# DuckDB-based CSV Search Pipeline

High-Performance Data Processing Documentation

## 1 Overview

**Project Summary**

This project is a scalable and efficient data processing pipeline built in Python, designed to:

- Convert large CSV files to optimized Parquet format
- Load and index the data into DuckDB
- Perform high-speed searches using indexed fields
- Export filtered results in multiple formats

## 2 Dependencies

**Required Dependencies:**

- Python 3.8+
- `duckdb`
- `pandas`
- `pyarrow`
- `tabulate`

**Installation:**

```
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Listing 1: Environment Setup

# 3 Pipeline Steps

## 3.1 1. CSV to Parquet Conversion

**Script:** `convert_to_parquet.py`

**Features:**

- Converts large CSV files into Parquet format using chunked streaming.
- Automatically creates directories and logs the process.

**Usage:**

```
python convert_to_parquet.py --csv data/sample.csv \
                             --parquet data/sample.parquet \
                             --chunk 1000000 \
                             --load yes
```

Listing 2: CSV to Parquet Conversion

## 3.2 2. Data Loading and Indexing

**Script:** `load_and_index.py`

**Functionality:**

- Loads Parquet file into DuckDB.
- Creates a persistent table and optionally recreates it.

**Note:** No CLI arguments are required for this script.

## 3.3 3. Search and Export

**Script:** `search_and_export.py`

**Capabilities:**

- Searches indexed or filterable fields using range or equality conditions.
- Supports multi-table joins with JOIN operations.
- Exports results in CSV, JSON, or Parquet.

# 4 Command Line Arguments

**Search and Export Arguments**

**Required Arguments:**

- `--tables table1 [table2]` - One or two table names to query
- `--field field_name` - Field to search on (must be indexed: col_0, col_1, or col_2)
- `--value search_value` - Value to search for
- `--columns col1 col2 ...` - Columns to extract in results

**Optional Arguments:**

- `--join-on column_name` - Common column to join on (required for two tables)
- `--format [csv|json|parquet]` - Export format (default: csv)
- `--output path` - Output file path without extension (default: data/search_output)

# 5 Usage Examples

**Single Table Search:**

```
python search_and_export.py --tables data_table \
                            --field col_0 \
                            --value 42 \
                            --columns col_0 col_1 col_2 \
                            --format csv
```

Listing 3: Basic Search with Single Value

**Multi-Table Join Search:**

```
python search_and_export.py --tables main_table lookup_table \
                            --join-on col_0 \
                            --field col_1 \
                            --value "search_value" \
                            --columns t1.col_0 t1.col_1 t2.description \
                            --format json
```

Listing 4: Join Two Tables

## 5.1 4. Logging

**Logging System**

Each script writes a timestamped log file under the `logs/` directory, and also streams logs to the terminal.

# 6 Error Handling

The pipeline includes validations and exception handling for:

- Missing or invalid files
- Invalid search fields or columns
- DuckDB connection issues
- Empty query results

All errors are logged in the `logs/` directory for audit and debugging purposes.

# 7 Logging Format

**Log Structure**

Each run produces a detailed log file that contains:

- Timestamp of execution
- CLI arguments passed
- Number of rows processed
- Any warnings or errors encountered

**Sample Log Output:**

```
2025-06-23 14:30:12 [INFO] Running query: SELECT col_1 FROM data WHERE col_0 =
    42
2025-06-23 14:30:12 [INFO] Found 3,920 rows.
2025-06-23 14:30:12 [INFO] Exported to data/search_output.csv
```

Listing 5: Example Log Output

# 8 Folder Structure

```
.
          data/
                sample.csv
                sample.parquet
                engine.duckdb
          logs/
                *.log
          convert_to_parquet.py
          load_and_index.py
          search_and_export.py
          requirements.txt
```

Listing 6: Project Directory Structure

# 9 Appendix: Sample Schema

**Dataset Schema**

The test dataset contains 50 columns named `col_0` to `col_49`, with a mix of types:

- `col_0, col_1, col_2` are indexed (INTEGER)
- `col_3` to `col_20` are FLOAT
- `col_21` to `col_45` are VARCHAR
- `col_46` to `col_49` may include TEXT/BLOB data

## Author Information

**Harsh Prakash**

Email: `harshprakash06@gmail.com`