EXP. NO:6	**Read Stock Market Data into a DataFrame, using Spark Window Functions**

Aim:

To read stock market data into a DataFrame, use window functions to calculate the moving average price for each stock, and display the results.

Theory:

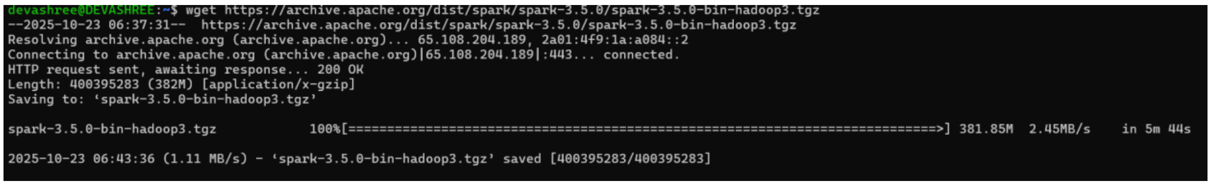
Apache Spark is a fast and general-purpose distributed computing system that provides APIs for large-scale data processing. Spark DataFrames allow users to manipulate structured data using a domain-specific language similar to SQL.

Window functions in Spark enable operations like running totals, ranking, and moving averages across a set of related rows.

Steps:

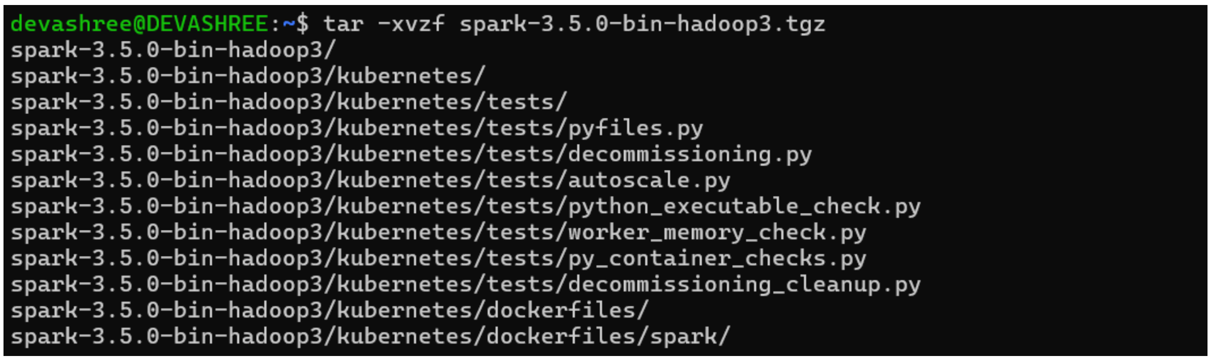
Step 1 : Download & extract Spark

wget https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz



```
devashree@DEVASHREE:~$ wget https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
--2025-10-23 06:37:31--  https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 400395283 (382M) [application/x-gzip]
Saving to: ‘spark-3.5.0-bin-hadoop3.tgz’

spark-3.5.0-bin-hadoop3.tgz          100%[==========================================================================>] 381.85M  2.45MB/s    in 5m 44s

2025-10-23 06:43:36 (1.11 MB/s) - ‘spark-3.5.0-bin-hadoop3.tgz’ saved [400395283/400395283]
```

tar -xvzf spark-3.5.0-bin-hadoop3.tgz

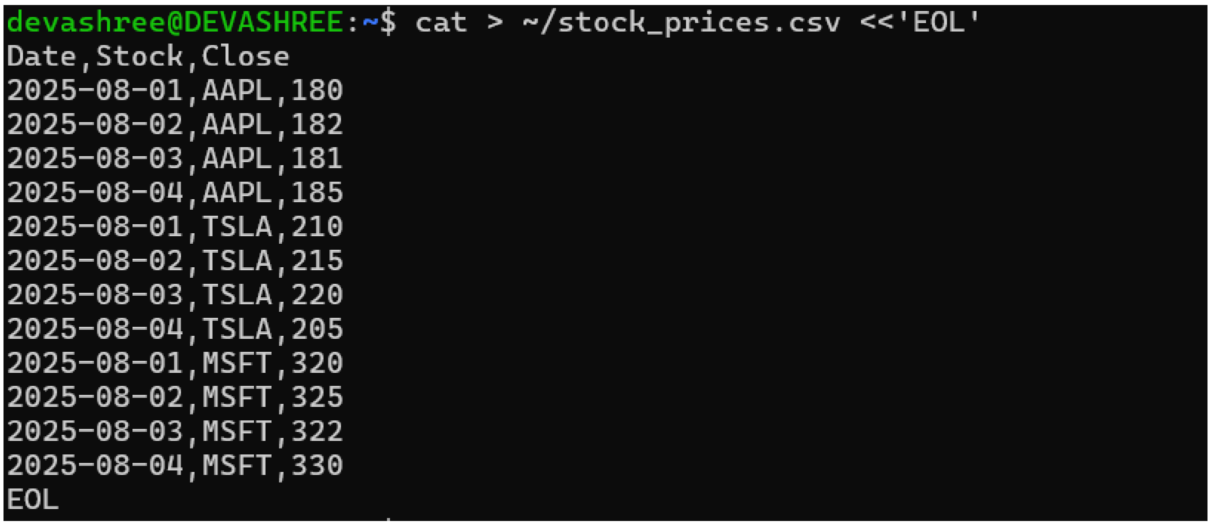


```
devashree@DEVASHREE:~$ tar -xvzf spark-3.5.0-bin-hadoop3.tgz
spark-3.5.0-bin-hadoop3/
spark-3.5.0-bin-hadoop3/kubernetes/
spark-3.5.0-bin-hadoop3/kubernetes/tests/
spark-3.5.0-bin-hadoop3/kubernetes/tests/pyfiles.py
spark-3.5.0-bin-hadoop3/kubernetes/tests/decommissioning.py
spark-3.5.0-bin-hadoop3/kubernetes/tests/autoscale.py
spark-3.5.0-bin-hadoop3/kubernetes/tests/python_executable_check.py
spark-3.5.0-bin-hadoop3/kubernetes/tests/worker_memory_check.py
spark-3.5.0-bin-hadoop3/kubernetes/tests/py_container_checks.py
spark-3.5.0-bin-hadoop3/kubernetes/tests/decommissioning_cleanup.py
spark-3.5.0-bin-hadoop3/kubernetes/dockerfiles/
spark-3.5.0-bin-hadoop3/kubernetes/dockerfiles/spark/
```

Step 2 : Create sample CSV (stock_prices.csv)

```
Date,Stock,Close
2025-08-01,AAPL,180
2025-08-02,AAPL,182
2025-08-03,AAPL,181
2025-08-04,AAPL,185
2025-08-01,TSLA,210
2025-08-02,TSLA,215
2025-08-03,TSLA,220
2025-08-04,TSLA,205
2025-08-01,MSFT,320
```

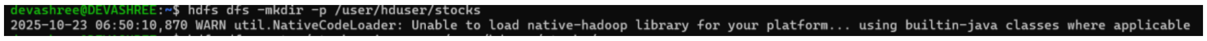
```
2025-08-02,MSFT,325
2025-08-03,MSFT,322
2025-08-04,MSFT,330
```



```
devashree@DEVASHREE:~$ cat > ~/stock_prices.csv <<'EOL'
Date,Stock,Close
2025-08-01,AAPL,180
2025-08-02,AAPL,182
2025-08-03,AAPL,181
2025-08-04,AAPL,185
2025-08-01,TSLA,210
2025-08-02,TSLA,215
2025-08-03,TSLA,220
2025-08-04,TSLA,205
2025-08-01,MSFT,320
2025-08-02,MSFT,325
2025-08-03,MSFT,322
2025-08-04,MSFT,330
EOL
```

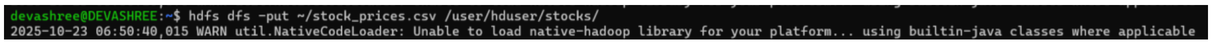
Step 3 : Upload CSV to HDFS

hdfs dfs -mkdir -p /user/hduser/stocks



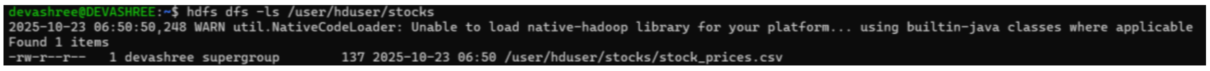
```
devashree@DEVASHREE:~$ hdfs dfs -mkdir -p /user/hduser/stocks
2025-10-23 06:50:10,870 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

hdfs dfs -put ~/stock_prices.csv /user/hduser/stocks/



```
devashree@DEVASHREE:~$ hdfs dfs -put ~/stock_prices.csv /user/hduser/stocks/
2025-10-23 06:50:40,015 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

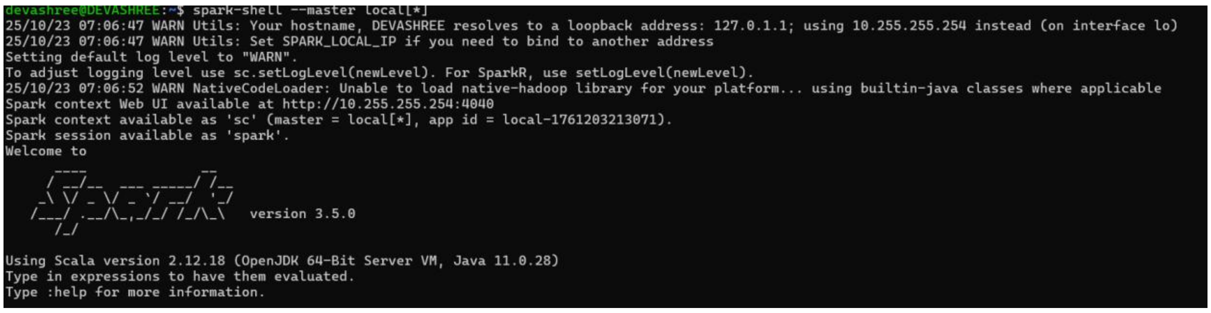
hdfs dfs -ls /user/hduser/stocks



```
devashree@DEVASHREE:~$ hdfs dfs -ls /user/hduser/stocks
2025-10-23 06:50:50,248 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 devashree supergroup        137 2025-10-23 06:50 /user/hduser/stocks/stock_prices.csv
```

Step 4 : Launch spark-shell (Scala)

spark-shell --master local[*]



```
devashree@DEVASHREE:~$ spark-shell --master local[*]
25/10/23 07:06:47 WARN Utils: Your hostname, DEVASHREE resolves to a loopback address: 127.0.1.1; using 10.255.255.254 instead (on interface lo)
25/10/23 07:06:47 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/10/23 07:06:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://10.255.255.254:4040
Spark context available as 'sc' (master = local[*], app id = local-1761203213071).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.28)
Type in expressions to have them evaluated.
Type :help for more information.
```

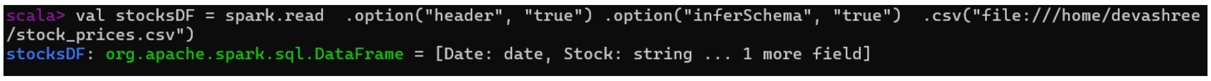
Step 5 : Load the data in Spark

val stocksDF = spark.read

  .option("header", "true")

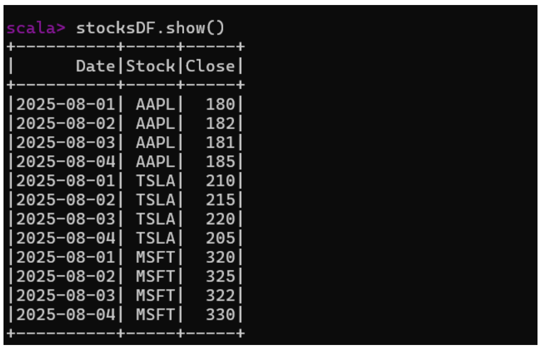
  .option("inferSchema", "true")

  .csv("file:///home/devashree/stock_prices.csv")



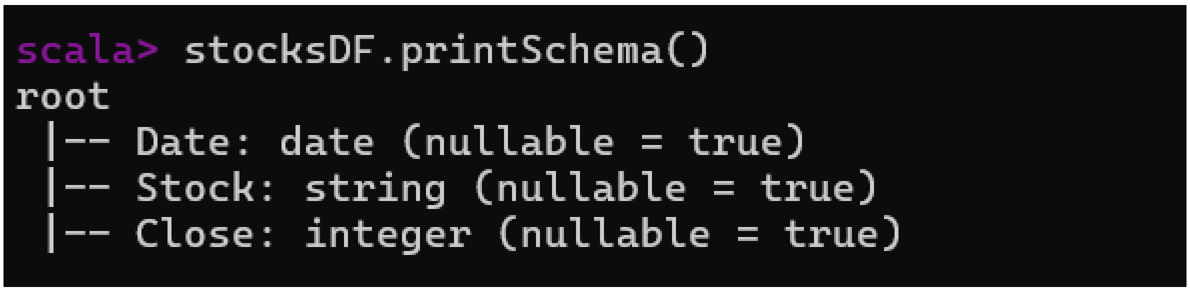
```
scala> val stocksDF = spark.read  .option("header", "true") .option("inferSchema", "true")  .csv("file:///home/devashree
/stock_prices.csv")
stocksDF: org.apache.spark.sql.DataFrame = [Date: date, Stock: string ... 1 more field]
```

stocksDF.show()



```
scala> stocksDF.show()
+----------+-----+-----+
|      Date|Stock|Close|
+----------+-----+-----+
|2025-08-01| AAPL|  180|
|2025-08-02| AAPL|  182|
|2025-08-03| AAPL|  181|
|2025-08-04| AAPL|  185|
|2025-08-01| TSLA|  210|
|2025-08-02| TSLA|  215|
|2025-08-03| TSLA|  220|
|2025-08-04| TSLA|  205|
|2025-08-01| MSFT|  320|
|2025-08-02| MSFT|  325|
|2025-08-03| MSFT|  322|
|2025-08-04| MSFT|  330|
+----------+-----+-----+
```

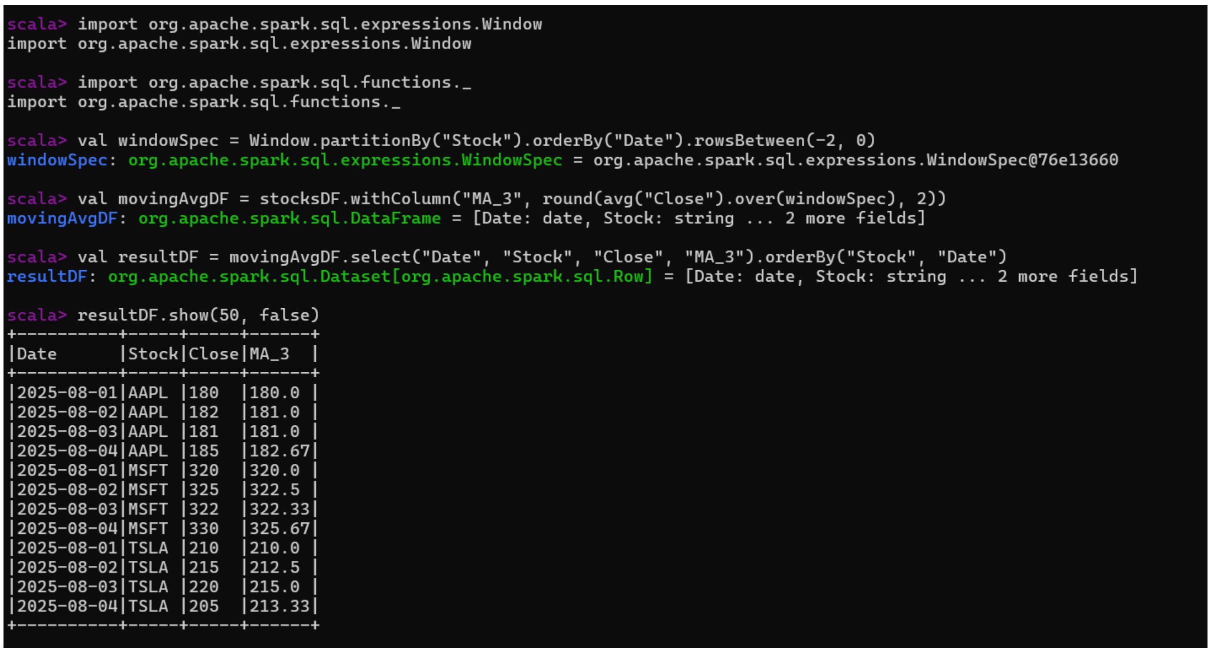
stocksDF.printSchema()



```
scala> stocksDF.printSchema()
root
 |-- Date: date (nullable = true)
 |-- Stock: string (nullable = true)
 |-- Close: integer (nullable = true)
```

