

# **DATA MIGRATION**

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE &  
ENGINEERING**

BY

**Pragya Sardar**

**EN18CS301171**

Under the Guidance of

**Dr. Kailash Chandra Bandhu**

**Mr. Binod Kumar Mishra**



**Department of Computer Science & Engineering**

**Faculty of Engineering**

**MEDI-CAPS UNIVERSITY, INDORE- 453331**

**JAN - JUN, 2022**

# **DATA MIGRATION**

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in COMPUTER  
SCIENCE & ENGINEERING**

BY

**Pragya Sardar  
EN18CS301171**

Under the Guidance of

**Dr. Kailash Chandra Bandhu  
Mr. Binod Kumar Mishra**



**Department of Computer Science & Engineering  
Faculty of Engineering  
MEDI-CAPS UNIVERSITY, INDORE- 453331**

**JAN - JUN, 2022**

## **Report Approval**

The project work “**DATA MIGRATION**” is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation

Affiliation

External Examiner

Name:

Designation

Affiliation

## **Declaration**

I hereby declare that the project entitled “**DATA MIGRATION**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science & Engineering’ completed under the supervision of **Dr. Kailash Chandra Bandhu, Associate Professor, Department of Computer Science & Engineering** and **Mr. Binod Kumar Mishra, Assistant Professor, Department of Computer Science & Engineering**, Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

**Pragya Sardar**

\_\_/\_\_/2022

## **Certificate**

I/We, **Dr. Kailash Chandra Bandhu** and **Mr. Binod Kumar Mishra** certify that the project entitled “**DATA MIGRATION**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Pragya Sardar (EN18CS301171)** is the record carried out by him/them under my/our guidance and that the work has not formed the basis of award of any other degree elsewhere.

---

**Dr. Kailash Chandra Bandhu**

Computer Science & Engineering

Medi-Caps University, Indore

---

**Mr. Binod Kumar Mishra**

Computer Science & Engineering

Medi-Caps University, Indore

---

**Dr. Pramod S. Nair**

**Head of the Department**

Computer Science & Engineering

Medi-Caps University, Indore

## **Offer Letter of the Project Work – II/Internship**



January 5, 2022

WD/OFF/1326

**Ms. Pragya Sardar  
D/O Virangan Sardar  
C-14/1, RRCAT Colony, Sukhniwas Colony,  
Indore Madhya Pradesh 452013**

**Dear Pragya,**

We are pleased to inform you that you have been selected for internship as a “**Technical Intern**” with Webdunia.com (India) Private Limited for the minimum duration of 6 Months or any further extension as agreed mutually.

Your joining date will be **Jan 5, 2022**. During internship your working hours shall be **8 Hours per day** at the stipend of **Rs. 5000/- per month**. The TDS if applicable will be deducted as per Govt. rules from the payable amount. Further as a policy, any leave taken during the internship period will be considered as Leave without pay. All the other terms of internship have already been discussed and agreed between us.

For any queries or clarifications, feel free to call the undersigned or any representative from our team. We look forward to welcoming you to the Company and wish you a successful and rewarding career.

**Sincerely**

**For RWS Moravia (India) Private Limited**

A handwritten signature in black ink, appearing to be 'SJ' or similar, written in a cursive style.

**Swapnil Jain**

**Director - Human Resources**

**I accept the above-mentioned terms and condition:**

**Name:** Pragya Sardar

**Signature & Date:**

## Completion Certificate/Letter



### TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Pragya Sardar** is pursuing training as a **Technical Intern** at RWS Moravia (India) Private Limited, in **Digital Technology** department. The internship tenure is from **05-Jan-22** to **1-Jul-22** and her project title is **Data Migration**. We found her work prompt and satisfactory and we wish her all the best for his future endeavors.

**For RWS Moravia (India) Private Limited**

A handwritten signature in black ink, appearing to be 'SJ' followed by a long horizontal stroke.

**Swapnil Jain**

**Director – Human Resources**

**Date: 05-May-22**

---

Corporate Office: RWS Moravia India Private Limited, 60/1, Baby Labh Chand Chhajani Marg, Indore - 452 009, Madhya Pradesh, India. T: +91-731-4865800 | F: +91-731-2548555 | E: [rwsmoravia-india@rws.com](mailto:rwsmoravia-india@rws.com)  
[www.rws.com](http://www.rws.com)

Registered Office: RWS Moravia India Private Limited, f.k.a. Webdunia.com (India) Private Limited, B-810, 8th Floor, BSEL Tech Park, 39/5 & 39/5A, Sector - 30A, Vashi, Navi Mumbai, Thane - 400 703, Maharashtra, India. T: +91-22-49786542 | CIN NO. U72900MH2000PTC1235281

## **Acknowledgements**

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal**, who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Dilip K. Patnaik**, Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda**, Pro Vice Chancellor, **Dr. Suresh Jain**, Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Prof. (Dr.) Pramod S. Nair** for his continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my **External Guide, Mr. Ankush Maheshwari**, Project Manager, RWS Moravia as well as to my Internal Guide, **Dr. Kailash Chandra Bandhu**, Associate Professor, Department of Computer Science & Engineering, Medi-Caps University, and **Mr. Binod Kumar Mishra**, Assistant Professor, Department of Computer Science & Engineering, Medi-Caps University, without whose continuous help and support, this project would ever have reached to the completion.

I would also like to thank to my team at **RWS Moravia, Mr. Ankush Maheshwari, Ms. Ronak Jain, Ms. Nishtha Jain, Ms. Aastha Katiyar, Ms. Sakina Saiffee** who extended their kind support and help toward the completion of this project.

It is their help and support, due to which I was able to complete the design and technical report. Without their support this report would not have been possible.

**Pragya Sardar (EN18CS301171)**

B.Tech. IV Year

Department of Computer Science & Engineering

Faculty of Engineering

Medi-Caps University, Indore



## **Executive Summary**

A common data quality problem is to have multiple different records that refer to the same entity but no unique identifier that ties these entities together.

For instance, customer data may have been entered multiple times in multiple different computer systems, with different spellings of names, different addresses, and other typos. The lack of a unique customer identifier presents challenges at all stages of data analysis - from basic questions such counting the number of unique customers, to feature engineering of customers' details for machine learning purposes.

There is a large body of theoretical and empirical work into this problem. The solution usually involves computing a new unique identifier column which allows entities to be linked and grouped, using a process of statistical estimation, machine learning. We usually need to link the records.

Record linkage, is simply the integration of information from two independent sources. Records from the two sources that are believed to relate to the same individual are matched in such a way that they may then be treated as a single record for that individual. Records brought together in this way are said to be linked.

The principles of record linkage may be applied to any field in which it is necessary to bring together information recorded about persons in different places or at different times.

We reached to solution which involves Spark framework to process the data, to identify duplicate customers and grouped them as single customer master record with relative key fields.

This project involves majorly data extraction, removing redundancy from customer data and entering them in new system after cleansing for business insights as well as to have accurate customer information across whole organization.

## **Table of Contents**

<b>Chapter No.</b>	<b>Chapter Name</b>	<b>Page No.</b>
	Report Approval	ii
	Declaration	iii
	Certificate	iv
	Offer Letter of the Project Work - II/Internship	v
	Completion Certificate/Letter	vi
	Acknowledgements	vii
	Executive Summary	viii
	Table of Contents	ix
	List of Figures	xi
	List of Tables	xii
	Abbreviations	xiii
	Notations & Symbols	xiv
Chapter 1	Introduction	1
	1.1 Introduction	1
	1.2 Literature Review	2
	1.3 Objectives	4
	1.4 Significance	4
Chapter 2	System Requirement analysis	5
	2.1 Activities involved in Software Requirement Analysis	5
	2.2 Experimental Set-up	7
	2.3 Requirement Gathering	15
	2.4 Feasibility Study	19
Chapter 3	Technology Description	22
	3.1 Scala	22
	3.2 Apache Spark	32

	3.3 SQL Server	42
Chapter 4	Designing & Implementation	44
	4.1 Procedure Adopted	44
	4.2 Proposed Algorithm	46
	4.3 Platform Specification	49
Chapter 5	Designing Methodology	50
	5.1 Designing Mechanism	50
	5.2 ER Diagram	51
	5.3 Data Flow Diagram	53
Chapter 6	Results and Discussions	55
Chapter 7	Summary & Conclusions	56
Chapter 8	Future scope	57
	Bibliography	58

## **List of Figures**

Figure 1.2.2.1 Jaccard Similarity.....	2
Figure 2.1 Software Requirement Analysis.....	5
Figure 2.4 Feasibility Analysis Steps.....	20
Figure 3.1.3 Scala Usage.....	23
Figure 3.2.1 Apache Spark VS Apache Hadoop.....	32
Figure 3.2.2 Apache Spark Features.....	33
Figure 3.2.3 Apache Spark Components.....	34
Figure 3.2.4.1 Apache Spark Architecture – Process.....	36
Figure 3.2.4.1 Apache Spark Architecture – Driver.....	36
Figure 3.2.4.5 Apache Spark RDD.....	38
Figure 3.3.1 SQL Server Architecture.....	43
Figure 5.2 ER Diagram.....	52
Figure 5.3 Data Flow Diagram – 1.....	53
Figure 5.3 Data Flow Diagram – 2.....	54

## **List of Tables**

Table 2.3 Functional VS Non-Functional Requirement.....	17
---	----

## **Abbreviations**

SRS	Software Requirement Specification
IDE	Integrated Development Environment
JDK	Java Development Kit
POM	Project Object Model
EPFL	Ecole Polytechnique Federale de Lausanne
JVM	Java Virtual Machine
SDK	Software Development Kit
API	Application Programming Interface
RDD	Resilient Distributed Dataset
SQL	Structured Query Language
ML	Machine Learning
HQL	Hive Query Language
JSON	JavaScript Object Notation
DAG	Directed Acyclic Graph
HDFS	Hadoop Distributed File System
SQLOS	SQL Server Operating System
GB	Giga Byte
TB	Tera Byte
SDLC	Software Design Life Cycle
ERD	Entity Relationship Diagram
DFD	Data Flow Diagram

## Notations & Symbols

S No.	Notation	Name
1	J	Jaccard Distance
2	A	Set 1
3	B	Set 2
4	m	Matching Character Number
5	t	Transposition Number
6	s1	Length of string 1
7	s2	Length of string 2

Symbol	Name	Description
$\cup$	Union	Belong to Set A or Set B
$\cap$	Intersection	Belong to both Set A and Set B
$ A \cup B $	Modulus Function	Gives the absolute value of Set A UNION Set B
$ A \cap B $	Modulus Function	Gives the absolute value of Set A INTERSECTION Set B

# **CHAPTER 1**

## **Introduction**

### **1.1 Introduction**

One of the main challenges for the transformation program is to be able to identify, evaluate and process existing Customer Data located in different systems that are also geographically dispersed. The data needs to be identified, quantified, and evaluated for processing. If data is to be transformed this data needs to be extracted, deduplicated and transformed which includes cleansing, matching and merging processes, ready to populate new systems.

For instance, a company has a global presence with many disparate data systems around the globe. Many business functions are duplicated, and these systems do not currently exist in a cohesive data ecosystem. Even, within each system the Customer Data is often of unreliable quality. There are no data governance procedures or policies in place. This result in huge difficulty in creating reports to provide valuable and necessary insights about the business.



## 1.2 Literature Review

### 1.2.1 System Review

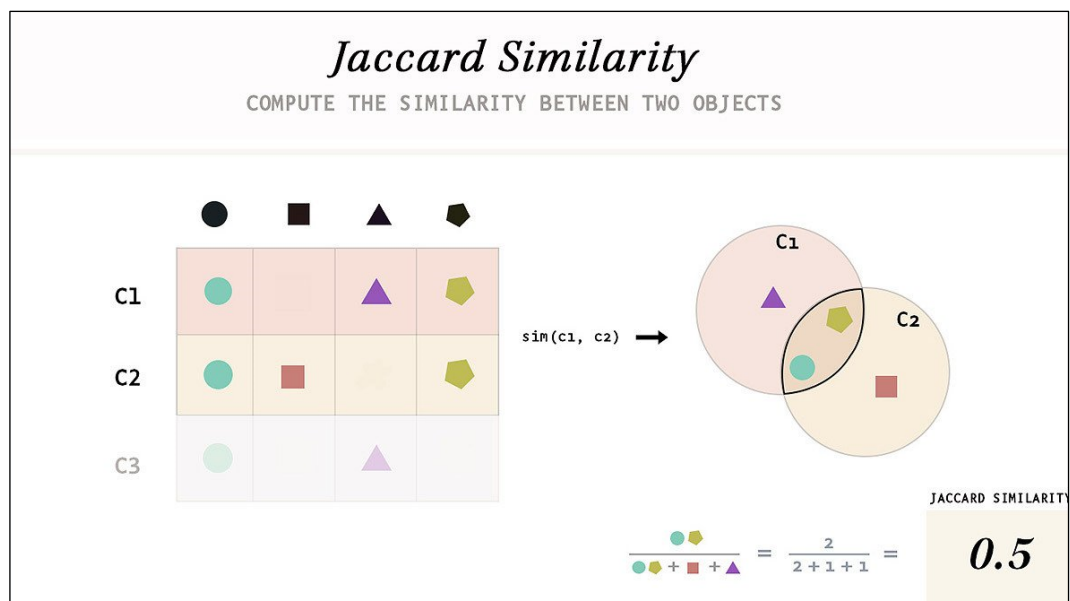
This survey is done to comprehend the need of a common data quality problem that has multiple different records that refer to the same entity but has no unique identifier that ties these entities together.

Based on the data, we made an audit that helped us to identify duplicate customers and grouped them as a single customer master record with relative key fields. We reached the decision that there is a need of such application and felt that there is a decent extent of progress in this field too.

### 1.2.2 Algorithms Used

#### 1.2.2.1 JACCARD SIMILARITY

Jaccard Similarity is used to measure the similarity between two sets of data to see which members are shared and distinct.



**Figure 1.2.2.1 Jaccard Similarity**

Jaccard Similarity can be used to find the similarity between two asymmetric binary vectors or between two sets.

It is calculated by dividing the number of observations in both sets by the number of observations in either set. In other words, the Jaccard similarity can be computed as the size of the intersection divided by the size of the union of two sets.

This can be written in set notation using intersection ( $A \cap B$ ) and unions ( $A \cup B$ ) of two sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$J$  = Jaccard distance

$A$  = set 1

$B$  = set 2

where  $|A \cap B|$  gives the number of members shared between both sets and  $|A \cup B|$  gives the total number of members in both sets (shared and un-shared). The Jaccard Similarity will be 0 if the two sets don't share any values and 1 if the two sets are identical. The set may contain either numerical values or strings.

#### 1.2.2.2 JARO - WINKLER

The **Jaro - Winkler distance** is a string metric measuring an edit distance between two sequences.

*The lower the Jaro - Winkler distance for two strings is, the more similar the strings are.*

The score is normalized such that 0 means an exact match and 1 means there is no similarity. The Jaro–Winkler similarity is the inversion,  $\{1 - (\text{Jaro–Winkler distance})\}$ .

## **1.3 Objective**

Record Linkage, Entity Resolution or Entity Matching is a crucial step in data cleaning which aims to find records (i.e., database tuples) that refer to the same real-world entity. Record Linkage is an expensive process that could take many hours or even days. The situation become more complicated especially for large datasets as it compares a pair of records using one or more similarity measures.

## **1.4 Significance**

The most important benefit that record linkage brings is that it helps companies build a single customer view. Without a record linkage system, you will find information about customers spread across different data sets and even multiple systems. Obviously, this makes it impossible to get useful insight and also makes it impossible for companies to make sound business decisions.

Record linkage provides the means for your company to always extract actionable data from databases and lists. With this data, you will be able to easily build a complete customer view. With a complete customer view, companies can significantly improve their marketing efforts, make more accurate business decisions, and increase overall customer loyalty.

