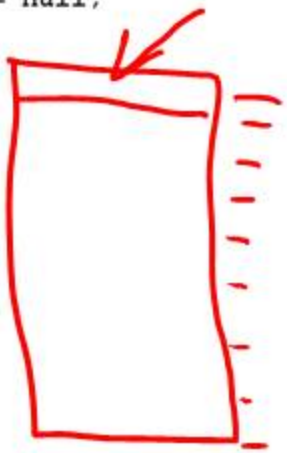
```
// return a connection to the database
public static Connection getDBConnection() {
  Connection conn = null;
  try {
    DriverManager.registerDriver(new com.mysql.jdbc.Driver());
    conn = (Connection) DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/flowershop", "root", "root");
  } catch (Exception e) {e.printStackTrace();}
  return conn;
}

// addDoc() takes a string and adds it to the index:
private static void addDoc(IndexWriter w, String v) throws IOException {
  Document doc = new Document();
  doc.add(new Field("title", v, Field.Store.YES, Field.Index.ANALYZED));
  w.addDocument(doc);
}
```

```
// executes a query and checks if it is empty or not
private static ResultSet getResultSetForSQL(String tSql) {
  Connection icon = null;  Statement istmt = null;   ResultSet myrs = null;
  try {
    icon = getDBConnection();   istmt = icon.createStatement();
    if (istmt.execute(tSql)) {
      if (istmt.getResultSet().isBeforeFirst() == false
              && istmt.getResultSet().isAfterLast() == false) {
        System.out.println("SQLBusiness.getResultSetForSQL--> "
                + "Empty Resultset at SQL:--> " + tSql);
        return null;
      } else {
        myrs = istmt.getResultSet();
      }
    }
  } catch (Exception e) {
    System.out.println("getResultSetForSQL--> Empty Resultset at SQL:--> " + tSql);
  } finally {
    return myrs;
  }
}
// fill the index (directory) with data (doc) resulted from ResultSet (rst)
private static void UploadSearchData(IndexWriter indexWriter, ResultSet rst) {
  try {
    while (rst.next()) {
      addDoc(indexWriter, rst.getString("rec"));
    }
    rst.close();  indexWriter.close();
  } catch (Exception e) {e.printStackTrace();}
}
```



documents	
docNo	title
1	1|Mixed Roses|Basket Of 100 Mixed Colored Roses
2	10|Pink Roses|Bunch Of 12 Pink Roses
3	100|White Lilies|Bunch Of White Colored Lilies
4	101|Red Roses|12 Red Roses With Some Ferns In A Vase
5	102|Colored Roses|Bunch Of 24 Mixed Colored Roses

SQL data
title
1|Mixed Roses|Basket Of 100 Mixed Colored Roses
10|Pink Roses|Bunch Of 12 Pink Roses
100|White Lilies|Bunch Of White Colored Lilies
101|Red Roses|12 Red Roses With Some Ferns In A Vase
102|Colored Roses|Bunch Of 24 Mixed Colored Roses

index	
Word	docNo
mix	1,
rose	1, 2, 4, 5
basket	1
color	1, 3, 5
pink	2
bunch	2, 3, 5
white	3
lilies	3
red	4
ferns	4
vase	4
1	1
10	2
12	2,3
100	1, 3
101	4
102	5
24	5

