DATA ANALYTICS AND AI

Department Industrial Vertical (DIV) #DAY2

Foundation of Data Engineering

Setting Enviroment

for Data Pipeline

To build a local ETL(Extract,Transform,Load) pipeline that reads data from:

-Mysql table(online sales)

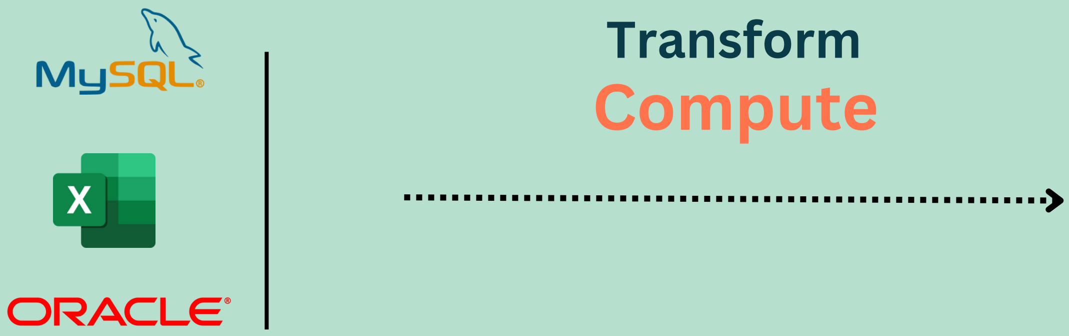
-An CSV file (offline sales)

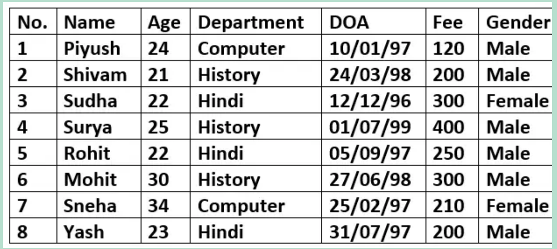
Extract

Transform

Compute

Load





No.	Name	Age	Department	DOA	Fee	Gender
1	Piyush	24	Computer	10/01/97	120	Male
2	Shivam	21	History	24/03/98	200	Male
3	Sudha	22	Hindi	12/12/96	300	Female
4	Surya	25	History	01/07/99	400	Male
5	Rohit	22	Hindi	05/09/97	250	Male
6	Mohit	30	History	27/06/98	300	Male
7	Sneha	34	Computer	25/02/97	210	Female
8	Yash	23	Hindi	31/07/97	200	Male

Raw Data from
different sources

Data are stored in
Mass storage
database like cloud

Tools Required for

Data Pipeline Process (ETL):

1. APACHE SPARK

A distributed computing system designed for fast data processing, analytics, and machine learning, ideal for handling large-scale data.

Prerequisites for Installing Apache Spark on Windows:

- **Java Development Kit (JDK) – Install JDK and set the JAVA_HOME environment variable.**
- **Python – Install Python 3.x (Anaconda includes Python).**
- **Anaconda (Optional) – Recommended for managing Python dependencies.**
- **Hadoop winutils – Required for running Spark on Windows.**
- **Set Environment Variables – Add Spark, Hadoop, and Java paths to the system environment variables.**

Installation Steps:

Create Folders:

C:\spark\ (for Spark)
C:\hadoop\bin\ (for winutils)

Download & Extract Spark:

Download Spark from https://spark.apache.org/downloads.html (Pre-built for Hadoop 3.3).

Extract and move it to C:\spark\.

Download Winutils:

Download winutils.exe from GitHub.