## CHAPTER 2

### System Requirement Analysis

#### 2.1 Activities involved in Software Requirement Analysis

Software requirement means requirement that is needed by software to increase quality of software product. These requirements are generally a type of expectation of user from software product that is important and need to be fulfilled by software. Analysis means to examine something in an organized and specific manner to know complete details about it.

Therefore, Software requirement analysis simply means complete study, analyzing, describing software requirements so that requirements that are genuine and needed can be fulfilled to solve problem.

There are several activities involved in analyzing Software requirements. Some of them are given below:

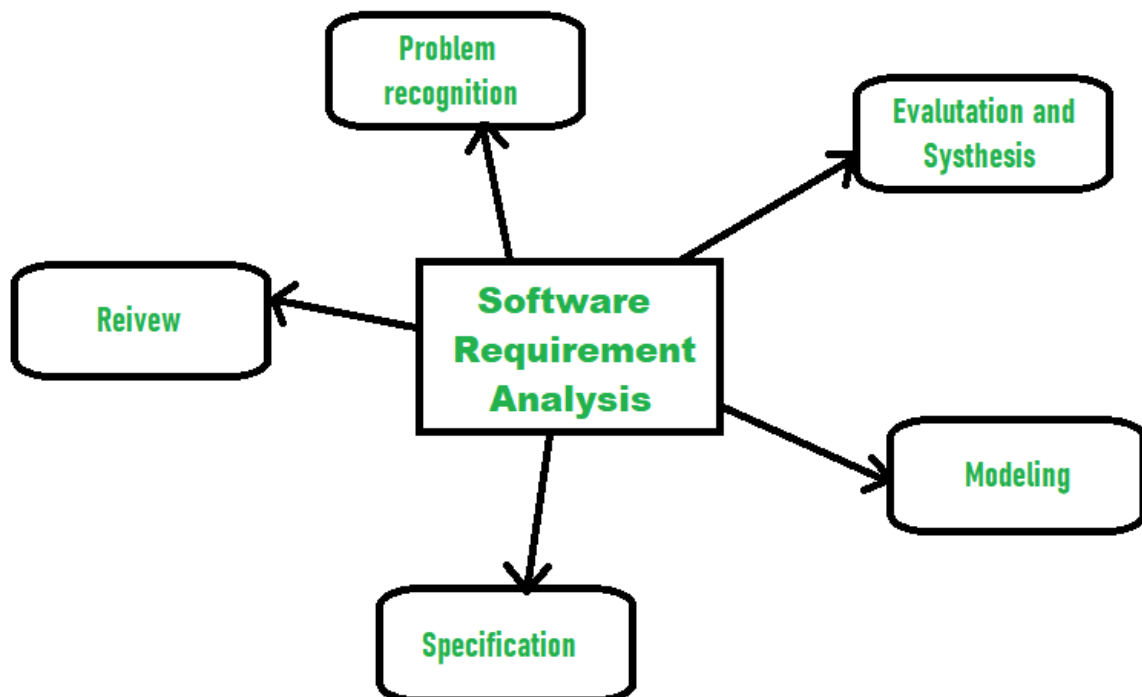


Figure 2.1 Software Requirement Analysis

### **1) Problem Recognition:**

The main aim of requirement analysis is to fully understand main objective of requirement that includes why it is needed, does it add value to product, will it be beneficial, does it increase quality of the project, does it will have any other effect. All these points are fully recognized in problem recognition so that requirements that are essential can be fulfilled to solve business problems.

### **2) Evaluation and Synthesis:**

Evaluation means judgement about something whether it is worth or not and synthesis means to create or form something. Here are some tasks are given that is important in the evaluation and synthesis of software requirement:

- To define all functions of software that necessary.
- To define all data objects that are present externally and are easily observable.
- To evaluate that flow of data is worth or not.

### **3) Modeling:**

After complete gathering of information from above tasks, functional and behavioral models are established after checking function and behavior of system using a domain model that also known as the conceptual model.

### **4) Specification:**

The software requirement specification (SRS) which means to specify the requirement whether it is functional or non-functional should be developed.

### **5) Review:**

After developing the SRS, it must be reviewed to check whether it can be improved or not and must be refined to make it better and increase the quality.

## 2.2 Experimental Set-Up

### 2.2.1 Scala IDE and Maven Project Set-Up:

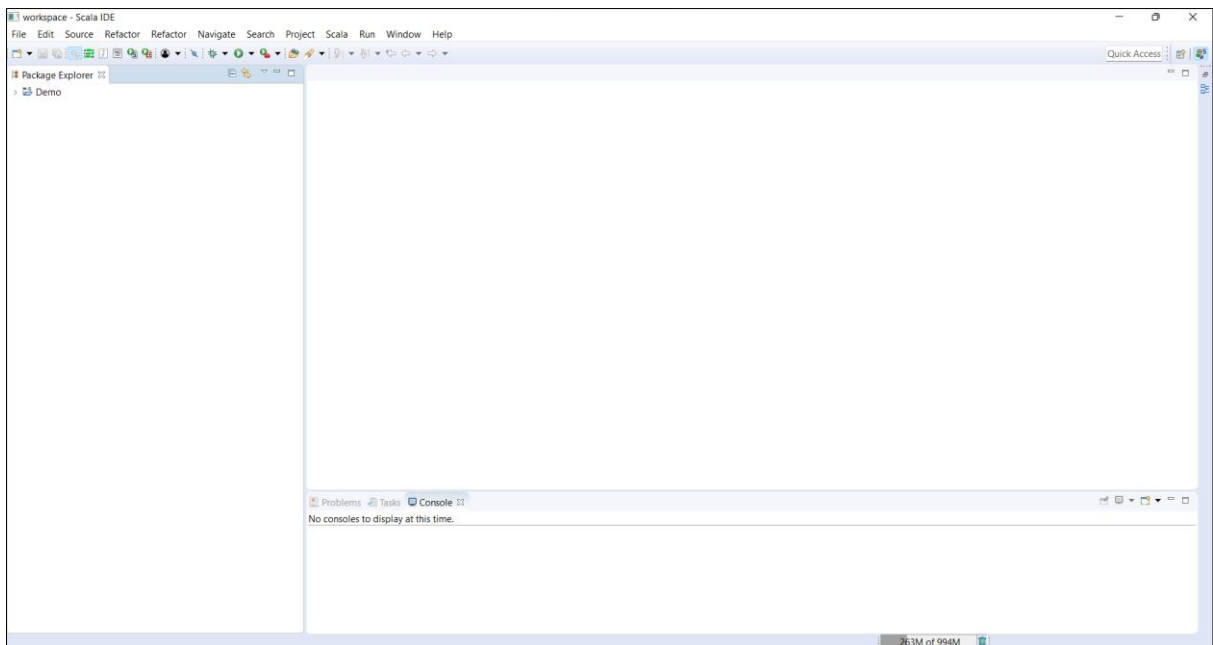
#### Prerequisites:

- 1) JDK must be installed on system.
- 2) Scala must be installed on system.

**STEP 1:** Download Scala IDE from official website.

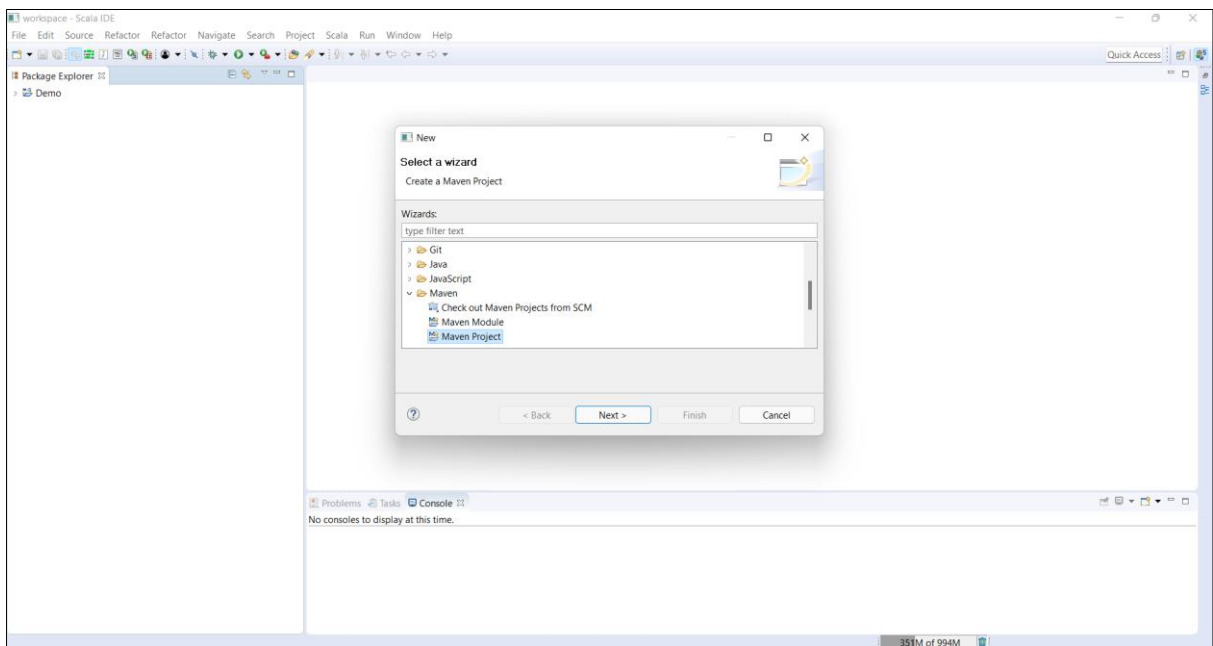
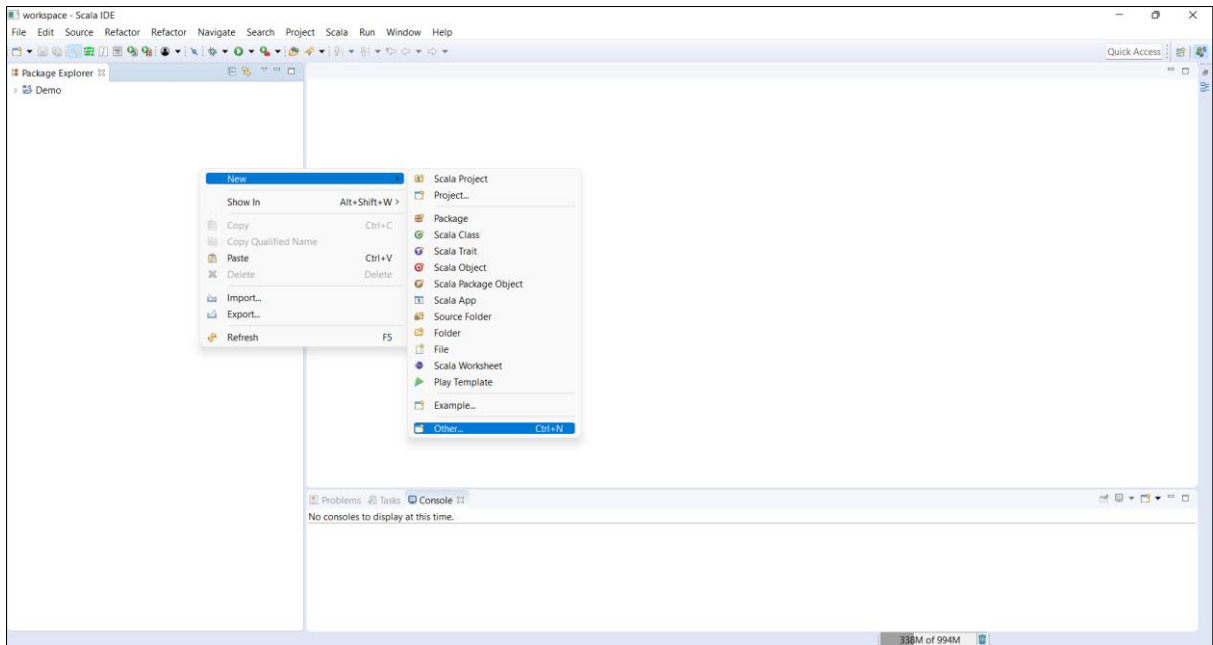
Link: [Download Scala IDE for Eclipse - Scala IDE for Eclipse \(scala-ide.org\)](https://scala-ide.org/)

**STEP 2:** Install this IDE

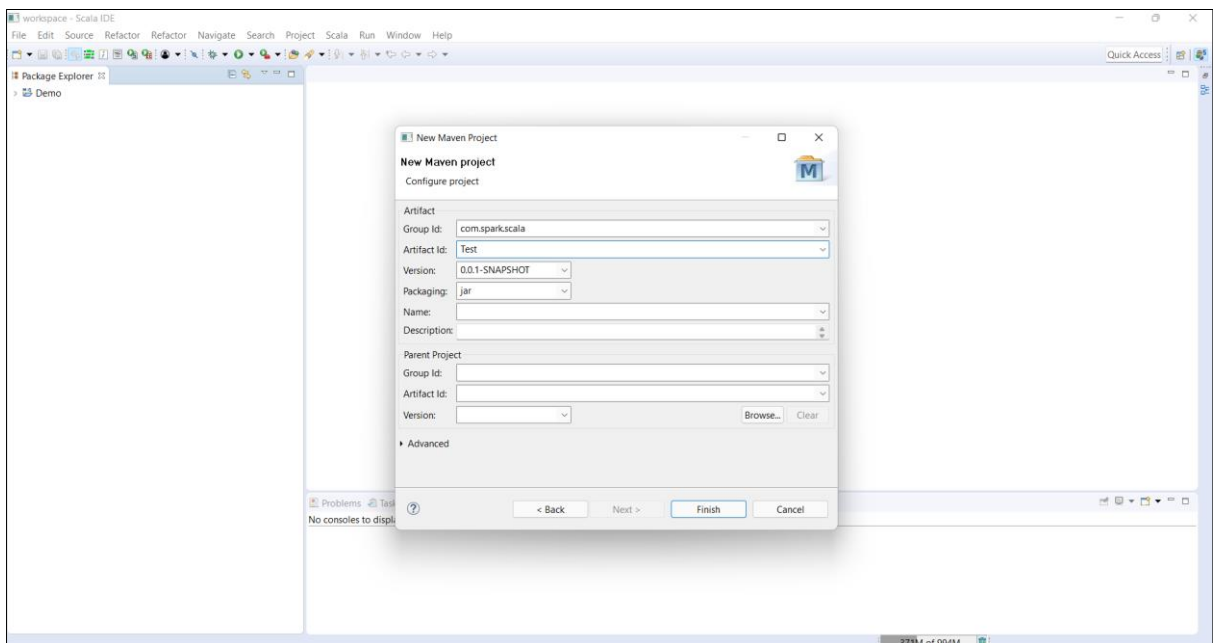
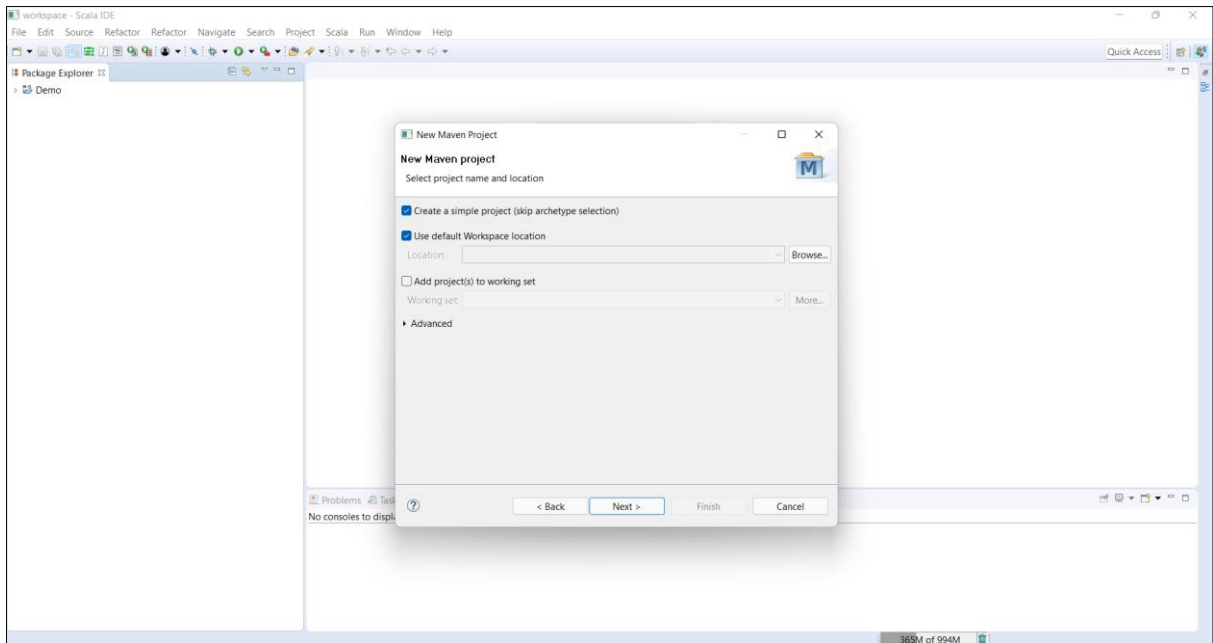


**STEP 3:** Project Creation

- i. Right click on left side window → go to “New” → go to “Other...” → select “Maven” → inside maven select “Maven Project” → click on “Next”.



