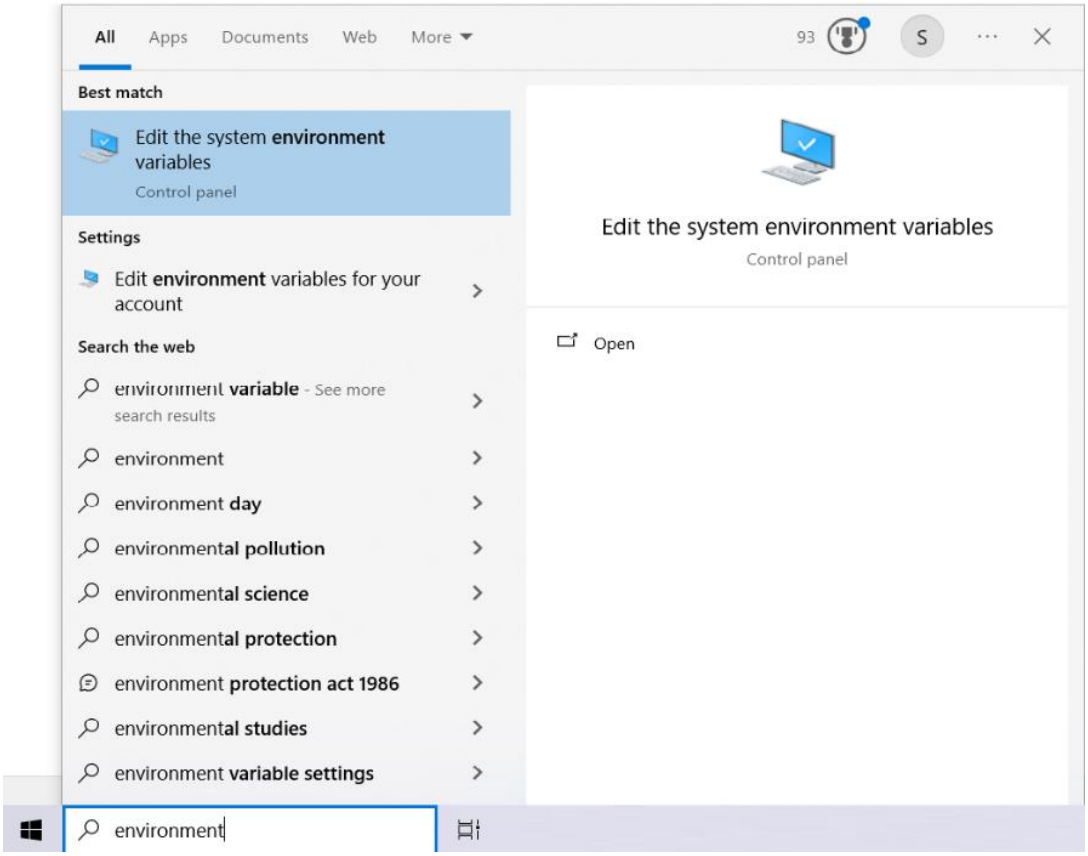
Unzip `db-derby-10.14.2.0-bin.zip` file using WinRAR or 7-Zip tool and it creates a directory named `db-derby-10.14.2.0-bin` in the installation folder with all necessary binaries of Derby.



2.3. Setup Derby Home:

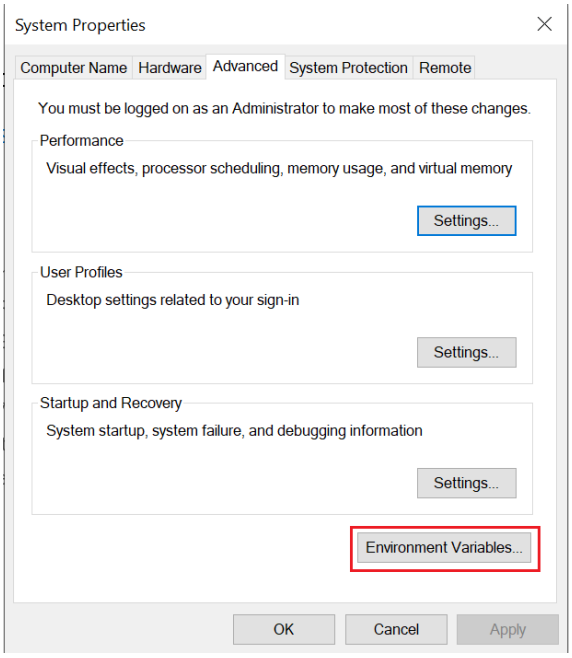
After Derby binaries are downloaded to your local machine, the next step is to set `DERBY_HOME` environment variable to the location where Derby is installed.

In the Windows search bar, start typing “environment variables” and select the first match which opens up **System Properties** dialog.



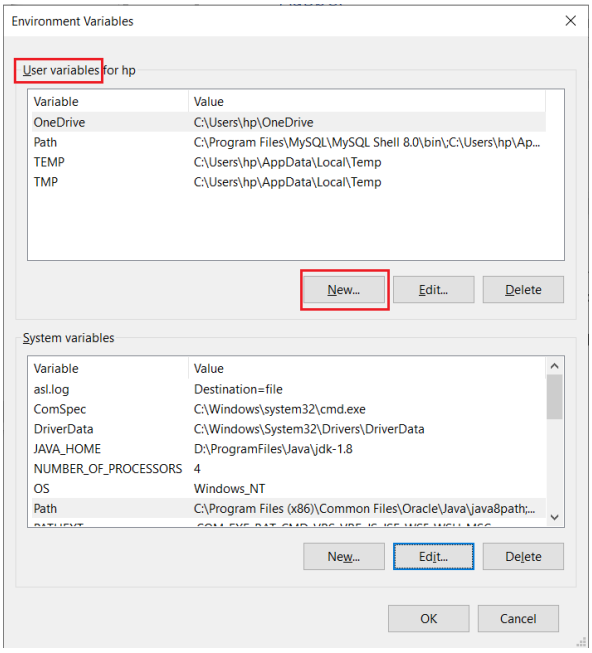
Alternatively (in Windows 8 or 9 versions), you can open **Control Panel** wizard and select **System and Security** setting. Then choose the **System** setting and click on it. In the **System Settings** window, click on **Advanced system settings** link to open up the **System Properties** window.