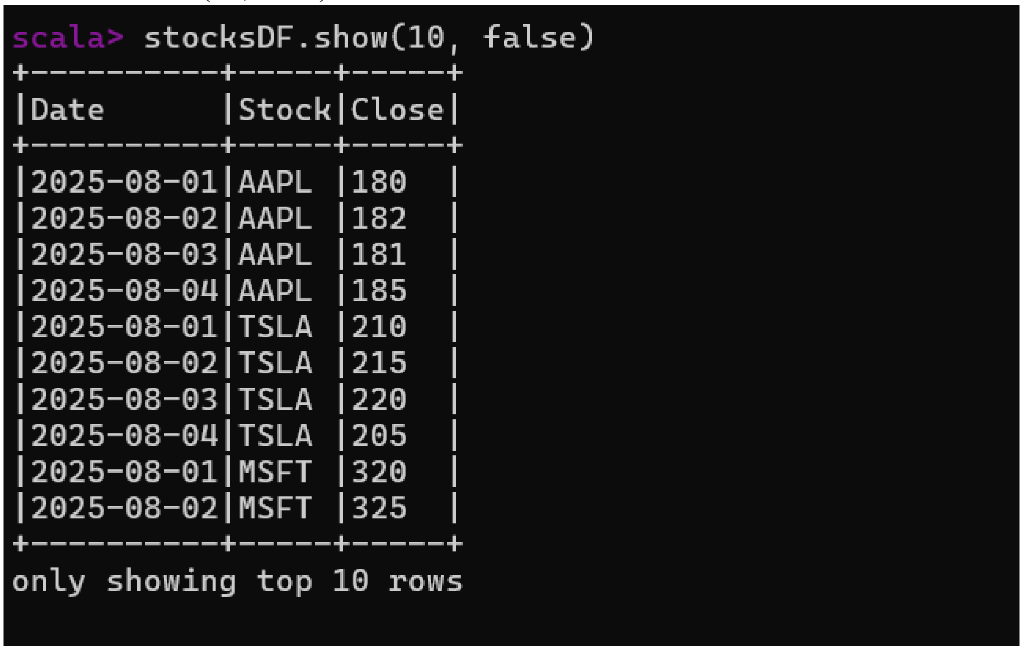
Step 6 : Compute 3-day moving average

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions._

val windowSpec = Window.partitionBy("Stock").orderBy("Date").rowsBetween(-2, 0)

val resultDF = stocksDF
  .withColumn("MA_3", round(avg(col("Close")).over(windowSpec), 2))
  .select("Date", "Stock", "Close", "MA_3")
  .orderBy("Stock", "Date")

resultDF.show(50, false)
```

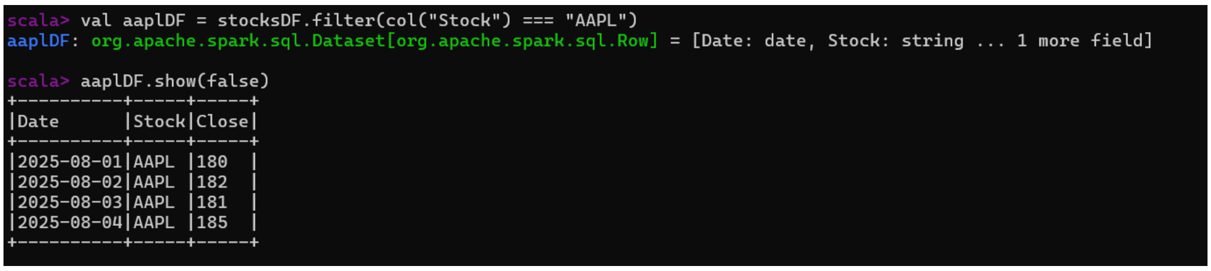

```
scala> import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.expressions.Window

scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> val windowSpec = Window.partitionBy("Stock").orderBy("Date").rowsBetween(-2, 0)
windowSpec: org.apache.spark.sql.expressions.WindowSpec = org.apache.spark.sql.expressions.WindowSpec@76e13660

scala> val movingAvgDF = stocksDF.withColumn("MA_3", round(avg("Close").over(windowSpec), 2))
movingAvgDF: org.apache.spark.sql.DataFrame = [Date: date, Stock: string ... 2 more fields]

scala> val resultDF = movingAvgDF.select("Date", "Stock", "Close", "MA_3").orderBy("Stock", "Date")
resultDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Date: date, Stock: string ... 2 more fields]

scala> resultDF.show(50, false)
+----------+-----+-----+------+
|Date      |Stock|Close|MA_3  |
+----------+-----+-----+------+
|2025-08-01|AAPL |180  |180.0 |
|2025-08-02|AAPL |182  |181.0 |
|2025-08-03|AAPL |181  |181.0 |
|2025-08-04|AAPL |185  |182.67|
|2025-08-01|MSFT |320  |320.0 |
|2025-08-02|MSFT |325  |322.5 |
|2025-08-03|MSFT |322  |322.33|
|2025-08-04|MSFT |330  |325.67|
|2025-08-01|TSLA |210  |210.0 |
|2025-08-02|TSLA |215  |212.5 |
|2025-08-03|TSLA |220  |215.0 |
|2025-08-04|TSLA |205  |213.33|
+----------+-----+-----+------+
```

EXERCISE:

1) **Load the stock data and display the first 10 rows of the DataFrame.**
   stocksDF.show(10, false)

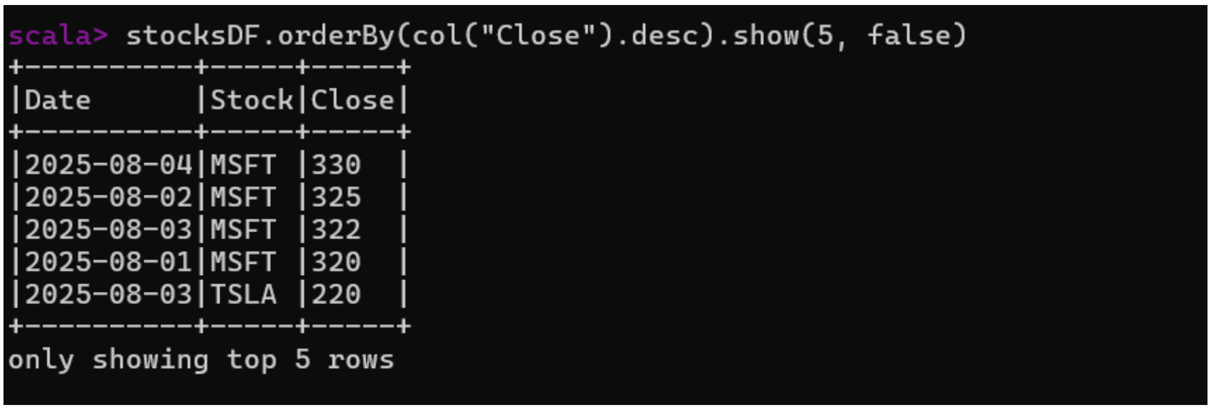

```
scala> stocksDF.show(10, false)
+----------+-----+-----+
|Date      |Stock|Close|
+----------+-----+-----+
|2025-08-01|AAPL |180  |
|2025-08-02|AAPL |182  |
|2025-08-03|AAPL |181  |
|2025-08-04|AAPL |185  |
|2025-08-01|TSLA |210  |
|2025-08-02|TSLA |215  |
|2025-08-03|TSLA |220  |
|2025-08-04|TSLA |205  |
|2025-08-01|MSFT |320  |
|2025-08-02|MSFT |325  |
+----------+-----+-----+
only showing top 10 rows
```

2) **Filter the DataFrame to show only the rows where the stock symbol is AAPL.**
   val aaplDF = stocksDF.filter(col("Stock") === "AAPL")
   aaplDF.show(false)


```
scala> val aaplDF = stocksDF.filter(col("Stock") === "AAPL")
aaplDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Date: date, Stock: string ... 1 more field]

scala> aaplDF.show(false)
+----------+-----+-----+
|Date      |Stock|Close|
+----------+-----+-----+
|2025-08-01|AAPL |180  |
|2025-08-02|AAPL |182  |
|2025-08-03|AAPL |181  |
|2025-08-04|AAPL |185  |
+----------+-----+-----+
```

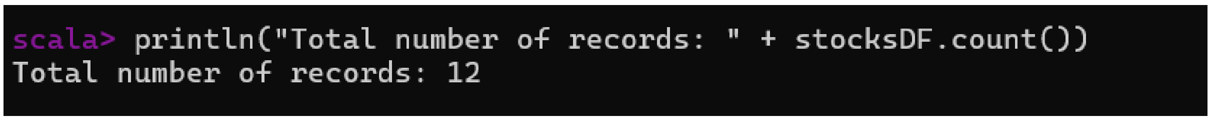
3) **Sort the data by the Close price in descending order and display the top 5 rows.**

stocksDF.orderBy(col("Close").desc).show(5, false)


```
scala> stocksDF.orderBy(col("Close").desc).show(5, false)
+----------+-----+-----+
|Date      |Stock|Close|
+----------+-----+-----+
|2025-08-04|MSFT |330  |
|2025-08-02|MSFT |325  |
|2025-08-03|MSFT |322  |
|2025-08-01|MSFT |320  |
|2025-08-03|TSLA |220  |
+----------+-----+-----+
only showing top 5 rows
```

4) **Count the total number of records in the stock DataFrame.**

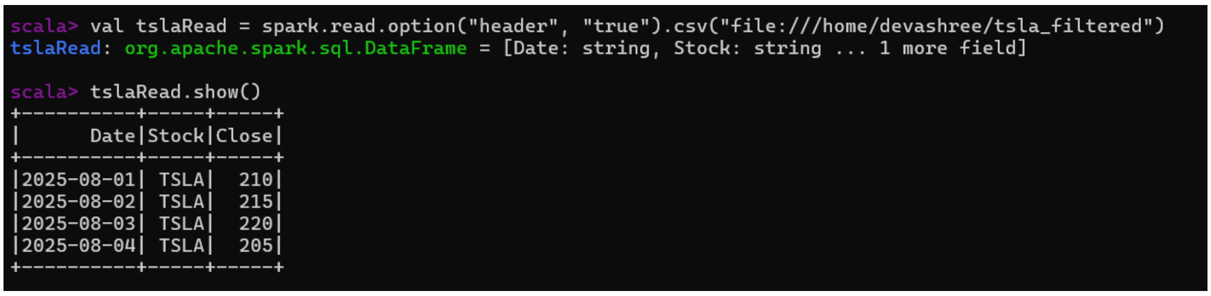
println("Total records: " + stocksDF.count())


```
scala> println("Total number of records: " + stocksDF.count())
Total number of records: 12
```

5) **Save the filtered data for TSLA into a separate directory in HDFS.**

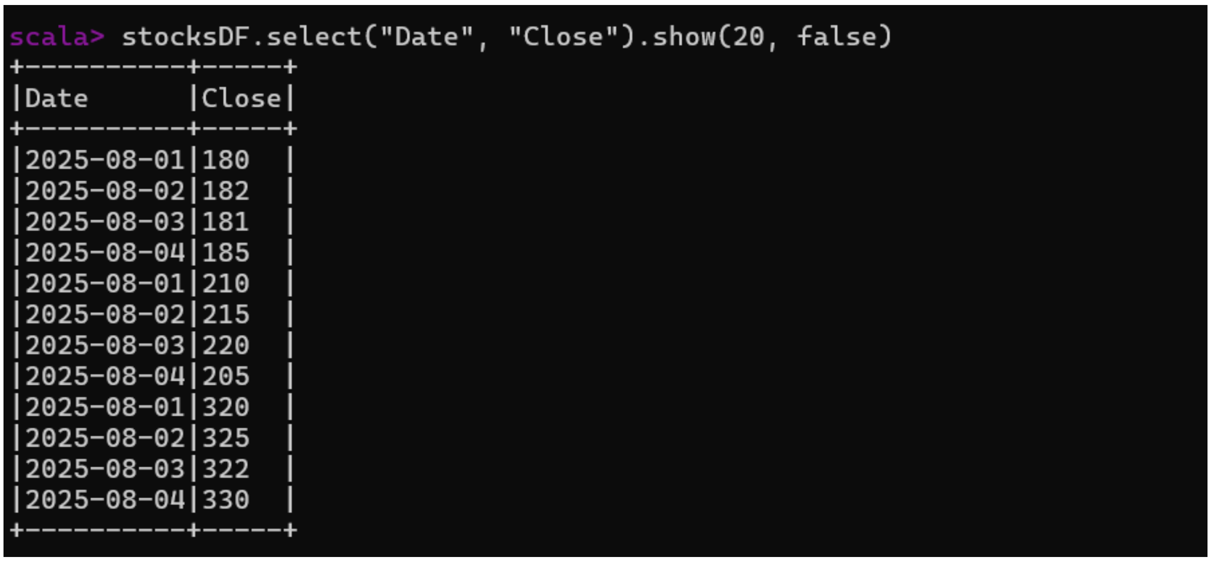
valtslaRead=spark.read.option("header","true").csv("file:///home/devashree/tsla_filtered")

tslaRead.show()


```
scala> val tslaRead = spark.read.option("header", "true").csv("file:///home/devashree/tsla_filtered")
tslaRead: org.apache.spark.sql.DataFrame = [Date: string, Stock: string ... 1 more field]

scala> tslaRead.show()
+----------+-----+-----+
|      Date|Stock|Close|
+----------+-----+-----+
|2025-08-01| TSLA|  210|
|2025-08-02| TSLA|  215|
|2025-08-03| TSLA|  220|
|2025-08-04| TSLA|  205|
+----------+-----+-----+
```

6) Display only the Date and Close columns for all stocks.

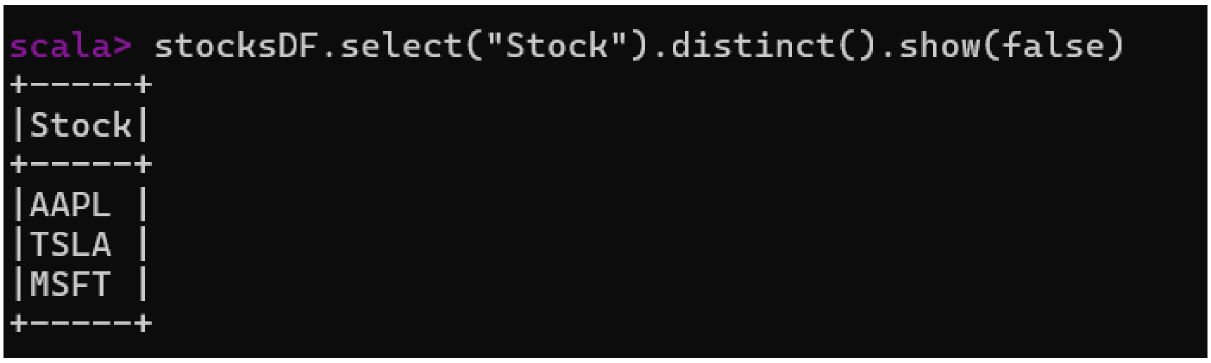
stocksDF.select("Date", "Close").show(20, false)



```
scala> stocksDF.select("Date", "Close").show(20, false)
+----------+-----+
|Date      |Close|
+----------+-----+
|2025-08-01|180  |
|2025-08-02|182  |
|2025-08-03|181  |
|2025-08-04|185  |
|2025-08-01|210  |
|2025-08-02|215  |
|2025-08-03|220  |
|2025-08-04|205  |
|2025-08-01|320  |
|2025-08-02|325  |
|2025-08-03|322  |
|2025-08-04|330  |
+----------+-----+
```

7) Find the unique stock symbols present in the dataset.

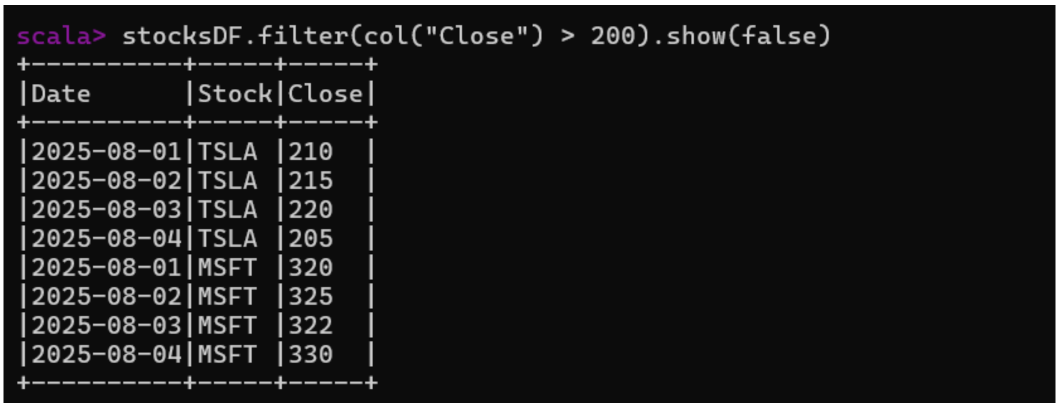
stocksDF.select("Stock").distinct().show(false)



