2/27/2017

**Installation Guide**

Sergei Gorianin, Irina Grigoreva, Walid Rhazadi, Oualid Manaï

big data project team

**Installation guide**

**Contents**

[1. Install and configure software 2](#_Toc476835392)

[1.1. Set up programs for remote work with Oracle Big Data Appliace (server) 2](#_Toc476835393)

[1.2. Configure installed programs 3](#_Toc476835394)

[1.2.1. Configure PuTTY 3](#_Toc476835395)

[1.2.2. Configure FileZilla Client for transferring files. 6](#_Toc476835396)

[1.2.3. Configure SQL Developer 8](#_Toc476835397)

[2. Introduction to technologies we will use 10](#_Toc476835398)

[3. Install MongoDB to the Server 11](#_Toc476835399)

[4. Import .csv data files to MongoDB 13](#_Toc476835400)

[5. Create MongoDB-Hive connection 14](#_Toc476835401)

[6. Create External MongoDB-Hive tables 15](#_Toc476835402)

[7. Import data files to Oracle NoSQL Database 17](#_Toc476835403)

[8. Create Oracle NoSQL – Hive external tables 18](#_Toc476835404)

[9. Import pollution data files to HDFS. 19](#_Toc476835405)

[10. Create HDFS external tables on Hive 20](#_Toc476835406)

[11. Import .csv data files to Oracle SQL Database 21](#_Toc476835407)

[12. Create Hive external tables on Oracle SQL Database 23](#_Toc476835408)

[13. Testing speed of executing queries 25](#_Toc476835409)

# **1. Install and configure software**

**1.1. Set up programs for remote work with Oracle Big Data Appliace (server)**

First, you need to install PuTTY client (PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.) for set up the remote connection to server.

Link to program: <http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> .

On the next step, you need to install the FTP Client for transferring files. We recommend use FileZilla for this.

Link to program: <https://filezilla-project.org/> .

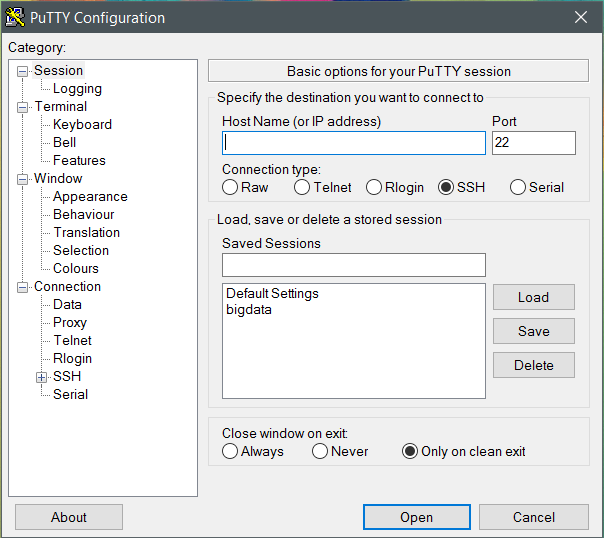
Also, we recommend to install SQL Developer for more comfortable work with Local data (tables) on Server.

Link to program: <http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html> .