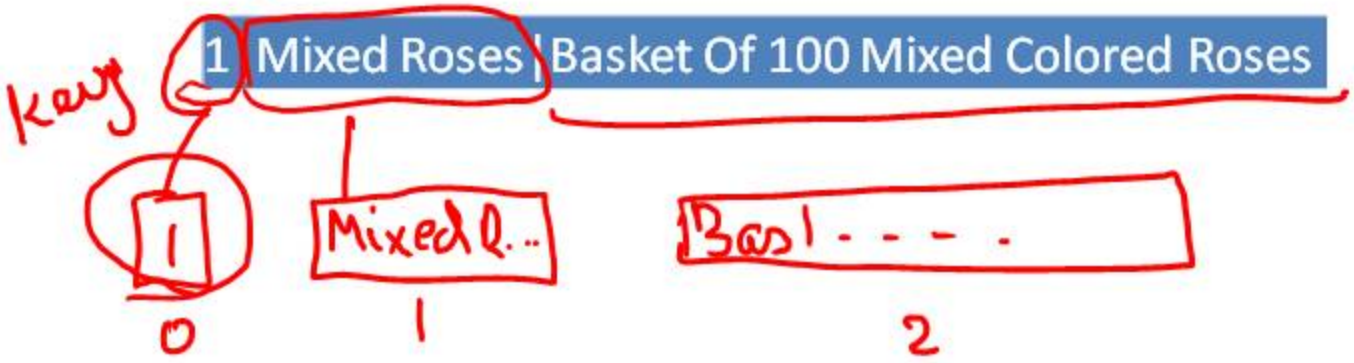
```
// return a String array of documents data
private static String[] getSearchResults(String Sql, String SearchStr,  int MaxNumOfHits) {
  String[] results = null;
  Directory directory = new RAMDirectory();
  StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_35);
  try {
    IndexWriterConfig conf = new IndexWriterConfig(Version.LUCENE_35, analyzer);
    IndexWriter indexWriter = new IndexWriter(directory, conf);
    // create documents and add them to the directory using indexWriter
    UploadSearchData(indexWriter, getResultSetForSQL(Sql));
    String querystr = SearchStr;
    Query query = new QueryParser(Version.LUCENE_35, "title", analyzer).parse(querystr);
    int hitsPerPage = MaxNumOfHits;
    IndexReader reader = IndexReader.open(directory);
    IndexSearcher searcher;
    searcher = new IndexSearcher(reader);
    TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
    searcher.search(query, collector);
    ScoreDoc[] hits = collector.topDocs().scoreDocs;
    results = new String[hits.length];
    for (int i = 0; i < hits.length; ++i) {
      int docId = hits[i].doc;
      Document d = searcher.doc(docId);
      results[i] = d.get("title");
    }
    searcher.close();
  } catch (Exception e) {e.printStackTrace();}
  return results;
}
```

```
// here we build the results that WE want from the search result (myresults)
public static String[] getSearchResultsKeys(String sqlDataSource, String Key, int MaxNumOfHits) {
  String[] ids = new String[0];
  if (!Key.equals("")) {
    // get the search results
    String[] myresults = getSearchResults(sqlDataSource, Key, MaxNumOfHits);
    // since we only want the ids from the search results
    // we create the array accordingly
    if (myresults != null) {
      ids = new String[myresults.length];
      // extract only the ids from the search result
      for (int i = 0; i < ids.length; i++)
        ids[i] = myresults[i].split("\\|", -1)[0];
    }
  } else ids = new String[0];
  return ids;
}
```

Specified

2|Roses|_

0 i [2]



```
public static void main(String[] args) throws IOException, ParseException {
  String[] searchResults = getSearchResults("select concat_ws('|', flid,ifnull(title,''), "
          + "ifnull(description,'')) as rec from flowers", "Red", 10);
  for (int i = 0; i < searchResults.length; i++) {
    String string = searchResults[i]; System.out.println(string);
  }
  String[] searchResultsIds = getSearchResultsKeys("select concat_ws('|', flid,ifnull(title,''), "
          + "ifnull(description,'')) as rec from flowers", "Red", 10);
  for (int i = 0; i < searchResultsIds.length; i++) {
    String string = searchResultsIds[i]; System.out.println(string);
  }
}
```

```
11
16
3
17
5
18
19
26
30
```

```
11|Red Roses|Bunch Of 12 Red Roses
163|Red Roses|Basket Of 24 Red Roses
175|Red Roses|Bunch Of 12 Red Roses
18|Red Roses|36 Red Roses In A Vase
19|Red Roses|Basket Of 50 Red Roses
26|Red Roses|Bunch Of 12 Red Roses
30|Red Roses|Bunch Of 100 Red Roses
45|Red Roses|Bunch Of 100 Red Roses
6|Red Roses|Bunch Of 24 Red Roses
```