```
scala> stocksDF.select("Stock").distinct().show(false)
+-----+
|Stock|
+-----+
|AAPL |
|TSLA |
|MSFT |
+-----+
```

8) Display only the records where the Close price is greater than 200.

stocksDF.filter(col("Close") > 200).show(false)



```
scala> stocksDF.filter(col("Close") > 200).show(false)
+----------+-----+-----+
|Date      |Stock|Close|
+----------+-----+-----+
|2025-08-01|TSLA |210  |
|2025-08-02|TSLA |215  |
|2025-08-03|TSLA |220  |
|2025-08-04|TSLA |205  |
|2025-08-01|MSFT |320  |
|2025-08-02|MSFT |325  |
|2025-08-03|MSFT |322  |
|2025-08-04|MSFT |330  |
+----------+-----+-----+
```

Preparation	**20**	
Implementation	**20**	
Viva	**15**	
Output	**10**	
Record	**10**	
Total	**75**	

Result:

Thus, the reading of stock market data into a DataFrame, use window functions to calculate the moving average price for each stock, and display the results.

EXP. NO:7(a)

Create a basic CI/CD pipeline for a sample application

Aim:

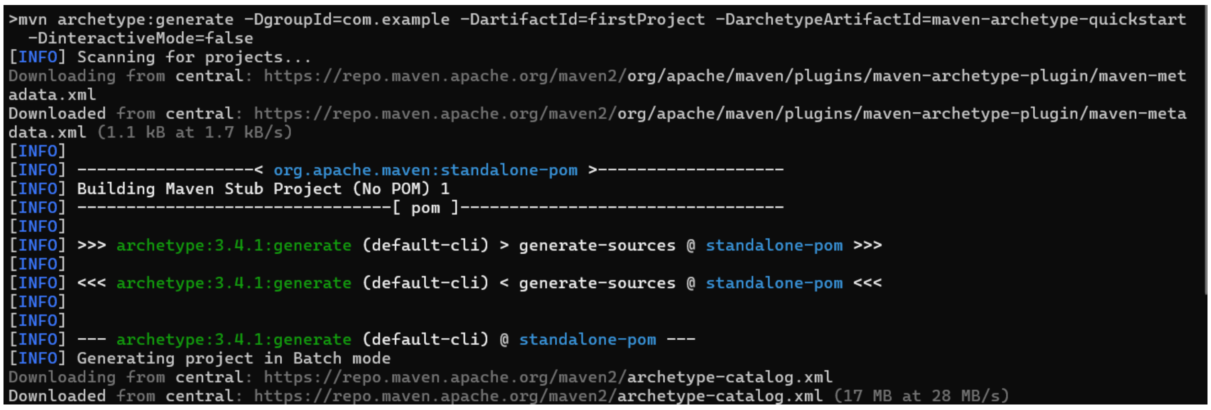
To Set up a version control repository (e.g., GitHub), configure a CI tool (e.g., Jenkins, GitLab CI), define stages for building, testing, and deploying the application, and trigger the pipeline on code commits.

Steps to be Implemented:

1) Create a Maven project (Hello World)

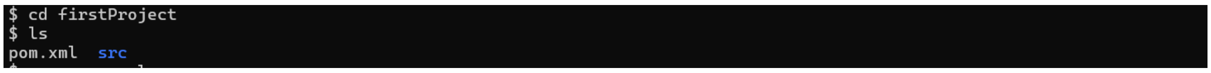
Open a terminal and run:

```
cd ~

mvn archetype:generate -DgroupId=com.example \

        -DartifactId=firstProject \

        -DarchetypeArtifactId=maven-archetype-quickstart \

        -DinteractiveMode=false

cd firstProject
```



```
>mvn archetype:generate -DgroupId=com.example -DartifactId=firstProject -DarchetypeArtifactId=maven-archetype-quickstart
  -DinteractiveMode=false
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/maven-met
adata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-archetype-plugin/maven-meta
data.xml (1.1 kB at 1.7 kB/s)
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> archetype:3.4.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< archetype:3.4.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- archetype:3.4.1:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
Downloading from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml
Downloaded from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml (17 MB at 28 MB/s)
```

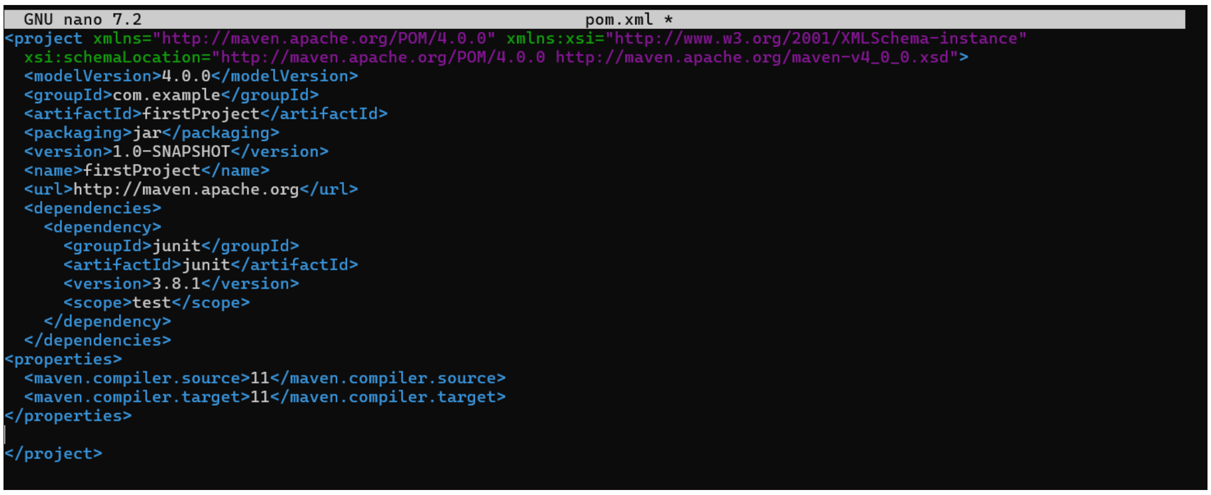
Verify: You should see a pom.xml and a src folder.



```
$ cd firstProject
$ ls
pom.xml  src
```

Set JDK version in pom.xml:

```
<properties>

  <maven.compiler.source>11</maven.compiler.source>

  <maven.compiler.target>11</maven.compiler.target>

</properties>
```

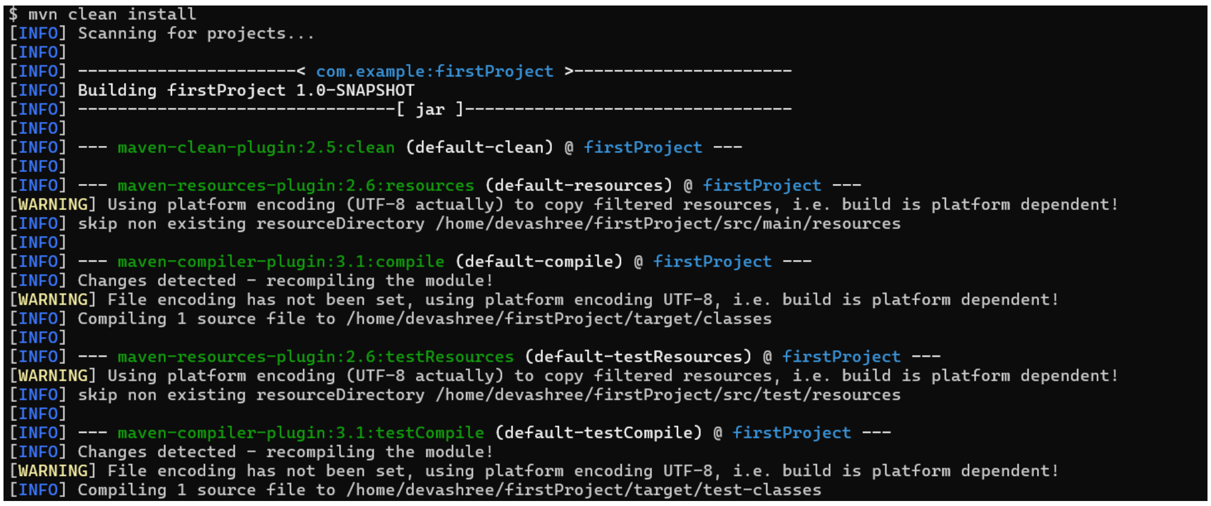


```
  GNU nano 7.2                                                    pom.xml *
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>firstProject</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>firstProject</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
</properties>

</project>
```

Verify: File compiles later via Maven without errors.

Build locally: mvn clean install

Verify: Look for BUILD SUCCESS and a JAR at target/firstProject-1.0-SNAPSHOT.jar.



```
$ mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.example:firstProject >----------------------
[INFO] Building firstProject 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ firstProject ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ firstProject ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/devashree/firstProject/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ firstProject ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /home/devashree/firstProject/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ firstProject ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/devashree/firstProject/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ firstProject ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /home/devashree/firstProject/target/test-classes
```

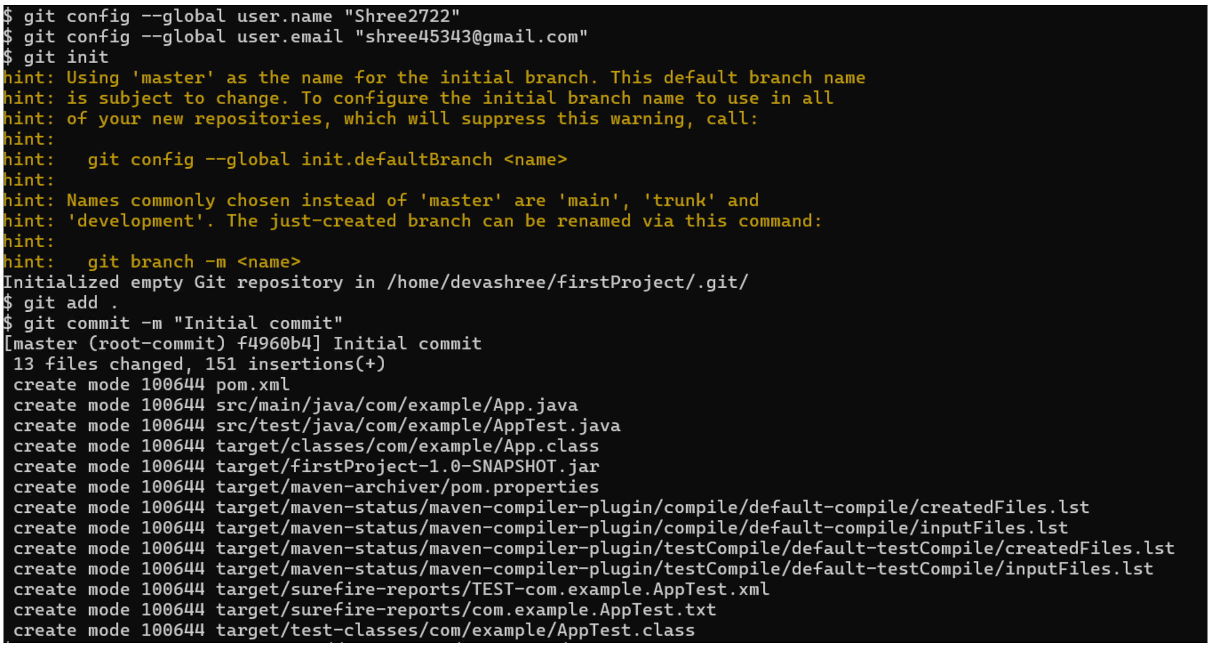
2) Initialize Git and make your first commit

Configure your Git identity (one-time):

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

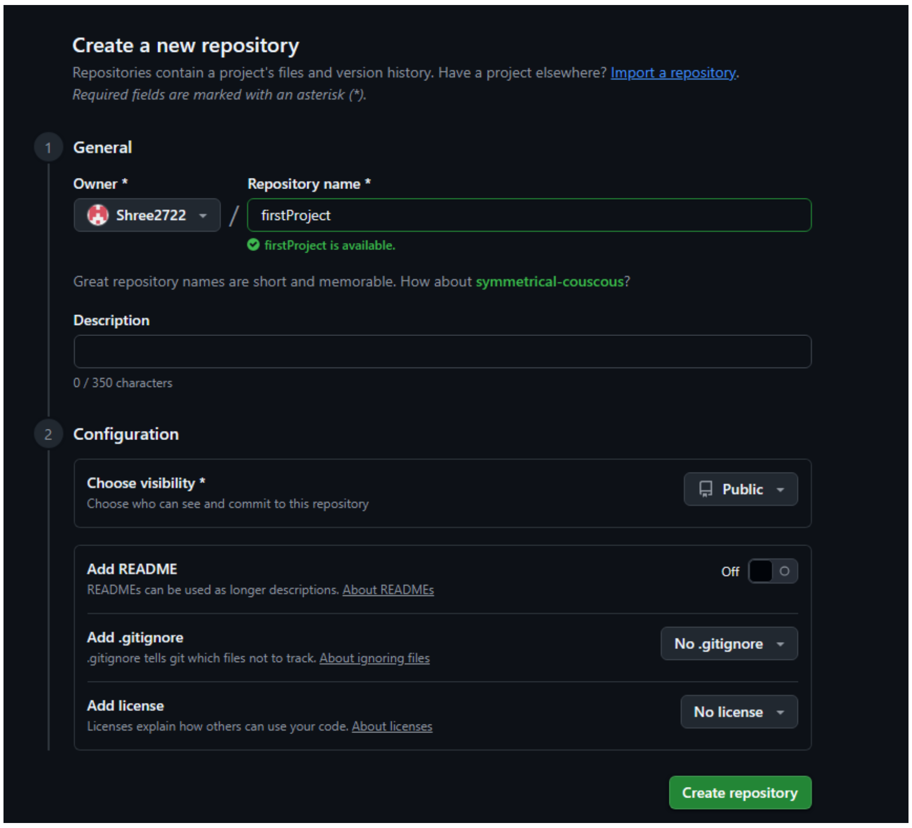
Initialize and commit:

```
git init
git add .
git commit -m "Initial commit"
```


```
$ git config --global user.name "Shree2722"
$ git config --global user.email "shree45343@gmail.com"
$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/devashree/firstProject/.git/
$ git add .
$ git commit -m "Initial commit"
[master (root-commit) f4960b4] Initial commit
 13 files changed, 151 insertions(+)
 create mode 100644 pom.xml
 create mode 100644 src/main/java/com/example/App.java
 create mode 100644 src/test/java/com/example/AppTest.java
 create mode 100644 target/classes/com/example/App.class
 create mode 100644 target/firstProject-1.0-SNAPSHOT.jar
 create mode 100644 target/maven-archiver/pom.properties
 create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/createdFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/inputFiles.lst
 create mode 100644 target/surefire-reports/TEST-com.example.AppTest.xml
 create mode 100644 target/surefire-reports/com.example.AppTest.txt
 create mode 100644 target/test-classes/com/example/AppTest.class
```

3) Push the project to GitHub

Create an empty GitHub repository named firstProject (no README).


Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

1 General

Owner *: Shree2722 / Repository name *: firstProject

firstProject is available.

Great repository names are short and memorable. How about symmetrical-couscous?

Description

0 / 350 characters

2 Configuration

Choose visibility * — Choose who can see and commit to this repository: Public

Add README — READMEs can be used as longer descriptions. About READMEs: Off

Add .gitignore — .gitignore tells git which files not to track. About ignoring files: No .gitignore

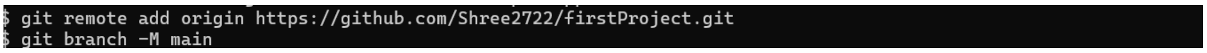
Add license — Licenses explain how others can use your code. About licenses: No license

Connect and push:

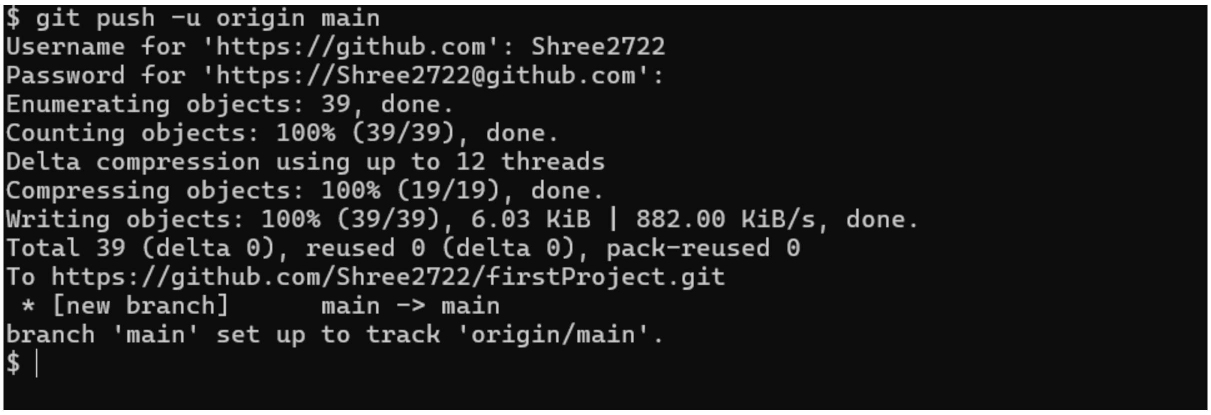
git remote add origin https://github.com/<your-username>/firstProject.git

git branch -M main

git push -u origin main



