A

Project on

Bureau of Labor Statistics Data Federation

By
Archana Jadala
Vidhya Moorthy
Vamshidhar Reddy vemula
Cyriacus Onuigbo

Project Specification

Purpose:To integrate current BLS data into the Systemic Reporting and Analytics Local server utilizing the BLS Application Programming Interface(API) version 2.0
Preferred development language: Any one of C++, Java, PHP. A final product must result in a compiled executable file.
ODS Database authentication: Users should authenticate via their Windows logon (DCCCD network login)

API Authentication: The programmer will need to acquire an API authentication from the service.
Desired BLS Series ID's:
Some of the data desired is still subject to discovery. Final information will be provided by 5/20/2018 to include complete Series ID's.
The first iteration of this project will address Local Area Unemployment Statistics in the Dallas-Fort Worth-Arlington MSA.
See https://www.bls.govnielp/hIpfonna.htm#LA to build Series ID's for this.

Seasonally Adjusted Data is desired. The following Measures are desired:

Measure

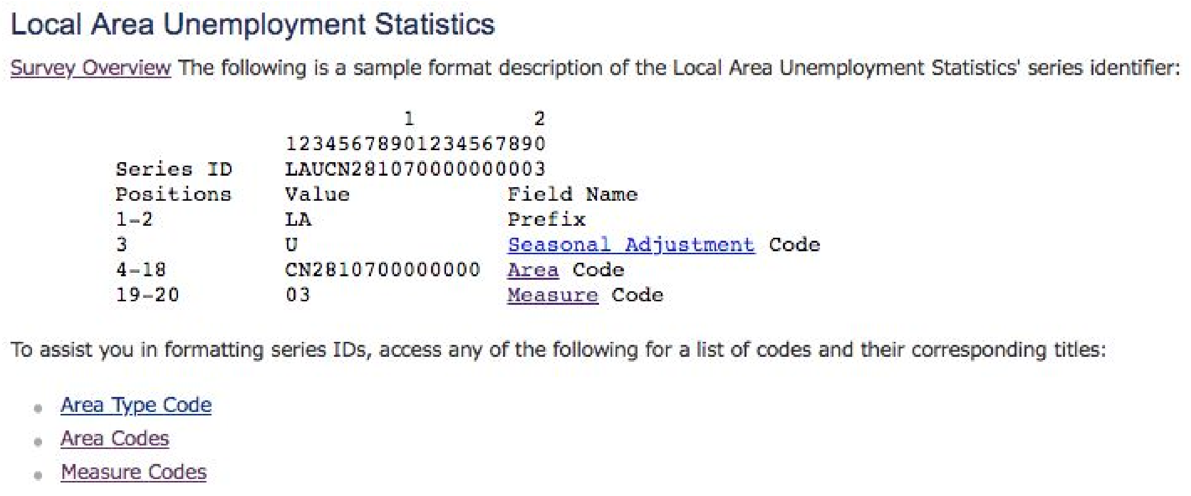
Indicates the unit of measurement:

· 06 Labor force,
· 05 Employment,
· 04 Unemployment, and
· 03 Unemployment rate.

Deliverables: A Windows application that allows any user (with appropriate database credentials) to select from amongst predetermined data sets (identified by a 'Series ID' and Year), receive a file from the API in JSON format and refresh tables within the Azure SQL ODS. The application should authenticate to the BLS API, retrieve the data requested and place it in a table in the BSDW 2 database. Both Source Code, and an executable file will be included. Code should be commented for clarity. The API authentication information should also be submitted.

Due Date: Due date is subject to negotiation and agreement. It is acknowledged that commencement of work is contingent on some need to learn API/REST programming.

The Project is all about occupation data from the Bureau of Labour Statistics or BLS. In this Project we're going to deal specifically about occupation employment statistics. You can get to the BLS website a couple of different ways using Google. The first iteration of this project will address Local Area Unemployment Statistics in the Dallas-Fort Worth-Arlington MSA. This link(https://www.bls.govnielp/hIpfonna.htm#LA) is a sample format description of the Local Area Unemployment Statistics series. The following is the screenshot of the link where there is the description of Series ID, list of codes and their corresponding titles.



