Unit Testing Framework Cheat Sheet

1. Commands and Cheat sheet

sbt Commands

Action	Command
Run all tests	sbt test
Run specific test class	sbt "testOnly com.example.MyTest"
Run tests continuously	sbt ~test

Common Assertions

Assertion	Usage
assert(a == b)	Assert equality
assertResult(expected)(actual)	Assert expected == actual
intercept[ExceptionType] { ... }	Check exception thrown
assert(a > b, "message")	Assert with custom message

Common Assertions

Assertion	Usage
assert(a == b)	Assert equality
assertResult(expected)(actual)	Assert expected == actual
intercept[ExceptionType] { ... }	Check exception thrown
assert(a > b, "message")	Assert with custom message

2. JUnit with Maven

Dependencies (`pom.xml`)

```
<dependency>
```

```
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
```

```
import org.junit.Test;
import static org.junit.Assert.*;

public class CalculatorTest {

  @Test
  public void testAddition() {
    assertEquals(5, 2 + 3);
  }
}
```

Maven Commands

Action	Command
Run all tests	mvn test
Run specific test class	mvn -Dtest=CalculatorTest test

```
#!/bin/bash

# Project name
PROJECT_NAME="spark-scala-unittest"

# Create directories
mkdir -p $PROJECT_NAME/src/{main,test}/scala/com/example
cd $PROJECT_NAME

# Create build.sbt
cat <<EOL > build.sbt
name := "SparkScalaUnitTest"
version := "0.1"
scalaVersion := "2.12.18"

libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % "3.5.1" % "provided",
  "org.apache.spark" %% "spark-sql" % "3.5.1" % "provided",
 "org.scalatest" %% "scalatest" % "3.2.18" % Test
)
EOL

# Sample ETL code
cat <<EOL > src/main/scala/com/example/SimpleETL.scala
package com.example

import org.apache.spark.sql.{DataFrame, SparkSession}

object SimpleETL {
 def transformData(df: DataFrame): DataFrame = {
   df.filter("age > 25")
 }

 def main(args: Array[String]): Unit = {
   val spark = SparkSession.builder()
     .appName("SimpleETL")
     .master("local[*]")
     .getOrCreate()

   import spark.implicits._
   val data = Seq(("Alice", 30), ("Bob", 20), ("Charlie", 40))
   val df = data.toDF("name", "age")

   transformData(df).show()

   spark.stop()
 }
}
EOL

# Unit test
cat <<EOL > src/test/scala/com/example/SimpleETLTest.scala
package com.example

import org.apache.spark.sql.SparkSession
import org.scalatest.funsuite.AnyFunSuite

class SimpleETLTest extends AnyFunSuite {
 val spark = SparkSession.builder()
   .appName("SimpleETLTest")
   .master("local[*]")
   .getOrCreate()

 import spark.implicits._

 test("transformData should filter out people age <= 25") {
   val df = Seq(("Aron", 30), ("Dani", 20), ("Brown", 40)).toDF("name", "age")
   val result = SimpleETL.transformData(df).collect()
   assert(result.length == 2)
```

```
  }
}
EOL

echo " Project setup complete in $PROJECT_NAME"
```

```
chmod +x setup_project.sh
./setup_project.sh


sbt test
```

https://www.scala-sbt.org/1.x/docs/Installing-sbt-on-Mac.html
https://docs.scala-lang.org/getting-started/intellij-track/building-a-scala-project-with-intellij-and-sbt.html
https://docs.scala-lang.org/getting-started/intellij-track/testing-scala-in-intellij-with-scalatest.html