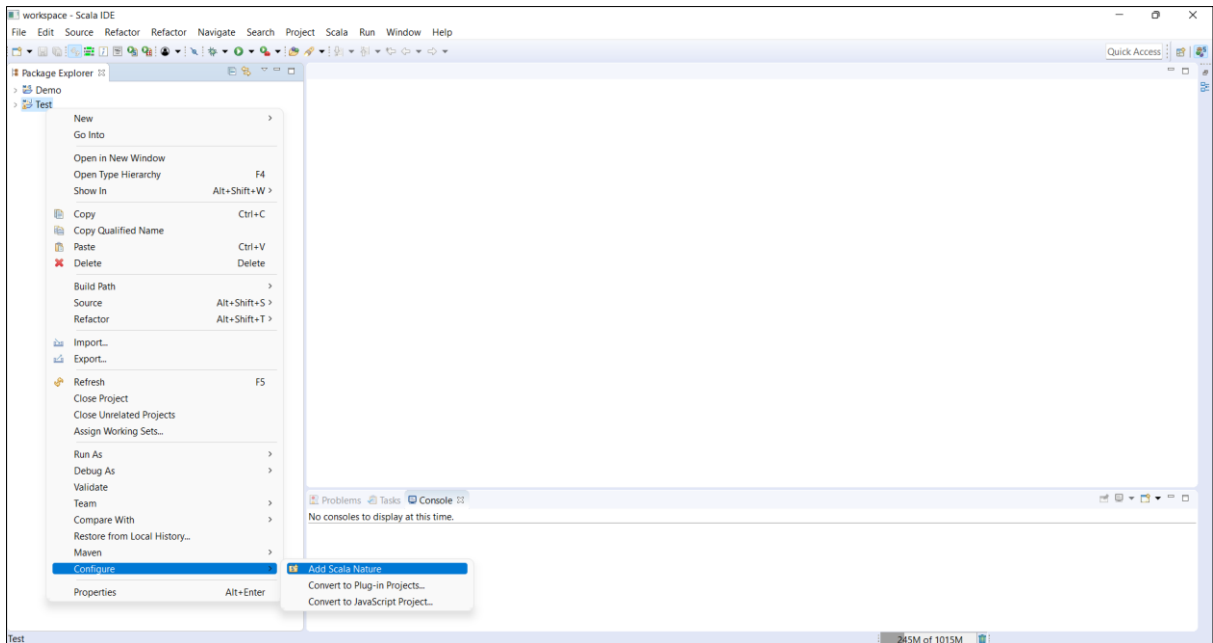
- ii. Then click on “Next” → click “Next” with default setting → click “Finish”.



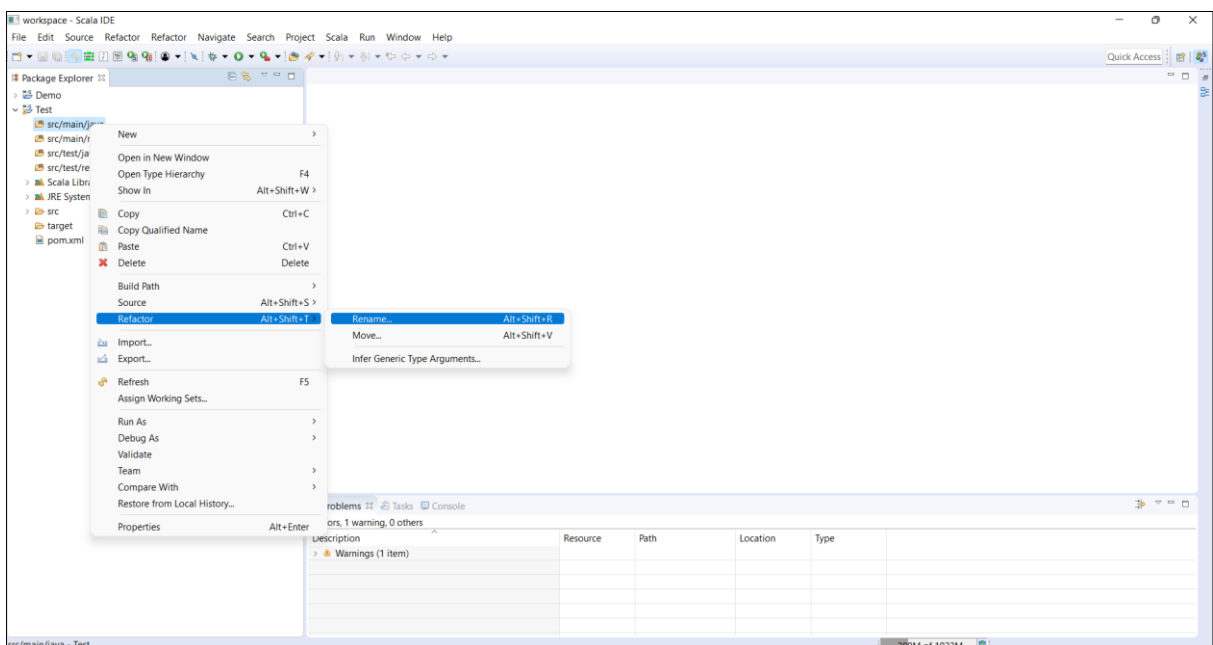
Now we having our project folder ready.

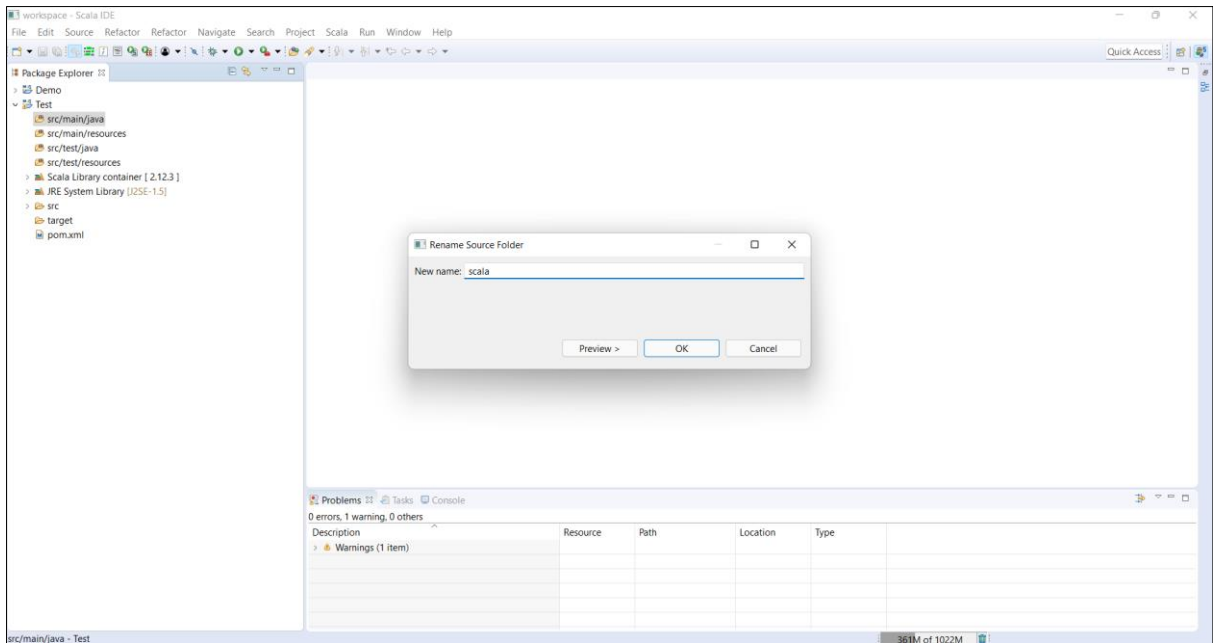
- iii. Right click on project (**Test**) → go to “Configure” → select “Add Scala Nature”



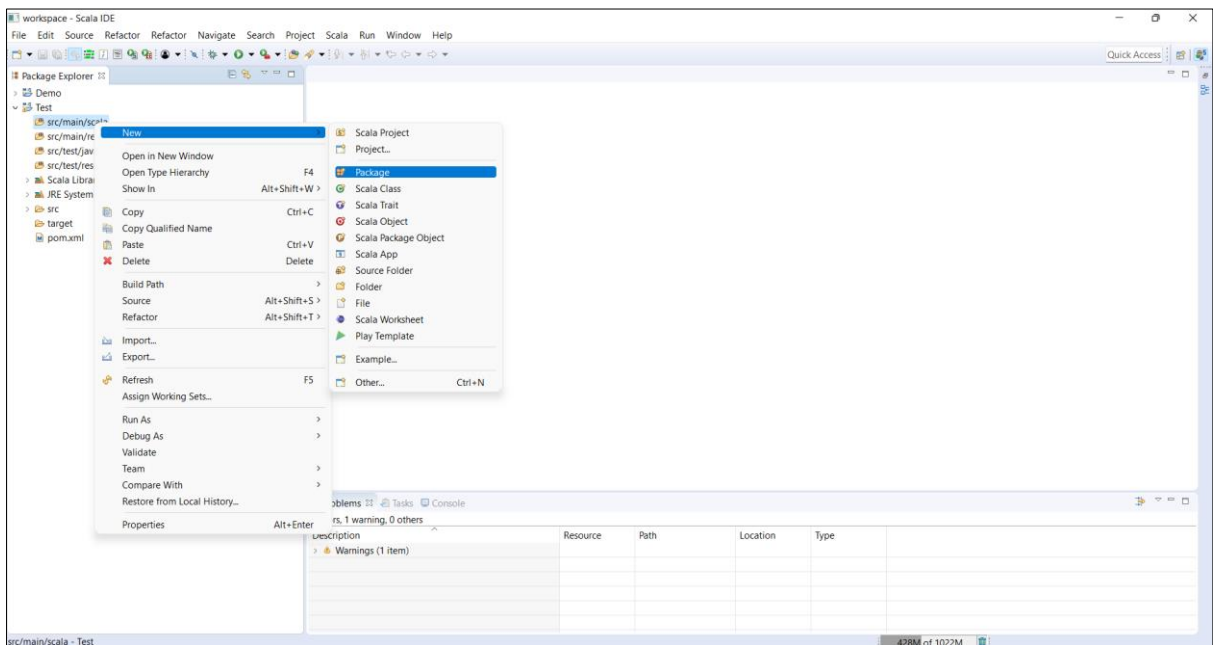


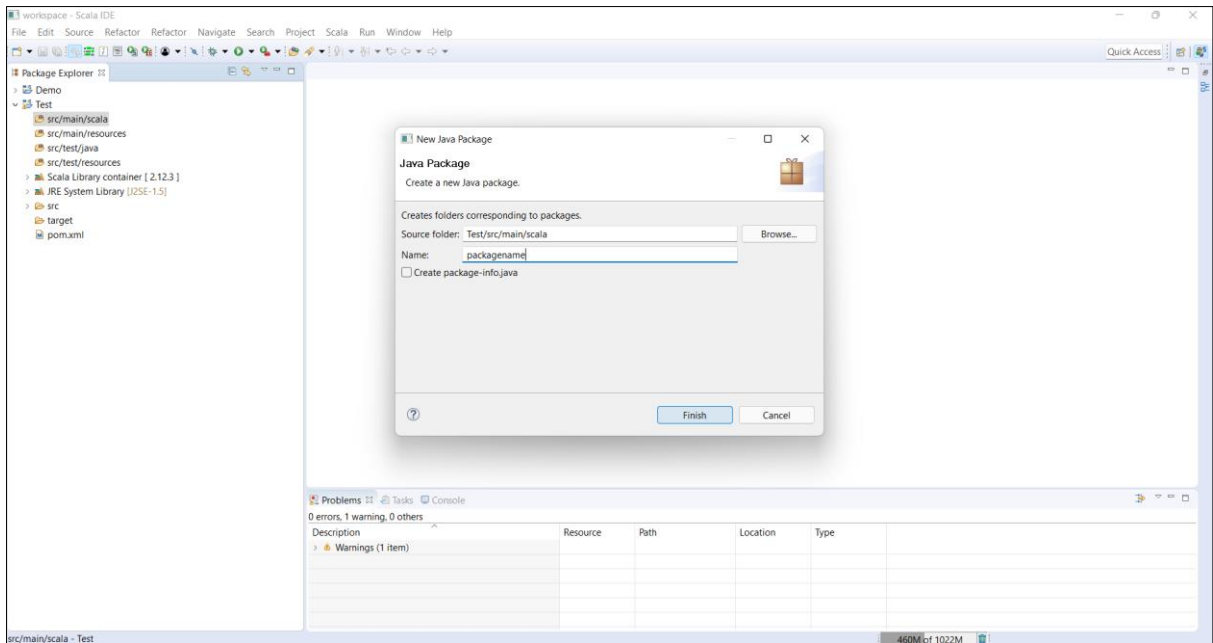
iv. Rename the folder “src/main/java” to “scala”