**1.2. Configure installed programs**

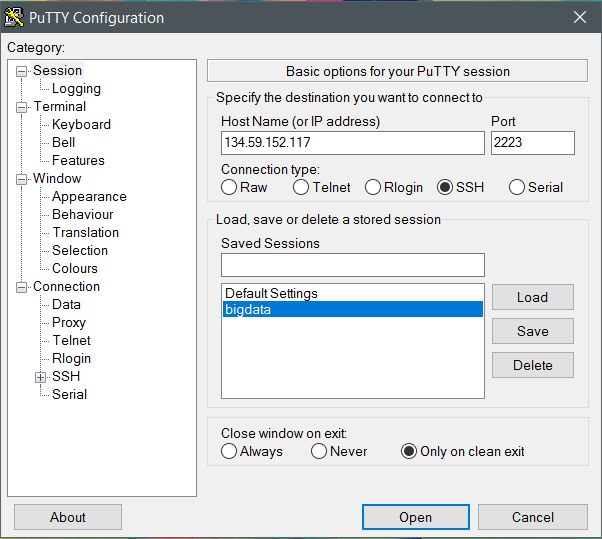
***1.2.1. Configure PuTTY***

1. Open PuTTY Client.

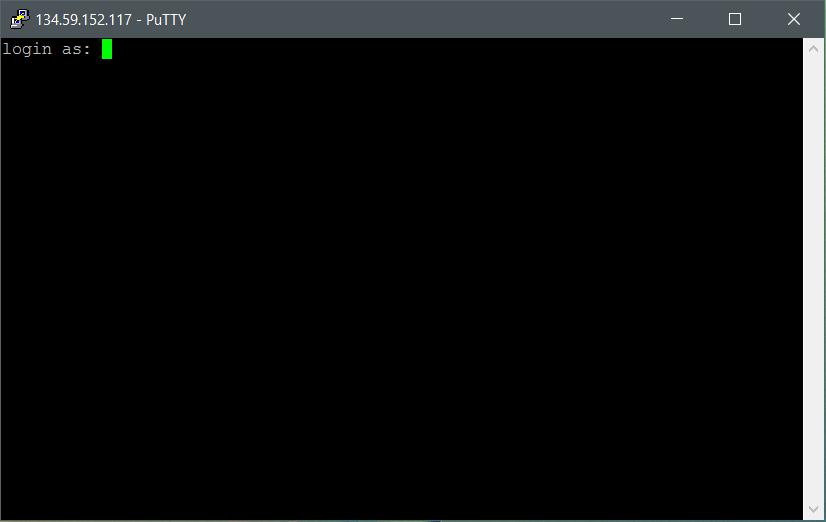


1. Print Host Name or IP address and Port in the same fields to connect to the server.

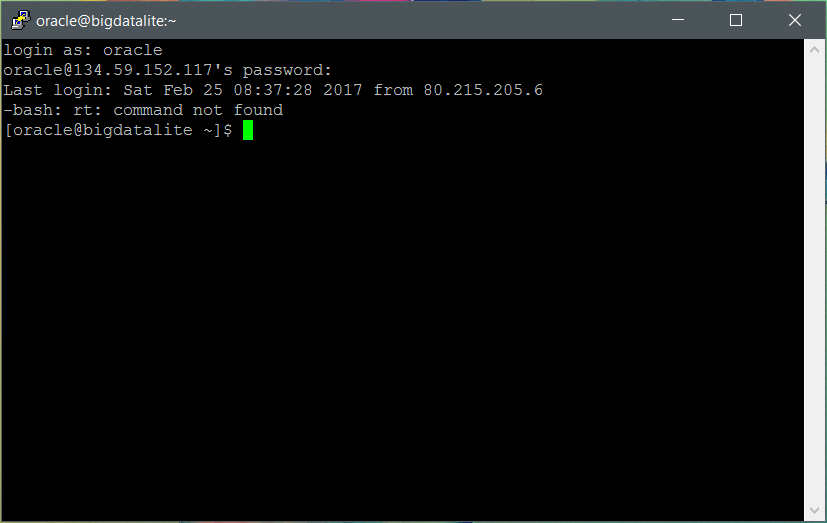
In this case: IP address: 134.59.152.117; Port: 2223.



1. Press Open to connect.

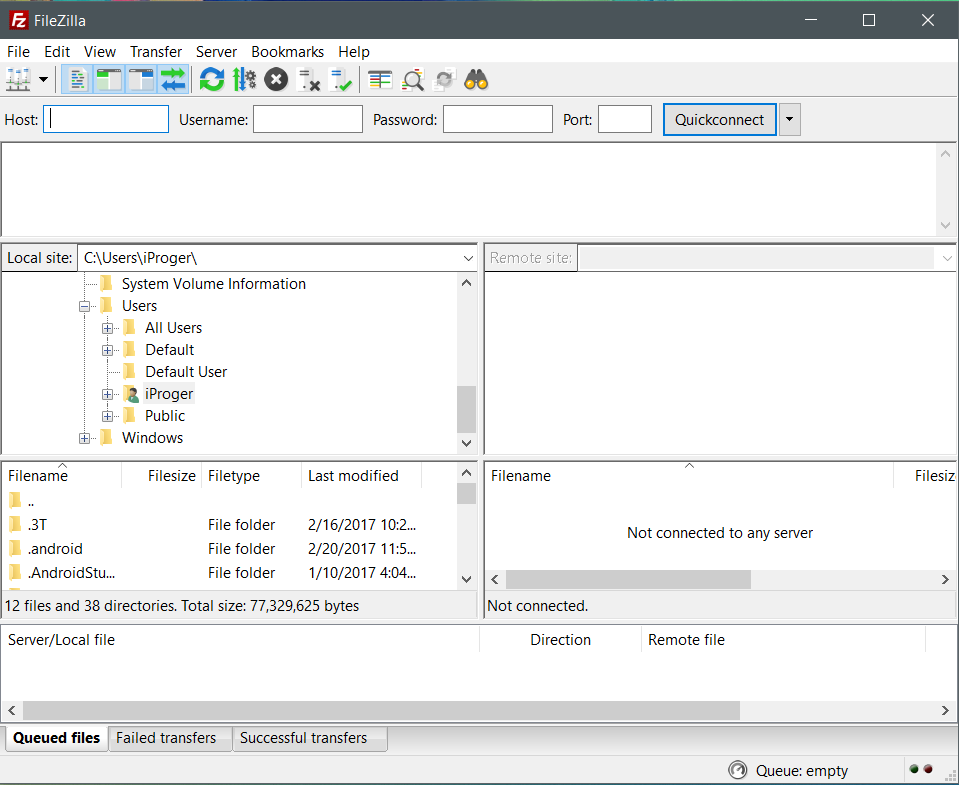


1. For creating connection to server, print login and password. In this case, print ***oracle*** as login, and ***welcome1*** as password.