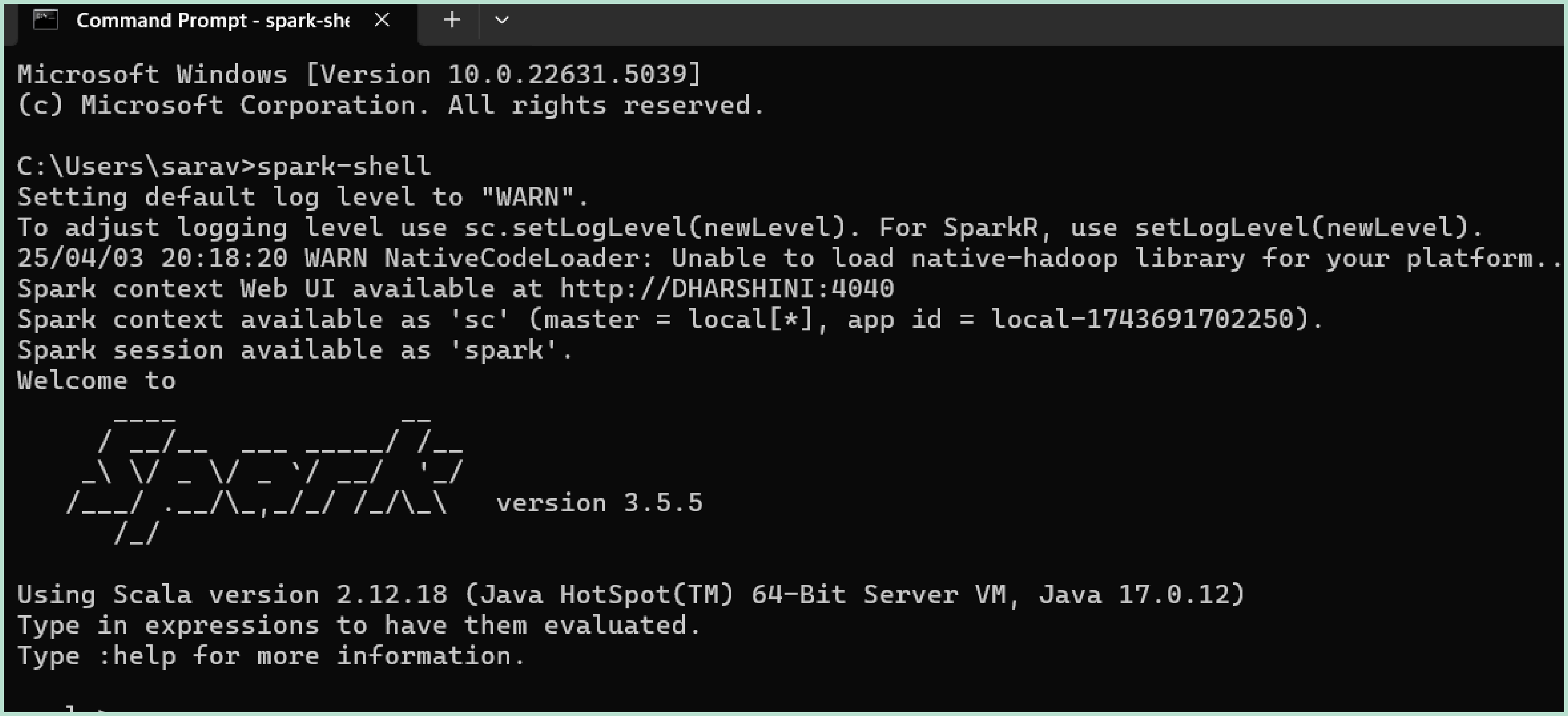
Copy it to C:\hadoop\bin\.

Set Environment Variables:

SPARK_HOME = C:\spark\spark-3.x.x-bin-hadoop3.3
HADOOP_HOME = C:\hadoop
JAVA_HOME = C:\Program Files\Java\jdk-XX.X.X
Add %SPARK_HOME%\bin and %HADOOP_HOME%\bin to Path.

Verify Installation in cmd:

```
java -version,  python --version,  spark-shell,  pyspark
```



```
Microsoft Windows [Version 10.0.22631.5039]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sarav>spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/03 20:18:20 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform..
Spark context Web UI available at http://DHARSHINI:4040
Spark context available as 'sc' (master = local[*], app id = local-1743691702250).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.5
      /_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 17.0.12)
Type in expressions to have them evaluated.
Type :help for more information.
```

2. MySQL

MySQL is used in a data pipeline to store, manage, and retrieve structured data efficiently with SQL queries. It integrates with Spark via JDBC, enabling seamless data processing, transformation, and analytics.

.