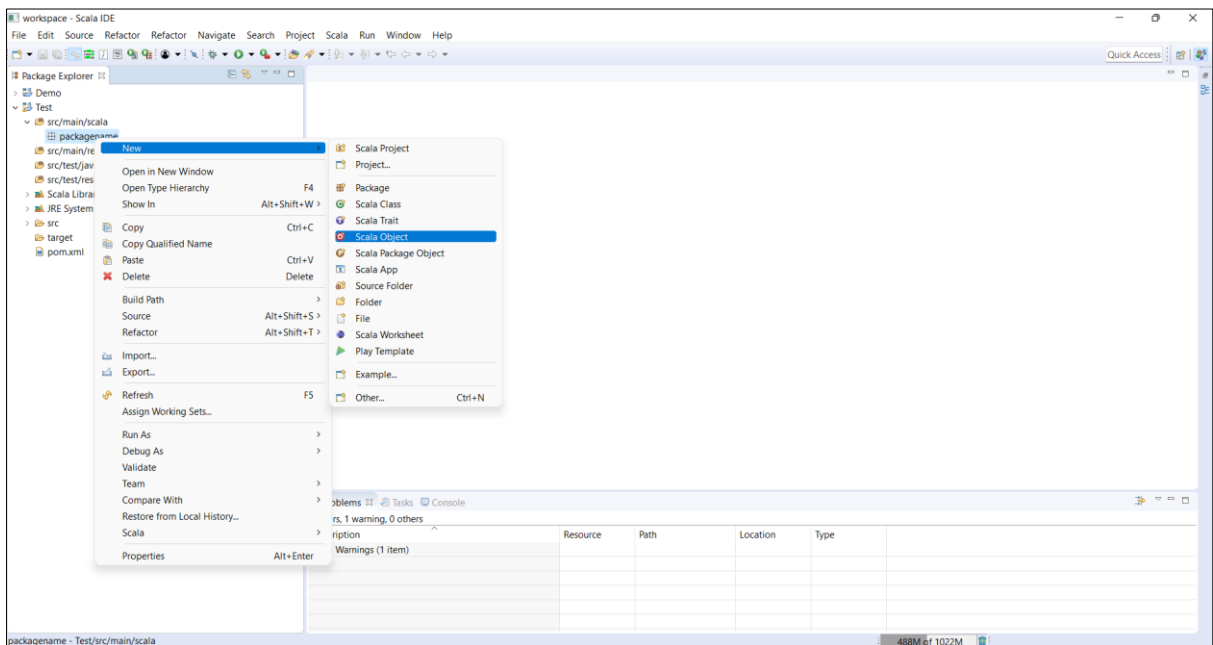


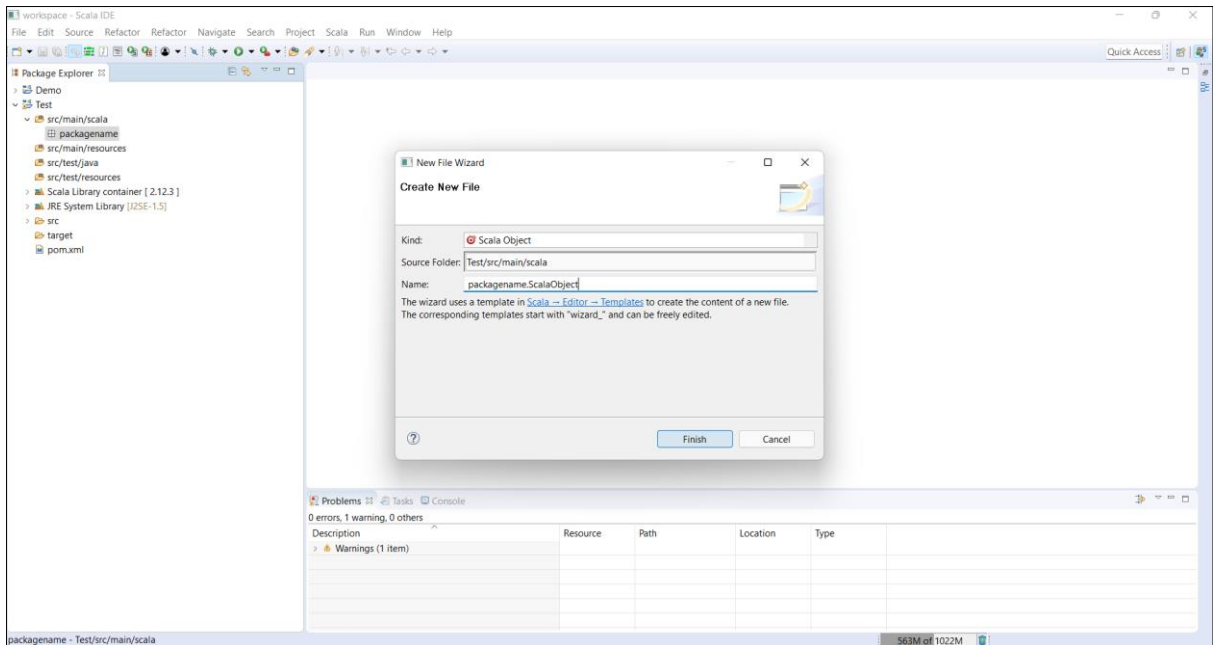
- v. Right click on “src/main/scala” → go to “New” → go to “Package” → create package “packagename”



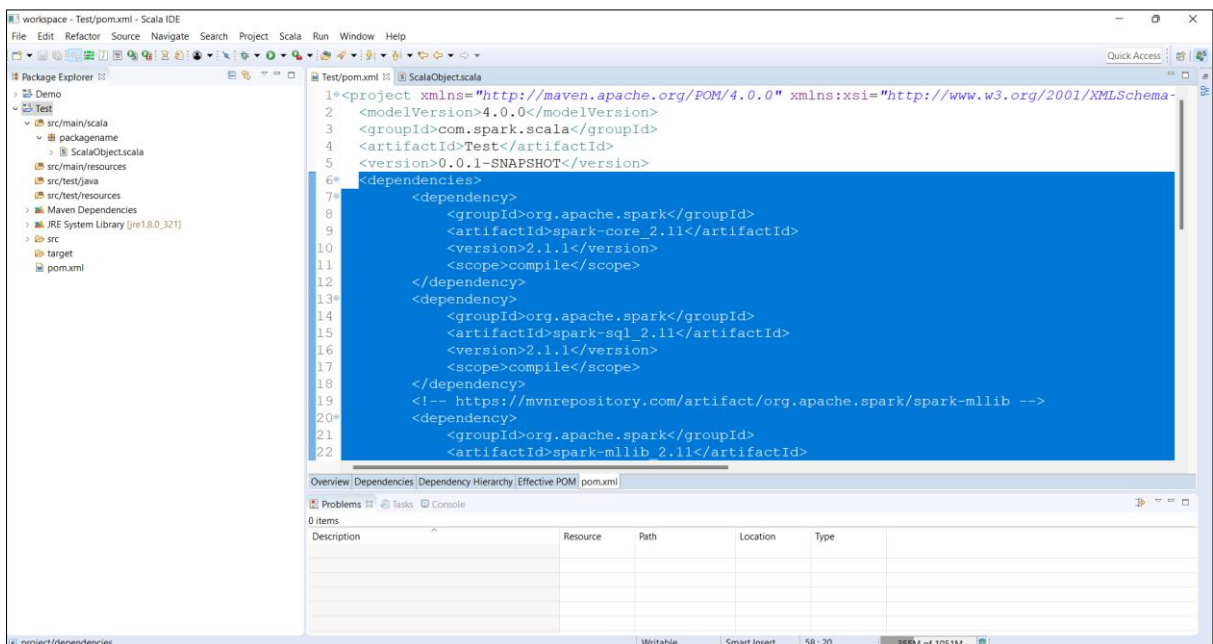


- vi. Right click on “packagename” → go to “New” → go to “Scala Object” → create object “ScalaObject”





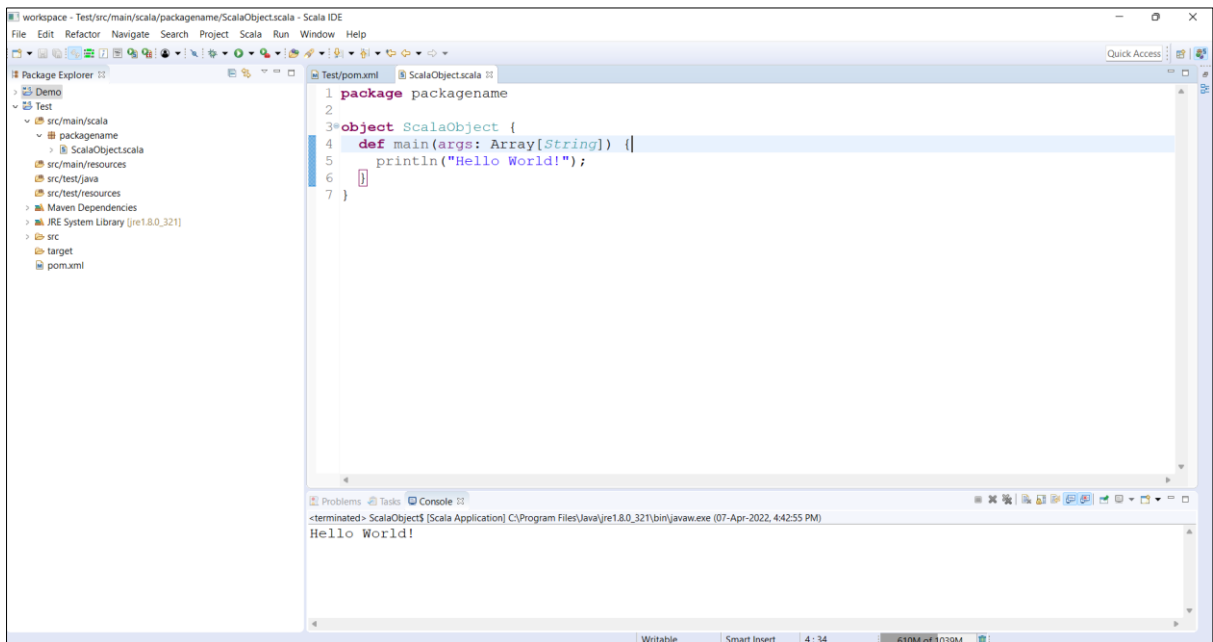
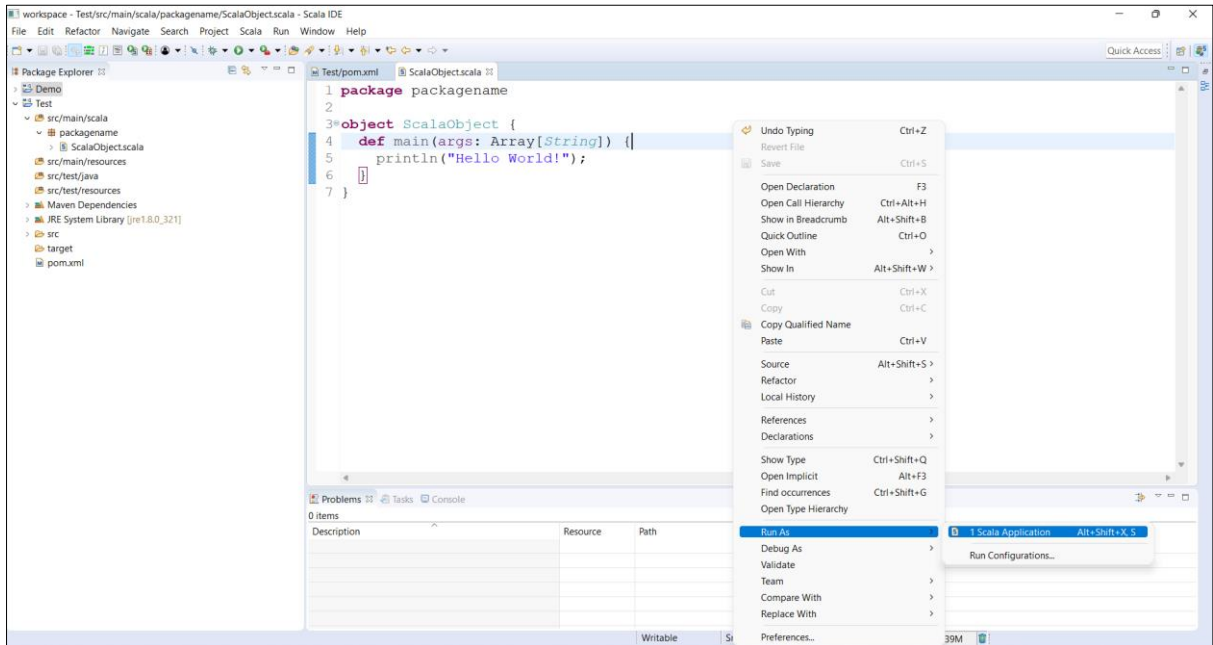
- vii. Now we have to set dependency for our project: Open “pom.xml” file → add the code in that file.



- viii. Click save on pom.xml file, it will download all the dependencies.  
NOTE: wait until dependency get downloaded.
- ix. Go to the Scala object that we created write your code and run it.

## STEP 4: Code Execution

Right click on code window → go to “Run As” → select “1 Scala Application”



## 2.3 Requirement Gathering

Requirements analysis is very critical process that enables the success of a system or software project to be assessed.

Requirements are generally split into two types: Functional and Non-functional requirements.

- 1) **Functional Requirements:** These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements describe the behaviour of the system as it correlates to the system's functionality.

Functional requirements should be written in a simple language, so that it is easily understandable. The examples of functional requirements are authentication, business rules, audit tracking, certification requirements, transaction corrections, etc.

These requirements allow us to verify whether the application provides all functionalities mentioned in the application's functional requirements. They support tasks, activities, user goals for easier project management.

There are a number of ways to prepare functional requirements. The most common way is that they are documented in the text form. Other formats of preparing the functional requirements are use cases, models, prototypes, user stories, and diagrams.

- 2) **Non - Functional Requirements:** These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

Non-functional requirements are not related to the software's functional aspect. They can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviours of the system.

Non-functional requirements specify the software's quality attribute. These requirements define the general characteristics, behaviour of the system, and features that affect the experience of the user. They ensure a better user experience, minimizes the cost factor. Non-functional requirements ensure that the software system must follow the legal and adherence rules.

The impact of the non-functional requirements is not on the functionality of the system, but they impact how it will perform. For a well-performing product, at least some of the non-functional requirements should be met.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

**Following are the differences between Functional and Non - Functional Requirements**

**Table 2.3 Functional VS Non-Functional Requirement**

<b>Functional Requirements</b>	<b>Non - Functional Requirements</b>
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies “What should the software system do?”	It places constraints on “How should the software system fulfill the functional requirements?”
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g., Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API Testing, etc. are done.	Non - Functional Testing like Performance, Stress, Usability, Security testing, etc. are done.
Usually easy to define.	Usually more difficult to define.



### 2.2.1 Functional Requirements

1. **Data Cleansing:** The data must be first cleaned that means if it has extra commas, blank spaces, duplicate values or improper data within the dataset it must be removed or fixed before processing.
2. **Data Extraction:** It is the process of identifying the useful and meaningful data from the dataset for the business insights.

### 2.2.2 Non - Functional Requirements

1. **Security:** The prevention of unauthorized access to programs and data.
2. **Confidentiality:** In all information systems there is a significant amount of information that is classified as confidential and must be accessible only by users possessing the proper authorisations.
3. **Integrity:** Data integrity deals with the integrity, consistency, and correctness of the data in the application.
4. **Availability of Information:** The data must be available whenever they are required.
5. **Access Control:** Every user must have the authorisation to access the system based on specific and well-defined rights.
6. **Portability:** It works in different environment if the underlying dependent framework stays the same.

## 2.4 Feasibility Study

A **Feasibility Study** summarizes results of the analysis and evaluations conducted to review the proposed solution and investigate project alternatives for the purpose of identifying if the project is really feasible, cost-effective and profitable. It enables us to determine the potential of existing system and designing new system and helps to meet user requirements.

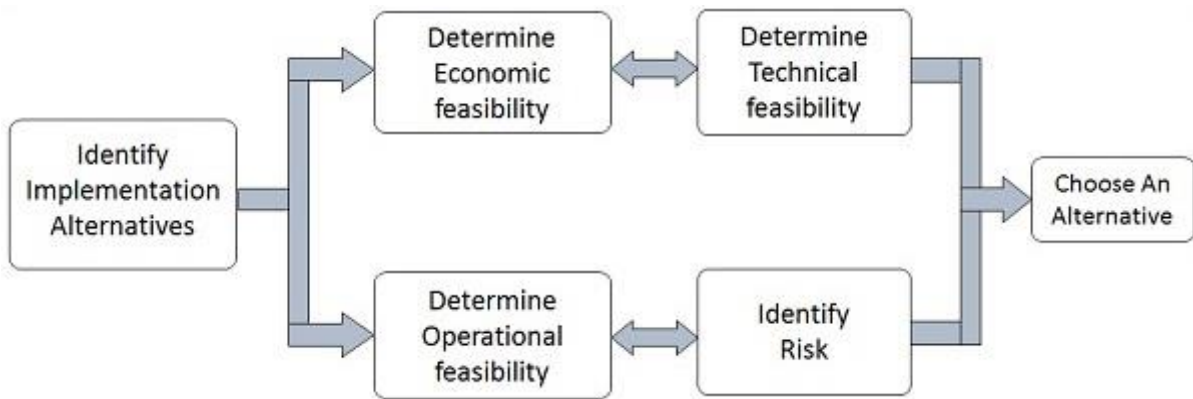
Conducting a feasibility study is one of the key activities within the project initiation phase. It aims to analyze and justify the project in terms of technical feasibility, business viability and cost-effectiveness.