Local Area Unemployment Statistics

Survey Overview The following is a sample format description of the Local Area Unemployment Statistics' series identifier:

```
                          1         2
                 12345678901234567890
Series ID        LAUCN281070000000003
Positions        Value                Field Name
1-2              LA                   Prefix
3                U                    Seasonal Adjustment Code
4-18             CN2810700000000      Area Code
19-20            03                   Measure Code
```

To assist you in formatting series IDs, access any of the following for a list of codes and their corresponding titles:

- Area Type Code
- Area Codes
- Measure Codes

The following measures are used in the project : 06 Labor force, 05 Employment, 04 Unemployment, and 03 Unemployment rate.

Software Requirements:

Programming Language: Java, SQL

Tools: NetBeans IDE 8.2, MySQL Workbench 8.0.11 (https://dev.mysql.com/downloads/mysql/)

Libraries used:

1. Commons-logging-1.2.jar (https://commons.apache.org/proper/commons-logging/download_logging.cgi)
2. Httpclient-4.5.5.jar (https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient/4.5.5)
3. Httpcore-4.4.9.jar (https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient/4.5.5)
4. Json-simple-1.1.jar (https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple/1.1.1)
5. Mysql-connector-java-8.0.12.jar (https://dev.mysql.com/downloads/connector/j/)

Our Project is a Windows application that allows any user (with appropriate database credentials) to select from amongst predetermined data sets (identified by a 'Series ID' and Year), receive a file from the API in JSON format and refresh tables within the MySQL Workbench.

Programming: The code is divided into three parts as follows

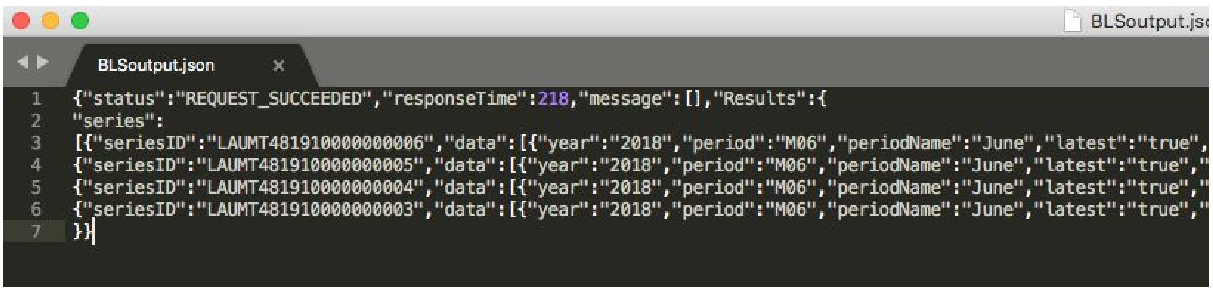
1. BLS Data Extractor
2. BLS Data Parser
3. Establishing connection to MYSQL Workbench

Part 1: **BLS Data Extractor** - Implemented using java where the data is extracted from the BLS content provider for predetermined Series ID's using the following code and a file is created (BLSoutput.json) and the extracted data is stored in this JSON file.

Code:

```
System.out.println("- Data Extraction Started from
https://api.bls.gov/publicAPI/v2/timeseries/data/ at "
+startCalendar.getTime() );

            HttpClient httpClient = new DefaultHttpClient();

            HttpPost httpPost = new
HttpPost("https://api.bls.gov/publicAPI/v2/timeseries/data/");

            String SeriesID1 = "LAUMT481910000000006";

            String SeriesID2 = "LAUMT481910000000005";

            String SeriesID3 = "LAUMT481910000000004";

            String SeriesID4 = "LAUMT481910000000003";

            StringEntity input = new StringEntity

("{\"seriesid\":[\"LAUMT481910000000006\",\"LAUMT481910000000005\",\"LAUMT4
81910000000004\",\"LAUMT481910000000003\"]}");

            input.setContentType("application/json");

            httpPost.setEntity(input);

            HttpResponse response = httpClient.execute(httpPost);

            HttpEntity entity = response.getEntity();

            String responseString = EntityUtils.toString(entity, "UTF-8");
            PrintWriter outputfile = new PrintWriter("BLSoutput.json");

            outputfile.print(responseString);

            outputfile.close();
```

Extracted Data is in this format:



```
{"status":"REQUEST_SUCCEEDED","responseTime":218,"message":[],"Results":{
"series":
[{"seriesID":"LAUMT481910000000006","data":[{"year":"2018","period":"M06","periodName":"June","latest":"true",
{"seriesID":"LAUMT481910000000005","data":[{"year":"2018","period":"M06","periodName":"June","latest":"true","
{"seriesID":"LAUMT481910000000004","data":[{"year":"2018","period":"M06","periodName":"June","latest":"true","
{"seriesID":"LAUMT481910000000003","data":[{"year":"2018","period":"M06","periodName":"June","latest":"true","
}}
```

Part 2: BLS Data Parser: The extracted data from the BLS content provider is stored in json file and this file is given as input to the BLS Data Parser where the extracted data is parsed according to the category. A file reader is used to read the input file and an object is created using a constructor (parse) of JSONParser class named obj.