```
$ git remote add origin https://github.com/Shree2722/firstProject.git
$ git branch -M main
```



```
$ git push -u origin main
Username for 'https://github.com': Shree2722
Password for 'https://Shree2722@github.com':
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Delta compression using up to 12 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (39/39), 6.03 KiB | 882.00 KiB/s, done.
Total 39 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Shree2722/firstProject.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
$ |
```

4) Install Jenkins (on Linux)

sudo apt update

sudo apt install openjdk-11-jdk -y

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins -y

sudo systemctl start jenkins

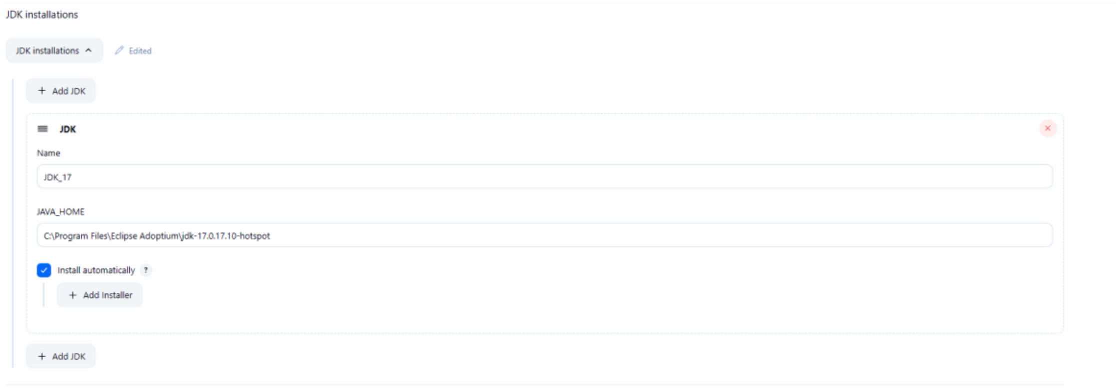
sudo systemctl enable jenkins

Verify:

- Visit http://<server-ip>:8080
- Unlock Jenkins with the initial admin password (/var/lib/jenkins/secrets/initialAdminPassword).

5) Configure tools and plugins in Jenkins

1. In Jenkins: **Manage Jenkins → Global Tool Configuration**
   - Add JDK 11 (or make sure "Install automatically" is set).
   - Add Maven (install automatically or specify a path).

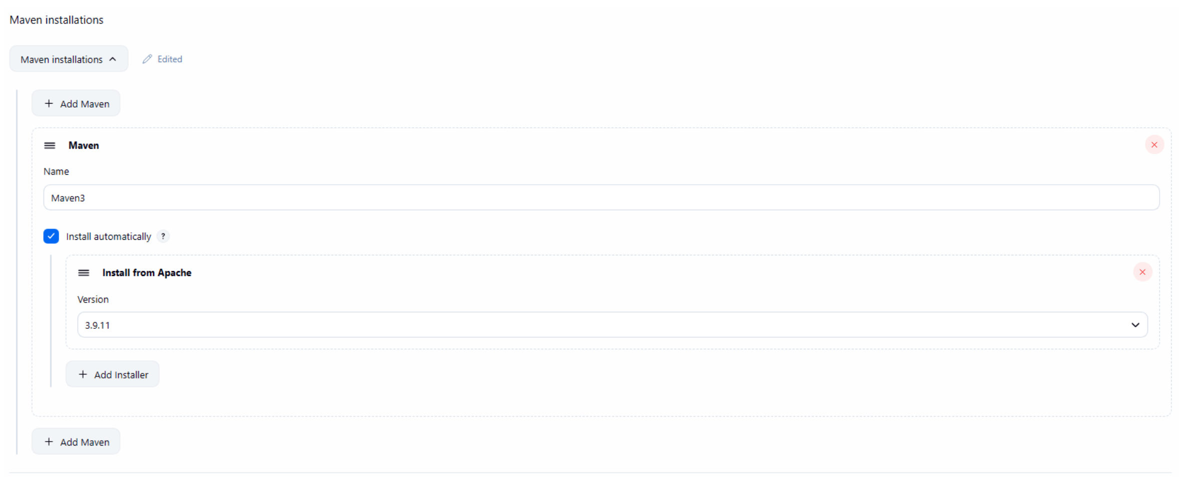


JDK installations

Name: JDK_17

JAVA_HOME: C:\Program Files\Eclipse Adoptium\jdk-17.0.17.10-hotspot

[✓] Install automatically


Maven installations

Maven installations ˄ Edited

+ Add Maven

Maven

Name

Maven3

☑ Install automatically ?

Install from Apache

Version

3.9.11

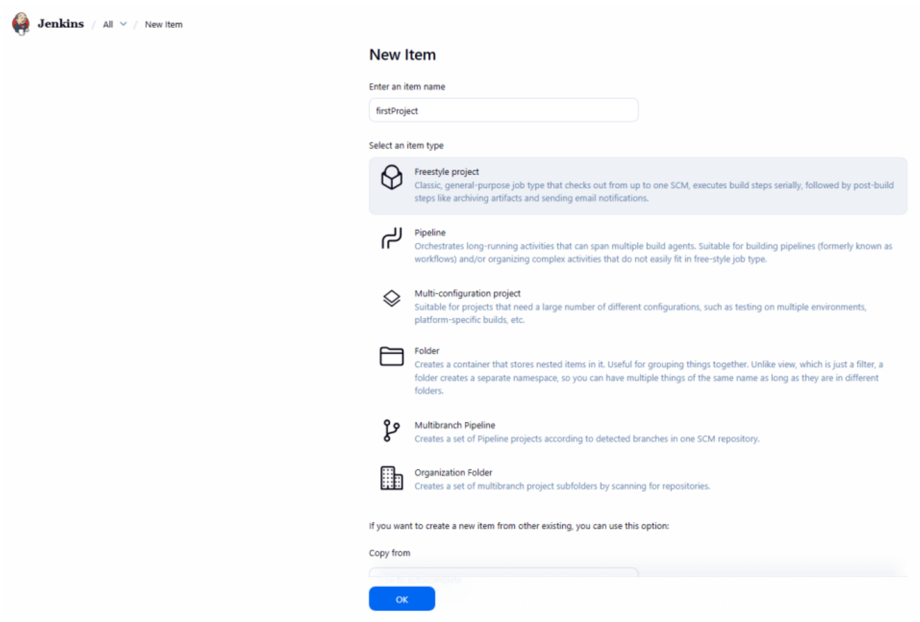
+ Add Installer

+ Add Maven

2. Install plugins: **Git** and **Maven Integration** (via **Manage Jenkins → Plugins**). Verify: You can see JDK and Maven listed under Global Tool Configuration without warnings.

6) Create a Jenkins Maven job

1. **New Item → Maven project**
   Name: firstProject


Jenkins / All / New Item

New Item

Enter an item name

firstProject

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

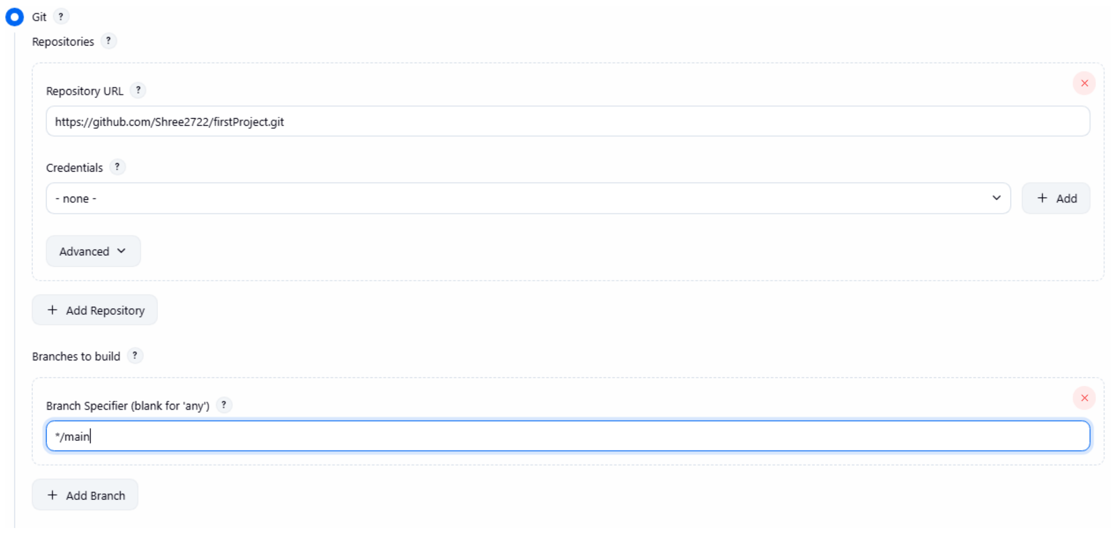
Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from

OK

2. **Source Code Management → Git**
   - Repository URL: https://github.com/<your-username>/firstProject.git
   - Credentials: Add your GitHub credentials if the repo is private
   - Branch: main


Git ?

Repositories ?

Repository URL ?

https://github.com/Shree2722/firstProject.git

Credentials ?

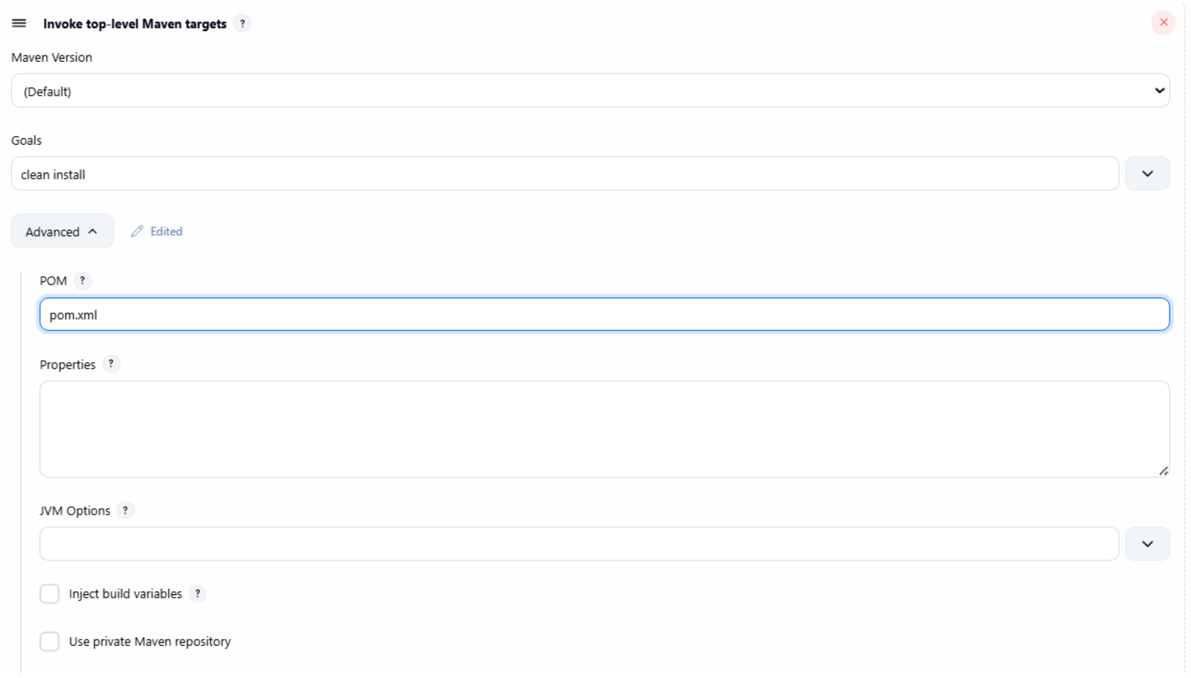
- none -

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

3. **Build**
   - Root POM: pom.xml
   - Goals: clean install
4. Save.
   Verify: Job shows your repo URL and branch correctly.


Invoke top-level Maven targets ?

Maven Version

(Default)

Goals

clean install

POM ?

pom.xml

Properties ?

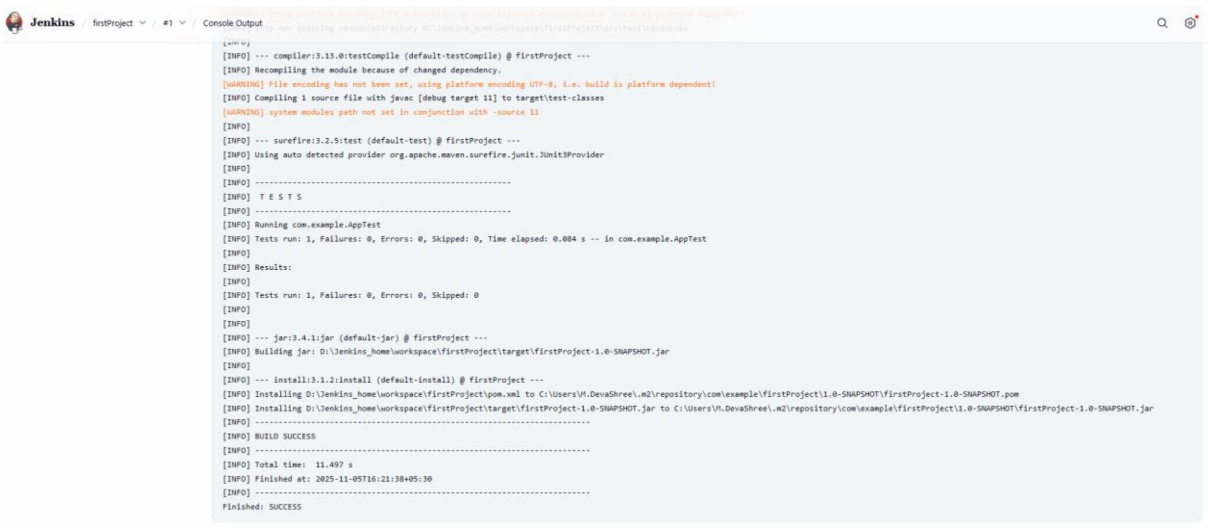
JVM Options ?

☐ Inject build variables ?

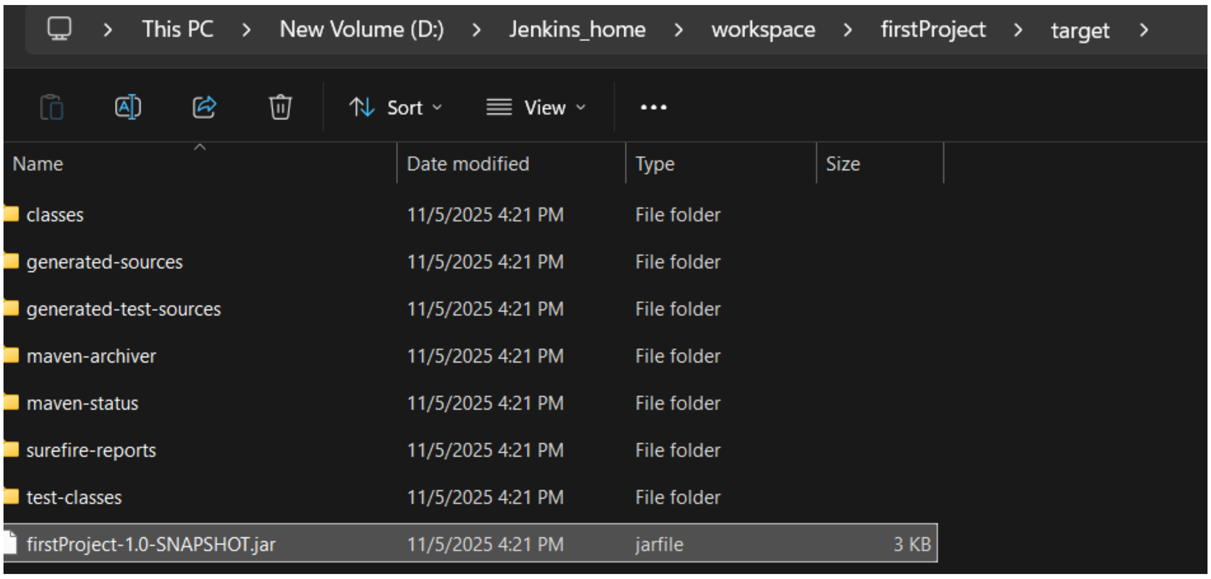
☐ Use private Maven repository

7) Run the pipeline and confirm artifacts

1. Click **Build Now**.
2. Open **Console Output**.
   Verify: You should see BUILD SUCCESS. Also check the workspace:
3. <JENKINS_HOME>/workspace/firstProject/target/firstProject-1.0-SNAPSHOT.jar



```
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ firstProject ---
[INFO] Recompiling the module because of changed dependency.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 11] to target\test-classes
[WARNING] system modules path not set in conjunction with -source 11
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ firstProject ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit.JUnit3Provider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.084 s -- in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ firstProject ---
[INFO] Building jar: D:\Jenkins_home\workspace\firstProject\target\firstProject-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- install:3.1.2:install (default-install) @ firstProject ---
[INFO] Installing D:\Jenkins_home\workspace\firstProject\pom.xml to C:\Users\M.DevaShree\.m2\repository\com\example\firstProject\1.0-SNAPSHOT\firstProject-1.0-SNAPSHOT.pom
[INFO] Installing D:\Jenkins_home\workspace\firstProject\target\firstProject-1.0-SNAPSHOT.jar to C:\Users\M.DevaShree\.m2\repository\com\example\firstProject\1.0-SNAPSHOT\firstProject-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  11.497 s
[INFO] Finished at: 2025-11-05T16:21:38+05:30
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```


This PC > New Volume (D:) > Jenkins_home > workspace > firstProject > target >
Sort
View
•••

Name	Date modified	Type	Size
classes	11/5/2025 4:21 PM	File folder	
generated-sources	11/5/2025 4:21 PM	File folder	
generated-test-sources	11/5/2025 4:21 PM	File folder	
maven-archiver	11/5/2025 4:21 PM	File folder	
maven-status	11/5/2025 4:21 PM	File folder	
surefire-reports	11/5/2025 4:21 PM	File folder	
test-classes	11/5/2025 4:21 PM	File folder	
firstProject-1.0-SNAPSHOT.jar	11/5/2025 4:21 PM	jarfile	3 KB

Preparation	**20**	
Implementation	**20**	
Viva	**15**	
Output	**10**	
Record	**10**	
Total	**75**	

Result:

Thus, the CI pipeline was successfully set up using GitHub and Jenkins, which automatically built and tested the application whenever new code was committed.