Feasibility Study can be considered as preliminary investigation that helps the management to take decision about whether study of system should be feasible for development or not.

- It identifies the possibility of improving an existing system, developing a new system, and produce refined estimates for further development of system.
- It is used to obtain the outline of the problem and decide whether feasible or appropriate solution exists or not.
- The main objective of a feasibility study is to acquire problem scope instead of solving the problem.
- The output of a feasibility study is a formal system proposal act as decision document which includes the complete nature and scope of the proposed system.

### Steps Involved in Feasibility Analysis

- Form a project team and appoint a project leader.
- Develop system flowcharts.
- Identify the deficiencies of current system and set goals.
- Enumerate the alternative solution or potential candidate system to meet goals.
- Determine the feasibility of each alternative such as technical feasibility, operational feasibility, etc.
- Weight the performance and cost effectiveness of each candidate system.
- Rank the other alternatives and select the best candidate system.
- Prepare a system proposal of final project directive to management for approval.



**Figure 2.4 Feasibility Analysis Steps**

**The project concept is feasible because of the following:**

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

### **2.4.1 TECHNICAL FEASIBILITY**

Technical feasibility study is the complete study of the project in terms of input, processes, output, fields, programs and procedures. It is a very effective tool for long term planning and trouble shooting. This project is technically feasible and to maximum extent the existing systems support the proposed system.

### **2.4.2 ECONOMICAL FEASIBILITY**

It includes the qualification or identification of all the benefits expected. This appraisal of typically involves cost and benefits of analysis. The motive of the economic feasibility appraisal is to determine the positive economic benefits to the organization that the suggested system will provide. The application takes care of the present existing system's data flow and procedures completely. It should be built as a standalone application.

### 2.4.3 OPERATIONAL FEASIBILITY

Operational feasibility refers to the measure of solving problems with the help of a new proposed system. It helps in taking advantage of the opportunities and fulfills the requirements as identified during the development of the project. It takes care that the management and the users support the project.

**The operational feasibilities of our project are:**

- **Availability** is an indicator of the planned-up time where the system is available for use and fully operational. Availability encompasses reliability, maintainability, and integrity.
- **Reliability** defines the probability of the system operating without failure for a specific period of time.
- **Maintainability** measures how easy is it to correct a defect or modify the system. It depends on how easily the system can be understood, changed, and tested.
- **Integrity** relates to security. This quality requirement category describes what is required to block unauthorized access to certain system functions, how to prevent information loss, how to ensure that the system is protected from virus infection, and how to protect the privacy and safety of data entered into the system.
- **Portability** addresses the effort required to migrate a system component from one operating environment to another.
- **Flexibility** measures how easy it is to add new capabilities to the product. You might refer to it as augmentation, extensibility, extendibility, or expandability.

## **CHAPTER 3**

### **Technology Description**

#### **3.1 Scala - The Scala Programming Language**

Scala, short for Scalable Language, is a hybrid functional programming language. It was created by Martin Odersky. Scala smoothly integrates the features of object-oriented and functional languages. Scala is compiled to run on the Java Virtual Machine.

Many existing companies, who depend on Java for business-critical applications, are turning to Scala to boost their development productivity, applications scalability and overall reliability.

##### **3.1.1 History of Scala**

Scala is a general-purpose programming language. It was created and developed by Martin Odersky. Martin started working on Scala in 2001 at the Ecole Polytechnique Federale de Lausanne (EPFL). It was officially released on January 20, 2004.

Scala is not an extension of Java, but it is completely interoperable with it. While compilation, Scala file translates to Java bytecode and runs on JVM (Java Virtual machine).

Scala was designed to be both object-oriented and functional. It is a pure object-oriented language in the sense that every value is an object and functional language in the sense that every function is a value. The name of scala is derived from word scalable which means it can grow with the demand of users.

##### **3.1.2 Popularity of Scala**

- Twitter have announced that it had switched large portions of its backend from Ruby to Scala and intended to convert the rest.
- Apple Inc. uses Scala in certain teams, along with Java and the Play framework.

- The New York Times revealed in 2014 that its internal content management system Blackbeard is built using Scala, Akka and Play Framework.
- There are teams within Google that use Scala, mostly due to acquisitions such as Firebase and Nest.
- The Walmart Canada Uses Scala for their back-end platform.

### 3.1.3 Where to use Scala



**Figure 3.1.3 Scala Usage**

### 3.1.4 Scala Features

- 1) **Scala is object-oriented** - Scala is a pure object-oriented language in the sense that every value is an object. Types and behavior of objects are described by classes and traits which will be explained in subsequent chapters.

- 2) **Scala is functional** - Scala is also a functional language in the sense that every function is a value and every value is an object so ultimately every function is an object.

Scala provides a lightweight syntax for defining anonymous functions, it supports higher-order functions, it allows functions to be nested, and supports currying. These concepts will be explained in subsequent chapters.

- 3) **Scala is statically typed** - Scala, unlike some of the other statically typed languages (C, Pascal, Rust, etc.), does not expect you to provide redundant type information. You don't have to specify a type in most cases, and you certainly don't have to repeat it.
- 4) **Scala runs on the JVM** - Scala is compiled into Java Byte Code which is executed by the Java Virtual Machine (JVM). This means that Scala and Java have a common runtime platform. You can easily move from Java to Scala.

The Scala compiler compiles your Scala code into Java Byte Code, which can then be executed by the 'scala' command. The 'scala' command is similar to the java command, in that it executes your compiled Scala code.

- 5) **Scala can Execute Java Code** - Scala enables you to use all the classes of the Java SDK and also your own custom Java classes, or your favorite Java open-source projects.
- 6) **Scala can do Concurrent & Synchronize processing** - Scala allows you to express general programming patterns in an effective way. It reduces the number of lines and helps the programmer to code in a type-safe way. It allows you to write codes in an immutable manner, which makes it easy to apply concurrency and parallelism (Synchronize).
- 7) **Type Inference:** In Scala, you don't require to mention data type and function return type explicitly. Scala is enough smart to deduce the type of data. The return type of function is determined by the type of last expression present in the function.
- 8) **Singleton object:** In Scala, there are no static variables or methods. Scala uses singleton object, which is essentially class with only one object in the source file. Singleton object is declared by using object instead of class keyword.
- 9) **Immutability:** Scala uses immutability concept. Each declared variable is immutable by default. Immutable means you can't modify its value. You can also create mutable variables which can be changed.
- 10) **Lazy Computation:** In Scala, computation is lazy by default. Scala evaluates expressions only when they are required. You can declare a lazy variable by using lazy keyword. It is used to increase performance.

**11) Case classes and Pattern matching:** Scala case classes are just regular classes which are immutable by default and decomposable through pattern matching. All the parameters listed in the case class are public and immutable by default. Case classes support pattern matching. So, you can write more logical code.

**12) Concurrency control:** Scala provides standard library which includes the actor model. You can write concurrency code by using actor. Scala provides one more platform and tool to deal with concurrency known as Akka. Akka is a separate open-source framework that provides actor-based concurrency. Akka actors may be distributed or combined with software transactional memory.

**13) String Interpolation:** Since Scala 2.10.0, Scala offers a new mechanism to create strings from your data. It is called string interpolation. String interpolation allows users to embed variable references directly in processed string literals. Scala provides three string interpolation methods: s, f and raw.

**14) Higher Order Functions:** Higher order function is a function that either takes a function as argument or returns a function. In other words, we can say a function which works with another function is called higher order function.

Higher order function allows you to create function composition, lambda function or anonymous function etc.

**15) Traits:** A trait is like an interface with a partial implementation. In Scala, trait is a collection of abstract and non-abstract methods. You can create trait that can have all abstract methods or some abstract and some non-abstract methods.

Traits are compiled into Java interfaces with corresponding implementation classes that hold any methods implemented in the traits.

**16) Rich Set of Collection:** Scala provides rich set of collection library. It contains classes and traits to collect data. These collections can be mutable or immutable. You can use it according to your requirement.

Scala.collection.mutable package contains all the mutable collections. You can add, remove and update data while using this package.

Scala.collection.immutable package contains all the immutable collections. It does not allow you to modify data.



### 3.1.5 How to install Scala on Windows?

#### STEP 1: Verifying Java Packages:

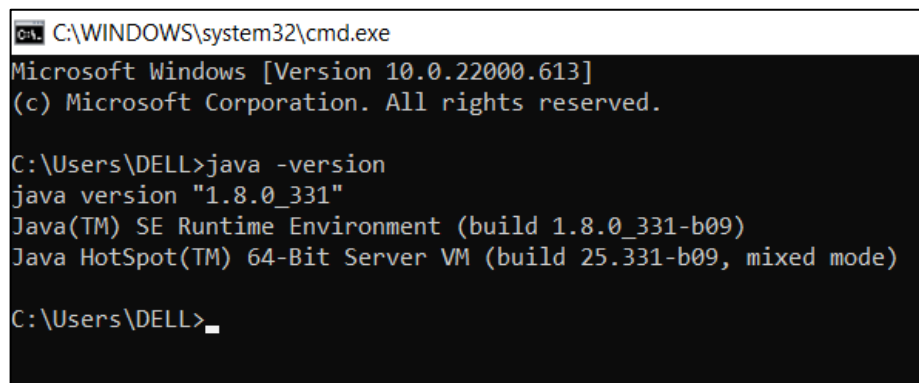
The first thing we need to have is a Java Software Development Kit (SDK) installed on the computer. We need to verify this SDK packages and if not installed then install them.

Just go to the **Command line** (Windows + **R**).

Now run the following command:

***java -version***

Once this command is executed the output will show the java version and the output will be as follows:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>java -version
java version "1.8.0_331"
Java(TM) SE Runtime Environment (build 1.8.0_331-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.331-b09, mixed mode)