Installation Steps:

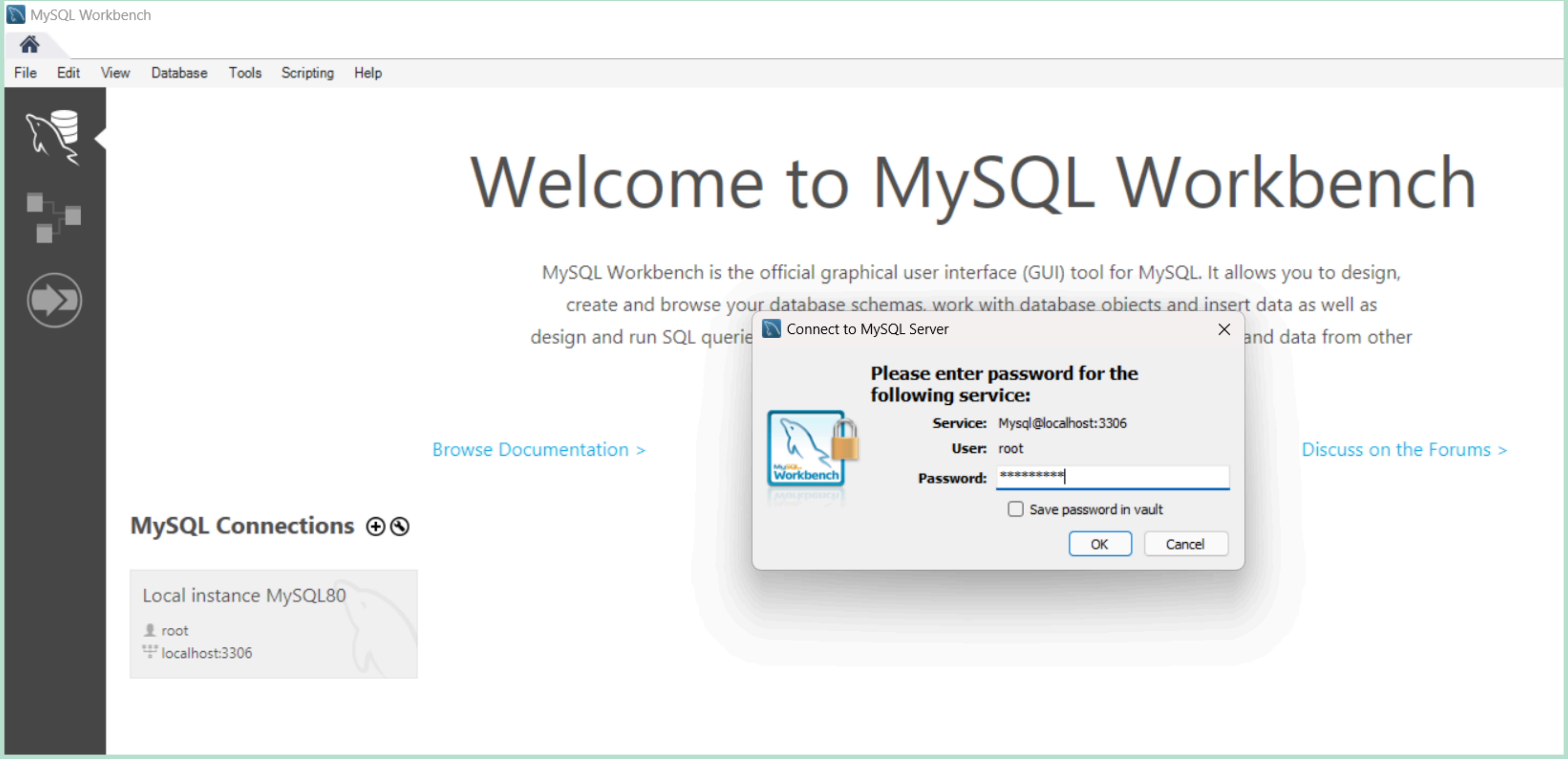
Download & Install MySQL (MySQL Installer)

Select MySQL Server & Workbench
→ Set root password → Complete installation.

Download MySQL JDBC Driver (Connector/J)

MySQL Connector/J (JDBC) is used to establish a connection between Apache Spark (or any Java-based application) and MySQL, enabling seamless data transfer.

Extract & copy mysql-connector-java-9.1.0.jar
to C:\spark\jars\.