EXP. NO:7(b)

Dockerize the application and deploy it using containers

Aim:

To write a Dockerfile to package the application into a container, set up a container registry (e.g., Docker Hub, Amazon ECR), and deploy the containerized application to a container orchestration platform (e.g., Kubernetes, Docker Swarm) using the CI/CD pipeline..

Prerequisites:

- **Docker:** A platform to build, package, and run applications in **containers**, which are lightweight, portable, and isolated environments.
- **Kubernetes (K8s):** A container orchestration system that manages deployment, scaling, and networking of containers across clusters.
- **CI/CD:** Continuous Integration and Continuous Deployment pipelines automate building, testing, and deploying code.

Requirements:

1. **Windows 10/11 with WSL2**
   - WSL2 allows Linux-based containers to run on Windows seamlessly.
2. **Docker Desktop**
   - Provides Docker Engine, CLI, and optional Kubernetes cluster for local testing.
3. **Node.js Project**
   - Application to deploy; in this manual, an Express.js app.

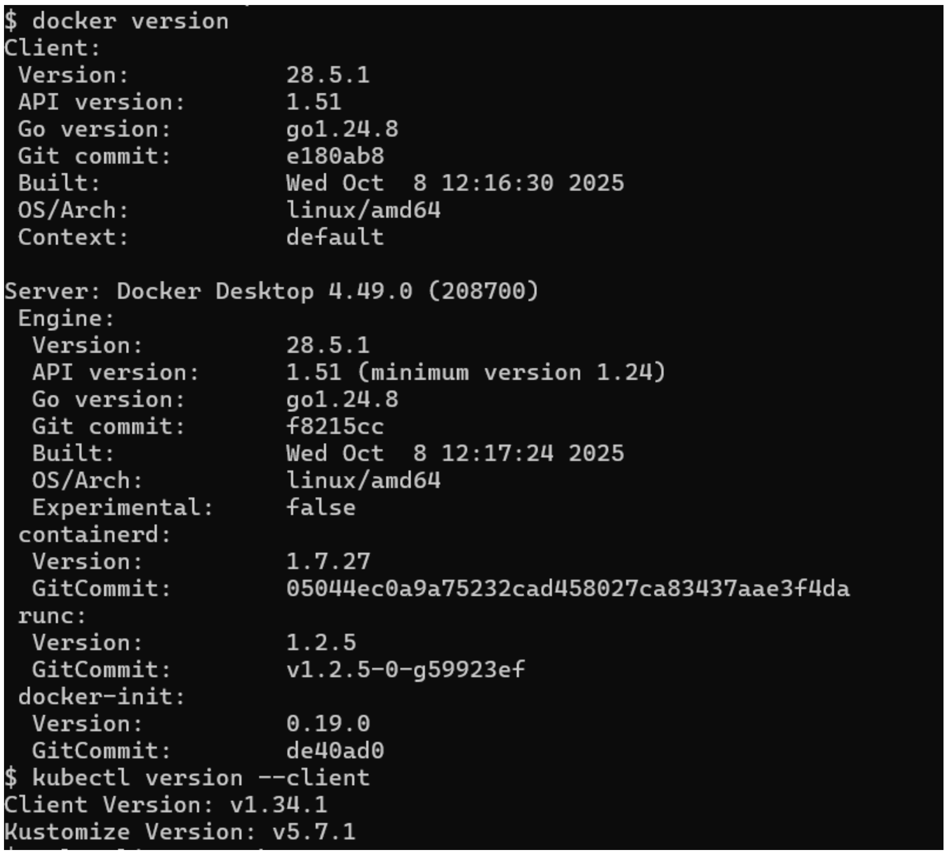
Step 1: Install Docker Desktop & Enable Kubernetes

- Docker Desktop includes Docker Engine + Docker CLI + optional Kubernetes cluster.
- Enabling Kubernetes allows you to test container orchestration locally without setting up a full cloud cluster.

Steps:-

1. Install Docker Desktop from docker.com.
2. Settings → Enable WSL2 Integration → Select your distro.
3. Settings → Kubernetes → Enable Kubernetes → Apply & Restart.
4. Verify:

```
docker version
kubectl version --client
wsl --list –verbose
```



```
$ docker version
Client:
 Version:           28.5.1
 API version:       1.51
 Go version:        go1.24.8
 Git commit:        e180ab8
 Built:             Wed Oct  8 12:16:30 2025
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Desktop 4.49.0 (208700)
 Engine:
  Version:          28.5.1
  API version:      1.51 (minimum version 1.24)
  Go version:       go1.24.8
  Git commit:       f8215cc
  Built:            Wed Oct  8 12:17:24 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.27
  GitCommit:        05044ec0a9a75232cad458027ca83437aae3f4da
 runc:
  Version:          1.2.5
  GitCommit:        v1.2.5-0-g59923ef
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
$ kubectl version --client
Client Version: v1.34.1
Kustomize Version: v5.7.1
```

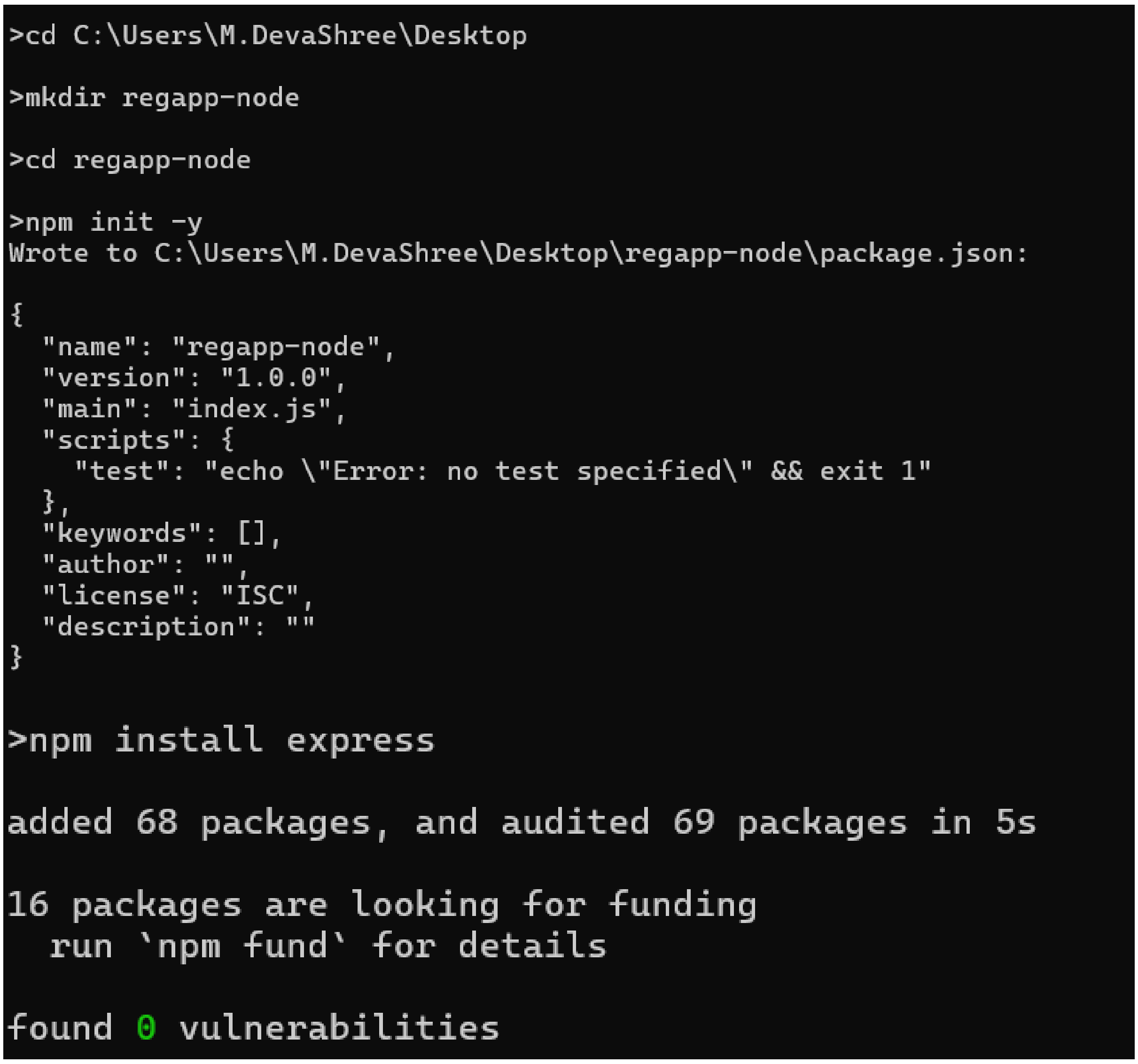
- docker version - checks Docker Engine and CLI.
- kubectl version - checks Kubernetes client connectivity.
- wsl --list --verbose - confirms WSL2 distro status.

Step 2: Create Node.js Project

- Node.js is a runtime environment for JavaScript outside the browser.
- Express.js is a minimal web framework for Node.js, used for handling HTTP requests.

Run the below commands:-

mkdir C:\Users\welcome\Desktop\regapp-node
cd C:\Users\welcome\Desktop\regapp-node
npm init -y
npm install express



```
>cd C:\Users\M.DevaShree\Desktop

>mkdir regapp-node

>cd regapp-node

>npm init -y
Wrote to C:\Users\M.DevaShree\Desktop\regapp-node\package.json:

{
  "name": "regapp-node",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

>npm install express

added 68 packages, and audited 69 packages in 5s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Create app.js:

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 8080;

app.get('/', (req, res) => res.send('Hello from Node.js Docker App!'));

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

- process.env.PORT → allows Kubernetes or Docker to map container ports dynamically.
- app.get('/', ...) → basic HTTP GET route.

Step 3: Dockerize Node.js App

- Dockerfile defines how to build a container image.
- Layers in Docker improve efficiency (cached during builds).

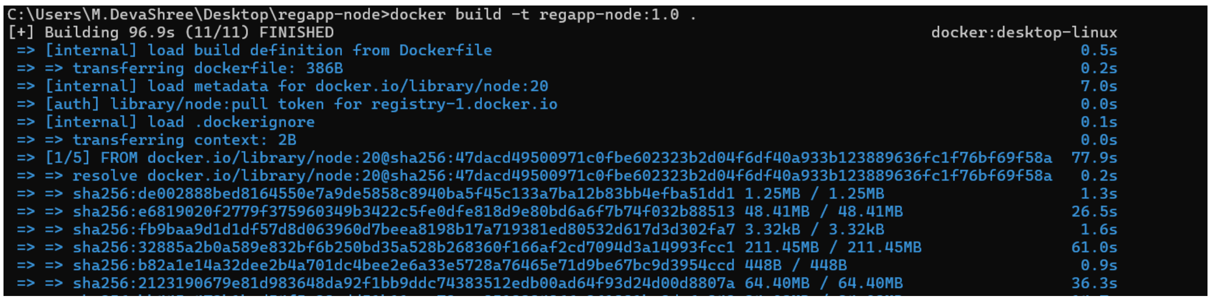
Create Dockerfile:
Paste the above code in the Dockerfile.

```
FROM node:20
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD ["npm", "start"]
```

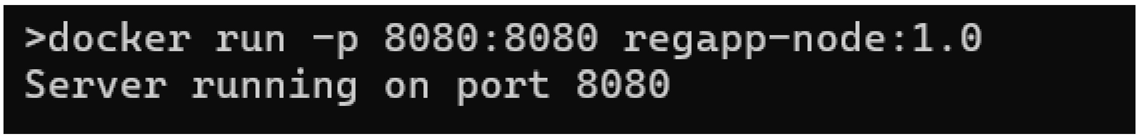
Run in the Powershell.

```
docker build -t regapp-node:1.0 .
docker run -p 8080:8080 regapp-node:1.0
```

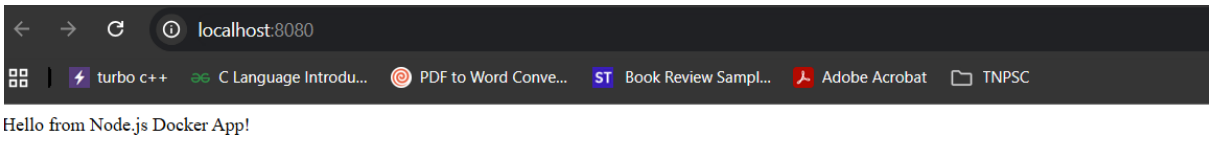
- EXPOSE → tells Docker which port the container listens on.
- CMD → default command executed when the container starts.
- docker run -p → maps container port to host port.