On the **System Properties** window, press **Environment Variables** button.

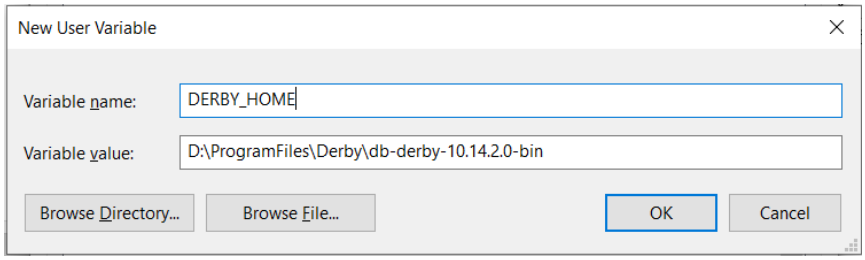


We will add DERBY_HOME variable to **User variables** since we are configuring Derby for a single user. If Derby needs to be configured for multiple users then add it to **System variables**.

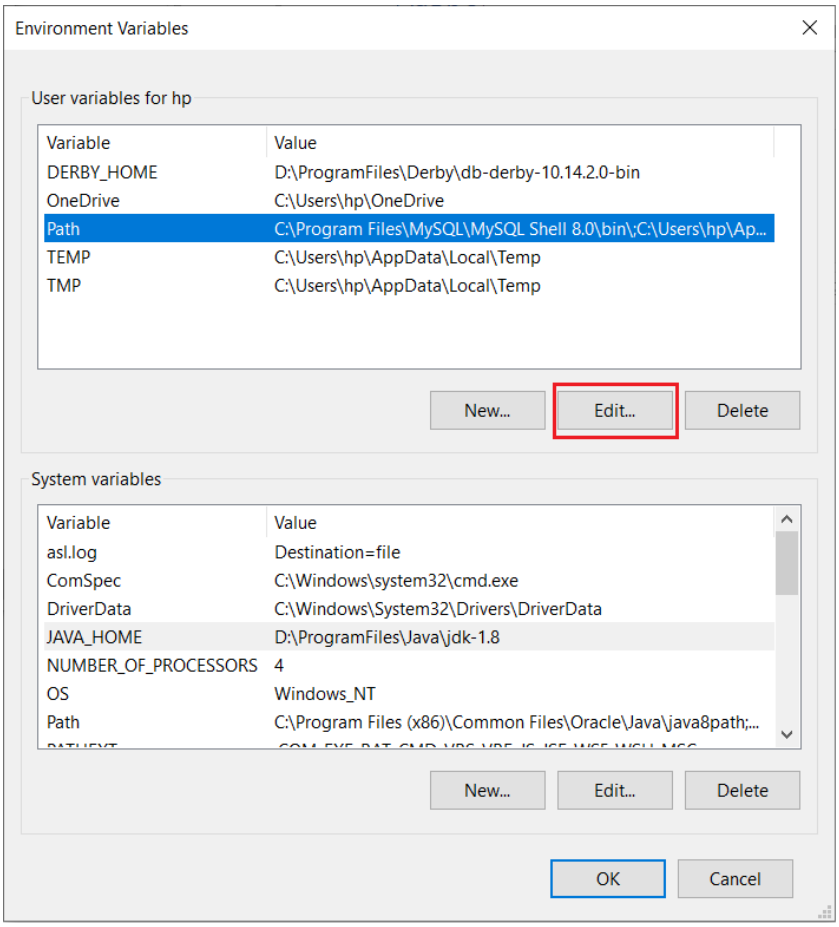
On the **Environment Variables** dialog, press **New** under **User variables** tab.



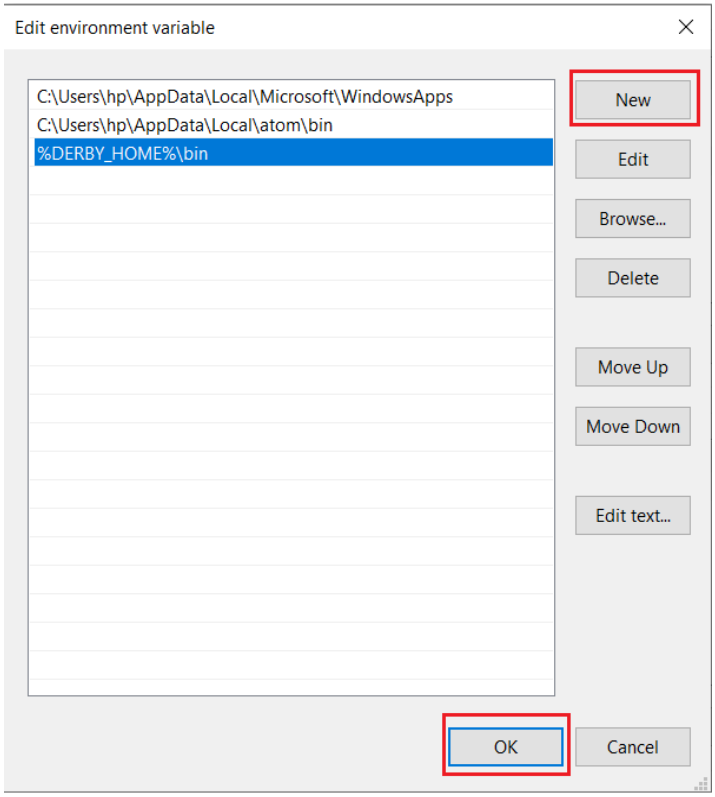
In the **New User Variable** window, enter the **Variable name** as `DERBY_HOME` and **Variable value** as `D:\ProgramFiles\Derby\db-derby-10.14.2.0-bin` where Derby was installed and then press OK.