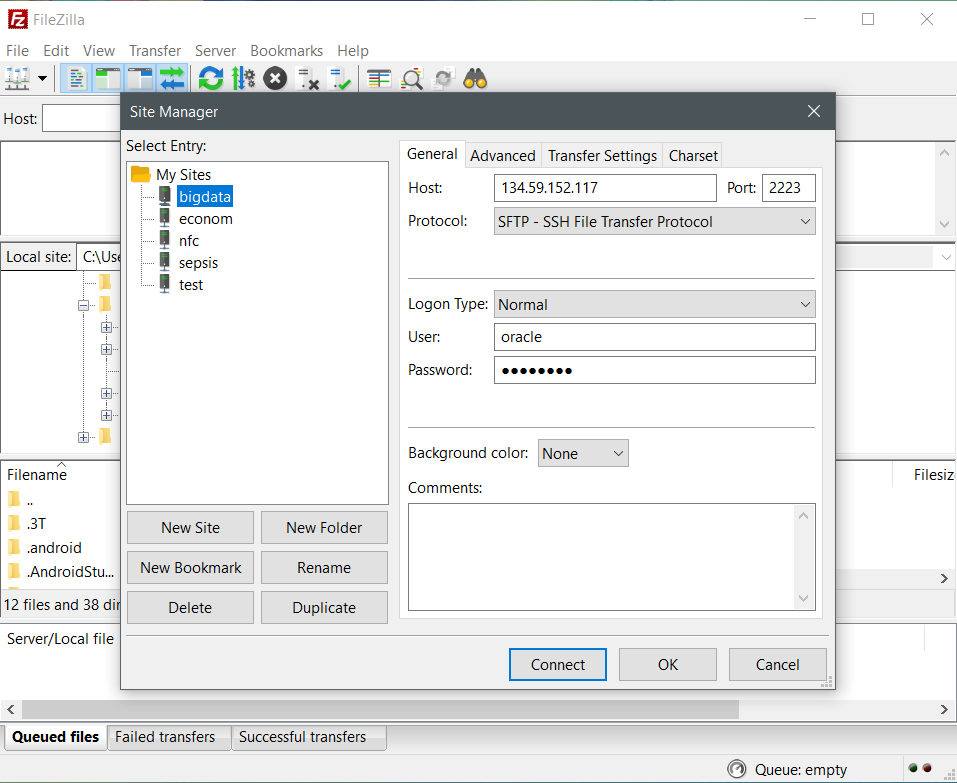


***1.2.2. Configure FileZilla Client for transferring files.***

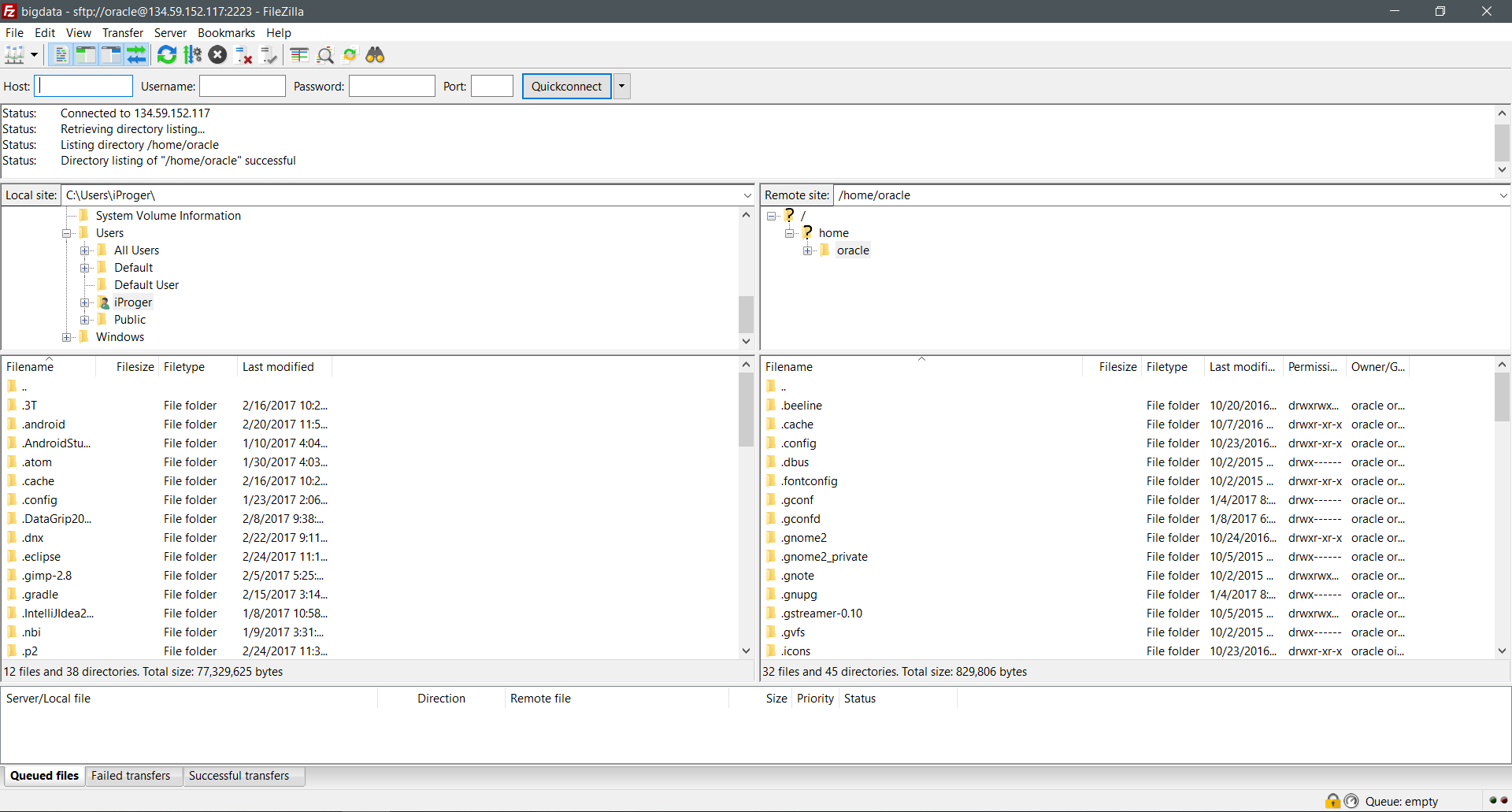
1. Open FileZilla Client.



