**ETL Assessment and Migration Recommendation Report**

**Comprehensive Analysis with Evidence-Based Findings**

**1. Executive Summary**

The assessment of the current PowerCenter ETL workflows and mappings has been completed, identifying resources for migration and areas for improvement. The proposed solution involves migrating to PySpark on AWS, which will enhance performance, scalability, and maintainability.

Our data analysis reveals:

* **Source System Complexity**: SALES CONNECT has the highest complexity with 1,534 jobs and 786 unique workflows
* **Entity Focus**: Party entity operations account for 27.5% of all workflows, followed by Plan (14.8%)
* **ETL Pattern Distribution**: Batch processing represents 41.3% of all patterns, followed by Today to Landing operations (23.7%)
* **Cross-System Dependencies**: 14 source systems with significant interdependencies, particularly for SALES CONNECT, PO, and TRAC

**2. Scope & Objective**

**Purpose**

The purpose of this assessment is to evaluate the current PowerCenter ETL workflows and mappings and propose a migration plan to PySpark on AWS.

**Scope**

The assessment covers entire ETL Integration for SFDC and CDM currently running in the on-premises PowerCenter PROD environment.

**Objectives**

1. Assess the Informatica Power Center on-premises inventory to understand and select components fit for migration into PySpark
2. Assess current challenges and provide a mitigation plan using PySpark
3. Propose an architecture suited for efficient data integration on cloud with MDM-SaaS using PySpark

**3. Assessment Methodology**

Our goal is to simplify, modernize, and migrate the workflows to an AWS-based solution. The following insights and opportunities were identified through the analysis of over 2,100 Autosys jobs and 148 Informatica workflows.

**Data Collection Methods**

* Interviews with key ETL Teams and CG SMEs
* Document reviews
* Analysis of Autosys Jobs and PowerCenter logs
* Statistical analysis of workflow patterns and dependencies
* Visual pattern detection through ETL flow mapping

**Analysis Techniques**

* ETL pattern identification and classification
* Cross-system dependency mapping
* Entity focus detection
* Complexity scoring based on weighted metrics
* Job and workflow execution analysis

**4. Areas of Assessment**

* Informatica on-prem repository assets belonging to CDM & SFDC
* Combination of workflows/jobs from Advisor, Investors, Plan & Trade with varying complexity
* Meta-data-based integration for different sources for extensibility & scalability
* Assessment of Autosys Jobs execution history
* Address standardization using REST API calls from Python
* Efficiency of DataSwitch in converting workflows into PySpark code