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL querie … and data from other

Connect to MySQL Server

Please enter password for the following service:

Service: Mysql@localhost:3306
User: root
Password: *********

□ Save password in vault

MySQL Connections

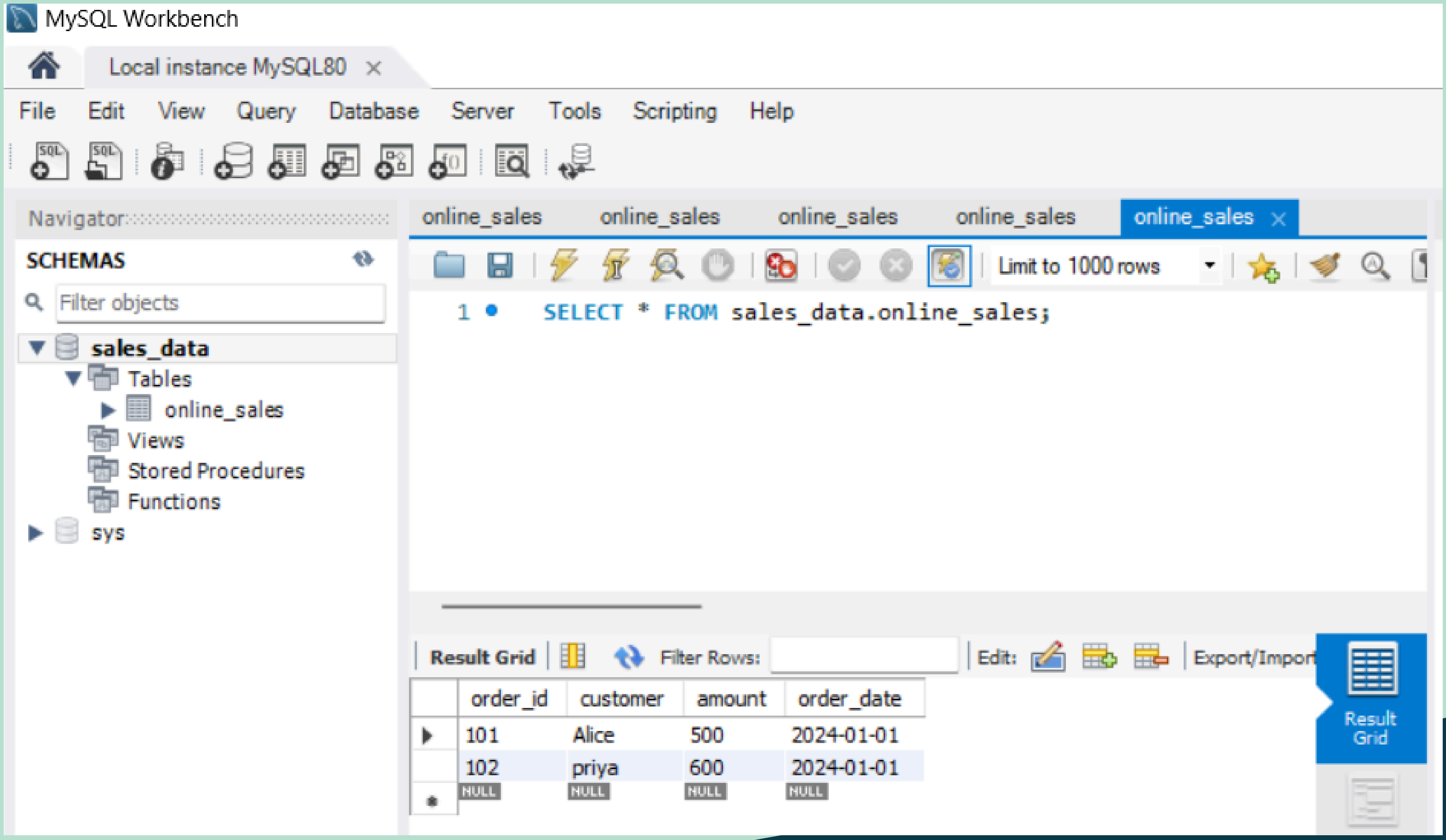
Local instance MySQL80
root
localhost:3306

Load Data:

```
CREATE DATABASE sales_data;
USE sales_data;

CREATE TABLE online_sales ( order_id INT,
 customer VARCHAR(50),
 amount FLOAT,
order_date DATE );

INSERT INTO online_sales VALUES
(101, 'Alice', 300.50, '2024-03-01'),
 (102, 'Charlie', 90.00, '2024-03-03');
```