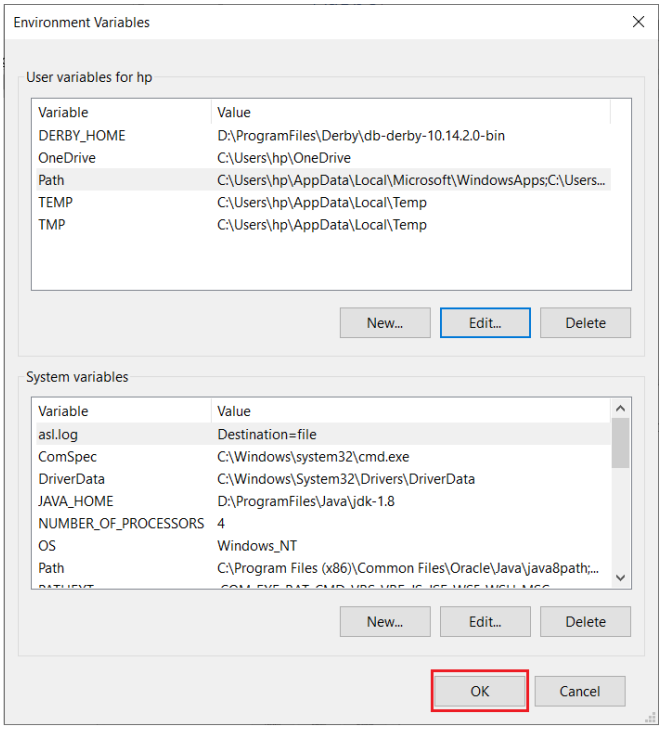
Now choose **Path** variable under **User variables** and click on Edit button.



Click on **New** and enter value as `%DERBY_HOME%\bin` and press **OK**



Again, press **OK** to apply the changes for Environment Variables and close the window.

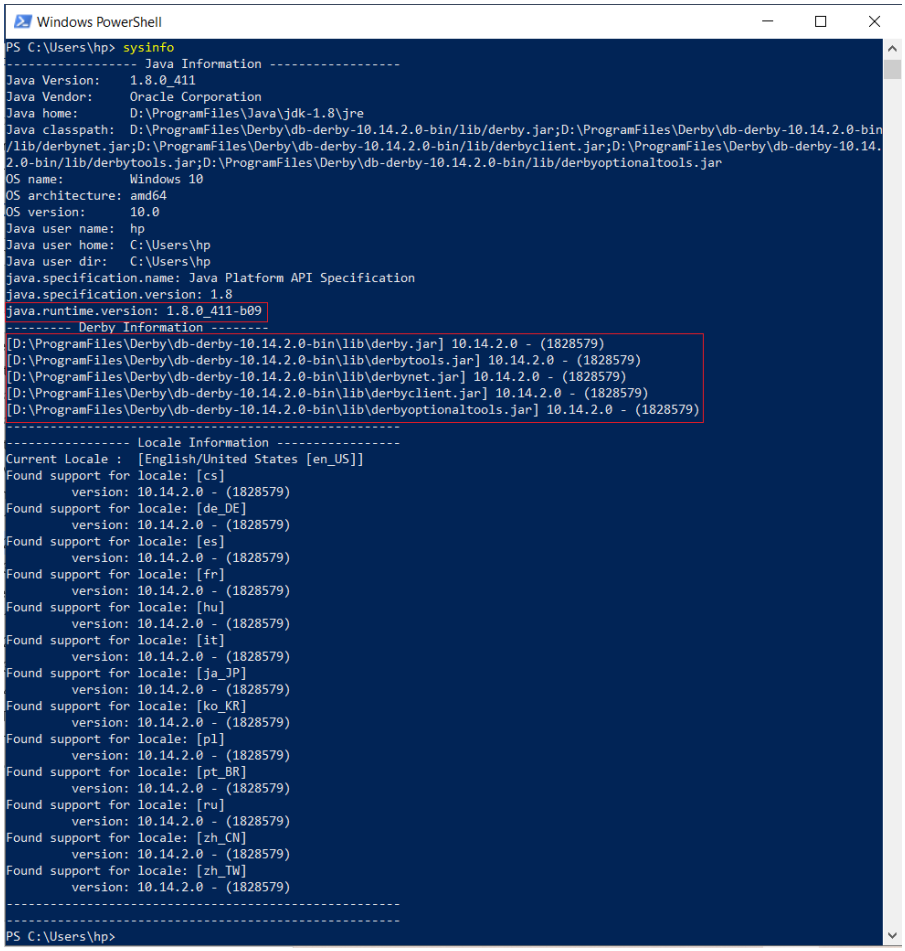


2.4. Verify Derby System:

Derby provides `sysinfo` script to get the Derby system information. This script is available at `DERBY_HOME\bin` location.

Simply run the following command in **Command Prompt** or **Windows PowerShell**.

```
sysinfo
```



3. Derby Modes:

Derby can be used in **Embedded Mode** and **Server Mode** (also called **Network Mode**)**.**

- In embedded mode, Derby runs within the JVM of the application in which case only that application can access the Derby database while other applications cannot access it.