1. Click File – Site Manager to create new connection.

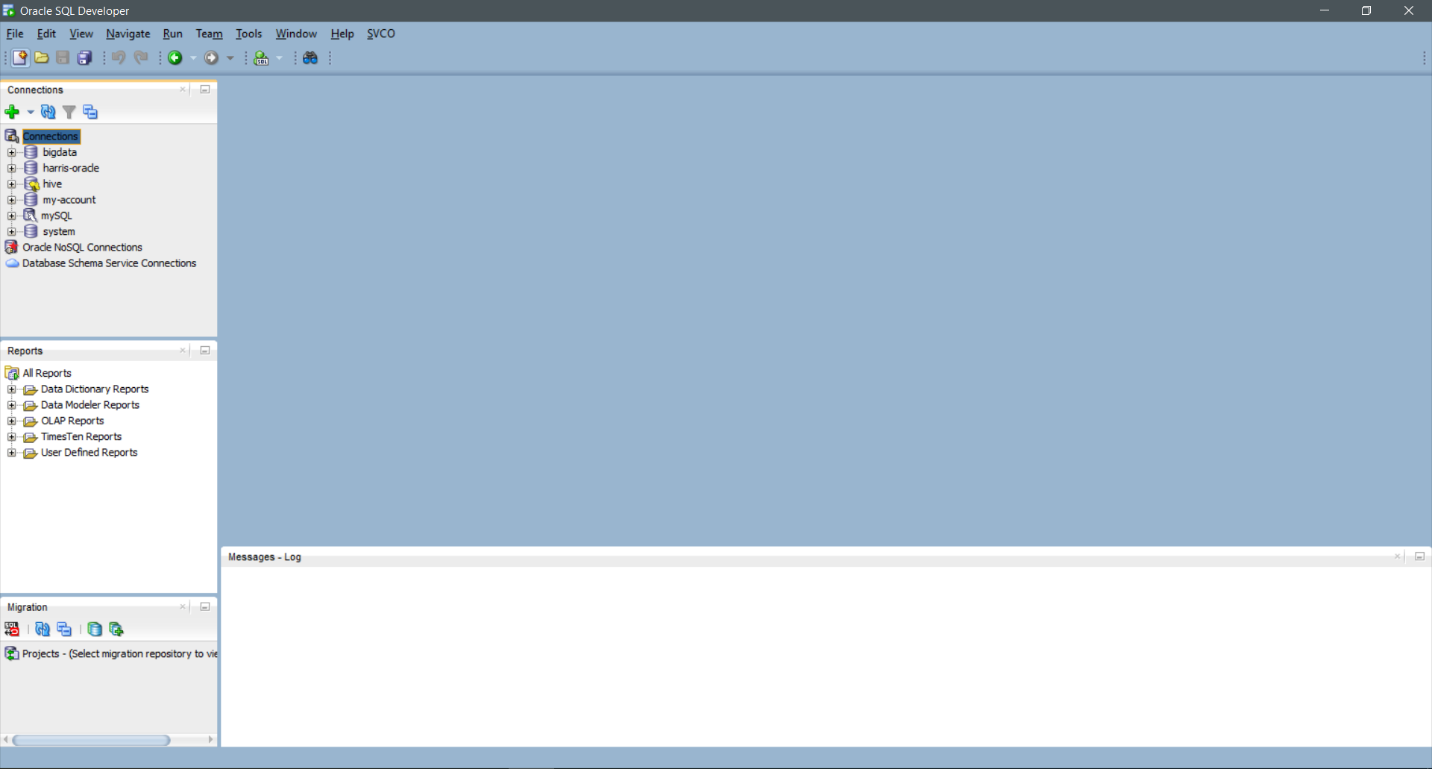


Click “*New Site*” button. Print name of new site, ad fill the connection fields: Host: ***134.59.152.117***; Port: ***2223***; Logon Type – select ***Normal***; User: ***oracle***; Password: ***welcome1***. Then, click “Connect” button. In the result, you will connect to Server, like on the screenshot below:

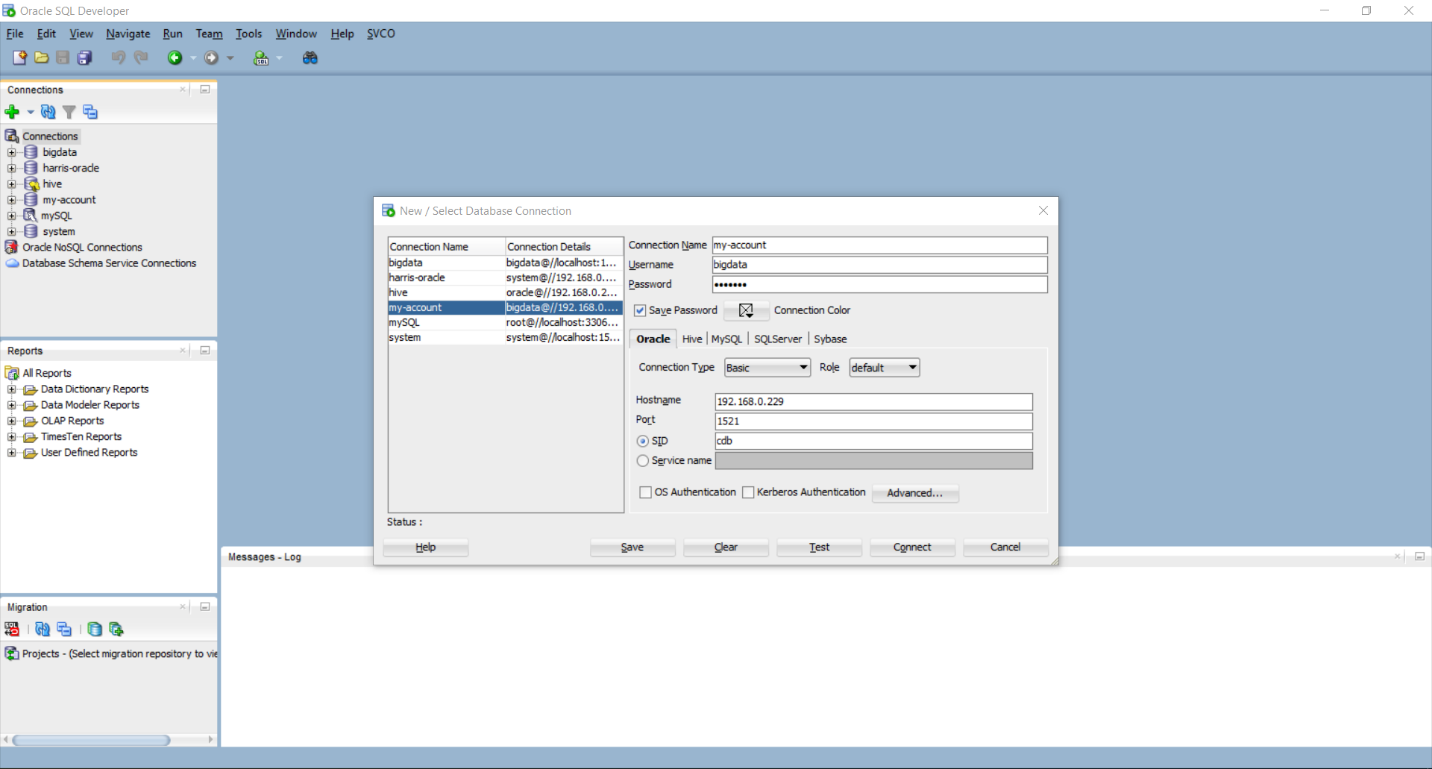


### ***1.2.3. Configure SQL Developer***

1. Start SQL Developer;



1. Right click on “Connections”, then choose “New Connection…”;
2. Fill the fields to create a connection to Oracle Database 12c;



Username: ***bigdata***;

Password: ***bigdata***;

Connection type: ***Basic***;

Role: ***default***;

Hostname: ***192.168.0.229***;

Port: ***1521***;

SID: ***cdb***.