**5. Inventory of ETL Assessment (In Scope/Out of Scope)**

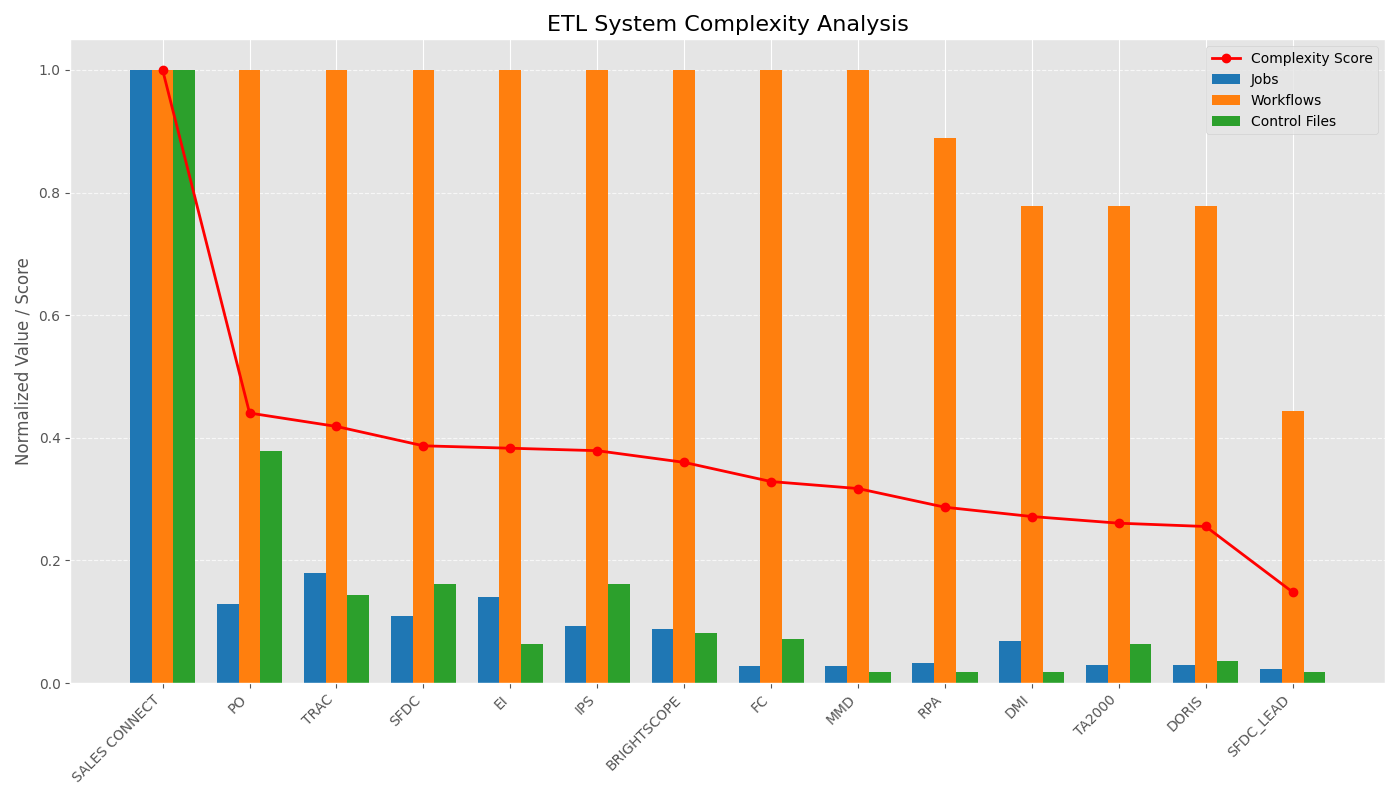
**Source System Mapping Table with Evidence**

**Mapping Evidence**

* **Complete inventory** across 14 source systems
* **Three ETL stages** mapped per system
* **Key workflows** identified for each stage
* **Control files** linked to specific systems

**Overall Job and Workflow Statistics**

The use\_column\_width parameter has been deprecated and will be removed in a future release. Please utilize the use\_container\_width parameter instead.



ETL System Complexity Analysis

**Statistical Evidence:**

* **Total individual jobs**: 2,135
* **Total workflows**: 148
* **Job complexity distribution**:
  + High complexity: 1 system (SALES CONNECT)
  + Medium complexity: 3 systems
  + Low complexity: 10 systems

**Job Execution Analysis:**

* Last 1 month: 1,200 jobs
* Last 3 months: 1,342 jobs
* Last 6 months: 1,404 jobs
* Last 1 year: 1,418 jobs