- In server mode, Derby Network server is started and it is responsible for handling multiple database requests for different applications.

3.1. Embedded Mode:

When an application accesses the Derby database using the Embedded JDBC driver, the Derby engine does not run on a separate process but runs inside the JVM of application.

In embedded mode, only one JVM can boot the database at a time and multiple applications running on different JVMs cannot access the same database at once. When one application booted the database and other application tries to access it, Derby gives us the following error: Another instance of Derby may have already booted the database.

3.1.1. Configure Embedded Derby:

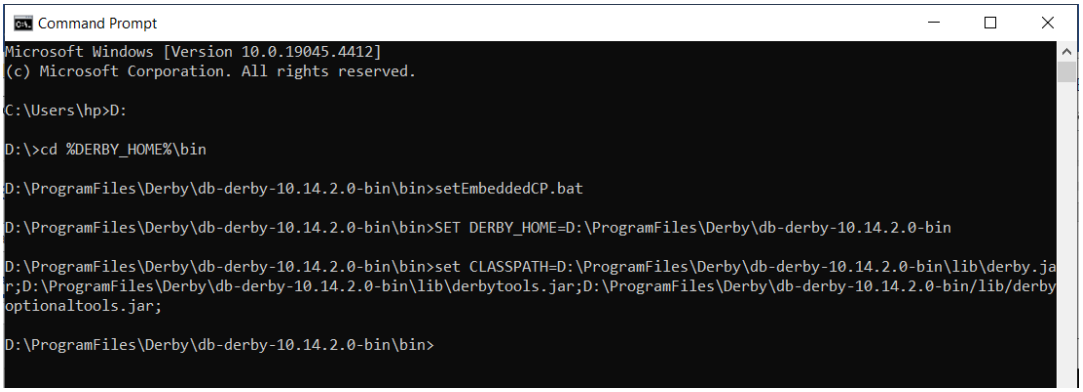
To use Derby in Embedded mode, first include the following jar files in the `CLASSPATH` variable.

- `derby.jar`: This contains the Derby engine and the Derby Embedded JDBC driver.
- `derbytools.jar`: This is optional, it provides the **ij** tool.

Derby provides `setEmbeddedCP.bat` script to set the `CLASSPATH` variable for embedded usage.

Open **Command Prompt**, navigate to Derby installation directory and run `setEmbeddedCP.bat` using the following commands

```
cd %DERBY_HOME%\bin
setEmbeddedCP.bat
```



3.1.2. Run Sample App:

Let us run the sample application `SampleApp.java` provided by Derby and is located in the `DERBY_HOME\demo\programs\simple` directory. By default, this application runs in Embedded mode.

The `SimpleApp.java` application performs the following operations:

- Load Derby Embedded JDBC driver and start the Derby engine using the following code:

```
public String driver = "org.apache.derby.jdbc.EmbeddedDriver";
Class.forName(driver).newInstance();
```

- Get an embedded connection by creating and connecting to `derbyDB` database using the following code:

```
public String protocol = "jdbc:derby:";
conn = DriverManager.getConnection(protocol + "derbyDB;create=true", props);
```

  The fully constructed embedded Derby JDBC URL would be `jdbc:derby:derbyDB;create=true`

- Create a table using the following code:

```
s = conn.createStatement();
s.execute("create table location(num int, addr varchar(40))");
```

- Insert data to table using the following code:

```
psInsert = conn.prepareStatement("insert into location values (?, ?)");
psInsert.setInt(1, 1956);
psInsert.setString(2, "Webster St.");
psInsert.executeUpdate();
```

- Update data in table using the following code:

```
psUpdate = conn.prepareStatement("update location set num=?, addr=? where
num=?");
psUpdate.setInt(1, 180);
psUpdate.setString(2, "Grand Ave.");
psUpdate.setInt(3, 1956);
psUpdate.executeUpdate();
```

- Select data from table by executing the following query:

```
rs = s.executeQuery("SELECT num, addr FROM location ORDER BY num");
```

- Drop the table using the following code:

```
s.execute("drop table location");
```

- Commit transaction using the following code:

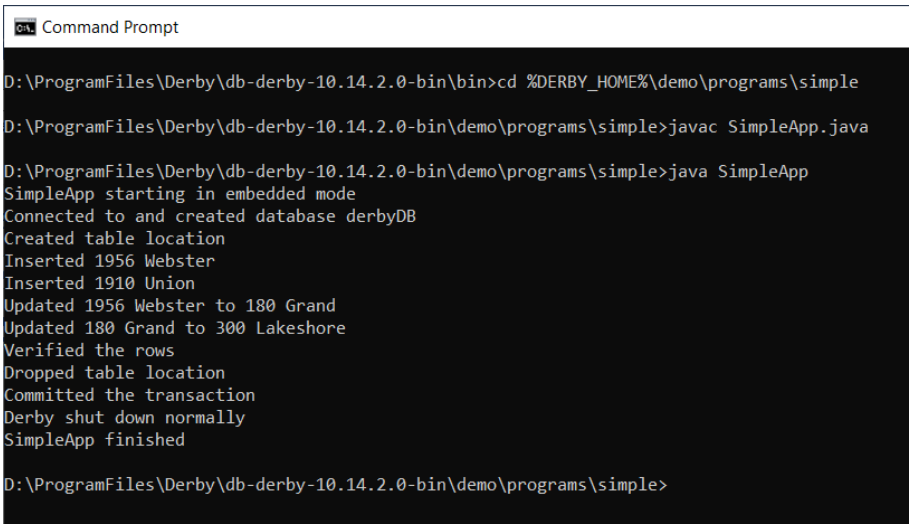
```
conn.commit();
```

- Shut down the Derby database to release resources. In embedded mode, the application should shut down the Derby database. If the database is not shutdown properly, Derby cannot perform the checkpoint when the JVM shuts down which makes longer time to reboot the database next time since Derby needs to perform recovery operation.

```
// Shutdown a specific database "MyDbTest"
DriverManager.getConnection("jdbc:derby:derbyDB;shutdown=true");
```