```
C:\Users\M.DevaShree\Desktop\regapp-node>docker build -t regapp-node:1.0 .
[+] Building 96.9s (11/11) FINISHED                                                     docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                    0.5s
 => => transferring dockerfile: 386B                                                                    0.2s
 => [internal] load metadata for docker.io/library/node:20                                              7.0s
 => [auth] library/node:pull token for registry-1.docker.io                                             0.0s
 => [internal] load .dockerignore                                                                       0.1s
 => => transferring context: 2B                                                                         0.0s
 => [1/5] FROM docker.io/library/node:20@sha256:47dacd49500971c0fbe602323b2d04f6df40a933b123889636fc1f76bf69f58a  77.9s
 => => resolve docker.io/library/node:20@sha256:47dacd49500971c0fbe602323b2d04f6df40a933b123889636fc1f76bf69f58a   0.2s
 => => sha256:de002888bed8164550e7a9de5858c8940ba5f45c133a7ba12b83bb4efba51dd1 1.25MB / 1.25MB           1.3s
 => => sha256:e6819020f2779f375960349b3422c5fe0dfe818d9e80bd6a6f7b74f032b88513 48.41MB / 48.41MB        26.5s
 => => sha256:fb9baa9d1d1df57d8d063960d7beea8198b17a719381ed80532d617d3d302fa7 3.32kB / 3.32kB           1.6s
 => => sha256:32885a2b0a589e832bf6b250bd35a528b268360f166af2cd7094d3a14993fcc1 211.45MB / 211.45MB      61.0s
 => => sha256:b82a1e14a32dee2b4a701dc4bee2e6a33e5728a76465e71d9be67bc9d3954ccd 448B / 448B               0.9s
 => => sha256:2123190679e81d983648da92f1bb9ddc74383512edb00ad64f93d24d00d8807a 64.40MB / 64.40MB        36.3s
```



```
>docker run -p 8080:8080 regapp-node:1.0
Server running on port 8080
```

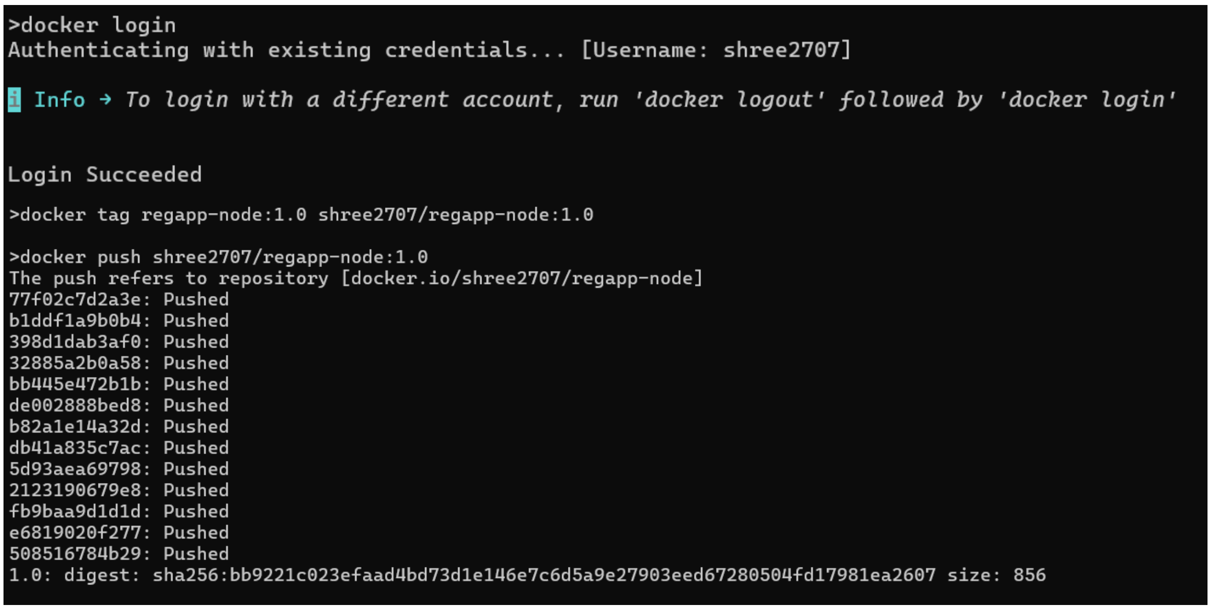


Hello from Node.js Docker App!

Step 4: Push Docker Image to Docker Hub

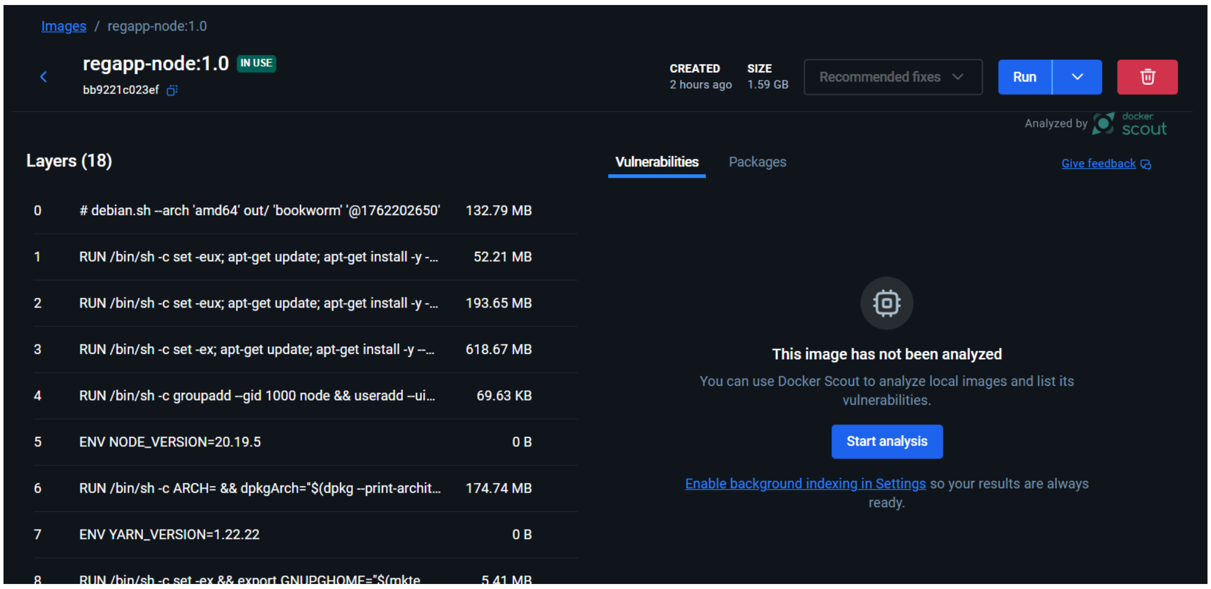
- Docker Hub is a **public container registry**.
- Pushing images allows any environment (like Kubernetes) to pull and run them.

docker login
docker tag regapp-node:1.0 <your-dockerhub-username>/regapp-node:1.0
docker push <your-dockerhub-username>/regapp-node:1.0



```
>docker login
Authenticating with existing credentials... [Username: shree2707]

i Info → To login with a different account, run 'docker logout' followed by 'docker login'


Login Succeeded

>docker tag regapp-node:1.0 shree2707/regapp-node:1.0

>docker push shree2707/regapp-node:1.0
The push refers to repository [docker.io/shree2707/regapp-node]
77f02c7d2a3e: Pushed
b1ddf1a9b0b4: Pushed
398d1dab3af0: Pushed
32885a2b0a58: Pushed
bb445e472b1b: Pushed
de002888bed8: Pushed
b82a1e14a32d: Pushed
db41a835c7ac: Pushed
5d93aea69798: Pushed
2123190679e8: Pushed
fb9baa9d1d1d: Pushed
e6819020f277: Pushed
508516784b29: Pushed
1.0: digest: sha256:bb9221c023efaad4bd73d1e146e7c6d5a9e27903eed67280504fd17981ea2607 size: 856
```

- Verify image on Docker Hub.



Images / regapp-node:1.0

regapp-node:1.0 IN USE
bb9221c023ef

CREATED 2 hours ago
SIZE 1.59 GB

Layers (18)

```
0   # debian.sh --arch 'amd64' out/ 'bookworm' '@1762202650'    132.79 MB
1   RUN /bin/sh -c set -eux; apt-get update; apt-get install -y -...   52.21 MB
2   RUN /bin/sh -c set -eux; apt-get update; apt-get install -y -...  193.65 MB
3   RUN /bin/sh -c set -ex; apt-get update; apt-get install -y --...  618.67 MB
4   RUN /bin/sh -c groupadd --gid 1000 node && useradd --ui...         69.63 KB
5   ENV NODE_VERSION=20.19.5                                                0 B
6   RUN /bin/sh -c ARCH= && dpkgArch="$(dpkg --print-archit...        174.74 MB
7   ENV YARN_VERSION=1.22.22                                                0 B
8   RUN /bin/sh -c set -ex && export GNUPGHOME="$(mkte                  5.41 MB
```

This image has not been analyzed

You can use Docker Scout to analyze local images and list its vulnerabilities.

Enable background indexing in Settings so your results are always ready.

Step 5: Kubernetes Deployment

- Deployment: defines pods (replicas of containers) and manages rolling updates.
- Service: exposes pods to other pods, external network, or both.
- LoadBalancer: allows external access (Docker Desktop assigns a local IP or use port-forward).

Deployment YAML (regapp-node-deploy.yml):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: regapp-node-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: regapp-node
  template:
    metadata:
      labels:
        app: regapp-node
    spec:
      containers:
      - name: regapp-node
        image: <your-dockerhub-username>/regapp-node:1.0
        ports:
        - containerPort: 8080
```

Service YAML (regapp-node-service.yml):

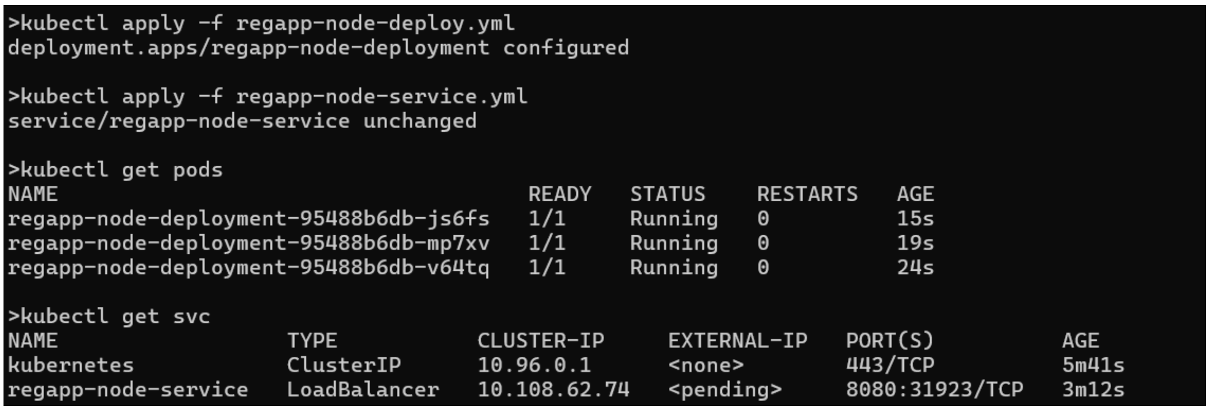
```
apiVersion: v1
kind: Service
metadata:
  name: regapp-node-service
spec:
  selector:
    app: regapp-node
  ports:
    - port: 8080
      targetPort: 8080
  type: LoadBalancer
```

Step 6: Apply Kubernetes Manifests

Run in the Powershell.

kubectl config use-context docker-desktop
kubectl apply -f regapp-node-deploy.yml
kubectl apply -f regapp-node-service.yml
kubectl get pods
kubectl get svc

- kubectl apply → applies the configuration in YAML.
- kubectl get pods → shows running pods.
- kubectl get svc → shows services and external access info.



```
>kubectl apply -f regapp-node-deploy.yml
deployment.apps/regapp-node-deployment configured

>kubectl apply -f regapp-node-service.yml
service/regapp-node-service unchanged

>kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
regapp-node-deployment-95488b6db-js6fs    1/1     Running   0          15s
regapp-node-deployment-95488b6db-mp7xv    1/1     Running   0          19s
regapp-node-deployment-95488b6db-v64tq    1/1     Running   0          24s

>kubectl get svc
NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP      10.96.0.1      <none>        443/TCP          5m41s
regapp-node-service   LoadBalancer   10.108.62.74   <pending>     8080:31923/TCP   3m12s
```

Step 7: Access Node.js App

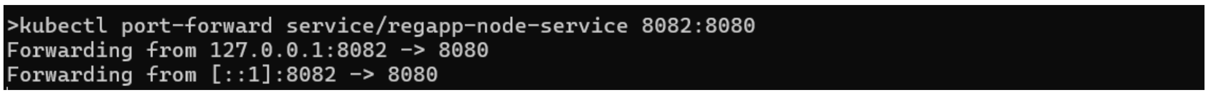
Port-forward:

kubectl port-forward service/regapp-node-service 8082:8080

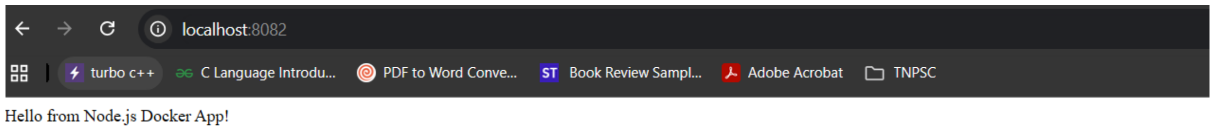
Open http://localhost:8082

verify the app is running.

- Port-forwarding is useful when LoadBalancer is not available locally.
- Pods are ephemeral; if a pod dies, Kubernetes automatically restarts it.



```
>kubectl port-forward service/regapp-node-service 8082:8080
Forwarding from 127.0.0.1:8082 -> 8080
Forwarding from [::1]:8082 -> 8080
```



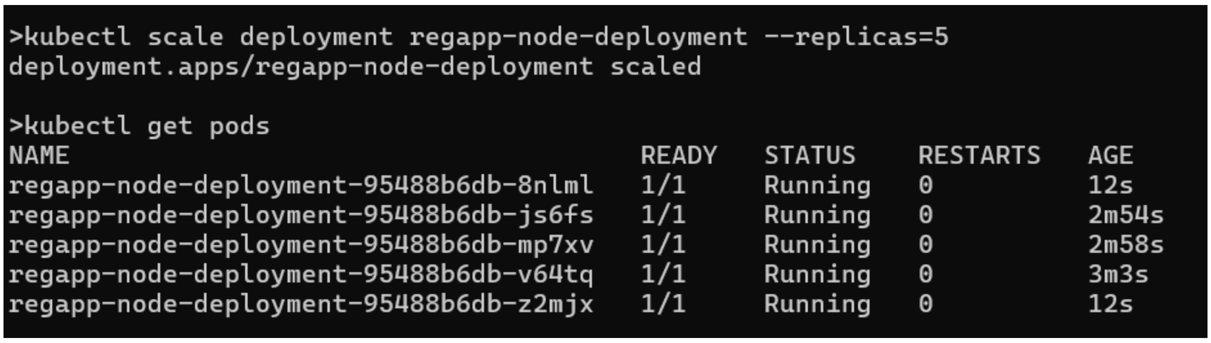
Hello from Node.js Docker App!

Step 8: Scale Deployment

Run in the Powershell.

```
kubectl scale deployment regapp-node-deployment --replicas=5
kubectl get pods
```

- Kubernetes scales pods automatically.
- ReplicaSets ensure the desired number of pods are running at all times.



```
>kubectl scale deployment regapp-node-deployment --replicas=5
deployment.apps/regapp-node-deployment scaled

>kubectl get pods
NAME                                      READY   STATUS    RESTARTS   AGE
regapp-node-deployment-95488b6db-8nlml   1/1     Running   0          12s
regapp-node-deployment-95488b6db-js6fs   1/1     Running   0          2m54s
regapp-node-deployment-95488b6db-mp7xv   1/1     Running   0          2m58s
regapp-node-deployment-95488b6db-v64tq   1/1     Running   0          3m3s
regapp-node-deployment-95488b6db-z2mjx   1/1     Running   0          12s
```

Step 9: Integrate CI/CD

The CI/CD pipeline operates as follows:

1. The developer writes or updates code and pushes it to the **main** branch of GitHub.
2. GitHub Actions automatically detects the change and triggers a workflow.
3. The workflow performs:
   - Code checkout
   - Environment setup
   - Dependency installation
   - Automatic build or test execution
4. If all steps succeed, the build is considered **successful**.

This ensures that the application is always in a **deployable and working state**.