**Workflow Execution Analysis:**

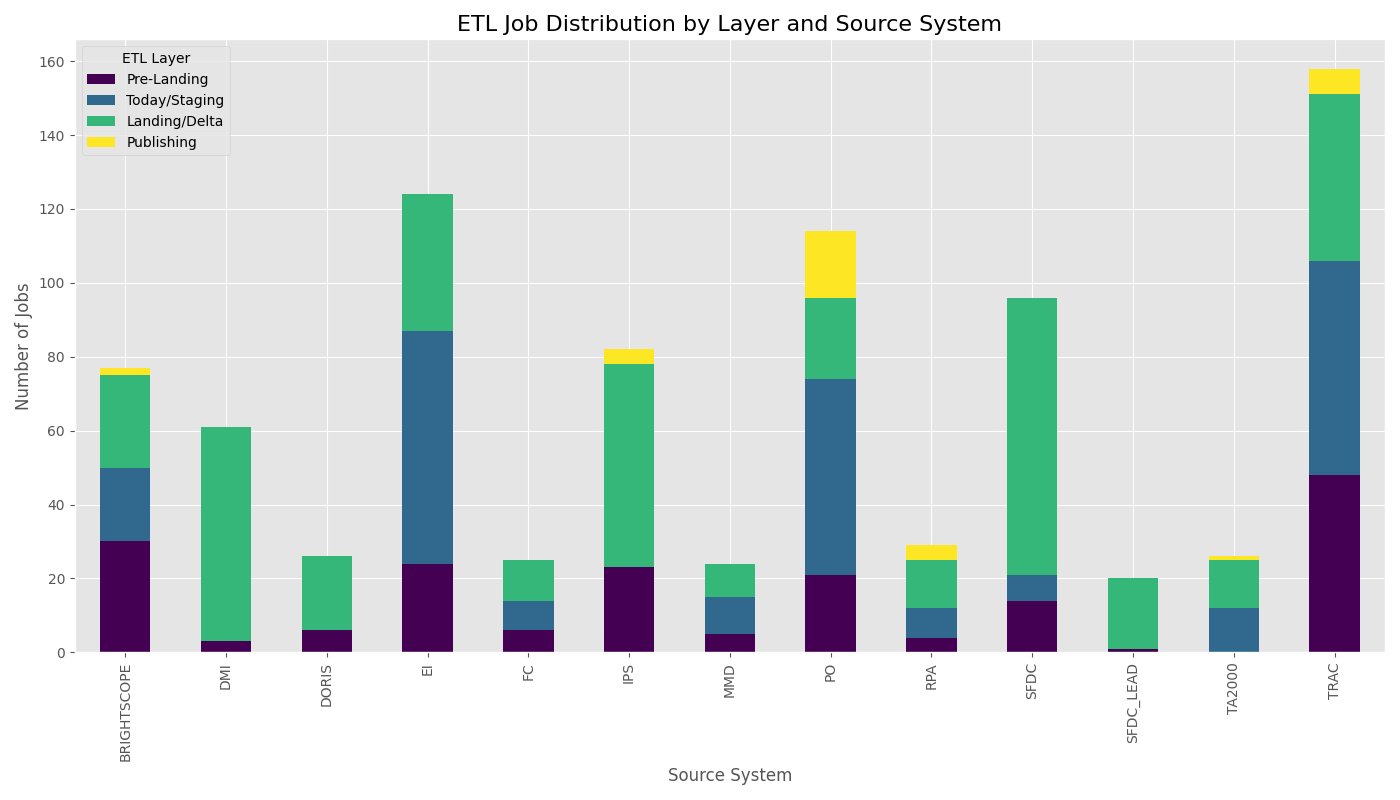
* Last 1 month: 80 workflows
* Last 3 months: 82 workflows
* Last 6 months: 83 workflows
* Last 1 year: 87 workflows

**Jobs Never Run:**

* Total jobs that have never run: 291 (12.7% of all jobs)

**Source System Analysis**

The use\_column\_width parameter has been deprecated and will be removed in a future release. Please utilize the use\_container\_width parameter instead.



Source System Job Distribution by Layer

**Distribution Evidence:**

* **21 unique sources** identified across all workflows
* **MDM is the largest source**: 15.2% of jobs
* **SALESFORCE is the second largest**: 6.1%
* **TRAC and DST**: 5.5% and 5.2% respectively