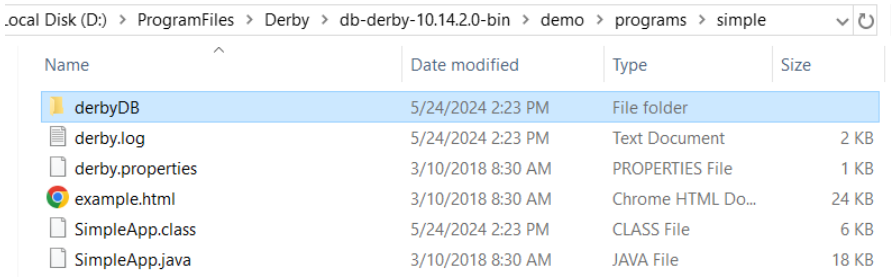
Now, go to `DERBY_HOME\demo\programs\simple` directory, compile `SimpleApp.java` application and run it using the following commands

```
cd %DERBY_HOME%\demo\programs\simple
javac SimpleApp.java
java SimpleApp
```



In case of any error thrown by the application, make sure `CLASSPATH` variable is set properly before running the application.

Note: After the successful execution of above application, it creates a `derbyDB` database in the location where ever `SimpleApp.java` got executed.



3.2. Network Mode:

Derby Network provides a framework that embeds Derby and handles database connections even from applications running in remote machines through network client.

We will see how to run a Sample application by connecting Derby in network mode.

3.2.1. Configure Network Derby:

To use Derby in Network mode, first include the following jar files in the `CLASSPATH` variable.

- `derbynet.jar`: This contains the Derby network server and reference to Derby engine jar file (`derby.jar`).
- `derbyclient.jar`: This contains the JDBC driver for Derby engine.
- `derbytools.jar`: This is optional, it provides the Derby tools.

Derby provides `setNetworkServerCP.bat` script to set the `CLASSPATH` variable for network usage.

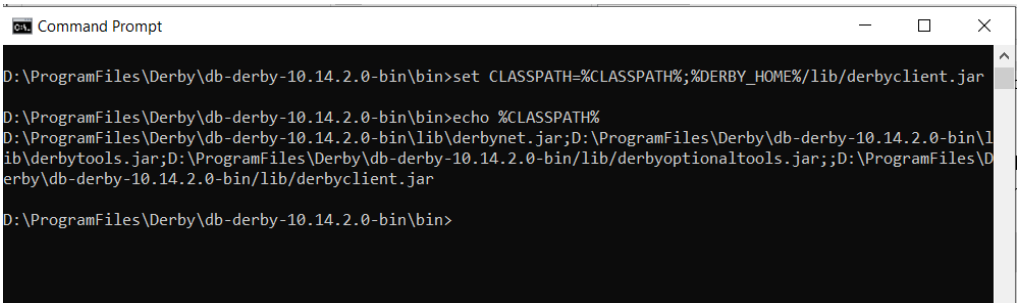
Open a new **Command Prompt**, navigate to Derby installation directory and run `setNetworkServerCP.bat` using the following commands

```
cd %DERBY_HOME%\bin
setNetworkServerCP.bat
```

Note that `setNetworkServerCP.bat` script does not set CLASSPATH variable to `derbyclient.jar` which is mandatory for Derby to know the JDBC driver to use.

Hence, we need to set `derbyclient.jar` to existing CLASSPATH variable manually using the following command.

```
set CLASSPATH=%CLASSPATH%;%DERBY_HOME%/lib/derbyclient.jar
```



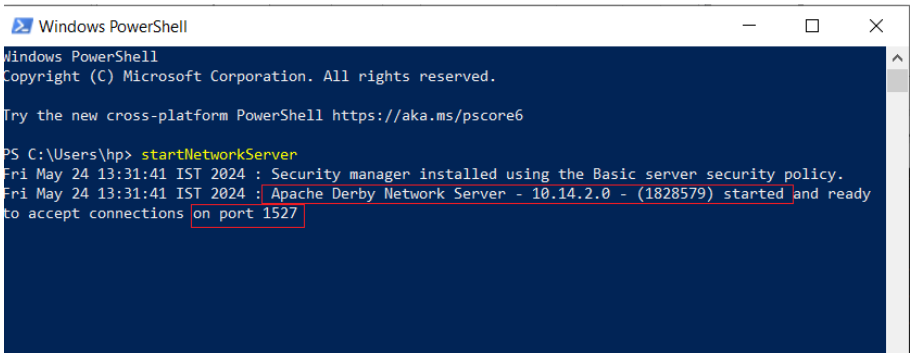
3.2.2. Start NetworkServer:

Next, start the Derby NetworkServer to accept multiple client connections.

There are multiple ways to start NetworkServer based on the need.