Test the connection and save it.

***You successfully install and configure all programs.***

# **2. Introduction to technologies we will use**

The server contains this list of services, we will be working with:

1. *ORCL* – Oracle Database 12c;
2. *Zookeeper*;
3. *HDFS*;
4. *HBase*;
5. *Hive*;
6. *Hue*;
7. *Impala* server, catalog, state-store;
8. *Kafka* server;
9. *NoSQL* – Oracle NoSQL Database;
10. *Oozie*;
11. *ORDS-Apex* – Oracle Rest Data Service Apex;
12. *Solr* server;
13. *Spatial* bdsg;
14. *Sqoop2* server;
15. *WebLogic-MovieDem*;
16. *YARN* Resource manager, nodemanager.

To show this list just print in PuTTY (while connected to Server) “services”.

Also, we have this list of data:

1. Health data, include .csv files with Asthma, Dyspnea and Common health diseases, like heart diseases.
2. Pollution data, include .csv files with:
3. Air pollution information from 2013 to 2015 years, information for each day;
4. Air pollution information from 2014 to 2016 years, for each day;
5. Air pollution information from 2013 to 2014 years, for each hour.

Because of the Health data will more static and will not update, we will use Oracle SQL Database 12c for storing it in **local tables**. For pollution information, we will use NoSQL database and HDFS. As a “bridge” for all data, we will use Hive.

As NoSQL Database, we will use Oracle NoSQL Database and MongoDB. Both databases will store the same data, but we will look at both type of connections between Hive and Oracle NoSQL Database and Hive and MongoDB.

# **3. Install MongoDB to the Server**

First, configure the package management system (***yum***). Create a ***/etc/yum.repos.d/mongodb-org-3.4.repo*** file to install then MongoDB directly, using yum.

Edit the file you created before, just add this text:

**[mongodb-org-3.4]**

**name=MongoDB Repository**

**baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.4/x86\_64/**

**gpgcheck=1**

**enabled=1**

**gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc**

Save file.

Then, we recommend change the “***bigdata.properties***” files. You can open it on terminal using command below:

**sudo nano /u01/bigdatasql\_config/bigdata.properties**

and add this jar to java.classpath.hadoop variable, like:

**java.classpath.hadoop=/usr/lib/hadoop/client/\*:/usr/lib/hadoop-mapreduce/\*:/usr/lib/hadoop-mapreduce/lib/\**:/home/oracle/bigdataProject/JARs/json-serde-1.3.6-SNAPSHOT-jar-with-dependencies.jar***

The last part is the location of JAR file what need to add. It locates on “**JARs**” directory in our project folder.

Save file and restart services.

Now, install MongoDB, using this command in terminal:

**sudo yum install -y mongodb-org**

To start MongoDB, use the command below:

**sudo service mongod start**

As needed, you can stop MongoDB, printing:

**sudo service mongod stop**

or restart, using command below:

**sudo service mongod restart**

To open MongoDB console, print in terminal command:

**mongo**

# **4. Import .csv data files to MongoDB**

To import .csv data file to MongoDB, in terminal print this command:

**mongoimport -d <database name> -c <collection-name> --type csv --file /path/to/csv/file/filename.csv --headerline**

Using **FileZilla** to transfer bigdataProject folder to default directory. The default path is: ***/home/oracle/bigdataProject***. This folder contains all data files, drivers, libraries.

So, you need to import each pollution .csv file to MongoDB.

Before importing, you can create database for pollution collections. We created the “***bigdataProjectDB***” database in MongoDB, using this command in MongoDB console:

**use bigDataProjectDB;**

Now, you should to close MongoDB console, using command:

**exit;**

Now, import all .csv data file to MongoDB. Print in terminal commands:

**mongoimport -d bigdataProjectDB -c pollution --type csv --file /home/oracle/bigdataProject/csv/pollution/pollution.csv --headerline**

Open MongoDB console again.

Print:

**use bigdataProjectDB;**

and then the command below to view a list of selected database collections:

**show collections;**

# **5. Create MongoDB-Hive connection**

In terminal print this command:

**hive**

to open hive console.

Next, give access to Hive to all needed drivers.

By default, any table created in Hive is HDFS-based; that is, the metadata and underlying rows of data associated with the table is stored in HDFS. Mongo-Hadoop now supports the creation of MongoDB-based Hive tables and BSON-based Hive tables. Both MongoDB-based Hive tables and BSON-based Hive tables can be:

* Queried just like HDFS-based Hive tables.
* Combined with HDFS-based Hive tables in joins and sub-queries

In your Hive script, use ADD JAR commands to include these JARs (core, hive, and the Java driver), e.g., ***ADD JAR /path-to/mongo-hadoop-hive-<version>.jar;***.

All files are in our project folder, in “drivers” directory. So, to add files, print in Hive console:

**ADD JAR /home/oracle/bigdataProject/mongo-java-driver-2.13.2.jar;**

**;**

**ADD JAR /home/oracle/bigdataProject/mongo-hadoop-hive-1.3.0.jar;**

**;**

**ADD JAR /home/oracle/bigdataProject/mongo-hadoop-core-1.3.0.jar;**

After that, close hive console, printing:

**exit;**

and restart Hive services, writing in terminal:

**services**

# **6. Create External MongoDB-Hive tables**

To create External tables, open Beeline console, printing in terminal:

**beeline**

Now, connect to database:

**!connect jdbc:hive2://localhost:10000**

and enter the login: ***oracle*** and the password: ***welcome1***.

