SpotifyDataAnalysis Implementation Guide

SpotifyDataAnalysis Implementation Guide

1. getSongWithMostStreams

```scala
def getSongWithMostStreams(l: List[Song]): Song =
  l.maxBy(_.streams)
```

---

2. getNameAndNumberOfTheArtistWithMostSongsInList

```scala
def getNameAndNumberOfTheArtistWithMostSongsInList(l: List[Song]): (String, Int) =
  l.groupBy(_.artist)
    .view.mapValues(_.size)
    .maxBy(_._2)
```

---

3. getArtistWithMostStreams

```scala
def getArtistWithMostStreams(l: List[Song]): (String, BigInt) =
  l.groupBy(_.artist)
    .view.mapValues(_.map(_.streams).sum)
    .maxBy(_._2)
```

---

4. getMinAndMaxAndAvgBPM

```scala
def getMinAndMaxAndAvgBPM(l: List[Song]): (Int, Int, Double) = {
  val (min, max, total, count) = l.foldLeft((Int.MaxValue, Int.MinValue, 0, 0)) {
    case ((min, max, sum, count), song) =>
      (math.min(min, song.bpm), math.max(max, song.bpm), sum + song.bpm, count + 1)
  }
  (min, max, total.toDouble / count)
}
```

---

5. getThe4MonthWithMostMinorSongs

```scala
```

```scala
def getThe4MonthWithMostMinorSongs(l: List[Song]): List[(Int, Double)] =
  l.filter(_.isMinor)
    .groupBy(song => song.releaseDate.split("-")(1).toInt)
    .view.mapValues(_.size.toDouble / l.size)
    .toList.sortBy(-_._2)
    .take(4)
```

---

6. getWords

```scala
def getWords(line: String): List[String] =
  line.replaceAll("[^a-zA-Z ]", "")
    .toLowerCase
    .split("\s+")
    .filter(_.nonEmpty)
    .toList
```

---

7. getAllWords

```scala
```

```scala
def getAllWords(l: List[Song]): List[String] =
  l.flatMap(song => getWords(song.title))
```

---

8. getThe4MostFrequentWordsInTitle

```scala
def getThe4MostFrequentWordsInTitle(l: List[Song]): List[(String, Int)] =
  getAllWords(l)
    .groupBy(identity)
    .view.mapValues(_.size)
    .toList.sortBy(-_._2)
    .take(4)
```

---

9. getThe20MostFrequentWordsInTitleWithFilter

```scala
def getThe20MostFrequentWordsInTitleWithFilter(l: List[Song], predicate: String => Boolean): List[(String, Int)] =
  getAllWords(l)
    .filter(predicate)
    .groupBy(identity)
    .view.mapValues(_.size)
    .toList.sortBy(-_._2)
    .take(20)
```

---

10. getAllWordsWithIndex

```scala
def getAllWordsWithIndex(l: List[Song]): Set[(Long, String)] =
  l.flatMap(song => getWords(song.title).map(word => (song.id, word))).toSet
```

---

11. createInverseIndex

```scala
def createInverseIndex(wwi: Set[(Long, String)]): Map[String, Set[Long]] =
  wwi.groupBy(_._2)
    .view.mapValues(_.map(_._1).toSet)
    .toMap
```

---

12. orConjunction

```scala
def orConjunction(words: List[String], invInd: Map[String, Set[Long]]): Set[Long] =
  words.flatMap(invInd.getOrElse(_, Set.empty)).toSet
```

---

13. andConjunction

```scala
def andConjunction(words: List[String], invInd: Map[String, Set[Long]]): Set[Long] =
  words.flatMap(invInd.get).reduceOption(_ intersect _).getOrElse(Set.empty)
```

---

14. findSongsWithAtLeast2WordsFromWordlist

```scala
def findSongsWithAtLeast2WordsFromWordlist(l: List[Song], words: List[String]): Set[(Long, String, Set[String])] = {
  val wordSet = words.toSet
  l.flatMap { song =>
```

```
    val titleWords = getWords(song.title).toSet

    val commonWords = titleWords intersect wordSet

    if (commonWords.size >= 2) Some((song.id, song.title, commonWords)) else None

  }.toSet

}
```