C:\Users\DELL>
```


In case we are not having the SDK installed then download the latest version according to the computer requirements from [oracle.com](https://www.oracle.com) and just proceed with the installation.

#### STEP 2: Downloading and Installing Scala


##### i. Downloading Scala:


Before starting with the installation process, you need to download it. For that, all versions of Scala for Windows are available on [scala-lang.org](https://scala-lang.org)


## Install or Upgrade Scala


 GET STARTED WITH SCALA


 or 


 PICK A SPECIFIC RELEASE

 Scala Book

 Tour of Scala

 Online Courses

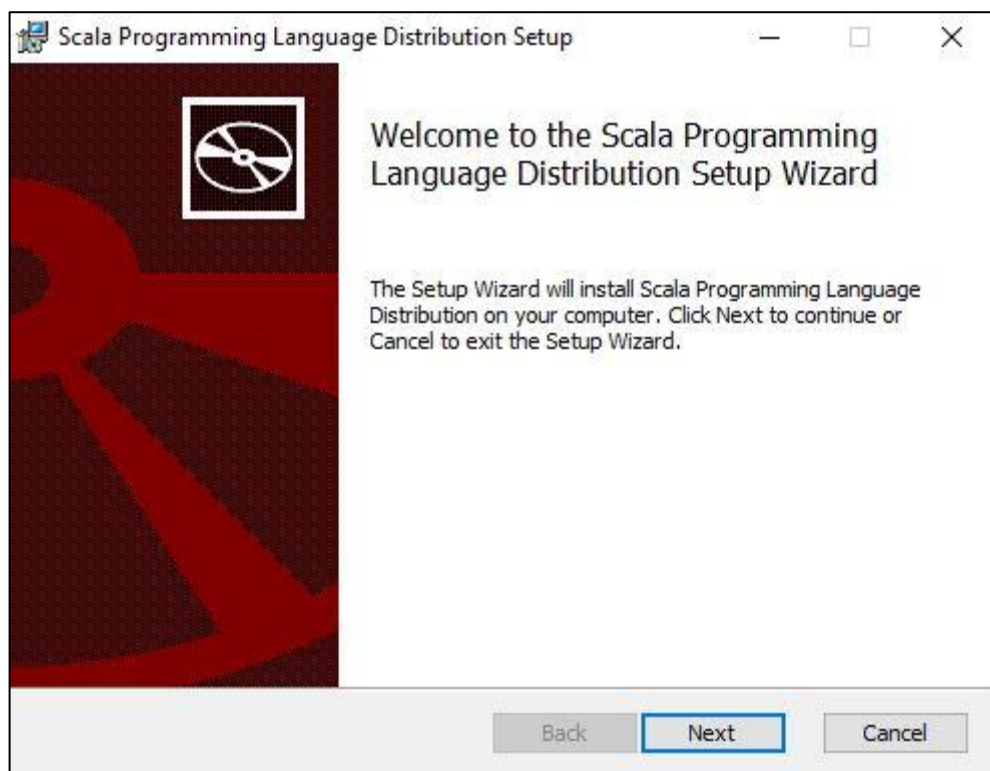
 Scala 3 documentation

 Migrating from Scala 2 to Scala 3

Download the Scala and follow the further instructions for the installation of Scala.

## ii. Beginning with the Installation:

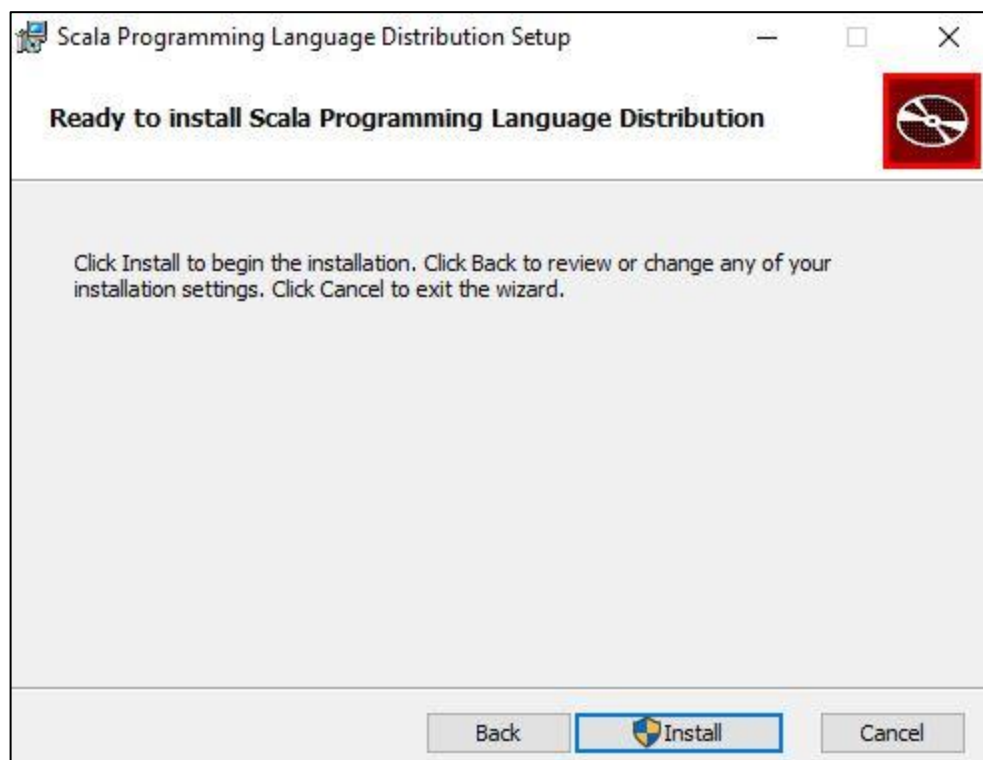
### ➤ Getting Started:



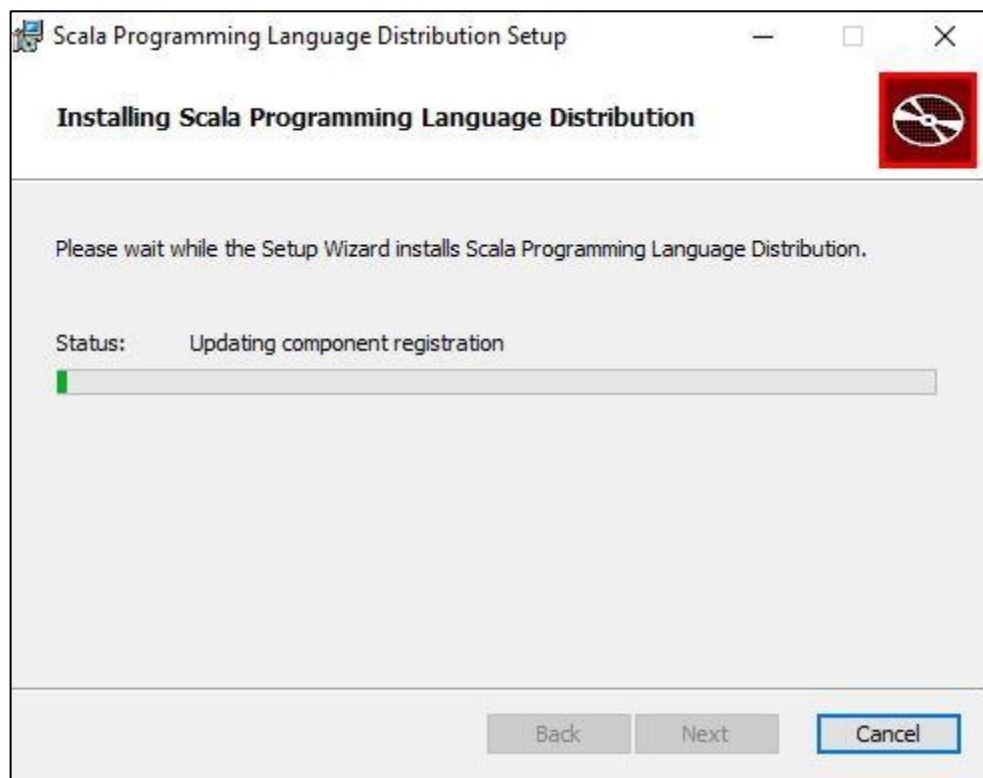
➤ **Getting done with the User's License Agreement:**



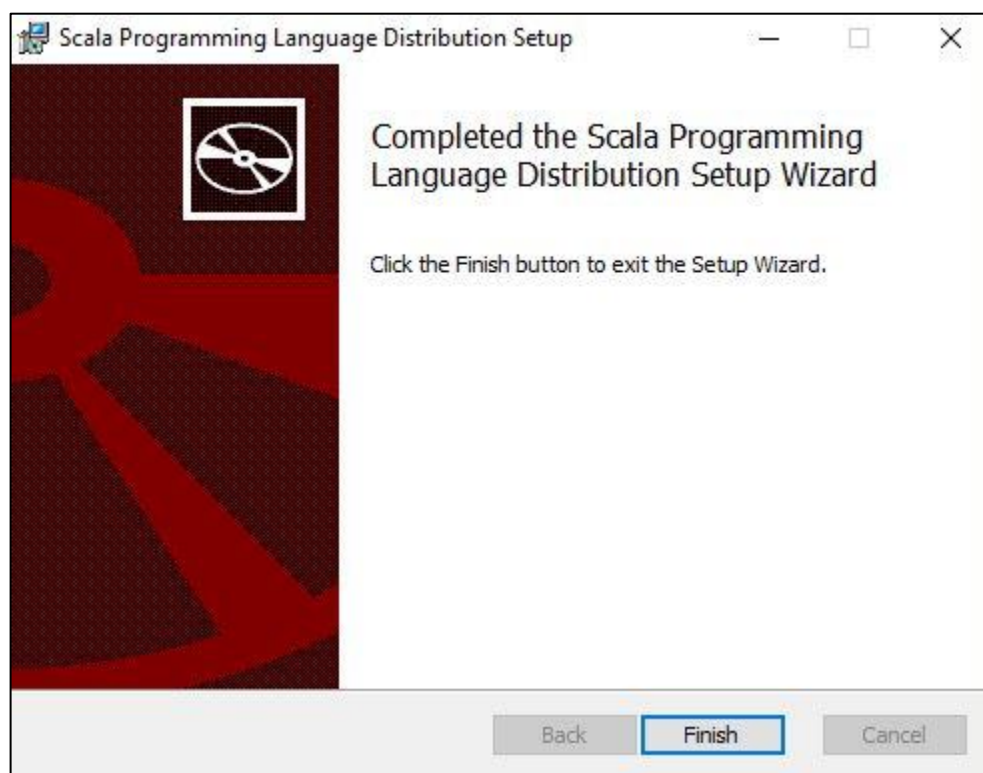
➤ **Move on to Installing:**



➤ **Installation Process:**



➤ **Installation Finished:**



## 3.1.6 First Scala Program

### 3.1.2.1 Basic Syntax

The following are the basic syntaxes and coding conventions in Scala programming.

- 1) **Case Sensitivity:** Scala is case-sensitive, which means identifier Hello and hello would have different meaning in Scala.
- 2) **Class Names:** For all class names, the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

Example – `class MyFirstScalaClass`.

- 3) **Method Names:** All method names should start with a Lower-Case letter. If multiple words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

Example – `def myMethodName()`

- 4) **Program File Name:** Name of the program file should exactly match the object name. When saving the file, you should save it using the object name (Remember Scala is case-sensitive) and append 'scala' to the end of the name. (If the file name and the object name do not match your program will not compile).

Example – Assume 'HelloWorld' is the object name. Then the file should be saved as 'HelloWorld.scala'.

- 5) **def main (args: Array[String]):** Scala program processing starts from the main() method which is a mandatory part of every Scala Program.

### Hello World Program:

```
object HelloWorld {  
  def main(args: Array[String]) {  
    println("Hello World!");  
  }  
}
```

### Output:

```
Hello World!
```

## 3.2 Apache Spark

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size. It provides development APIs in Java, Scala, Python and R, and supports code reuse across multiple workloads—batch processing, interactive queries, real-time analytic, machine learning, and graph processing.

### 3.2.1 Apache Spark VS Apache Hadoop

		
Processing data in real time	✗	✓
Handle input for multiple sources	✓	✓
Easy to use	✗	✓
Faster Processing	✗	✓

**Figure 3.2.1 Apache Spark VS Apache Hadoop**

### 3.2.2 Spark Features

- 1. Fast Processing:** Using Apache Spark, we achieve a high data processing speed of about 100x faster in memory and 10x faster on the disk. This is made possible by reducing the number of read-write to disk.

2. **Flexibility:** Apache Spark supports multiple languages and allows developers to write applications in Java, Scala, R, or Python.
3. **In-memory computation:** Spark stores data in the RAM of servers, which allows it to access data quickly, and in-turn this accelerates the speed of analytic.
4. **Real Time Processing:** Spark is able to process real-time streaming data. Unlike MapReduce, which processes the stored data, Spark is able to process the real-time data and hence is able to produce instant outcomes.
5. **Fault Tolerance in Spark:** It provides fault-tolerance through Spark abstraction-RDD. Spark RDD are designed to handle the failure of any worker node in the cluster. Thus, it ensures that the loss of data reduces to zero.
6. **Advance Analytics:** Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

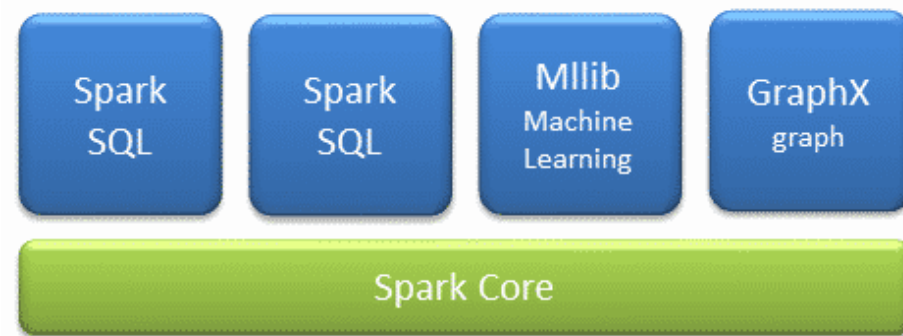


**Figure 3.2.2 Apache Spark Features**



### 3.2.3 Spark Components

The Spark project consists of different types of tightly integrated components. At its core, Spark is a computational engine that can schedule, distribute and monitor multiple applications.



**Figure 3.2.3 Apache Spark Components**

- 1) **Spark Core:** The Spark Core is the heart of Spark and performs the core functionality. It holds the components for task scheduling, fault recovery, interacting with storage systems and memory management.
- 2) **Spark SQL:** The Spark SQL is built on the top of Spark Core. It provides support for structured data. It allows to query the data via SQL (Structured Query Language) as well as the Apache Hive variant of SQL called the HQL (Hive Query Language). It also supports various sources of data like Hive tables, Parquet, and JSON.
- 3) **Spark Streaming:** Spark Streaming is a Spark component that supports scalable and fault-tolerant processing of streaming data. It uses Spark Core's fast scheduling capability to perform streaming analytic. It accepts data in mini-batches and performs RDD transformations on that data. Its design ensures that the applications written for streaming data can be reused to analyze batches of historical data with little modification. The log files generated by web servers can be considered as a real-time example of a data stream.

- 4) **MLlib:** The MLlib is a Machine Learning library that contains various machine learning algorithms. It is nine times faster than the disk-based implementation used by Apache Mahout.
- 5) **GraphX:** The GraphX is a library that is used to manipulate graphs and perform graph-parallel computations. It facilitates to create a directed graph with arbitrary properties attached to each vertex and edge. To manipulate graph, it supports various fundamental operators like subgraph, join vertices and aggregated messages.

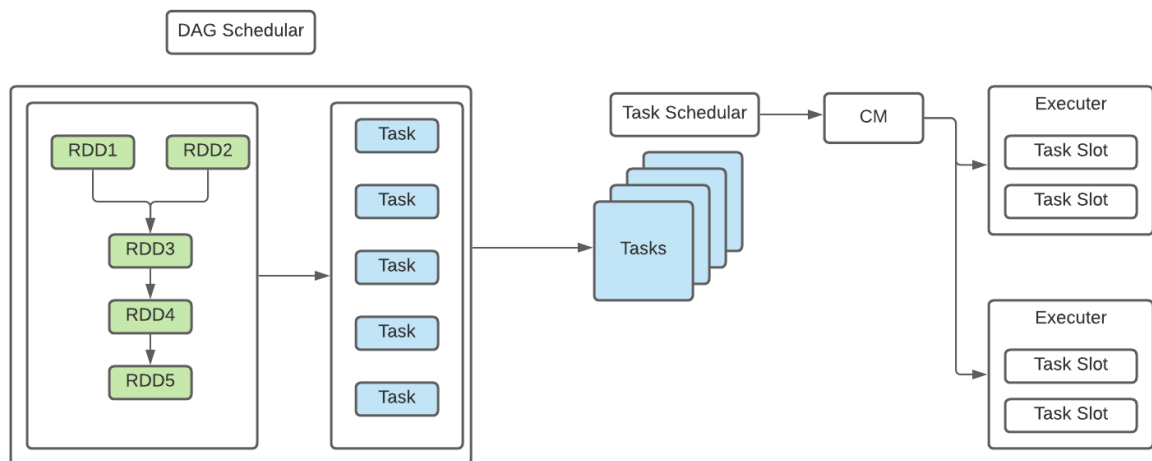
### 3.2.4 Spark Architecture

#### 3.2.4.1 Process

When a client submits a spark user application code, the driver implicitly converts the code containing transformations and actions into a logical directed acyclic graph (DAG). At this stage, the driver program also performs certain optimizations like pipelining transformations and then it converts the logical DAG into physical execution plan with set of stages. After creating the physical execution plan, it creates small physical execution units referred to as tasks under each stage. Then tasks are bundled to be sent to the Spark Cluster.

The driver program then talks to the cluster manager and negotiates for resources. The cluster manager then launches executors on the worker nodes on behalf of the driver. At this point the driver sends tasks to the cluster manager based on data placement. Before executors begin execution, they register themselves with the driver program so that the driver has holistic view of all the executors.

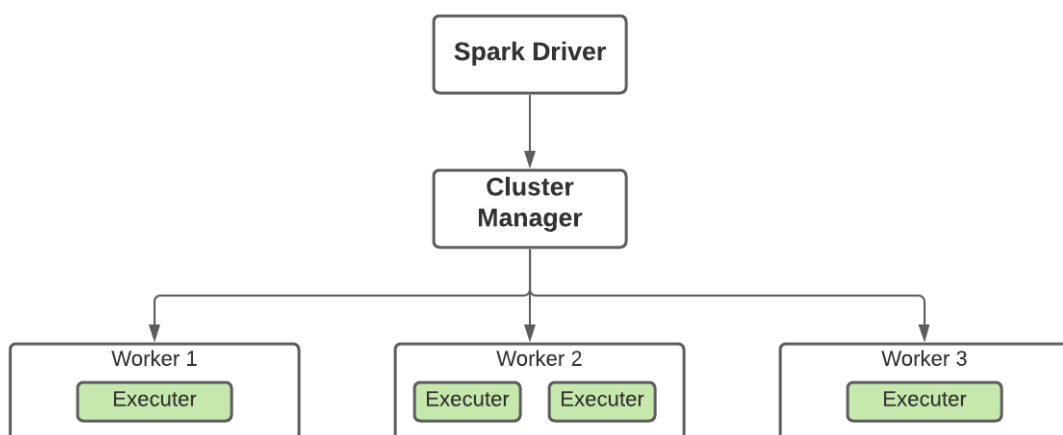
Now executors start executing the various tasks assigned by the driver program. At any point of time when the spark application is running, the driver program will monitor the set of executors that run. Driver program in the spark architecture also schedules future tasks based on data placement by tracking the location of cached data. When driver programs main () method exits or when it calls the stop () method of the Spark Context, it will terminate all the executors and release the resources from the cluster manager.



**Figure 3.2.4.1 Apache Spark Architecture - Process**

### 3.2.4.2 Driver in Spark Architecture

It is the entry point of the Spark Shell. The driver program runs the `main()` function of the application and is the place where the Spark Context and RDDs are created, and also where transformations and actions are performed. Spark Driver performs two main tasks: Converting user programs into tasks and planning the execution of tasks by executors. Driver translates the RDD's into the execution graph and splits the graph into multiple stages.



**Figure 3.2.4.1 Apache Spark Architecture - Driver**

### 3.2.4.3 Executor in Spark Architecture

An executor is a distributed agent responsible for the execution of tasks. Every spark application has its own executor process.

- Executor performs all the data processing and returns the results to the Driver.
- Reads from and writes data to external sources.
- Executor stores the computation results in data in-memory, cache or on hard disk drives.
- Interacts with the storage systems.

Count of executor depend upon workload, if workload is high then cluster manager can increase the executors.

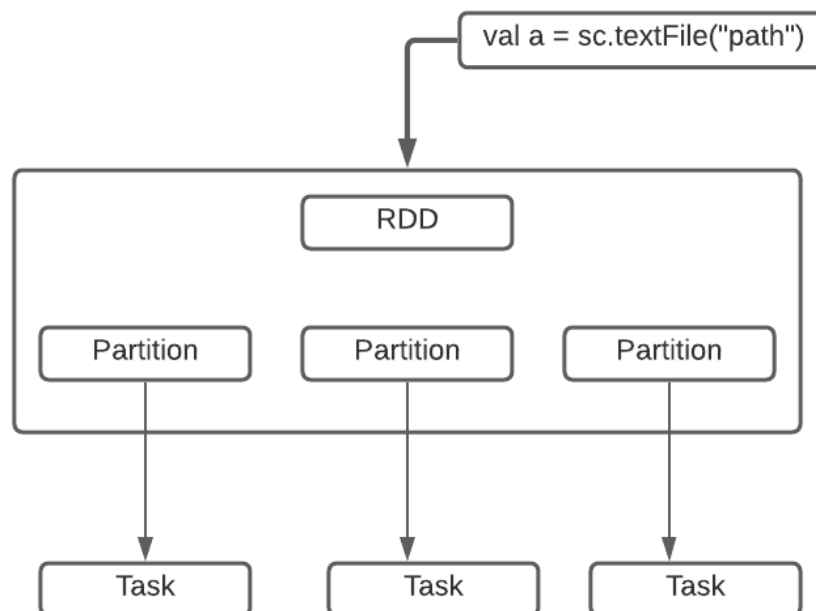
### 3.2.4.4 Cluster Manager

An external service is responsible for acquiring resources on the Spark cluster and allocating them to a spark job. Cluster Manager keeps track of the available resources (nodes) available in the cluster. Multiple spark applications could be run on a single cluster. We have different cluster manager like, Hadoop YARN, Apache Mesos, Kubernetes, or the simple standalone spark cluster manager.

- 1) **Standalone Cluster Manager:** Standalone cluster manager is a simple cluster manager that comes included with the Spark. It contains one master and several workers, each having a configured size of memory and CPU cores. It allows running applications in client mode and cluster mode.
- 2) **Hadoop YARN:** YARN is another option for Cluster Manager in Spark. It was introduced in Hadoop 2.0 and supports utilizing varied data processing frameworks on a distributed resource pool. It allows running applications in client mode and cluster mode.
- 3) **Apache Mesos:** This cluster manager is a common-purpose cluster manager that can perform both analytics workloads and long-running utilities (e.g., web applications or key/value stores) on a cluster. It allows running applications only in client mode.

### 3.2.4.5 Spark RDD - Resilient Distributed Dataset

- RDD is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. It can contain any type of Python, Java, or Scala objects.
- There are two ways to create RDDs – parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.



**Figure 3.2.4.5 Apache Spark RDD**

#### 3.2.4.5.1 Features of RDD

- 1) RDD is Immutable.
- 2) It performs lazy Evaluation.
- 3) In memory computation.
- 4) RDD is fault tolerant.

### 3.2.4.5.2 RDD Creation