For out project we created new database, called bigdataprojectdb. In this tutorial we will use it.

To create new database, print in beeline console:

**create database bigdataprojectdb;**

To view a list of all databases, print:

**show databases;**

To select new database for working with it, write:

**use bigdataprojectdb;**

To show a list of tables of selected database, print the command below:

**show tables;**

Now, we will create MongoDB-Hive External tables. Copy these commands to create each table.

**CREATE EXTERNAL TABLE pollution\_mongo(**

**station string,**

**pollutant string,**

**description string,**

**units string,**

**date\_pol string,**

**value\_pol int**

**)**

**ROW FORMAT SERDE**

**'com.mongodb.hadoop.hive.BSONSerDe'**

**STORED BY**

**'com.mongodb.hadoop.hive.MongoStorageHandler'**

**WITH SERDEPROPERTIES (**

**'mongo.columns.mapping'='{"station":"station",**

**"pollutant":"pollutant",**

**"description":"description",**

**"units":"units",**

**"date\_pol":"date\_pol",**

**"value\_pol":"value\_pol"}')**

**TBLPROPERTIES (**

**'mongo.uri'='mongodb://localhost:27017/bigdataProjectDB.pollution');**

Now, you can check tables and work with them, using beehive connection. For example, to show first 5 rows of table “pollution\_mongo”, printing:

**select \* from pollution\_mongo limit 5;**

# **7. Import data files to Oracle NoSQL Database**

First, connect to Oracle NoSQL database. For this, in terminal write:

**java -jar $KVHOME/lib/kvstore.jar runadmin -port 5000 -host bigdatalite.localdomain**

and then connect to store, printing:

**connect store -name kvstore**

Then, create table, using commands below:

**execute 'create table pollution (station string, pollutant string, description string, units string, date\_pol string, value\_pol integer, primary key(shard(date\_pol), station, pollutant, value\_pol))'**

All data stored in .json files. They are in “json” directory in our project folder.

The next step is importing pollution data to created tables. For this, write in console commands:

**put table -name pollution -file /home/oracle/bigdataProject/json/pollution.json**

To check importing data, use command below:

**get table -name pollution;**

**show tables //show the list of all tables from kvstore;**

# **8. Create Oracle NoSQL – Hive external tables**

First, open the Beeline console, using commands, we described before. Connect to database and select our database for next work.

Now, copy these commands to create external tables:

**create external table pollution\_kv (**

**station string,**

**pollutant string,**

**description string,**

**units string,**

**date\_pol string,**

**value\_pol int)**

**stored by 'oracle.kv.hadoop.hive.table.TableStorageHandler'**

**tblproperties**

**('oracle.kv.kvstore'='kvstore',**

**'oracle.kv.host'='bigdatalite.localdomain:5000',**

**'oracle.kv.hadoop.host'='bigdatalite.localdomain/127.0.0.1',**

**'oracle.kv.tableName'='pollution');**

# **9. Import pollution data files to HDFS.**

Also, we can use HDFS for working with data in Hive.

First, we need to create directories for each data file. For this, in terminal print the commands below:

**hdfs dfs -mkdir /pollution**

Then, put the .csv files into these directories. For that, first, you need to go to our project directory in terminal, using command:

**cd /home/oracle/bigdataProject/csv/hdfs**

**“hdfs”** folder contains .csv files for importing to HDFS. Put these files, using commands:

**hdfs dfs -put pollution.csv /pollution**

To check directories, use this command:

**hdfs dfs -ls /directory-name //in our case, it’s /pollution**

# **10. Create HDFS external tables on Hive**

Open Beeline console and connect to database. Choose out database and create external tables for HDFS data, using commands below:

