CSV Export Feature for Final Report Generation

Overview

The `src/workflows/generate_final_report.py` script now includes functionality to export all generated tables as individual CSV files. This feature makes it easy to collect and analyze the final results in a structured format.

New Functionality

CSV Export Method

- **Method**: `export_tables_to_csv(output_dir: Path)`
- **Location**: `ReportGenerator` class
- **Purpose**: Exports all generated tables as individual CSV files

Output Directory

- **Default Location**: `data/report/` (configurable via `CSV_OUTPUT_DIR`)
- **Auto-creation**: Directory is created automatically if it doesn't exist

Generated CSV Files

The following CSV files are generated when the script runs:

Basic Statistics Tables

1. `match_type_counts.csv` - Count of matches by type (name, semantic, etc.)

2. `avg_score_by_type.csv` - Average similarity scores by match type
3. `matches_by_framework.csv` - Number of matches by source framework
4. `matches_by_project.csv` - Number of matches by target project

Pattern Analysis Tables (if patterns exist)

1. `source_pattern_analysis.csv` - Where patterns originate from
2. `adoption_pattern_analysis.csv` - Where patterns are adopted/used
3. `patterns_by_match_count.csv` - Patterns sorted by match frequency
4. `avg_score_by_pattern.csv` - Average similarity scores by pattern
5. `patterns_in_[framework].csv` - Patterns found in each framework (e.g., `patterns_in_qiskit.csv`)

Additional Tables

1. `top_matched_concepts.csv` - Top 20 most frequently matched concepts
2. `unmatched_patterns.csv` - List of patterns not found in any project (if any)

Usage

The CSV export is automatically triggered when running the main script:

```
# The export happens automatically in the main() function
python src/workflows/generate_final_report.py
```

Or programmatically:

```
from src.workflows.generate_final_report import ReportGenerator
import pandas as pd

# Load your data
df = pd.read_csv("your_data.csv")
patterns = {"pattern1", "pattern2"}

# Create generator and export
```

```
generator = ReportGenerator(df, patterns)
generator.export_tables_to_csv(Path("output/directory"))
```

File Structure

```
data/
└── report/
    ├── match_type_counts.csv
    ├── avg_score_by_type.csv
    ├── matches_by_framework.csv
    ├── matches_by_project.csv
    ├── source_pattern_analysis.csv
    ├── adoption_pattern_analysis.csv
    ├── patterns_by_match_count.csv
    ├── avg_score_by_pattern.csv
    ├── patterns_in_qiskit.csv
    ├── patterns_in_pennylane.csv
    ├── patterns_in_classiq.csv
    ├── top_matched_concepts.csv
    └── unmatched_patterns.csv
```

Benefits

1. **Easy Data Collection**: All results in one organized location
2. **Structured Format**: CSV files can be easily imported into Excel, R, Python, etc.
3. **Individual Analysis**: Each table can be analyzed separately
4. **Automated**: No manual intervention required
5. **Consistent Naming**: Clear, descriptive filenames for easy identification

Integration

The CSV export is seamlessly integrated into the existing workflow: - Runs automatically after generating TXT and Markdown reports - Uses the same data structures already computed for the reports - Maintains consistency with the existing report generation process