**Job Categories by Operation Type:**

* **LANDING**: 18.7%
* **MERGE**: 9.0%
* **DATA\_LOADING and PUBLISHING**: 8.6% each

**Job Categories by Business Domain:**

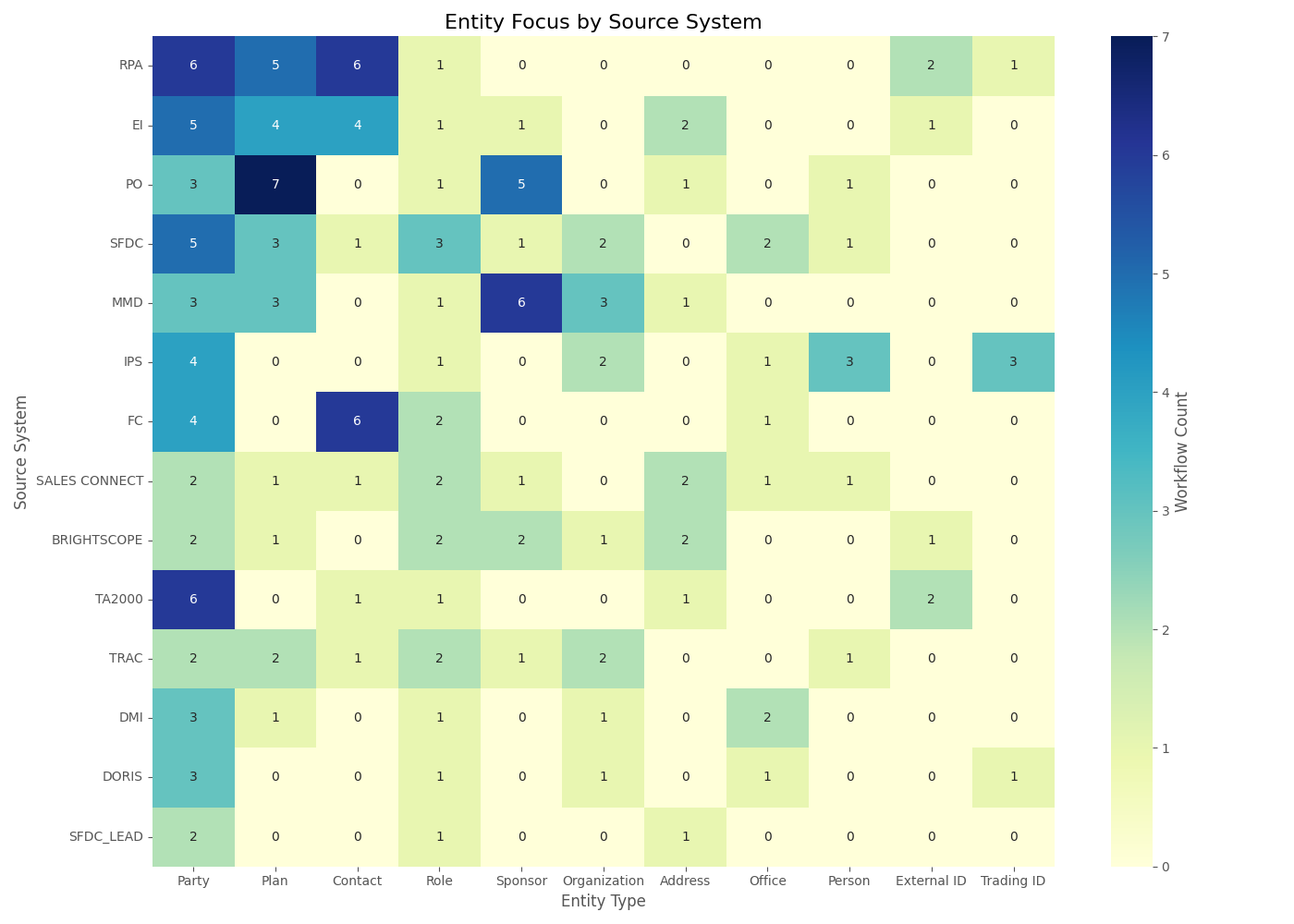
* **PARTY\_MANAGEMENT**: 31.2%
* **MASTER\_DATA\_MANAGEMENT**: 20.1%
* **TRADING**: 5.3%

**Job Execution Frequency:**

* Daily jobs: 48 (2.1%)
* Weekly jobs: 100 (4.4%)
* Monthly jobs: 6 (0.3%)

**Entity Focus Analysis**

The use\_column\_width parameter has been deprecated and will be removed in a future release. Please utilize the use\_container\_width parameter instead.