**create external table pollution\_hdfs (**

**station string,**

**pollutant string,**

**description string,**

**units string,**

**date\_pol string,**

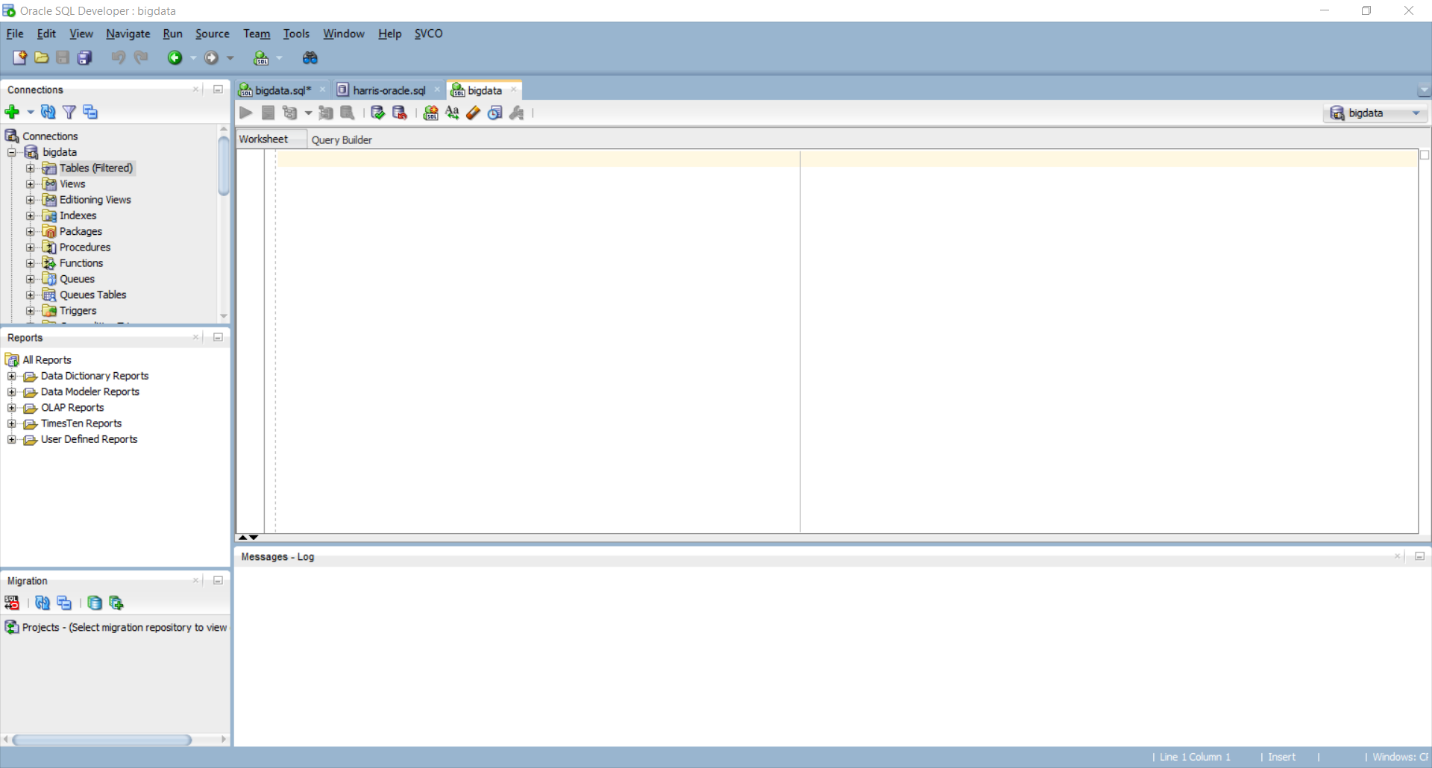
**value\_pol int)**

**row format delimited fields terminated by ','**

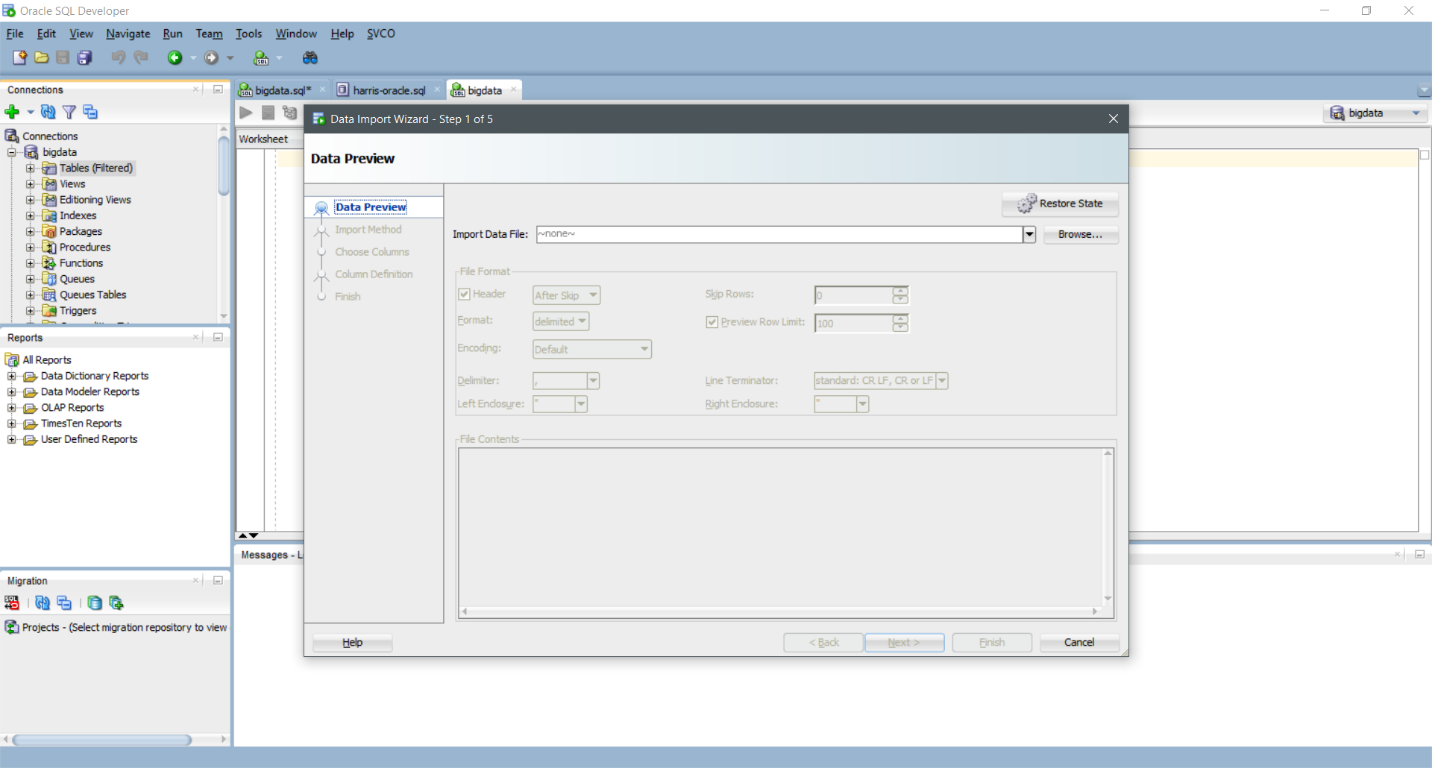
**stored as textfile location 'hdfs:/pollution/';**

# **11. Import .csv data files to Oracle SQL Database**

Open SQL Developer on your PC and connect to Oracle SQL Database. Connection we created at the beginning of this tutorial.



Right click on “**Tables**” field and select “**Import data…**” to open “**Data Import Wizard**”:



Follow these steps in “Data Import Wizard” to import .csv files with health data as *local* *tables* to Oracle SQL Database. Files for importing are in “../csv/health-data/” directory in our project folder. You need to download them, before start importing to Oracle SQL Database.