JSON (JavaScript Object Notation) is a lightweight, text-based, language-independent data exchange format that is easy for humans and machines to read and write. JSON can represent two structured types: **objects and arrays**. An object is an unordered collection of zero or more name/value pairs. An array is an ordered sequence of zero or more values. The values can be strings, numbers, booleans, null, and these two structured types.

JSON Processing in Java : The Java API for JSON Processing JSON.simple is a simple Java library that allow parse, generate, transform, and query JSON. You need to download the json-simple-1.1 jar and put it in your CLASSPATH before compiling and running the code.

JSON-Simple API : It provides object models for JSON object and array structures. These JSON structures are represented as object models using types JSONObject and JSONArray. JSONObject provides a Map view to access the unordered collection of zero or more name/value pairs from the model. Similarly, JSONArray provides a List view to access the ordered sequence of zero or more values from the model.

JSON Array: [{"seriesID":"LAUMT481910000000006"}, {"year":"2018"}]

JSON Object: {"status":"REQUEST_SUCCEEDED"}

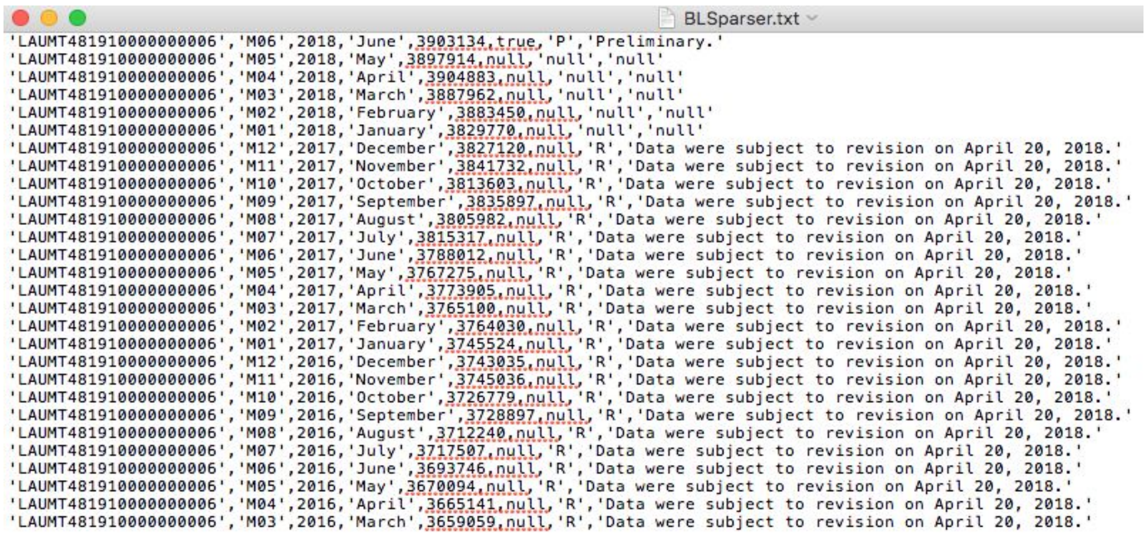
Thus we created two different JSON objects for parsing the data JSON Array and JSON Object. The parsed data is stored in a text file named “BLSparser.txt”.