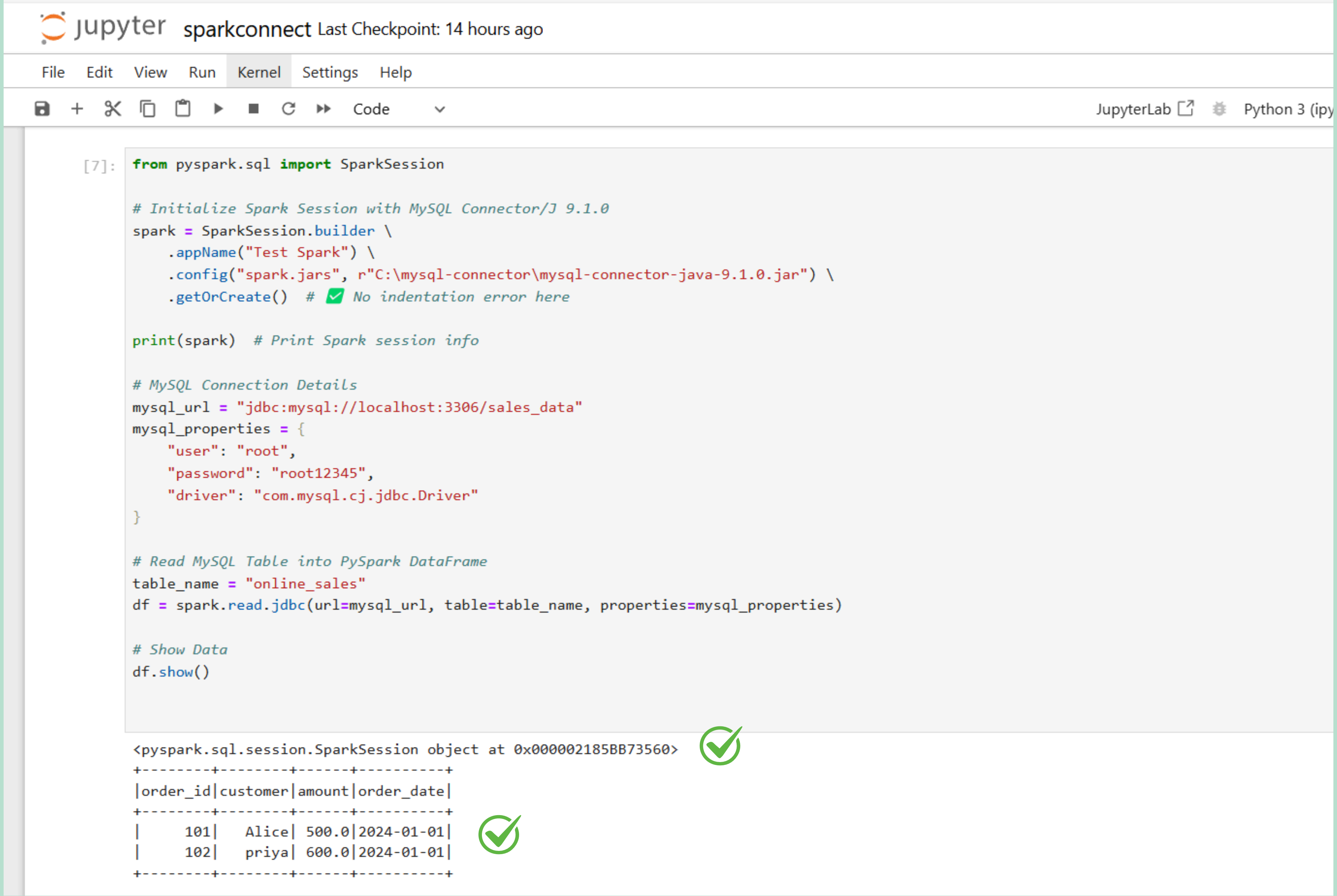
```
SELECT * FROM sales_data.online_sales;
```

order_id	customer	amount	order_date
101	Alice	500	2024-01-01
102	priya	600	2024-01-01
NULL	NULL	NULL	NULL

3. pySpark code (MySql)

- MySQL Connector/J (JDBC) in PySpark enables seamless reading and writing of MySQL data using Spark DataFrames. It allows efficient SQL-based data processing, making it essential for ETL and analytics pipelines