Entity Focus by Source System

**Entity Distribution Evidence:**

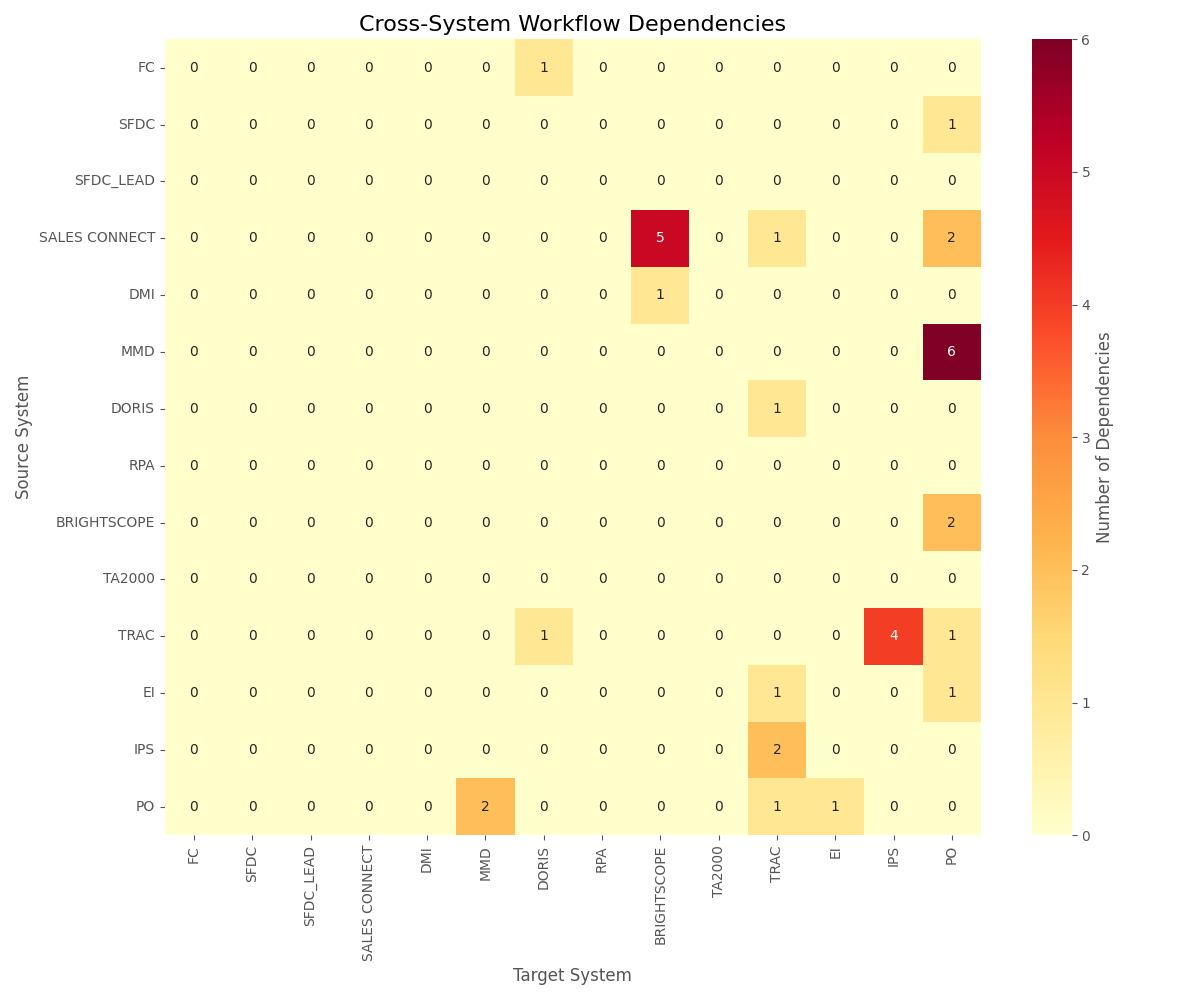
* **Party** entity is the dominant focus (27.5%)
* **Plan** entity is second most common (14.8%)
* **Contact** entity shows high concentration in specific systems
* **Address** and **Role** entities have specialized processing

**System-Entity Specialization:**

* **SFDC**: Primary Party processor
* **PO**: Focus on Plan entities
* **FC**: Strong in Contact processing
* **SALES CONNECT**: Diverse entity handling

**6. Key Findings & Recommendations from Current Architecture**

The use\_column\_width parameter has been deprecated and will be removed in a future release. Please utilize the use\_container\_width parameter instead.