GitHub Actions CI Workflow

The CI workflow is defined in: .github/workflows/ci.yml

Workflow File Contents

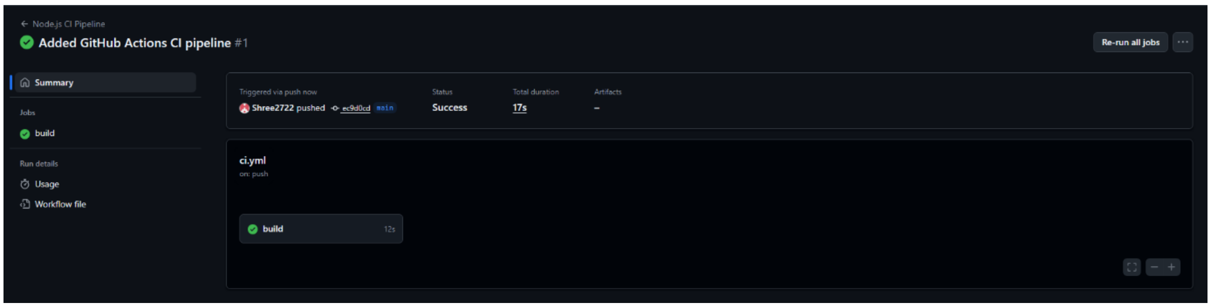
```
name: CI Pipeline

on:
  push:
    branches: ["main"]
  pull_request:
    branches: ["main"]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout Repository
      uses: actions/checkout@v3

    - name: Set up Node
      uses: actions/setup-node@v3
      with:
        node-version: 18

    - name: Install Dependencies
      run: npm install

    - name: Run Tests (Optional)
      run: npm test || echo "No tests available"

    - name: Build Application (Optional)
      run: npm run build || echo "No build step configured"
```

GitHub Actions CD Workflow

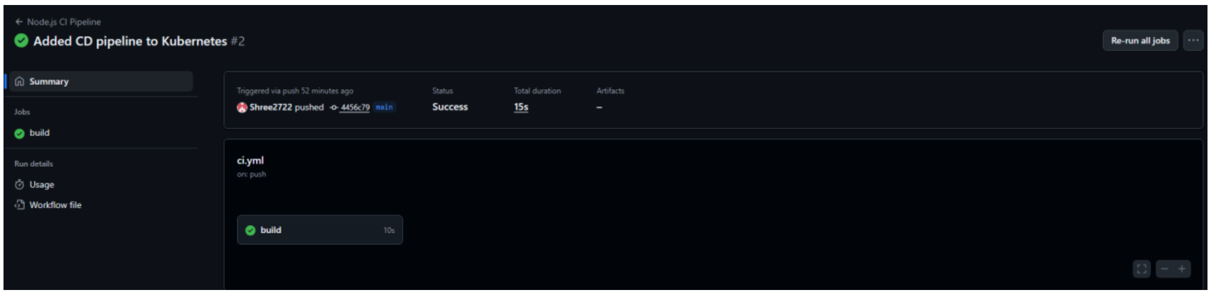
The CD workflow is defined in: .github/workflows/cd.yml

Workflow File Contents

```
name: CD to Docker Desktop Kubernetes

on:
  push:
    branches: ["main"]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Login to Docker Hub
        run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin

      - name: Build Docker Image
        run: docker build -t ${{ secrets.DOCKER_USERNAME }}/regapp-node:latest .

      - name: Push Docker Image
        run: docker push ${{ secrets.DOCKER_USERNAME }}/regapp-node:latest
      - name: Set up kubectl
        uses: azure/setup-kubectl@v3

      - name: Configure kubeconfig
        run: |
          mkdir -p ~/.kube
          echo "${{ secrets.KUBE_CONFIG }}" > ~/.kube/config

      - name: Deploy to Kubernetes
        run: |
          kubectl set image deployment/regapp-node-deployment regapp-node=${{
secrets.DOCKER_USERNAME }}/regapp-node:latest
          kubectl rollout status deployment/regapp-node-deployment
```



```
← Node.js CI Pipeline
Added GitHub Actions CI pipeline #1
Triggered via push now            Status     Total duration   Artifacts
Shree2722 pushed -o- ec9d0cd main Success    17s              –
ci.yml
on: push
build 12s
```



```
← Node.js CI Pipeline
Added CD pipeline to Kubernetes #2
Triggered via push 52 minutes ago  Status     Total duration   Artifacts
Shree2722 pushed -o- 4456c79 main  Success    15s              –
ci.yml
on: push
build 10s
```

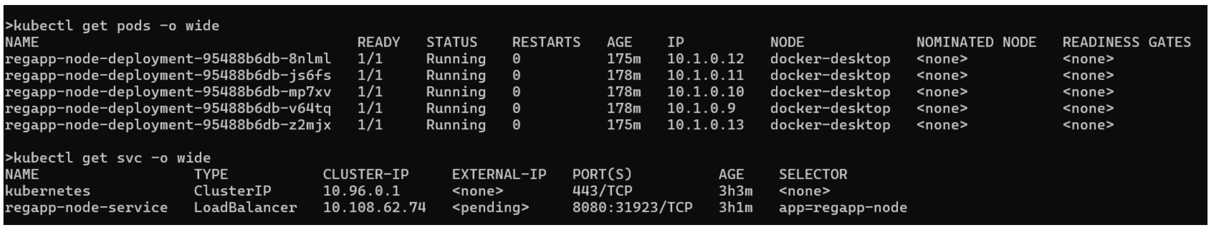
Step 10: Verification

Run in the Powershell.

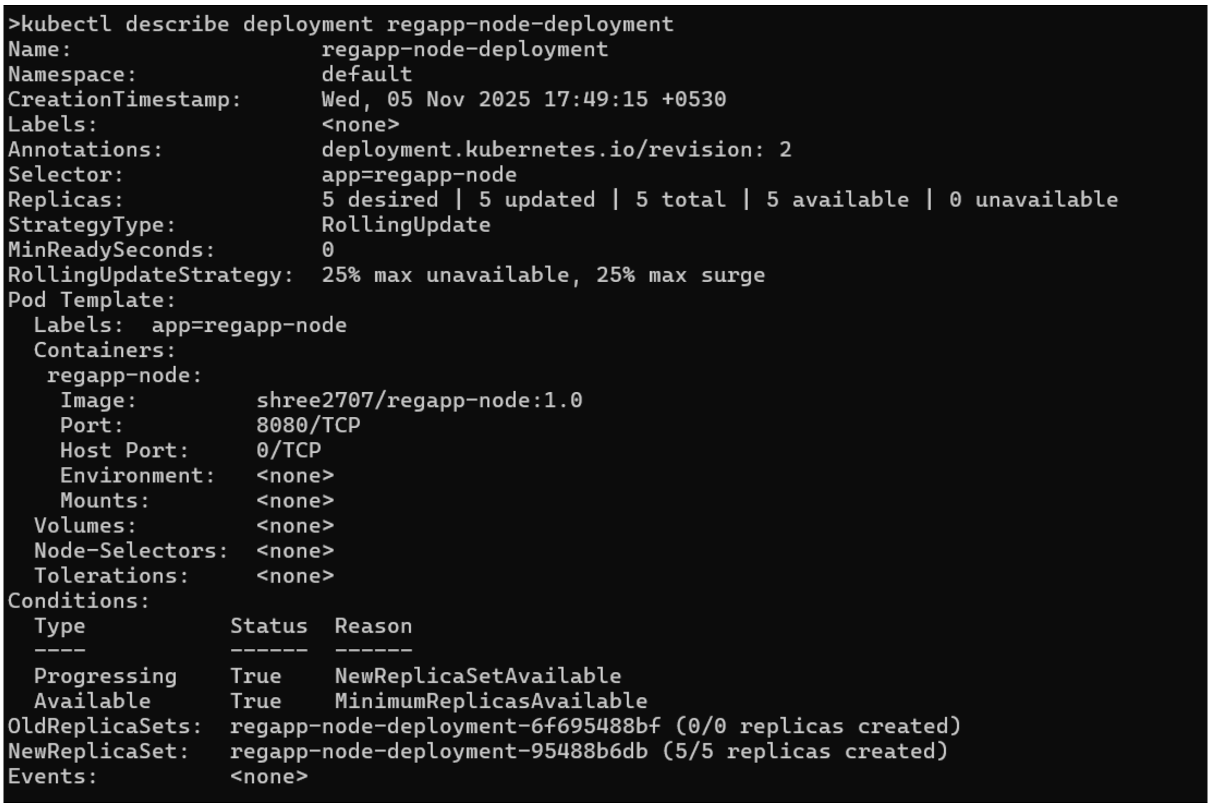
kubectl get pods -o wide

kubectl get svc -o wide

kubectl describe deployment regapp-node-deployment



```
>kubectl get pods -o wide
NAME                                     READY   STATUS    RESTARTS   AGE    IP          NODE             NOMINATED NODE   READINESS GATES
regapp-node-deployment-95488b6db-8nlml   1/1     Running   0          175m   10.1.0.12   docker-desktop   <none>           <none>
regapp-node-deployment-95488b6db-js6fs   1/1     Running   0          178m   10.1.0.11   docker-desktop   <none>           <none>
regapp-node-deployment-95488b6db-mp7xv   1/1     Running   0          178m   10.1.0.10   docker-desktop   <none>           <none>
regapp-node-deployment-95488b6db-v64tq   1/1     Running   0          178m   10.1.0.9    docker-desktop   <none>           <none>
regapp-node-deployment-95488b6db-z2mjx   1/1     Running   0          175m   10.1.0.13   docker-desktop   <none>           <none>

>kubectl get svc -o wide
NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE    SELECTOR
kubernetes            ClusterIP      10.96.0.1      <none>        443/TCP          3h3m   <none>
regapp-node-service   LoadBalancer   10.108.62.74   <pending>     8080:31923/TCP   3h1m   app=regapp-node
```



```
>kubectl describe deployment regapp-node-deployment
Name:                   regapp-node-deployment
Namespace:              default
CreationTimestamp:      Wed, 05 Nov 2025 17:49:15 +0530
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=regapp-node
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=regapp-node
  Containers:
   regapp-node:
    Image:         shree2707/regapp-node:1.0
    Port:          8080/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  regapp-node-deployment-6f695488bf (0/0 replicas created)
NewReplicaSet:   regapp-node-deployment-95488b6db (5/5 replicas created)
Events:          <none>
```

Preparation	**20**	
Implementation	**20**	
Viva	**15**	
Output	**10**	
Record	**10**	
Total	**75**	

Result:

Thus ,the application was successfully containerized using a Dockerfile, which packaged the source code along with all necessary dependencies into a portable container image.

EXP. NO: 8	Retail Sales Data ETL using MySQL

Aim:

To Develop an ETL process using SQL scripts to extract data from a source system (e.g., transactional database), transform it (e.g., aggregate, cleanse, join), and load it into a data warehouse.

1. Open MySQL

On Windows CMD or Terminal

mysql -u root -p

Then enter your MySQL password. After login, you'll see:

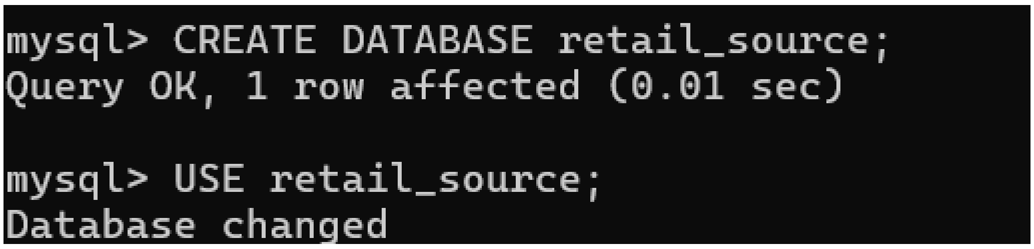
mysql>

2. Create Source Database (Transactional System)

Create the database that represents the retail transactions.

CREATE DATABASE retail_source;

USE retail_source;

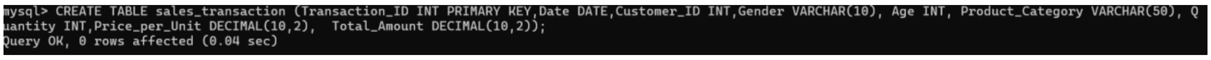


```
mysql> CREATE DATABASE retail_source;
Query OK, 1 row affected (0.01 sec)

mysql> USE retail_source;
Database changed
```

3. Create Source Table

This table stores raw sales transactions.

```
CREATE TABLE sales_transaction (
    Transaction_ID INT PRIMARY KEY,
    Date DATE,
    Customer_ID INT,
    Gender VARCHAR(10),
    Age INT,
    Product_Category VARCHAR(50),
    Quantity INT,
    Price_per_Unit DECIMAL(10,2),
    Total_Amount DECIMAL(10,2)
);
```



```
mysql> CREATE TABLE sales_transaction (Transaction_ID INT PRIMARY KEY,Date DATE,Customer_ID INT,Gender VARCHAR(10), Age INT, Product_Category VARCHAR(50), Q
uantity INT,Price_per_Unit DECIMAL(10,2),  Total_Amount DECIMAL(10,2));
Query OK, 0 rows affected (0.04 sec)
```

4. Insert Data

Insert a larger dataset to make ETL.
Sample Data:

INSERT INTO sales_transaction VALUES

(1, '2025-01-01', 101, 'Male',   275, 'Electronics', 2, 500.00, 1000.00),

(2, '2025-01-02', 102, 'Female', 30, 'Grocery',   5, 50.00, 250.00),

(3, '2025-01-02', 103, 'Male',   40, 'Clothing',   3, 200.00, 600.00),

(4, '2025-01-03', 104, 'Female', 22, 'Furniture',   1, 800.00, 800.00),

(5, '2025-01-03', 105, 'Male',   35, 'Clothing',   4, 150.00, 600.00),

(6, '2025-01-04', 106, 'Female', 28, 'Electronics', 1, 700.00, 700.00),

(7, '2025-01-04', 107, 'Male',   42, 'Grocery',   10, 40.00, 400.00),

(8, '2025-01-05', 108, 'Female', 33, 'Furniture',   2,1200.00, 2400.00),

(9, '2025-01-05', 109, 'Male',   27, 'Beauty',   5, 100.00, 500.00),

(10, '2025-01-06', 110, 'Female', 38, 'Electronics', 1, 900.00, 900.00),

(11, '2025-01-07', 111, 'Male',   29, 'Furniture',   1,1500.00, 1500.00),

(12, '2025-01-07', 112, 'Female', 24, 'Beauty',   3, 100.00, 300.00),

(13, '2025-01-08', 113, 'Male',   31, 'Beauty',   2, 200.00, 400.00),

(14, '2025-01-08', 114, 'Female', 41, 'Electronics', 2, 850.00, 1700.00),

(15, '2025-01-09', 115, 'Male',   37, 'Grocery',   8, 60.00, 480.00),

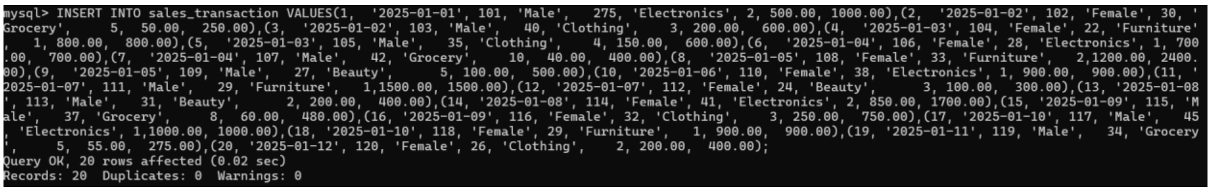
(16, '2025-01-09', 116, 'Female', 32, 'Clothing',   3, 250.00, 750.00),

(17, '2025-01-10', 117, 'Male',   45, 'Electronics', 1,1000.00, 1000.00),

(18, '2025-01-10', 118, 'Female', 29, 'Furniture',   1, 900.00, 900.00),

(19, '2025-01-11', 119, 'Male',   34, 'Grocery',   5, 55.00, 275.00),

(20, '2025-01-12', 120, 'Female', 26, 'Clothing',   2, 200.00, 400.00);



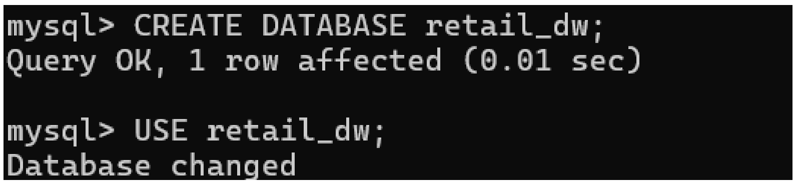
```
mysql> INSERT INTO sales_transaction VALUES(1,  '2025-01-01', 101, 'Male',   275, 'Electronics', 2, 500.00, 1000.00),(2,  '2025-01-02', 102, 'Female', 30, '
Grocery',     5,  50.00,  250.00),(3,  '2025-01-02', 103, 'Male',   40, 'Clothing',    3, 200.00,  600.00),(4,  '2025-01-03', 104, 'Female', 22, 'Furniture'
,   1, 800.00,  800.00),(5,  '2025-01-03', 105, 'Male',   35, 'Clothing',    4, 150.00,  600.00),(6,  '2025-01-04', 106, 'Female', 28, 'Electronics', 1, 700
.00,  700.00),(7,  '2025-01-04', 107, 'Male',   42, 'Grocery',    10,  40.00,  400.00),(8,  '2025-01-05', 108, 'Female', 33, 'Furniture',   2,1200.00, 2400.
00),(9,  '2025-01-05', 109, 'Male',   27, 'Beauty',      5, 100.00,  500.00),(10, '2025-01-06', 110, 'Female', 38, 'Electronics', 1, 900.00,  900.00),(11, '
2025-01-07', 111, 'Male',   29, 'Furniture',   1,1500.00, 1500.00),(12, '2025-01-07', 112, 'Female', 24, 'Beauty',      3, 100.00,  300.00),(13, '2025-01-08
', 113, 'Male',   31, 'Beauty',      2, 200.00,  400.00),(14, '2025-01-08', 114, 'Female', 41, 'Electronics', 2, 850.00, 1700.00),(15, '2025-01-09', 115, 'M
ale',   37, 'Grocery',     8,  60.00,  480.00),(16, '2025-01-09', 116, 'Female', 32, 'Clothing',    3, 250.00,  750.00),(17, '2025-01-10', 117, 'Male',   45
, 'Electronics', 1,1000.00, 1000.00),(18, '2025-01-10', 118, 'Female', 29, 'Furniture',   1, 900.00,  900.00),(19, '2025-01-11', 119, 'Male',   34, 'Grocery
',     5,  55.00,  275.00),(20, '2025-01-12', 120, 'Female', 26, 'Clothing',    2, 200.00,  400.00);
Query OK, 20 rows affected (0.02 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

5. Create Data Warehouse Database

This is where **transformed data** will be stored.

CREATE DATABASE retail_dw;

USE retail_dw;