***\*\*Note: while importing, we recommend use for all fields Data Type – VarChar2, because some numeric fields in .csv files contain “-“ character.***

# **12. Create Hive external tables on Oracle SQL Database**

Start PuTTY Client and open the terminal. Print the command:

**sqlplus**

to start SQL Plus. Then, enter the login: ***bigdata*** and the password: ***bigdata***.

After that, print the command below to connect to database:

**connect bigdata/bigdata**

Now, copy the commands below to create external tables:

**create table pollution\_kv (**

**station varchar2(55),**

**pollutant varchar2(10),**

**description varchar2(55),**

**units varchar2(12),**

**date\_pol varchar2(32),**

**value\_pol numeric(8))**

**organization external (**

**type oracle\_hive**

**default directory oracle\_bigdata\_config**

**access parameters (**

**com.oracle.bigdata.tablename = bigdataprojectdb.pollution\_kv)) reject limit unlimited;**

**create table pollution\_hdfs (**

**station varchar2(55),**

**pollutant varchar2(10),**

**description varchar2(55),**

**units varchar2(12),**

**date\_pol varchar2(32),**

**value\_pol numeric(8))**

**organization external (**

**type oracle\_hive**

**default directory oracle\_bigdata\_config**

**access parameters (**

**com.oracle.bigdata.tablename = bigdataprojectdb.pollution\_hdfs)) reject limit unlimited;**

***\*\*Note: we importing one Oracle NoSQL Table and one HDFS Table to test what data storing method if faster in ours case. So, we import (create external tables) the “pollution\_hdfs” and “pollution\_kv” tables from hive to Oracle SQL Warehouse.***

# **13. Testing speed of executing queries**

So, we have two external tables with pollution data: pollution\_hdfs (HDFS external table) and pollution\_kv (Oracle NoSQL external table). Now, we will be testing the SQL queries time to choose just one data store for pollution data – Oracle NoSQL or HDFS.

We will execute simple query in SQL Developer and set time, using command:

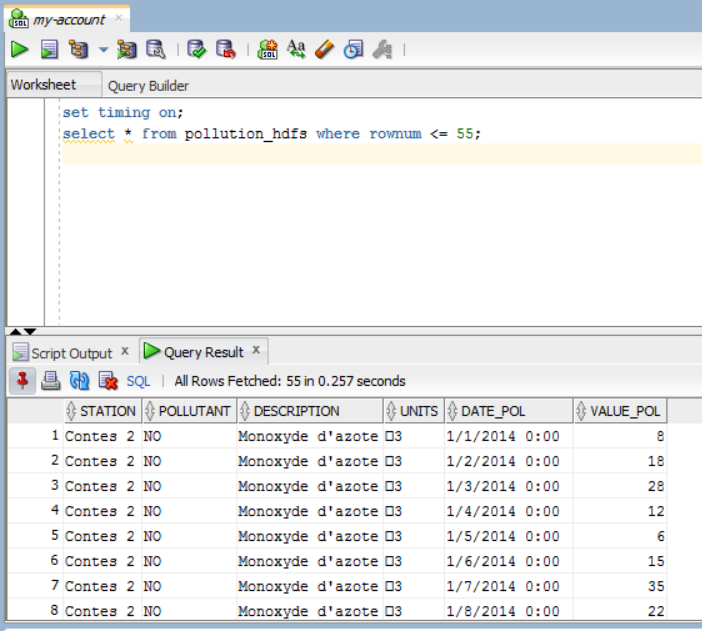
**set timing on;**

The query is:

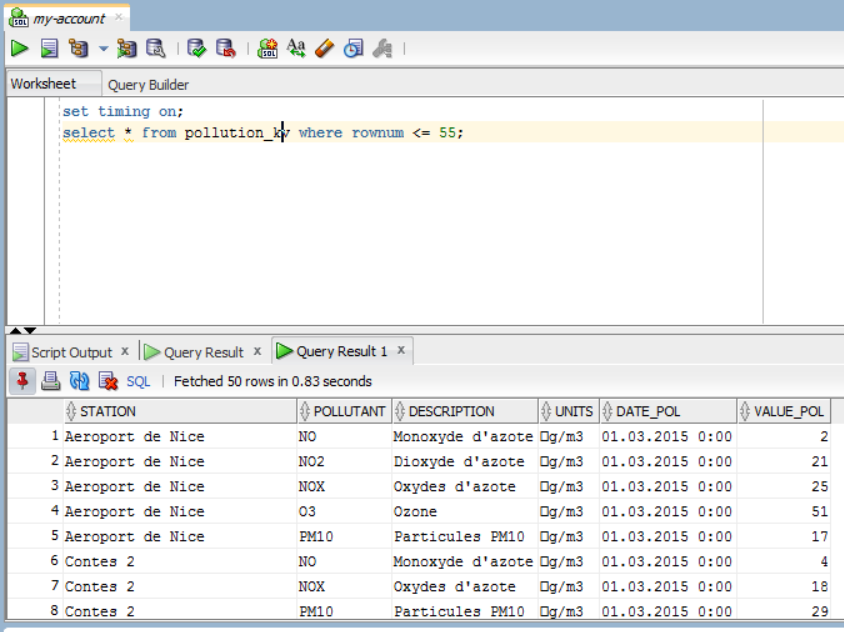
**select \* from <table-name> where rownum <= 55;**

The results of this query for both tables you can see on screenshots below:

**HDFS Result**



**Oracle NoSQL Result**



As you can see, query executes for HDFS in **0.257 secs**. For Oracle NoSQL the result is **0.83 secs**. Further, we will use HDFS external table. So, we will remove Oracle NoSQL external table, using command:

**drop table pollution\_kv;**

For more comfortable work, we recommend to rename table “pollution\_hdfs” to “pollution”, using command:

**alter table pollution\_hdfs rename to pollution;**