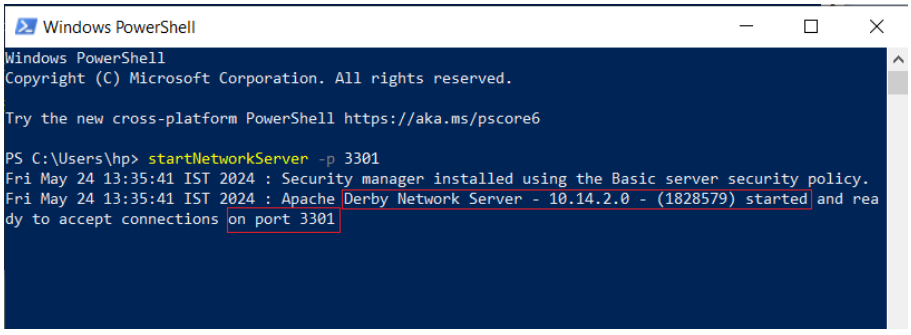
- **Start NetworkServer on Default Port:**
  Run the following command in **Command Prompt** or **Windows PowerShell** to start the Derby NetworkServer which runs on default port 1527.

```
startNetworkServer
```



- **Start NetworkServer on Custom Port:**
  To start the NetworkServer on a different port, use `-p` option as below

```
startNetworkServer -p 3301
```



Note that Derby NetworkServer accepts multiple connections from the localhost only by default.

- **Start NetworkServer to Accept Different Host Connections:**
  To make the Derby NetworkServer taking multiple connections from a specific host, provide the specific host IP address using `-h` option in the `StartNetworkServer` command. The NetworkServer will then accept connections from another host onlyl as the localhost.

```
startNetworkServer -h IP_of_host_name
```

- **Start NetworkServer to Accept all Host Connections:**
  If NetworkServer is required to accept connections from localhost as well as from any other server, then start it using the following command.

```
startNetworkServer -h 0.0.0.0
```

3.2.3. Run Sample App:

We will use the same example application `SampleApp.java` from `%DERBY_HOME%\demo\programs\simple` directory to run Derby in Network mode.

By default, `SampleApp.java` application runs in embedded mode but when we pass the `derbyclient` argument at run time, it will create and connect to the database using the Derby Network Client JDBC driver instead.

This `SimpleApp.java` application performs the following operations when executed with `derbyclient` argument:

- Load the Client JDBC driver and start the Network Derby engine using the following code:

```
driver = "org.apache.derby.jdbc.ClientDriver";
Class.forName(driver).newInstance();
```

- Get a network connection by creating and connecting to `derbyDB` database using the following code:

```
protocol = "jdbc:derby://localhost:1527/";
conn = DriverManager.getConnection(protocol + "derbyDB;create=true", props);
```

The fully constructed embedded Derby JDBC URL would be `jdbc:derby://localhost:1527/derbyDB;create=true`

To connect to an existing `derby` database located at `D:\ProgramFiles\Derby\db-derby-10.14.2.0-bin\bin`, then use the following JDBC URL `jdbc:derby://localhost:1527/d:\programs\derby\db-derby-10.14.2.0-bin\bin\derbyDB;create=true`

- Create a table using the following code:

```
s = conn.createStatement();
s.execute("create table location(num int, addr varchar(40))");
```

- Insert data using the following code:

```
psInsert = conn.prepareStatement("insert into location values (?, ?)");
psInsert.setInt(1, 1956);
psInsert.setString(2, "Webster St.");
psInsert.executeUpdate();
```

- Update data using the following code:

```
psUpdate = conn.prepareStatement("update location set num=?, addr=? where
num=?");
psUpdate.setInt(1, 180);
psUpdate.setString(2, "Grand Ave.");
psUpdate.setInt(3, 1956);
psUpdate.executeUpdate();
```

- Select data from the table by executing the following query:

```
rs = s.executeQuery("SELECT num, addr FROM location ORDER BY num");
```

- Drop a table using the following code:

```
s.execute("drop table location");
```

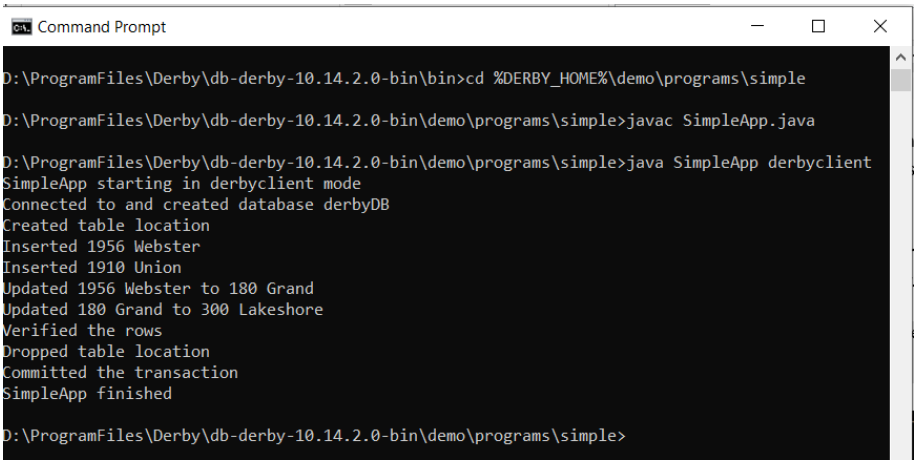
- Commits transaction using the following code:

```
conn.commit();
```

- Do not shut down the Derby database. When connecting via Network Server, we should not shut down the Derby database because other applications might be accessing the same database.

Now, run the following commands in **Command Prompt** to execute `SimpleApp.java` application connected to Network Derby.

