# Stored Procedures Requiring Refactoring Review

This document contains detailed technical analysis of stored procedures with complexity scores greater than 3. These procedures have been flagged for potential refactoring to improve maintainability, performance, and code quality.

## uspSearchCandidateResumes (Complexity: 4)

### Business Function

1. The stored procedure 'uspSearchCandidateResumes' serves the business function of searching through candidate resumes based on a provided search string, with options to use inflectional forms of words and/or a thesaurus for more comprehensive results.  
2. It works with data from the 'JobCandidate' table in the 'HumanResources' database, specifically the 'JobCandidateID' and 'RANK' fields, and produces a list of candidate IDs and their corresponding ranks based on the search criteria.  
3. The procedure implements business rules to handle different search scenarios: if both inflectional and thesaurus options are selected, it uses a 'FREETEXTTABLE' search; if only the thesaurus option is selected, it uses a 'CONTAINSTABLE' search with thesaurus forms; if only the inflectional option is selected, it uses a 'CONTAINSTABLE' search with inflectional forms; and if neither option is selected, it performs a basic 'CONTAINSTABLE' search.

### Technical Analysis & Refactoring Recommendations

1. \*\*Code Structure Analysis\*\*: The code is structured in a clear and readable manner. It uses conditional logic to determine the type of search to be performed based on the parameters passed. However, the repeated use of similar SELECT statements for different conditions can be improved to reduce redundancy.  
  
2. \*\*Performance Concerns\*\*: The performance of this stored procedure largely depends on the size of the [HumanResources].[JobCandidate] table and the efficiency of the full-text search. If the table is large and not properly indexed, the full-text search could be slow. Also, the use of the wildcard "\*" in the CONTAINSTABLE and FREETEXTTABLE functions could potentially slow down the search as it searches all columns in the table.  
  
3. \*\*Maintainability Issues\*\*: The repeated use of similar SELECT statements for different conditions makes the code longer than necessary and could make it harder to maintain. If a change is needed in the SELECT statement, it would have to be made in multiple places.  
  
4. \*\*Best Practices Violations\*\*: The use of the wildcard "\*" in the CONTAINSTABLE and FREETEXTTABLE functions is generally not recommended as it can lead to slower performance. It would be better to specify the columns to be searched.  
  
5. \*\*Refactoring Recommendations\*\*:   
 - Reduce code duplication by creating a separate procedure for the SELECT statement and passing the necessary parameters to it.  
 - Specify the columns to be searched in the CONTAINSTABLE and FREETEXTTABLE functions to improve performance.  
 - Consider adding error handling to catch and handle any potential errors during the execution of the stored procedure.  
 - Review the indexing on the [HumanResources].[JobCandidate] table to ensure it is optimized for full-text search.  
  
6. \*\*Risk Assessment\*\*: The risk level of refactoring this procedure is Medium. While the changes recommended are not overly complex, care must be taken to ensure that the functionality of the stored procedure is not affected. Also, any changes to the indexing of the [HumanResources].[JobCandidate] table could potentially impact other queries or procedures that use this table.

### Complexity Factors

Lines of Code: 56

Contributing Factors: Contains JOIN operations (1.5x multiplier); Contains conditional logic; Calls other procedures/functions; Creates database objects; Medium-sized procedure (56 lines)

──────────────────────────────────────────────────

## Summary

Total procedures analyzed: 10  
Procedures flagged for refactoring review: 1  
Percentage requiring attention: 10.0%