- Jupyter Notebook is commonly used with PySpark for interactive development, debugging, and visualization. It allows executing PySpark code step by step, making it easier to analyze MySQL data in a local data pipeline.



```python
from pyspark.sql import SparkSession

# Initialize Spark Session with MySQL Connector/J 9.1.0
spark = SparkSession.builder \
    .appName("Test Spark") \
    .config("spark.jars", r"C:\mysql-connector\mysql-connector-java-9.1.0.jar") \
    .getOrCreate()  # ✅ No indentation error here

print(spark)  # Print Spark session info

# MySQL Connection Details
mysql_url = "jdbc:mysql://localhost:3306/sales_data"
mysql_properties = {
    "user": "root",
    "password": "root12345",
    "driver": "com.mysql.cj.jdbc.Driver"
}

# Read MySQL Table into PySpark DataFrame
table_name = "online_sales"
df = spark.read.jdbc(url=mysql_url, table=table_name, properties=mysql_properties)

# Show Data
df.show()
```

```
<pyspark.sql.session.SparkSession object at 0x000002185BB73560>
+--------+--------+------+----------+
|order_id|customer|amount|order_date|
+--------+--------+------+----------+
|     101|   Alice| 500.0|2024-01-01|
|     102|   priya| 600.0|2024-01-01|
+--------+--------+------+----------+
```

4. pySpark code (CSV file)