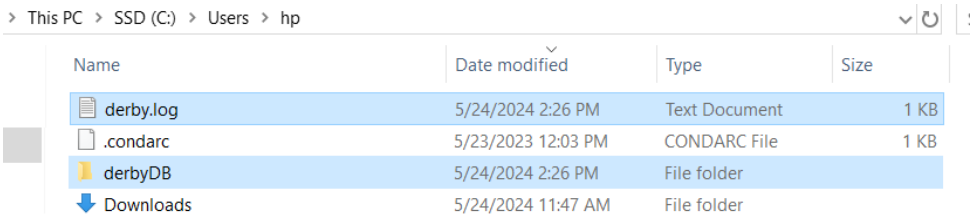
```
cd %DERBY_HOME%\demo\programs\simple
javac SimpleApp.java
java SimpleApp derbyclient
```



In case of any error thrown by the application, make sure the `CLASSPATH` variable is set properly before running the application

Note: After the successful execution of above application, it creates a `derbyDB` database in the location where ever **NetworkServer** is started.

For example, I started NetworkServer from `C:\Users\hp` location and so `derbyDB` database was created here.



3.2.4. Stop NetworkServer:

We can directly stop the NetworkServer using `stopNetworkServer.bat` script available in `DERBY_HOME\bin` location.

```
cd %DERBY_HOME%\bin
stopNetworkServer.bat
```

4. Derby Tools:

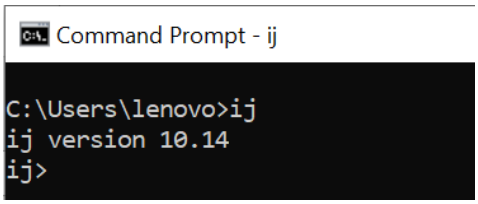
Derby provides various tools such as `ij, dblook` etc

4.1. ij Tool:

ij is Derby's interactive SQL scripting tool that allows to run scripts or interactive queries against the Derby database. This tool can be used with both Derby Embedded JDBC driver and Derby Network Client driver.

To start `ij` tool, open a Command Prompt and run the following command:

```
cd %DERBY_HOME%\bin
ij
```



Now, we will us see how to connect Derby database in embedded mode and server mode from ij tool.

Connect to Embedded Derby:

On the ij tool, execute the following command to connect to `derbyDB` located at `D:\ProgramFiles\Derby\db-derby-10.14.2.0-bin\demo\programs\simple` directory in embedded mode.

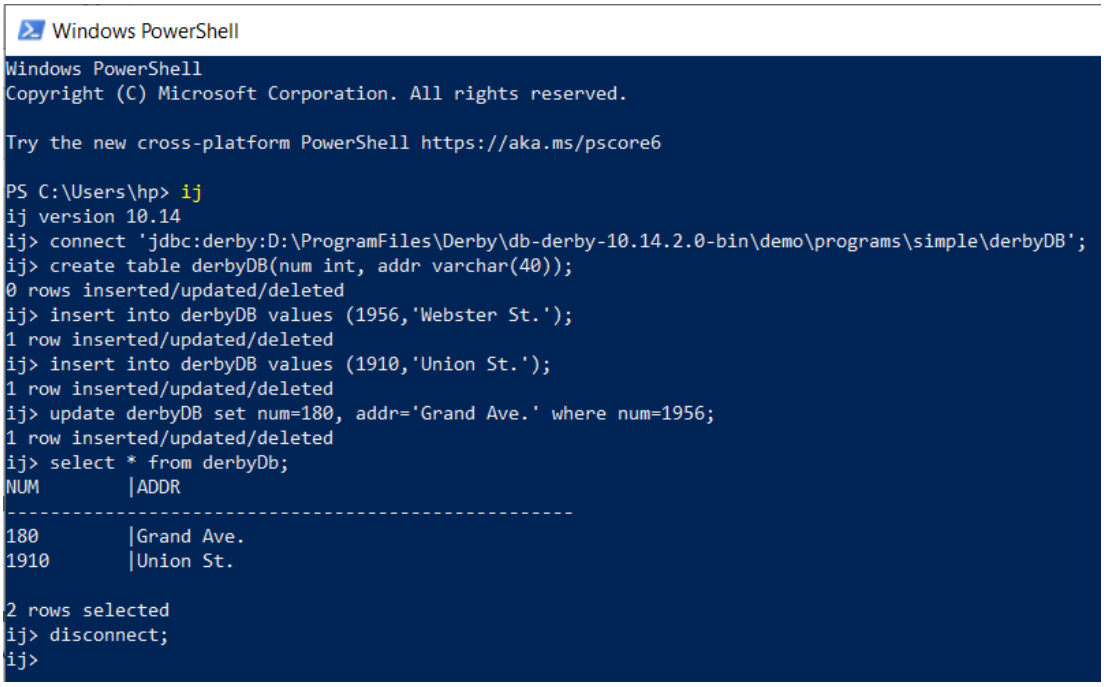
```
connect 'jdbc:derby:D:\ProgramFiles\Derby\db-derby-10.14.2.0-
bin\demo\programs\simple\derbyDB';
```

Once the database is connected we can run the regular SQL queries to create a table, insert data, update data and select data etc.

```
create table derbyDB(num int, addr varchar(40));
insert into derbyDB values (1956,'Webster St.');
insert into derbyDB values (1910,'Union St.');
update derbyDB set num=180, addr='Grand Ave.' where num=1956;
select * from derbyDb;
```

To exit from the current database, use `disconnect` command.

```
disconnect;
```