Code:

```
Object obj = new JSONParser().parse(new FileReader("BLSoutput.json"));

            PrintWriter outputfile = new PrintWriter("BLSparser.txt");

            // typecasting obj to JSONObject

            JSONObject jo = (JSONObject) obj;

            // getting status

            //System.out.println("BLS Data Parsed Output");

            String status = (String) jo.get("status");

            System.out.println("-- Request status:" + status);

            // getting responseTime

            long responseTime = (long) jo.get("responseTime");

            System.out.println("-- Request responseTime:"+ responseTime);

            // getting message

            JSONArray message =  (JSONArray)jo.get("message");

            //System.out.println("-- Message:"+ message.get(0));

            // Creating JSON object for Results

            JSONObject Results =  (JSONObject)jo.get("Results");

            // Creating JSON array object for Series

            JSONArray series = (JSONArray) Results.get("series");

             Iterator itr = series.iterator();

            while (itr.hasNext())

            {

                Object ItrSeriesObject = itr.next();

                JSONObject jsonObject2 = (JSONObject) ItrSeriesObject;

                String seriesID = (String)jsonObject2.get("seriesID");
                System.out.println("\t\tseriesID:" + seriesID);

                String level0 = "'"+seriesID+"'";

                JSONArray data = (JSONArray)jsonObject2.get("data");

                Iterator itrData = data.iterator();

                while (itrData.hasNext())

                {

                    Object dataIterObject = itrData.next();

                    JSONObject jsonObject3 = (JSONObject) dataIterObject;


                    String period = (String)jsonObject3.get("period");

                    //System.out.println("\t\tperiod:" + period);


                    String year = (String)jsonObject3.get("year");

                    //System.out.println("\t\tyear:" + year);


                    String periodName =
(String)jsonObject3.get("periodName");

                    //System.out.println("\t\tperiodName:" + periodName);


                    String value = (String)jsonObject3.get("value");

                    //System.out.println("\t\tvalue:" + value);


                    JSONArray footnotes =
(JSONArray)jsonObject3.get("footnotes");

                    //System.out.println("data:" + footnotes);
                    String latest = (String)jsonObject3.get("latest");

                    //System.out.println("\t\tlatest:" + latest);



                    String level1 =
"'"+period+"',"+year+",'"+periodName+"',"+value+","+latest;



                    Iterator itrFootNotes = footnotes.iterator();



                    while(itrFootNotes.hasNext())

                    {

                        Object footNotesIterObject = itrFootNotes.next();

                        JSONObject jsonObject4 =
(JSONObject)footNotesIterObject;



                        String code = (String)jsonObject4.get("code");

                        //System.out.println("\t\t\tcode:" + code);



                        String text = (String)jsonObject4.get("text");

                        //System.out.println("\t\t\ttext:" + text);



                        String level2 =
(level0+","+level1+",'"+code+"','"+text+"'");



                        outputfile.println(level2);



                        String sql = "INSERT INTO seriesData VALUES " + "("
+ level2 + ")";
```

```
                                stmt.executeUpdate(sql);

                        }

                }
```

BLS Parsed Data:

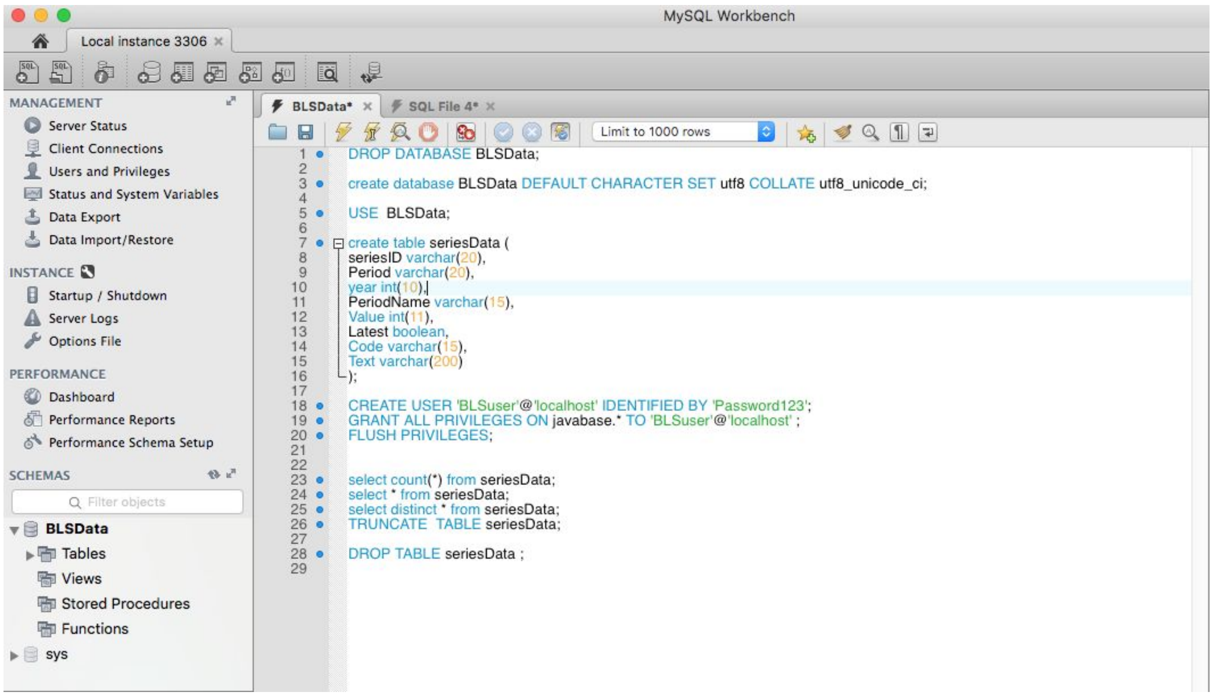


BLSparser.txt

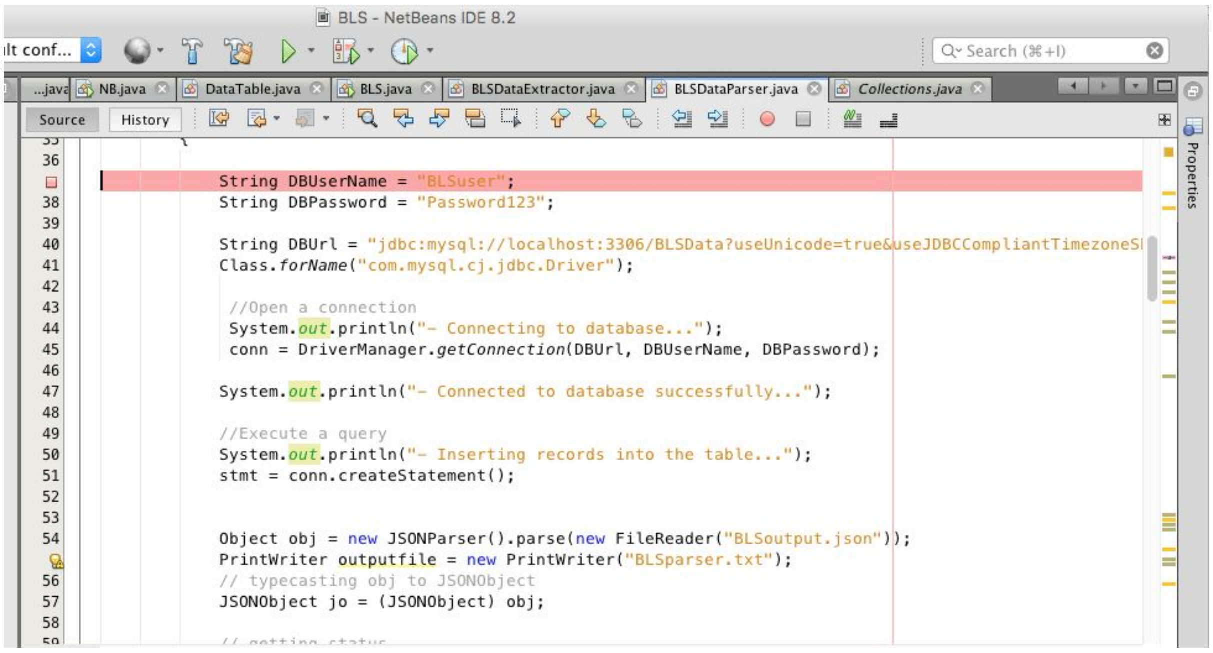
```
'LAUMT481910000000006','M06',2018,'June',3903134,true,'P','Preliminary.'
'LAUMT481910000000006','M05',2018,'May',3897914,null,'null','null'
'LAUMT481910000000006','M04',2018,'April',3904883,null,'null','null'
'LAUMT481910000000006','M03',2018,'March',3887962,null,'null','null'
'LAUMT481910000000006','M02',2018,'February',3883450,null,'null','null'
'LAUMT481910000000006','M01',2018,'January',3829770,null,'null','null'
'LAUMT481910000000006','M12',2017,'December',3827120,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M11',2017,'November',3841732,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M10',2017,'October',3813603,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M09',2017,'September',3835897,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M08',2017,'August',3805982,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M07',2017,'July',3815317,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M06',2017,'June',3788012,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M05',2017,'May',3767275,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M04',2017,'April',3773905,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M03',2017,'March',3765100,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M02',2017,'February',3764030,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M01',2017,'January',3745524,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M12',2016,'December',3743035,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M11',2016,'November',3745036,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M10',2016,'October',3726779,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M09',2016,'September',3728897,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M08',2016,'August',3712240,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M07',2016,'July',3717507,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M06',2016,'June',3693746,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M05',2016,'May',3670094,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M04',2016,'April',3665141,null,'R','Data were subject to revision on April 20, 2018.'
'LAUMT481910000000006','M03',2016,'March',3659059,null,'R','Data were subject to revision on April 20, 2018.'
```