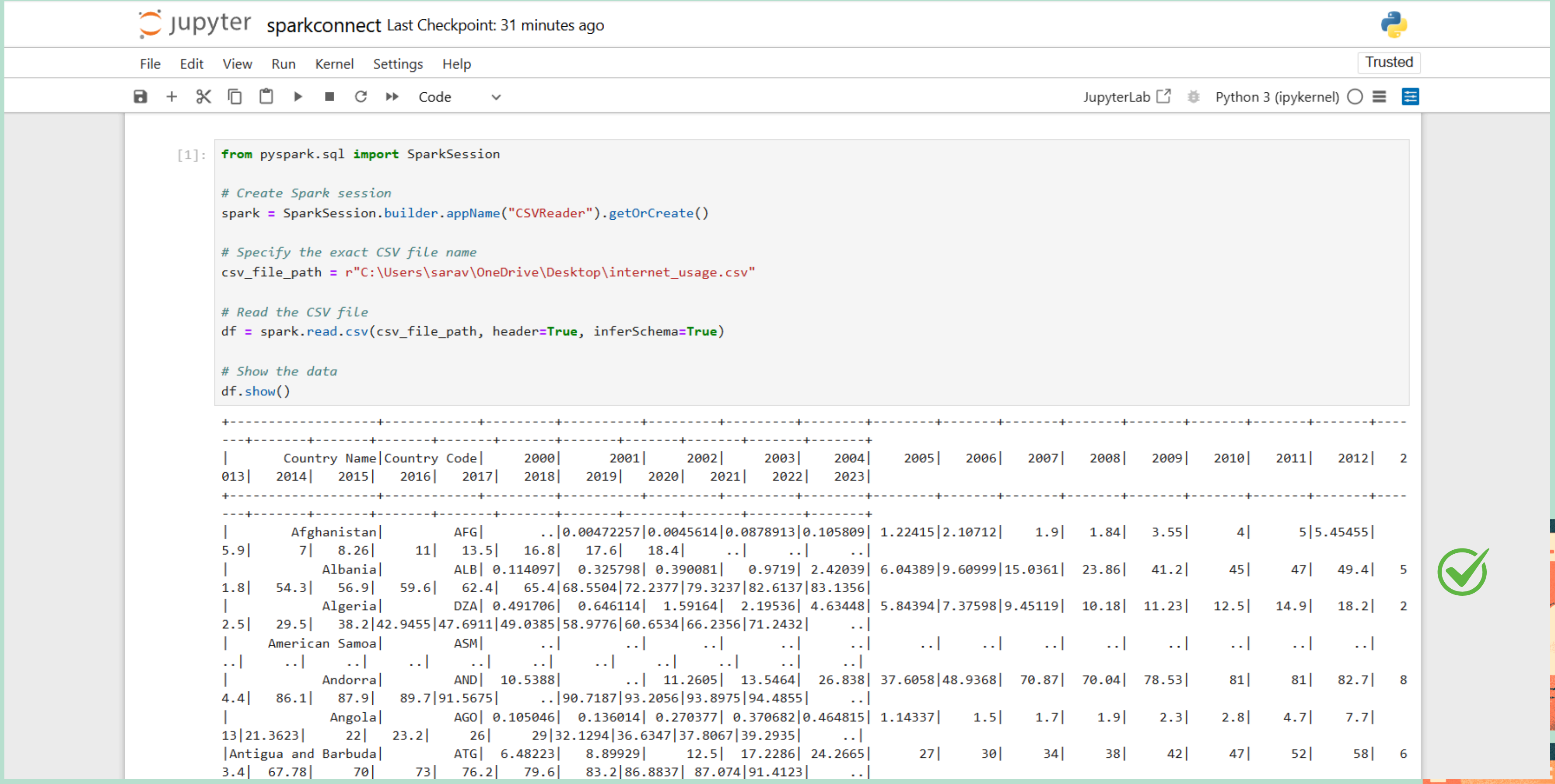
Loading a CSV File in PySpark

Why Load CSV in PySpark?

Handles Large Datasets – PySpark efficiently processes large CSV files in a distributed manner.

Schema Inference & Customization – Supports automatic and manual schema definitions.

Flexible Data Processing – Allows filtering, transformation, and SQL-like operations on CSV data.



```
from pyspark.sql import SparkSession

# Create Spark session
spark = SparkSession.builder.appName("CSVReader").getOrCreate()

# Specify the exact CSV file name
csv_file_path = r"C:\Users\sarav\OneDrive\Desktop\internet_usage.csv"

# Read the CSV file
df = spark.read.csv(csv_file_path, header=True, inferSchema=True)

# Show the data
df.show()
```

```
+------------------+------------+---------+----------+---------+---------+--------+--------+-------+-------+-------+-------+-------+-------+-------+----
---+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|      Country Name|Country Code|     2000|      2001|     2002|     2003|    2004|    2005|   2006|   2007|   2008|   2009|   2010|   2011|   2012|   2
013|   2014|   2015|   2016|   2017|   2018|   2019|   2020|   2021|   2022|   2023|
+------------------+------------+---------+----------+---------+---------+--------+--------+-------+-------+-------+-------+-------+-------+-------+----
---+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|       Afghanistan|         AFG|       ..|0.00472257|0.0045614|0.0878913|0.105809| 1.22415|2.10712|    1.9|   1.84|   3.55|      4|      5|5.45455|    
5.9|      7|   8.26|     11|   13.5|   16.8|   17.6|   18.4|     ..|     ..|     ..|
|           Albania|         ALB| 0.114097|  0.325798| 0.390081|   0.9719| 2.42039| 6.04389|9.60999|15.0361|  23.86|   41.2|     45|     47|   49.4|   5
1.8|   54.3|   56.9|   59.6|   62.4|   65.4|68.5504|72.2377|79.3237|82.6137|83.1356|
|           Algeria|         DZA| 0.491706|  0.646114|  1.59164|  2.19536| 4.63448| 5.84394|7.37598|9.45119|  10.18|  11.23|   12.5|   14.9|   18.2|   2
2.5|   29.5|   38.2|42.9455|47.6911|49.0385|58.9776|60.6534|66.2356|71.2432|     ..|
|    American Samoa|         ASM|       ..|        ..|       ..|       ..|      ..|      ..|     ..|     ..|     ..|     ..|     ..|     ..|     ..|    
..|     ..|     ..|     ..|     ..|     ..|     ..|     ..|     ..|     ..|     ..|
|           Andorra|         AND|  10.5388|        ..|  11.2605|  13.5464|  26.838| 37.6058|48.9368|  70.87|  70.04|  78.53|     81|     81|   82.7|   8
4.4|   86.1|   87.9|   89.7|91.5675|     ..|90.7187|93.2056|93.8975|94.4855|     ..|
|            Angola|         AGO| 0.105046|  0.136014| 0.270377| 0.370682|0.464815| 1.14337|    1.5|    1.7|    1.9|    2.3|    2.8|    4.7|    7.7|    
13|21.3623|     22|   23.2|     26|     29|32.1294|36.6347|37.8067|39.2935|     ..|
|Antigua and Barbuda|        ATG|  6.48223|   8.89929|     12.5|  17.2286| 24.2665|      27|     30|     34|     38|     42|     47|     52|     58|   6
3.4|  67.78|     70|     73|   76.2|   79.6|   83.2|86.8837| 87.074|91.4123|     ..|
```