```
mysql> CREATE DATABASE retail_dw;
Query OK, 1 row affected (0.01 sec)

mysql> USE retail_dw;
Database changed
```

6. Create Dimension Tables

1. Customer Dimension

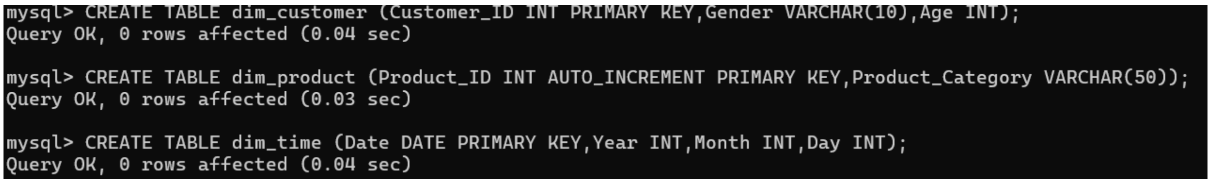
CREATE TABLE dim_customer (
Customer_ID INT PRIMARY KEY,
Gender VARCHAR(10),
Age INT
);

2. Product Dimension

CREATE TABLE dim_product (
Product_ID INT AUTO_INCREMENT PRIMARY KEY,
Product_Category VARCHAR(50)
);

3. Time Dimension

CREATE TABLE dim_time (
Date DATE PRIMARY KEY,
Year INT,
Month INT,
Day INT
);

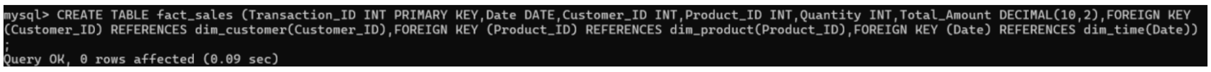


```
mysql> CREATE TABLE dim_customer (Customer_ID INT PRIMARY KEY,Gender VARCHAR(10),Age INT);
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE dim_product (Product_ID INT AUTO_INCREMENT PRIMARY KEY,Product_Category VARCHAR(50));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE dim_time (Date DATE PRIMARY KEY,Year INT,Month INT,Day INT);
Query OK, 0 rows affected (0.04 sec)
```

7. Create Fact Table

This stores the measurable data the sales transactions.

CREATE TABLE fact_sales (
Transaction_ID INT PRIMARY KEY,
Date DATE,
Customer_ID INT,
Product_ID INT,
Quantity INT,
Total_Amount DECIMAL(10,2),
FOREIGN KEY (Customer_ID) REFERENCES dim_customer(Customer_ID),
FOREIGN KEY (Product_ID) REFERENCES dim_product(Product_ID),
FOREIGN KEY (Date) REFERENCES dim_time(Date));



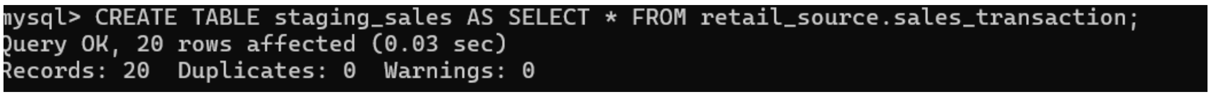
```
mysql> CREATE TABLE fact_sales (Transaction_ID INT PRIMARY KEY,Date DATE,Customer_ID INT,Product_ID INT,Quantity INT,Total_Amount DECIMAL(10,2),FOREIGN KEY
(Customer_ID) REFERENCES dim_customer(Customer_ID),FOREIGN KEY (Product_ID) REFERENCES dim_product(Product_ID),FOREIGN KEY (Date) REFERENCES dim_time(Date))
;
Query OK, 0 rows affected (0.09 sec)
```

8. Extract Data

Move raw data into staging tables before cleaning.

CREATE TABLE staging_sales AS

*SELECT * FROM retail_source.sales_transaction;*

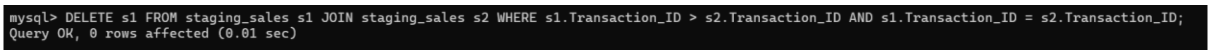


```
mysql> CREATE TABLE staging_sales AS SELECT * FROM retail_source.sales_transaction;
Query OK, 20 rows affected (0.03 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

9. Transform Data

->Remove duplicates

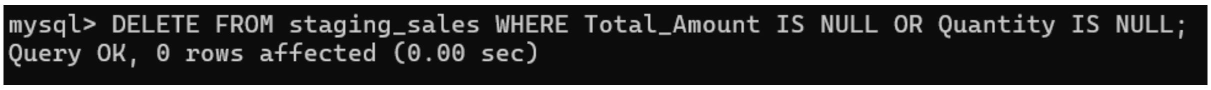
DELETE s1 FROM staging_sales s1
JOIN staging_sales s2
WHERE s1.Transaction_ID > s2.Transaction_ID
AND s1.Transaction_ID = s2.Transaction_ID;



```
mysql> DELETE s1 FROM staging_sales s1 JOIN staging_sales s2 WHERE s1.Transaction_ID > s2.Transaction_ID AND s1.Transaction_ID = s2.Transaction_ID;
Query OK, 0 rows affected (0.01 sec)
```

->Handle missing or invalid values

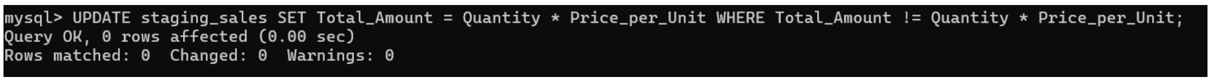
DELETE FROM staging_sales
WHERE Total_Amount IS NULL OR Quantity IS NULL;



```
mysql> DELETE FROM staging_sales WHERE Total_Amount IS NULL OR Quantity IS NULL;
Query OK, 0 rows affected (0.00 sec)
```

->Check inconsistencies

UPDATE staging_sales
*SET Total_Amount = Quantity * Price_per_Unit*
*WHERE Total_Amount != Quantity * Price_per_Unit;*



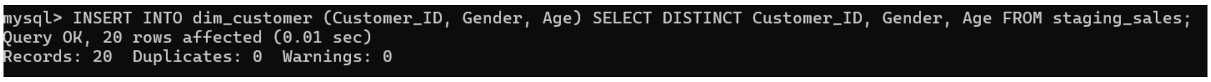
```
mysql> UPDATE staging_sales SET Total_Amount = Quantity * Price_per_Unit WHERE Total_Amount != Quantity * Price_per_Unit;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

10. Load Data into Dimensions

Customer

INSERT INTO dim_customer (Customer_ID, Gender, Age)

SELECT DISTINCT Customer_ID, Gender, Age FROM staging_sales;

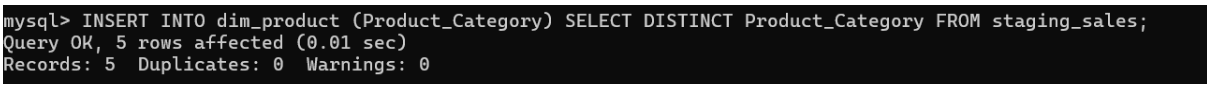


```
mysql> INSERT INTO dim_customer (Customer_ID, Gender, Age) SELECT DISTINCT Customer_ID, Gender, Age FROM staging_sales;
Query OK, 20 rows affected (0.01 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

Product

INSERT INTO dim_product (Product_Category)

SELECT DISTINCT Product_Category FROM staging_sales;



```
mysql> INSERT INTO dim_product (Product_Category) SELECT DISTINCT Product_Category FROM staging_sales;
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Time

INSERT INTO dim_time (Date, Year, Month, Day)

SELECT DISTINCT

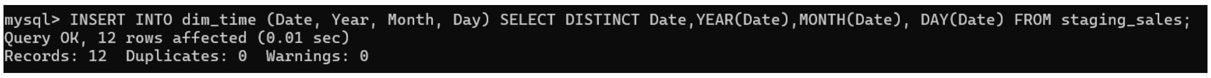
Date,

YEAR(Date),

MONTH(Date),

DAY(Date)

FROM staging_sales;



```
mysql> INSERT INTO dim_time (Date, Year, Month, Day) SELECT DISTINCT Date,YEAR(Date),MONTH(Date), DAY(Date) FROM staging_sales;
Query OK, 12 rows affected (0.01 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

11. LOAD Data into Fact Table

Join the product dimension to get Product_ID.

INSERT INTO fact_sales (Transaction_ID, Date, Customer_ID, Product_ID, Quantity, Total_Amount)

SELECT

s.Transaction_ID,

s.Date,

s.Customer_ID,

p.Product_ID,

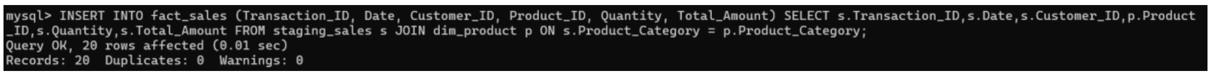
s.Quantity,

s.Total_Amount

FROM staging_sales s

JOIN dim_product p

ON s.Product_Category = p.Product_Category;

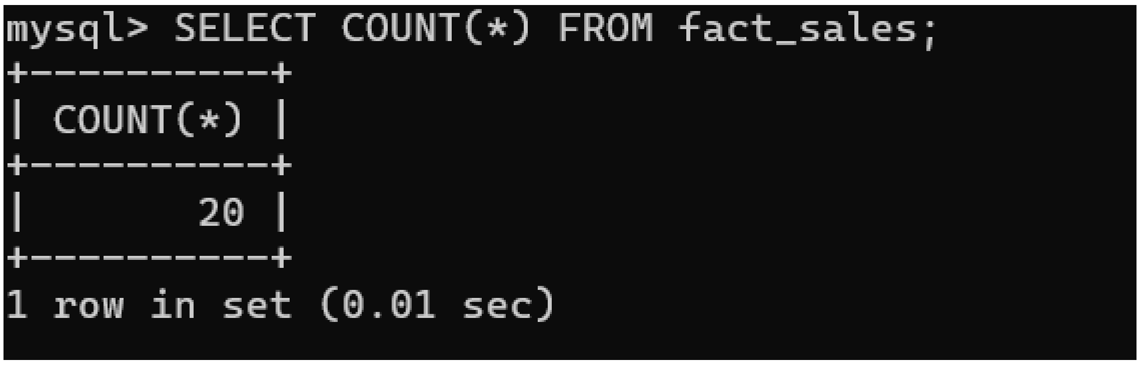


```
mysql> INSERT INTO fact_sales (Transaction_ID, Date, Customer_ID, Product_ID, Quantity, Total_Amount) SELECT s.Transaction_ID,s.Date,s.Customer_ID,p.Product
_ID,s.Quantity,s.Total_Amount FROM staging_sales s JOIN dim_product p ON s.Product_Category = p.Product_Category;
Query OK, 20 rows affected (0.01 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

12. Data Quality Check

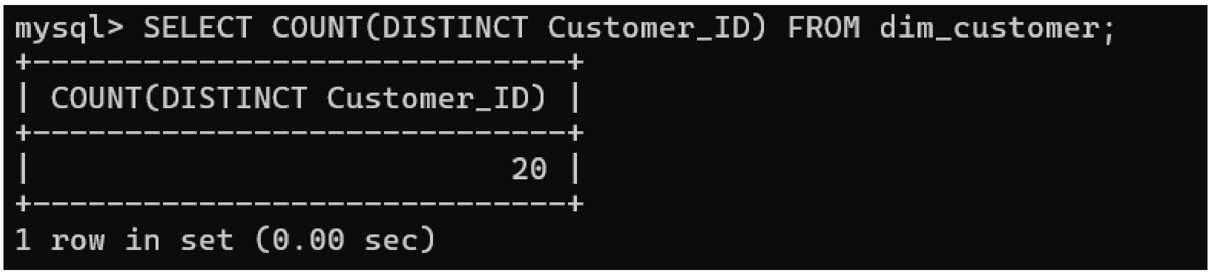
Count checks

SELECT COUNT(*) FROM fact_sales;



```
mysql> SELECT COUNT(*) FROM fact_sales;
+----------+
| COUNT(*) |
+----------+
|       20 |
+----------+
1 row in set (0.01 sec)
```

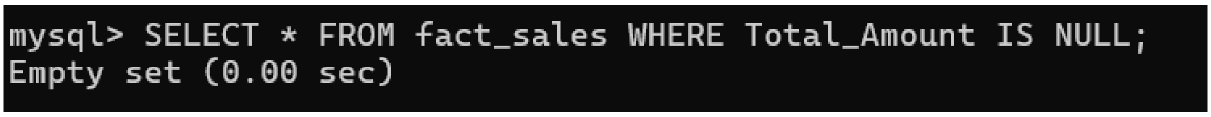
SELECT COUNT(DISTINCT Customer_ID) FROM dim_customer;



```
mysql> SELECT COUNT(DISTINCT Customer_ID) FROM dim_customer;
+-----------------------------+
| COUNT(DISTINCT Customer_ID) |
+-----------------------------+
|                          20 |
+-----------------------------+
1 row in set (0.00 sec)
```

Null checks

*SELECT * FROM fact_sales WHERE Total_Amount IS NULL;*



```
mysql> SELECT * FROM fact_sales WHERE Total_Amount IS NULL;
Empty set (0.00 sec)
```

13. Reporting Queries

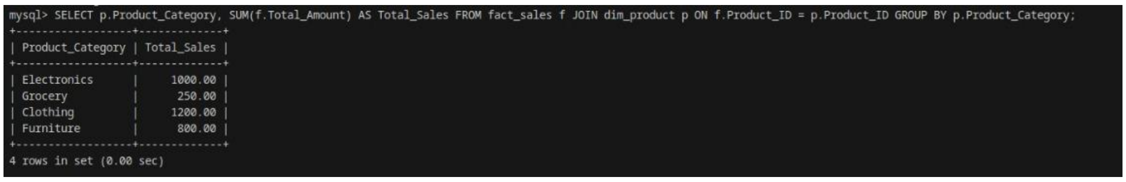
1. Total Sales Revenue by Product Category

SELECT p.Product_Category, SUM(f.Total_Amount) AS Total_Sales

FROM fact_sales f

JOIN dim_product p ON f.Product_ID = p.Product_ID

GROUP BY p.Product_Category;



```
mysql> SELECT p.Product_Category, SUM(f.Total_Amount) AS Total_Sales FROM fact_sales f JOIN dim_product p ON f.Product_ID = p.Product_ID GROUP BY p.Product_Category;
+------------------+-------------+
| Product_Category | Total_Sales |
+------------------+-------------+
| Electronics      |     1000.00 |
| Grocery          |      250.00 |
| Clothing         |     1200.00 |
| Furniture        |      800.00 |
+------------------+-------------+
4 rows in set (0.00 sec)
```

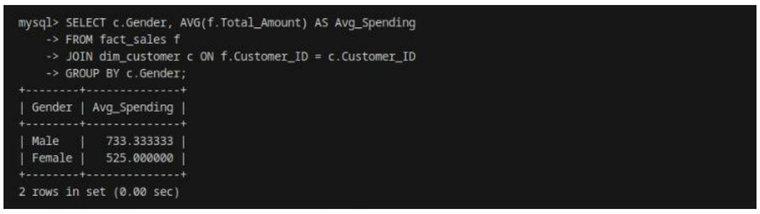
2. Customer Demographics (Average Spending by Gender)

SELECT c.Gender, AVG(f.Total_Amount) AS Avg_Spending

FROM fact_sales f

JOIN dim_customer c ON f.Customer_ID = c.Customer_ID

GROUP BY c.Gender;



```
mysql> SELECT c.Gender, AVG(f.Total_Amount) AS Avg_Spending
    -> FROM fact_sales f
    -> JOIN dim_customer c ON f.Customer_ID = c.Customer_ID
    -> GROUP BY c.Gender;
+--------+--------------+
| Gender | Avg_Spending |
+--------+--------------+
| Male   |   733.333333 |
| Female |   525.000000 |
+--------+--------------+
2 rows in set (0.00 sec)
```

3. Sales Trend Over Time

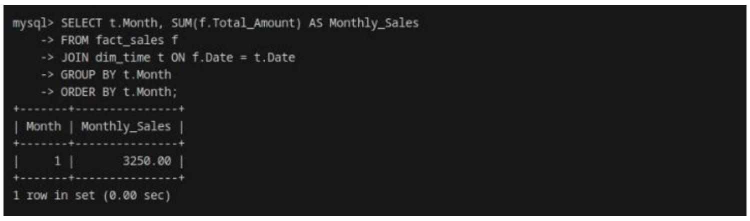
SELECT t.Month, SUM(f.Total_Amount) AS Monthly_Sales

FROM fact_sales f

JOIN dim_time t ON f.Date = t.Date

GROUP BY t.Month

ORDER BY t.Month;



```
mysql> SELECT t.Month, SUM(f.Total_Amount) AS Monthly_Sales
    -> FROM fact_sales f
    -> JOIN dim_time t ON f.Date = t.Date
    -> GROUP BY t.Month
    -> ORDER BY t.Month;
+-------+---------------+
| Month | Monthly_Sales |
+-------+---------------+
|     1 |       3250.00 |
+-------+---------------+
1 row in set (0.00 sec)
```

Preparation	**20**	
Implementation	**20**	
Viva	**15**	
Output	**10**	
Record	**10**	
Total	**75**	

Result:

Thus, the ETL process successfully extracted raw transactional data, cleansed and transformed it, and loaded it into the data warehouse.