Part 3: Establishing Connection to MYSQL Workbench:

Go to https://dev.mysql.com/downloads/windows/installer/8.0.html and download mysql (mysql-installer-community-8.0.11.0.msi) skip the signup and just download and save file to desktop. Then double click the installer and install. Choose Custom setup type, in Select Product and features select MySQL Servers and for further instructions follow this Youtube link and easy installation (https://youtu.be/Ddx13KlW8yQ?t=67) or (https://youtu.be/aY6LiTbfckA?t=119) Use the code below to create a user in your local database, then create tables to store the data. Connect to the local database with with previously created Username and password in Java code. Below are the screenshot of code in database and in java.



```sql
DROP DATABASE BLSData;

create database BLSData DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;

USE  BLSData;

create table seriesData (
seriesID varchar(20),
Period varchar(20),
year int(10),
PeriodName varchar(15),
Value int(11),
Latest boolean,
Code varchar(15),
Text varchar(200)
);

CREATE USER 'BLSuser'@'localhost' IDENTIFIED BY 'Password123';
GRANT ALL PRIVILEGES ON javabase.* TO 'BLSuser'@'localhost' ;
FLUSH PRIVILEGES;


select count(*) from seriesData;
select * from seriesData;
select distinct * from seriesData;
TRUNCATE  TABLE seriesData;

DROP TABLE seriesData ;
```



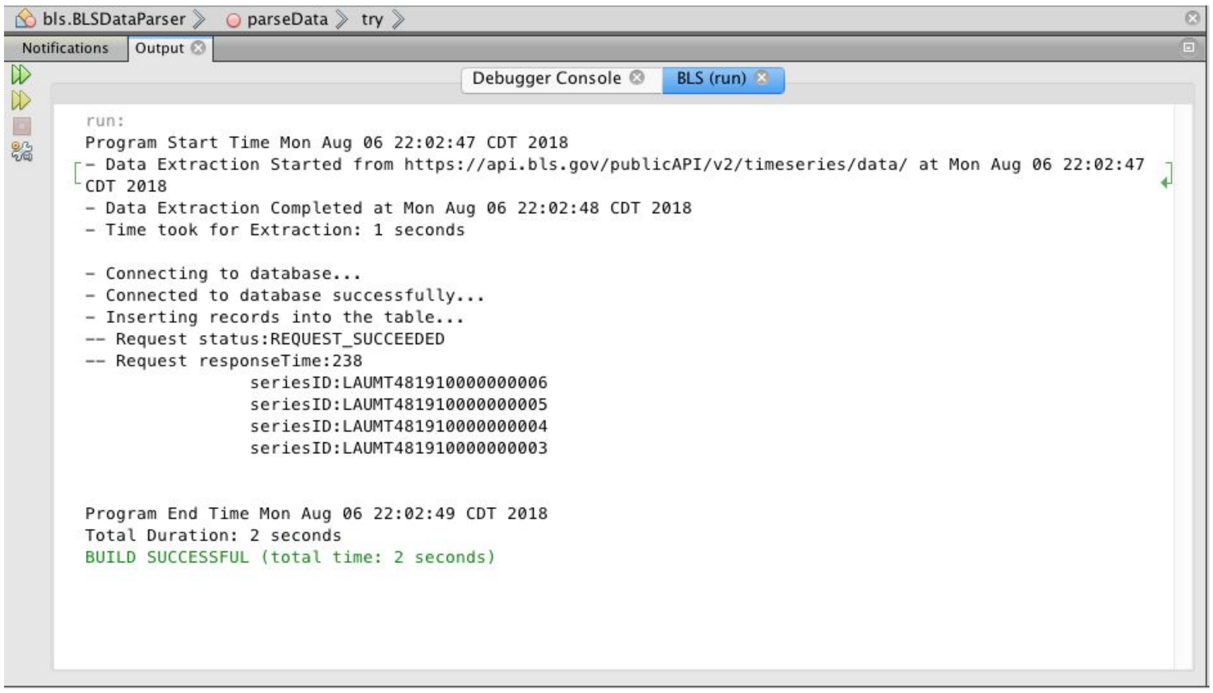
```java
        String DBUserName = "BLSuser";
        String DBPassword = "Password123";

        String DBUrl = "jdbc:mysql://localhost:3306/BLSData?useUnicode=true&useJDBCCompliantTimezoneS
        Class.forName("com.mysql.cj.jdbc.Driver");

         //Open a connection
         System.out.println("- Connecting to database...");
         conn = DriverManager.getConnection(DBUrl, DBUserName, DBPassword);

        System.out.println("- Connected to database successfully...");

        //Execute a query
        System.out.println("- Inserting records into the table...");
        stmt = conn.createStatement();


        Object obj = new JSONParser().parse(new FileReader("BLSoutput.json"));
        PrintWriter outputfile = new PrintWriter("BLSparser.txt");
        // typecasting obj to JSONObject
        JSONObject jo = (JSONObject) obj;
```