```
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.Session

object RDDParallelize {

  def main(args: Array[String]): Unit = {
    val spark:Session = Session.builder().master("local[1]")
      .appName("SparkByExamples.com")
      .getOrCreate()

    val rdd:RDD[Int] = spark.sparkContext.parallelize(List(1,2,3,4,5))
    val rddCollect:Array[Int] = rdd.collect()
    println("Number of Partitions: "+rdd.getNumPartitions)
    println("Action: First element: "+rdd.first())
    println("Action: RDD converted to Array[Int]: ")
    rddCollect.foreach(println)
  }
}
```

### Output

```
22/05/05 02:26:52 INFO TaskSchedulerImpl: Adding task set 0.0 with 1 tasks
22/05/05 02:26:52 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS_LOCAL, 5901 bytes)
22/05/05 02:26:52 INFO Executor: Running task 0.0 in stage 0.0 (TID 0)
22/05/05 02:26:52 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 1001 bytes result sent to driver
22/05/05 02:26:52 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 183 ms on localhost (executor driver) (1/1)
22/05/05 02:26:52 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
22/05/05 02:26:52 INFO DAGScheduler: ResultStage 0 (collect at RDDParallelize.scala:13) finished in 0.216 s
22/05/05 02:26:52 INFO DAGScheduler: Job 0 finished: collect at RDDParallelize.scala:13, took 0.841771 s
Number of Partitions: 1
22/05/05 02:26:52 INFO SparkContext: Starting job: first at RDDParallelize.scala:15
22/05/05 02:26:52 INFO DAGScheduler: Got job 1 (first at RDDParallelize.scala:15) with 1 output partitions
22/05/05 02:26:52 INFO DAGScheduler: Final stage: ResultStage 1 (first at RDDParallelize.scala:15)
22/05/05 02:26:52 INFO DAGScheduler: Parents of final stage: List()
22/05/05 02:26:52 INFO DAGScheduler: Missing parents: List()
22/05/05 02:26:52 INFO DAGScheduler: Submitting ResultStage 1 (ParallelCollectionRDD[0] at parallelize at RDDParallelize.scala:12), which has no missing parents
22/05/05 02:26:52 INFO MemoryStore: Block broadcast_1_piece0 stored as values in memory (estimated size 1408.0 B, free 877.2 MB)
22/05/05 02:26:52 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 950.0 B, free 877.2 MB)
22/05/05 02:26:52 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on 192.168.179.28:58627 (size: 950.0 B, free: 877.2 MB)
22/05/05 02:26:52 INFO SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:996
22/05/05 02:26:52 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 1 (ParallelCollectionRDD[0] at parallelize at RDDParallelize.scala:12)
22/05/05 02:26:52 INFO TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
22/05/05 02:26:52 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, localhost, executor driver, partition 0, PROCESS_LOCAL, 5899 bytes)
22/05/05 02:26:52 INFO Executor: Running task 0.0 in stage 1.0 (TID 1)
22/05/05 02:26:52 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1). 903 bytes result sent to driver
22/05/05 02:26:52 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 73 ms on localhost (executor driver) (1/1)
22/05/05 02:26:52 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
22/05/05 02:26:52 INFO DAGScheduler: ResultStage 1 (first at RDDParallelize.scala:15) finished in 0.074 s
22/05/05 02:26:52 INFO DAGScheduler: Job 1 finished: first at RDDParallelize.scala:15, took 0.088778 s
Action: First element: 1
Action: RDD converted to Array[Int]:
1
2
3
4
5
22/05/05 02:26:52 INFO SparkContext: Invoking stop() from shutdown hook
22/05/05 02:26:52 INFO BlockManagerInfo: Removed broadcast_0_piece0 on 192.168.179.28:58627 in memory (size: 924.0 B, free: 877.2 MB)
22/05/05 02:26:52 INFO SparkUI: Stopped Spark web UI at http://192.168.179.28:4040
22/05/05 02:26:52 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/05/05 02:26:52 INFO MemoryStore: MemoryStore cleared
22/05/05 02:26:52 INFO BlockManager: BlockManager stopped
```

### 3.2.4.1 RDD Transformation Function

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf

object Transformations {
  def main(args: Array[String]) {

    val sparkConf = new SparkConf()
    sparkConf.setAppName("firstApp")
    sparkConf.setMaster("local")

    val sc = new SparkContext(sparkConf)

    // Map
    val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5))
    val maprdd1 = rdd1.collect().map(x => x + 5)
    maprdd1.foreach(println)

    // Filter
    val rdd2 = sc.parallelize(Array(1, 2, 3, 4, 3))
    val filterrdd2 = rdd2.filter(x => x != 3);
    filterrdd2.foreach(println)

    // map
    val rdd3 = sc.parallelize(Array(1, 2))
    val maprdd = rdd3.map(x => x * 3);
    maprdd.foreach(println)

    // distinct
    val rdd4 = sc.parallelize(Array(1, 2, 2, 3, 3, 4, 4, 5))
    val distictrdd = rdd4.distinct()
    distictrdd.foreach(println)

    // Union
    val unionrdd = rdd3.union(rdd4)
    unionrdd.foreach(println)

    // intersection
    val intersectionrdd = rdd3.intersection(rdd4)
    intersectionrdd.foreach(println)

  }
}
```

## Output

6  
7  
8  
9  
10

1  
2  
4

3  
6

4  
1  
3  
5  
2

1  
2

1  
2  
2  
3  
3  
4  
4  
5

1  
2



## 3.3 SQL Server

SQL Server is a relational database management system, or RDBMS, developed and marketed by Microsoft.

Similar to other RDBMS software, SQL Server is built on top of SQL, a standard programming language for interacting with relational databases. SQL Server is tied to Transact-SQL, or T-SQL, the Microsoft's implementation of SQL that adds a set of proprietary programming constructs.

SQL Server works exclusively on the Windows environment for more than 20 years. In 2016, Microsoft made it available on Linux. SQL Server 2017 became generally available in October 2016 that ran on both Windows and Linux.

### 3.3.1 SQL Server Architecture

SQL Server consists of two main components:

#### 1) Database Engine

The core component of the SQL Server is the Database Engine. The Database Engine consists of a relational engine that processes queries and a storage engine that manages database files, pages, indexes, etc.

The database objects such as stored procedures, views, and triggers are also created and executed by the Database Engine.

**Relational Engine:** The Relational Engine contains the components that determine the best way to execute a query. The relational engine is also known as the query processor.

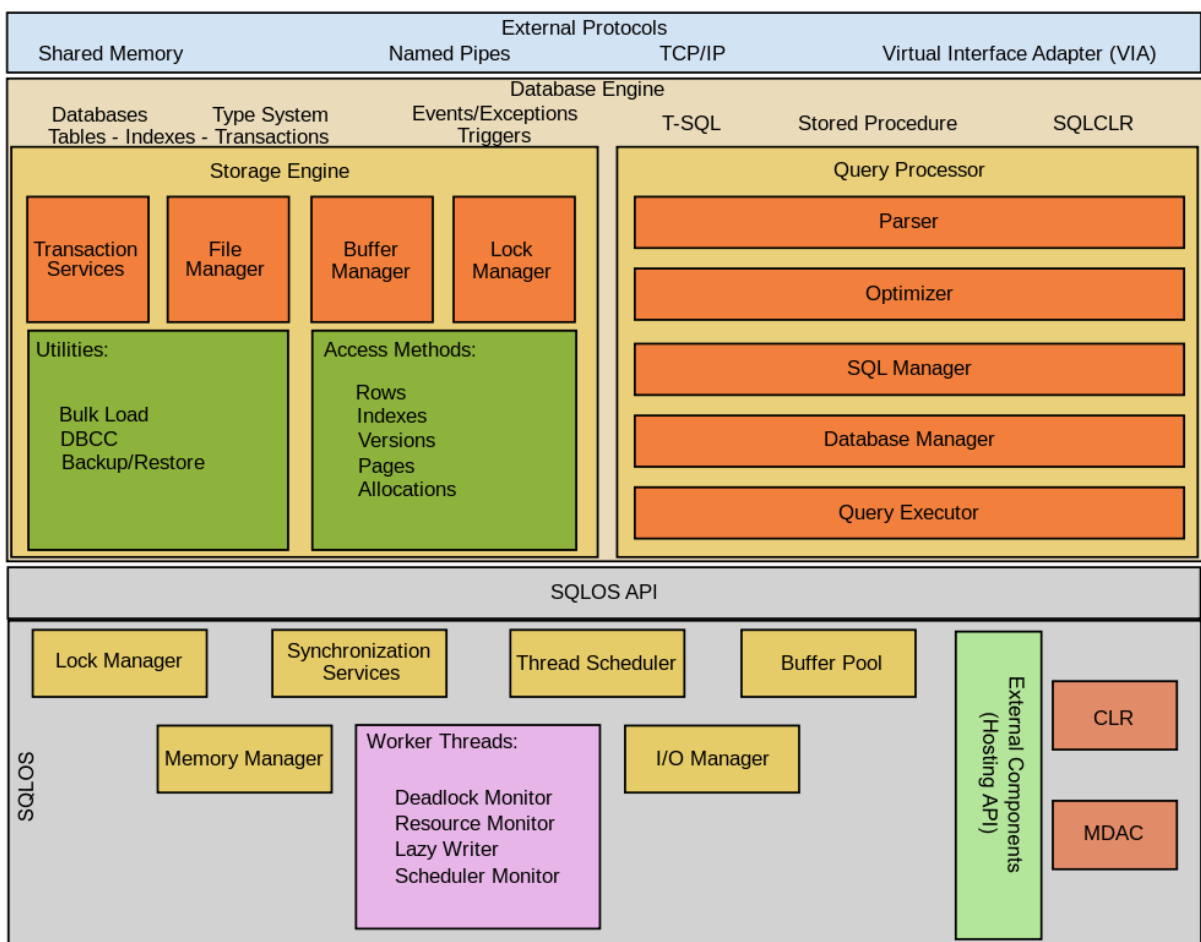
- The relational engine requests data from the storage engine based on the input query and processed the results.
- Some tasks of the relational engine include querying processing, memory management, thread and task management, buffer management, and distributed query processing.

**Storage Engine:** The storage engine is in charge of storage and retrieval of data from the storage systems such as disks and SAN.

## 2) SQLOS

Under the relational engine and storage engine is the SQL Server Operating System or SQLOS. SQLOS provides many operating system services such as memory and I/O management. Other services include exception handling and synchronization services.

The following diagram illustrates the architecture of the SQL Server:



**Figure 3.3.1 SQL Server Architecture**

# **CHAPTER 4**

## **Designing & Implementation**

### **4.1 Procedure Adopted**

#### **4.1.1 Proposed Work**

This project involves majorly data extraction, removing redundancy from customer data and entering them in new system after cleansing for business insights as well as to have accurate customer information across whole organization.

#### **4.1.2 Approach**

##### **➤ STEP 1**

First step in the DATA MIGRATION project is analysis of the data that is received. This data is first categorized as structured or unstructured data and the whole process is first documented.

##### **➤ STEP 2**

Once the analysis of the data is completed and validated, we extract the useful and meaningful data by cleaning it, that includes removing the blank spaces, extra commas, hyphens, or fixing duplicate, inappropriate or corrupted data.

##### **➤ STEP 3**

After the validation of the whole cleansing process, next step was to map the records that seemed similar. For comparing the different records, we used 2 algorithms: Jaccard Similarity and Jaro - Winkler.

These two algorithms acted according to the needs and conditions, in few places Jaccard Similarity is being used and in other Jaro - Winkler.

#### ➤ **STEP 4**

Next step was to load or migrate the mapped data into the target system for that we then identified which all attributes are necessary for the analysis or deduplication. We created a GLOBAL ENTITY table for that.

#### ➤ **STEP 5**

Once the data is loaded into the target system it requires two validations:

**1) Technical Validation**

**2) Business Validation**

#### ➤ **STEP 6**

After all these steps the data gets ready for the business insights.

## 4.2 Proposed Algorithm

### 4.2.1 JACCARD SIMILARITY

Jaccard Similarity is used to measure the similarity between two sets of data to see which members are shared and distinct. Jaccard Similarity can be used to find the similarity between two asymmetric binary vectors or between two sets.

It is calculated by dividing the number of observations in both sets by the number of observations in either set. In other words, the Jaccard similarity can be computed as the size of the intersection divided by the size of the union of two sets.

This can be written in set notation using intersection ( $A \cap B$ ) and unions ( $A \cup B$ ) of two sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$J$  = Jaccard distance

$A$  = set 1

$B$  = set 2

where  $|A \cap B|$  gives the number of members shared between both sets and  $|A \cup B|$  gives the total number of members in both sets (shared and un-shared). The Jaccard Similarity will be 0 if the two sets don't share any values and 1 if the two sets are identical. The set may contain either numerical values or strings.

### 4.2.2 JARO - WINKLER

The **Jaro - Winkler distance** is a string metric measuring an edit distance between two sequences.

*The lower the Jaro - Winkler distance for two strings is, the more similar the strings are.*

The score is normalized such that 0 means an exact match and 1 means there is no similarity. The Jaro–Winkler similarity is the inversion,  $\{1 - (\text{Jaro–Winkler distance})\}$ .