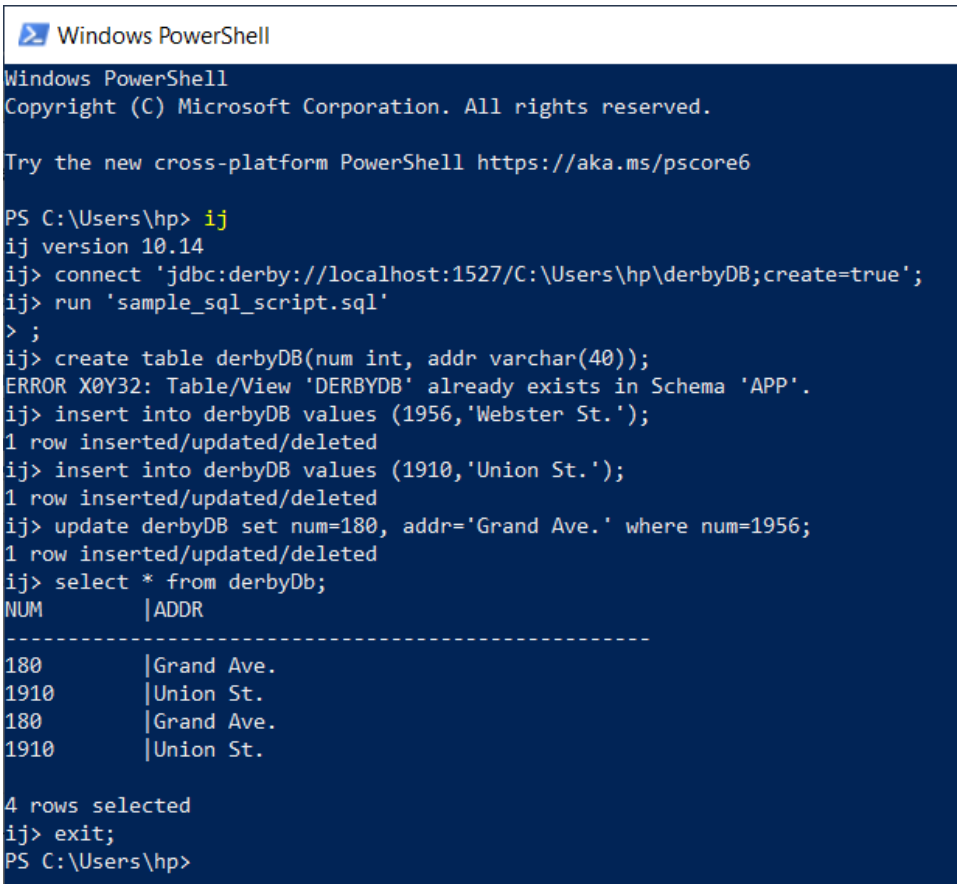
Connect to Network Derby:
On the ij tool, execute the following to command to connect `derbyDB` located at `C:\Users\hp\derbyDB` in network mode. Make sure that NetworkServer is running before connecting in network mode.

```
connect 'jdbc:derby://localhost:1527/C:\Users\hp\derbyDB;create=true';
```

Once the database is connected, we can directly run regular SQL queries or run the SQL script file.

For example, I am running `sample_sql_script.sql` file available in the current directory. We can execute SQL script from ij tool using `run` command.

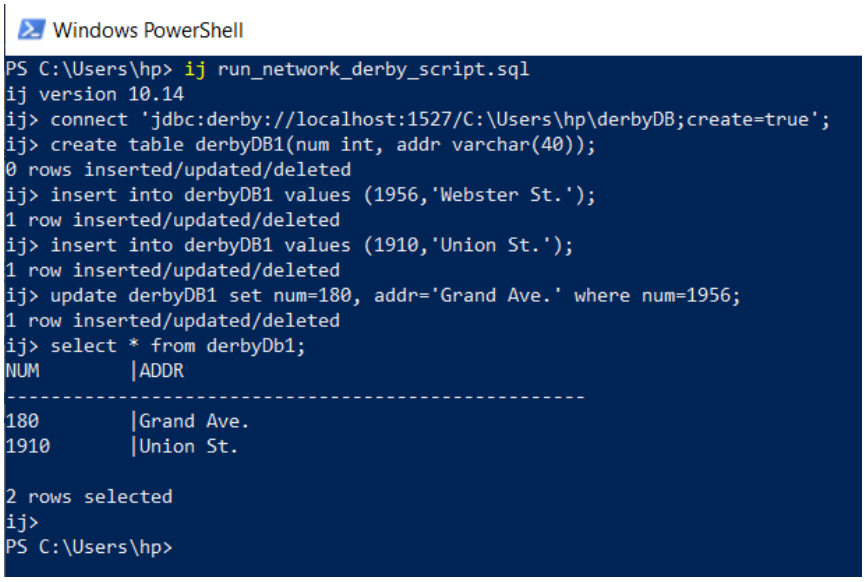
```
run 'sample_sql_script.sql';
```



To come out of ij tool, simple type `exit;` command

We can also execute the SQL script by passing it to ij tool directly

```
ij run_network_derby_script.sql
```



4.2. dblook Tool:

dblook is another Derby's tool that provides the SQL dump of the database schema. This tool can be used with both Derby Embedded JDBC driver and Derby Network Client driver.

`dblook` expects `-d` argument with a database URL that it needs to connect and can have the following options:

`-z <schema name>`: Specify a schema to which the DDL generation should be limited. Only database objects with that schema will have their DDL generated.

`-t <table one> <table two>`: Specify a list of tables for which the DDL will be generated; any tables not in the list will be ignored.

`-td <value>`: Specify what should be appended to the end of each DDL statement. This defaults to ';'.

`-noview`: Prevents the generation of DDL for views.

`-append`: It keeps from overwriting the output files.

`-verbose`: It ensures error messages are printed to the console (in addition to the log file). If not specified, errors will only be printed to the log file.

`-o <filename>`: Specify the file name to which the generated DDL will be written. If not specified, default is the console.

For example, run the following `dblook` command to write the table emp to the file `dblook_emp_ddl.sql`

```
cd %DERBY_HOME%\bin
dblook -d 'jdbc:derby://localhost:1527/C:\Users\hp\derbyDB' -t emp1 -o
C:\Users\hp\dblook_emp_ddl.sql
```