Code:

```
DROP DATABASE BLSData;//Drops Previous Database

create database BLSData DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;

USE  BLSData;

create table seriesData (

      seriesID varchar(20),

      Period varchar(20),

      year int(10),

      PeriodName varchar(15),

      Value int(11),

      Latest boolean,

      Code varchar(15),

      Text varchar(200)

);

CREATE USER 'BLSuser'@'localhost' IDENTIFIED BY 'Password123';//Creates
BLSuser with password 'Password123'

GRANT ALL PRIVILEGES ON javabase.* TO 'BLSuser'@'localhost' ;

FLUSH PRIVILEGES;

select count(*) from seriesData;

select * from seriesData;

select distinct * from seriesData;

TRUNCATE  TABLE seriesData;

DROP TABLE seriesData ;
```

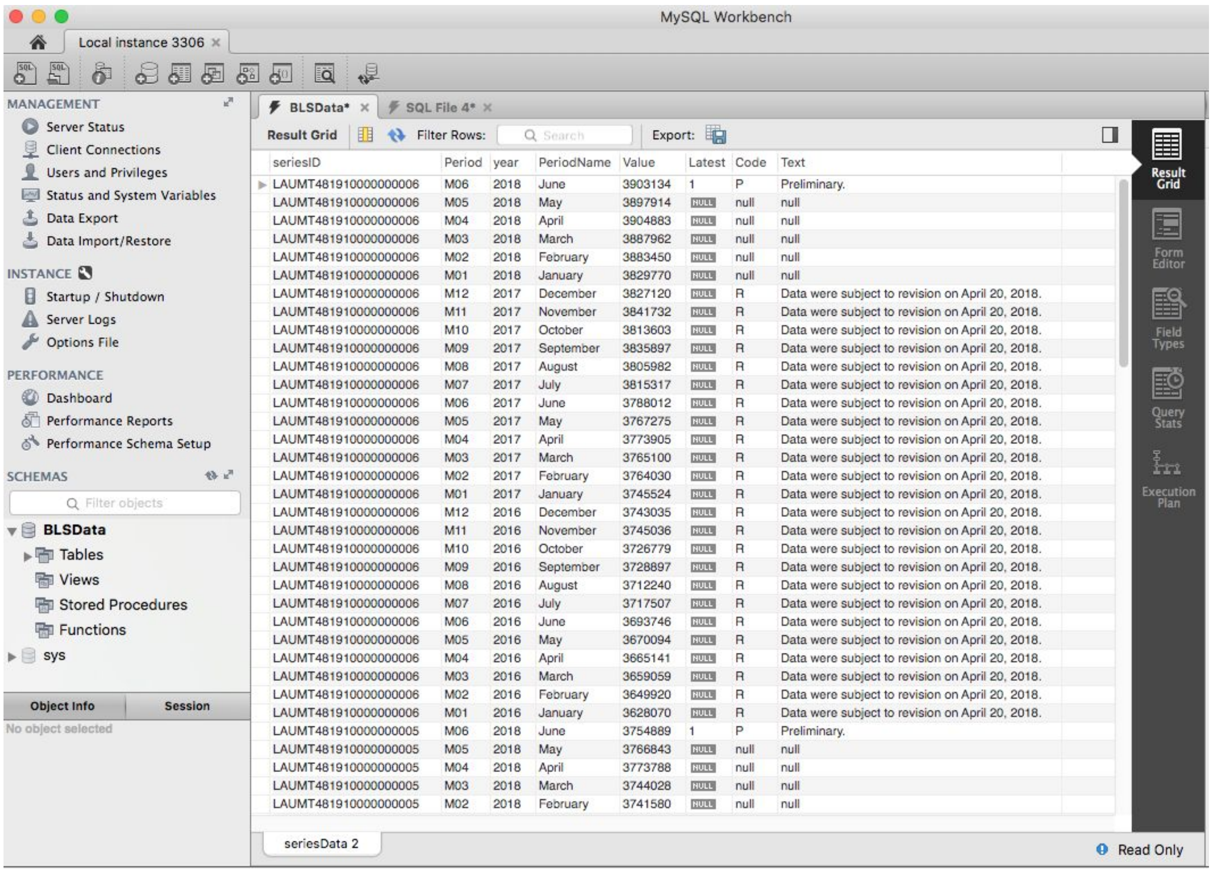
OutPut:

After java code execution



```
run:
Program Start Time Mon Aug 06 22:02:47 CDT 2018
- Data Extraction Started from https://api.bls.gov/publicAPI/v2/timeseries/data/ at Mon Aug 06 22:02:47
CDT 2018
- Data Extraction Completed at Mon Aug 06 22:02:48 CDT 2018
- Time took for Extraction: 1 seconds

- Connecting to database...
- Connected to database successfully...
- Inserting records into the table...
-- Request status:REQUEST_SUCCEEDED
-- Request responseTime:238
                seriesID:LAUMT481910000000006
                seriesID:LAUMT481910000000005
                seriesID:LAUMT481910000000004
                seriesID:LAUMT481910000000003


Program End Time Mon Aug 06 22:02:49 CDT 2018
Total Duration: 2 seconds
BUILD SUCCESSFUL (total time: 2 seconds)
```

and two files will be created in project folder "BLSoutput.json", "BLSparser.txt" on the other side in Database the final records are shown as



seriesID	Period	year	PeriodName	Value	Latest	Code	Text
LAUMT481910000000006	M06	2018	June	3903134	1	P	Preliminary.
LAUMT481910000000006	M05	2018	May	3897914	NULL	null	null
LAUMT481910000000006	M04	2018	April	3904883	NULL	null	null
LAUMT481910000000006	M03	2018	March	3887962	NULL	null	null
LAUMT481910000000006	M02	2018	February	3883450	NULL	null	null
LAUMT481910000000006	M01	2018	January	3829770	NULL	null	null
LAUMT481910000000006	M12	2017	December	3827120	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M11	2017	November	3841732	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M10	2017	October	3813603	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M09	2017	September	3835897	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M08	2017	August	3805982	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M07	2017	July	3815317	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M06	2017	June	3788012	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M05	2017	May	3767275	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M04	2017	April	3773905	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M03	2017	March	3765100	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M02	2017	February	3764030	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M01	2017	January	3745524	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M12	2016	December	3743035	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M11	2016	November	3745036	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M10	2016	October	3726779	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M09	2016	September	3728897	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M08	2016	August	3712240	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M07	2016	July	3717507	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M06	2016	June	3693746	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M05	2016	May	3670094	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M04	2016	April	3665141	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M03	2016	March	3659059	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M02	2016	February	3649920	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000006	M01	2016	January	3628070	NULL	R	Data were subject to revision on April 20, 2018.
LAUMT481910000000005	M06	2018	June	3754889	1	P	Preliminary.
LAUMT481910000000005	M05	2018	May	3766843	NULL	null	null
LAUMT481910000000005	M04	2018	April	3773788	NULL	null	null
LAUMT481910000000005	M03	2018	March	3744028	NULL	null	null
LAUMT481910000000005	M02	2018	February	3741580	NULL	null	null

References:

1. https://dev.mysql.com/downloads/installer/
2. https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient/4.5.5
3. https://www.youtube.com/watch?v=aY6LiTbfckA
4. https://www.geeksforgeeks.org/parse-json-java/
5. https://www.bls.gov/help/hlpforma.htm#LA
6. https://www.bls.gov/developers/api_java.htm#java2
7. https://stackoverflow.com