### 4.2.3 Working

The Jaccard similarity index (sometimes called the Jaccard similarity *coefficient*) compares members for two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, and can be normalized in between 0 - 1.

Jaccard Similarity will be 0 if the two sets don't share any values and 1 if the two sets are identical. The set may contain either numerical values or strings.

The formula to find the Index is:

$$J(X, Y) = |X \cap Y| / |X \cup Y|$$

In Steps, that's:

1. Count the number of members which are shared between both sets.
2. Count the total number of members in both sets (shared and un-shared).
3. Divide the number of shared members (1) by the total number of members (2).

#### Examples:

The Jaro Similarity is calculated using the following formula

$$Jaro\ similarity = \begin{cases} 0, & \text{if } m=0 \\ \frac{1}{3} \left( \frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m} \right), & \text{for } m \neq 0 \end{cases}$$

where:

- **m** is the number of matching characters
- **t** is half the number of transpositions
- **|s1|** and **|s2|** are the lengths of strings s1 and s2 respectively

The characters are said to be matching if they are the same and the characters are not further than  $\left\lfloor \frac{\max(|s1|, |s2|)}{2} \right\rfloor - 1$ . Transpositions are half the number of matching characters in both strings but in a different order.

### **Calculation:**

Let  $s1 = \text{"arnab"}$ ,  $s2 = \text{"raanb"}$ , so the maximum distance to which each character is matched is 1.

It is evident that both the strings have 5 matching characters, but the order is not the same, so the number of characters that are not in order is 4, so the number of transpositions is 2.

Therefore, Jaro similarity can be calculated as follows:

$$\text{Jaro Similarity} = (1/3) * \{(5/5) + (5/5) + (5-2)/5\} = \mathbf{0.86667}$$

## **4.3 Platform Specification**

- Operating system: Windows 10
- Back-End: Spark, Scala
- Database: SQL Server

### **4.3.1 Software requirements**

- Scala IDE
- Apache Spark

### **4.3.2 Hardware requirements**

- RAM – 8.00 GB
- Processor – i5 11<sup>th</sup> Gen Intel Core Processor
- Hard Drive Space – 1TB 256 GB SSD



## **CHAPTER 5**

### **Designing Methodology**

#### **5.1 Designing Mechanism**

Software design is a mechanism to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. It deals with representing the client's requirement, as described in SRS (Software Requirement Specification) document, into a form, i.e., easily implementable using programming language.

The software design phase is the first step in SDLC (Software Design Life Cycle), which moves the concentration from the problem domain to the solution domain. In software design, we consider the system to be a set of components or modules with clearly defined behaviours & boundaries.

## 5.2 ER Diagram: Entity Relationship Diagram

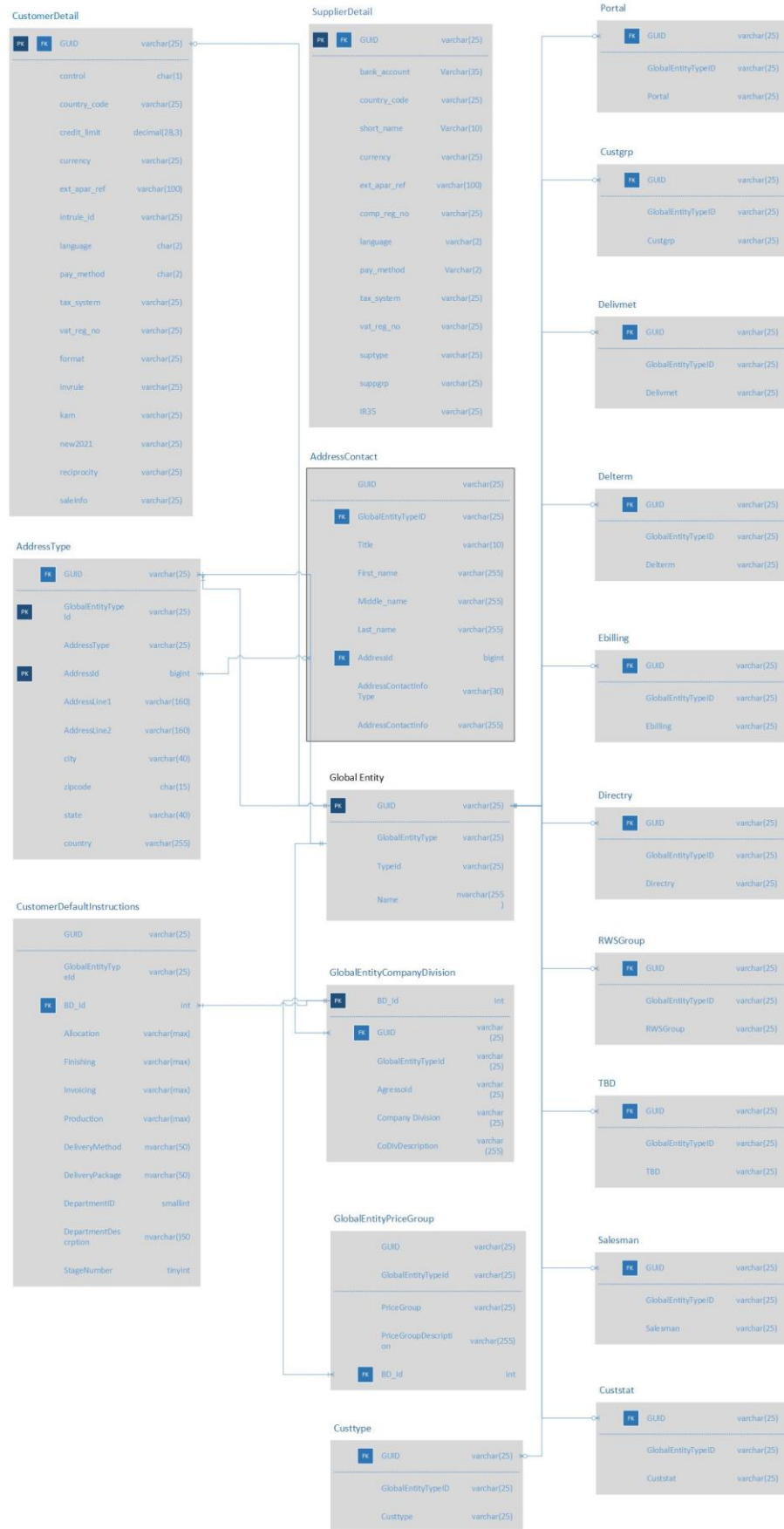
ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling method are called Entity-Relationship Diagrams or ER diagrams or ERDs.

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

### Purpose of ERD

- The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD.
- The ERD serves as a documentation tool.
- Finally, the ERD is used to connect the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.

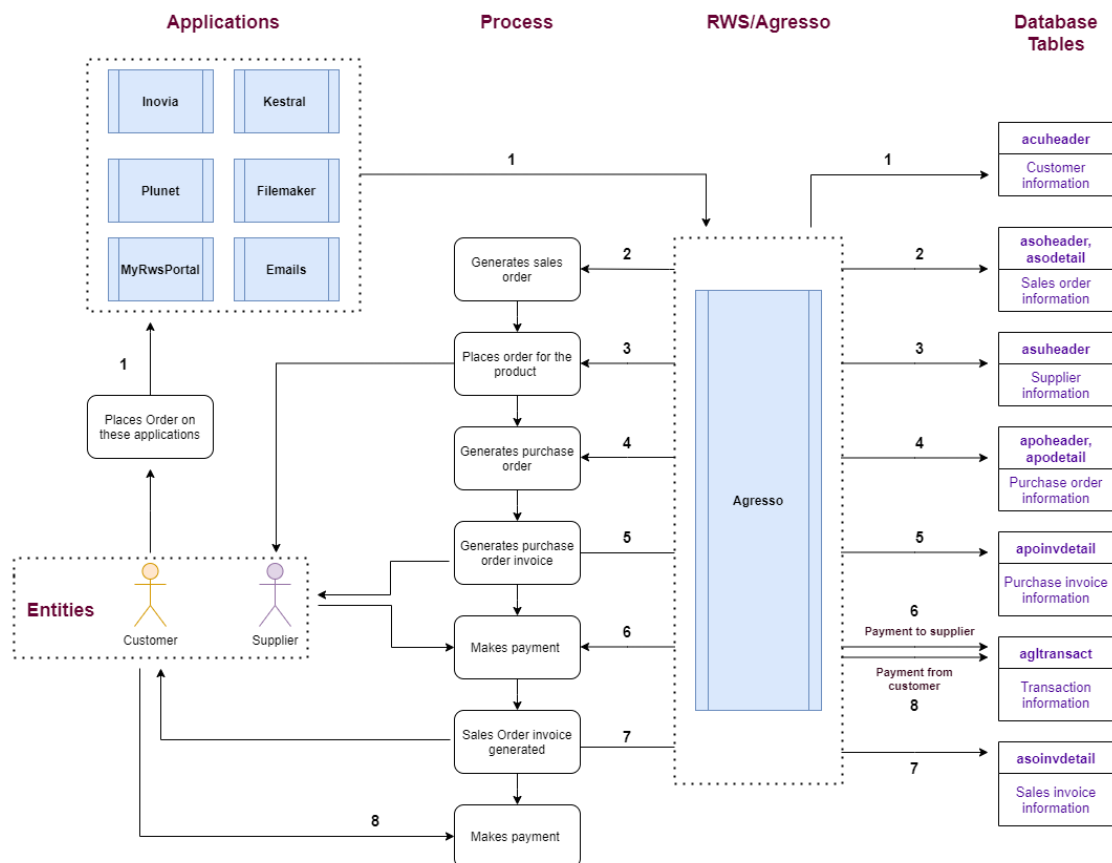


**Figure 5.2 ER Diagram**

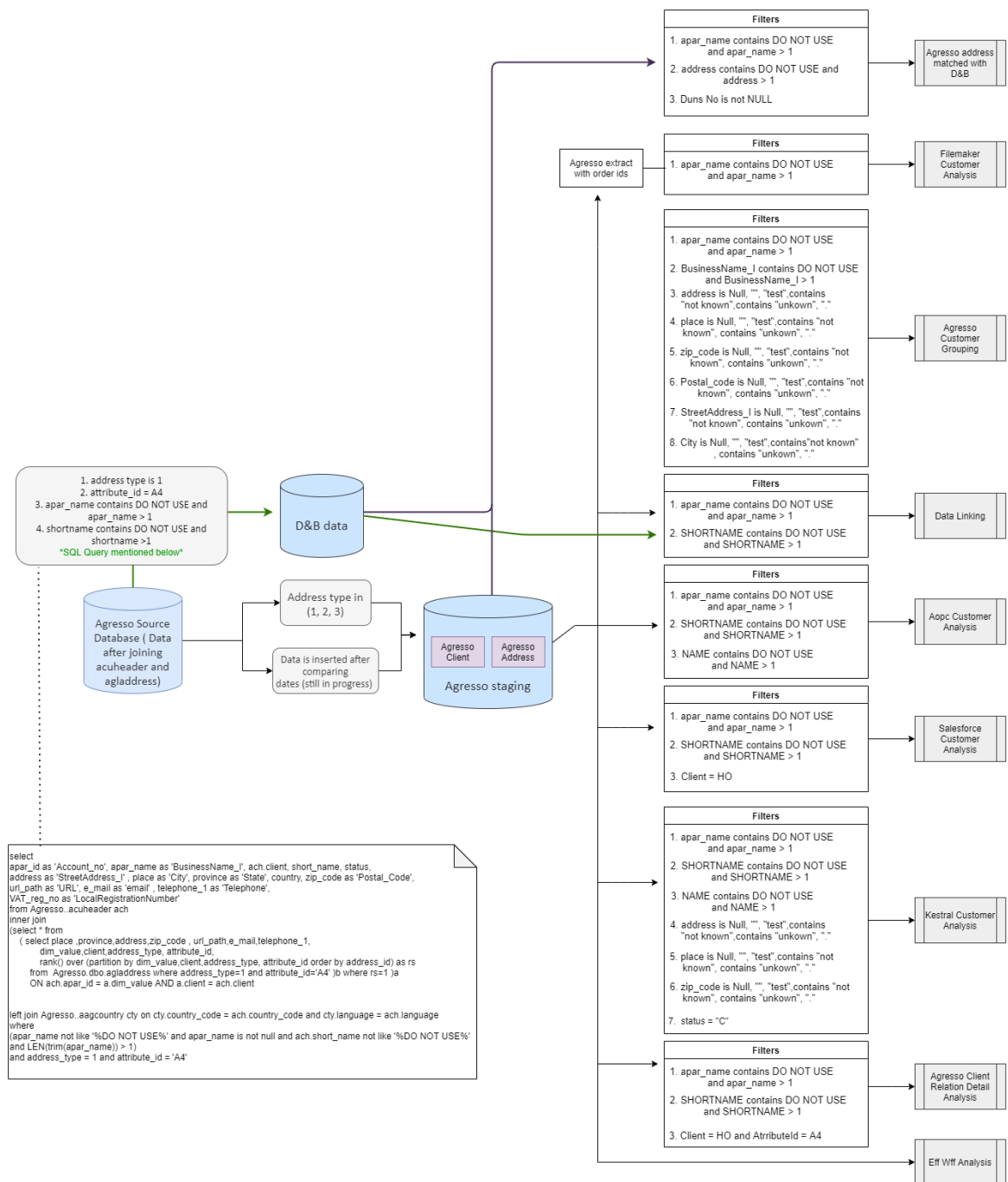
## 5.3 Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.



**Figure 5.3 Data Flow Diagram – 1**



***Figure 5.3 Data Flow Diagram – 2***

## **CHAPTER 6**

### **Results & Discussion**

Data Migration Project is capable to identify duplicate customers records and capable to group that duplicate records as a single customer master record with relative key fields.

The algorithms used for matching the duplicate customer data or fuzzy matching in the system are Jaccard Similarity and Jaro - Winkler that are used to compare the various strings - the customer records and returns a normalized value between 0 to 1. The score is normalized such that 0 means an exact match and 1 means there is no similarity. The lower the Jaro - Winkler distance for two strings is, the more similar the strings are.

These records are then combined and grouped together as a single customer master record that has the relative key fields.

## **CHAPTER 7**

### **Summary & Conclusion**

#### **Summary**

This project involves majorly data extraction, removing redundancy from customer data and entering them in new system after cleansing for business insights as well as to have accurate customer information across whole organization.

We achieved this using Apache Spark framework to process the data, to identify duplicate customers and grouped them as single customer master record with relative key fields.

#### **Conclusion**

Recent years have witnessed a common data quality problem in which we have multiple duplicate records distributed over multiple business domains in an application which doesn't has any validation.

There is a large body of theoretical and empirical work into this problem. The solution usually involves computing a new unique identifier column which allows entities to be linked and grouped, using a process of statistical estimation, machine learning into the target table i.e., the customer master record.

## **CHAPTER 8**

### **Future Scope**

#### **Future Scope**

Where customers are concerned, data migration can take center stage when it comes to analyzing their demands and behaviors as the customer experience is known to be one of the key focal points. This is where business and IT are able to meet from a transformational perspective. However, it's more than just the functions that start from front-end to customer-facing interactions, including the touchpoints. It's about linking the entire enterprise to shift as one giant phase that optimizes the culture, organization, operations and last but not least, the overall customer experience.



## Bibliography

- [https://www.statisticshowto.com/jaccard-index/#:~:text=The%20Jaccard%20similarity%20index%20\(sometimes,more%20similar%20the%20two%20populations.](https://www.statisticshowto.com/jaccard-index/#:~:text=The%20Jaccard%20similarity%20index%20(sometimes,more%20similar%20the%20two%20populations.)
- [https://en.wikipedia.org/wiki/Jaccard\\_index](https://en.wikipedia.org/wiki/Jaccard_index)
- [https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler\\_distance](https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance) <https://srinivas>
- <https://databricks.com/spark/about>