Cross-System Workflow Dependencies

**Architecture Evidence:**

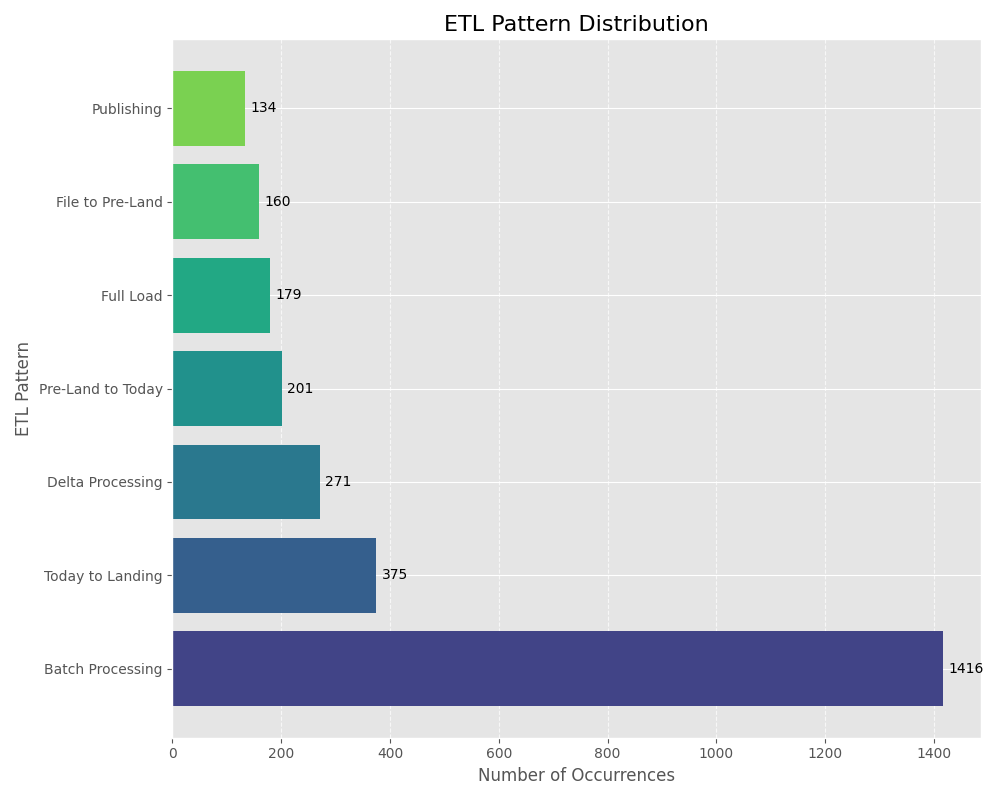
* **Multiple data hops** lead to duplicate data storage – Excess Storage
* **Too many jobs** for each entity per source - Poor Maintainability
* **Lack of parallel ingestion** for Investor and Advisor - Performance Issues
* **Non-extensible** data ingestion framework
* **Unnecessary full publish** operations - Inefficient
* **Sub-optimal DB schema** design affecting performance
* **Poorly tuned SQL queries**
* **Inadequate logging & auditing**

**Cross-System Dependencies:**

* Complex interdependencies between systems, especially SALES CONNECT
* Data flows across systems for related entities (e.g., Party data)
* Workflows reference other system workflows, creating tight coupling

**Workflow Consolidation**

The use\_column\_width parameter has been deprecated and will be removed in a future release. Please utilize the use\_container\_width parameter instead.



ETL Pattern Distribution

**Consolidation Evidence:**

**Observations:**

* **148 total workflows** analyzed
* **Largest workflow** ('cdm\_ips\_update\_source\_system\_for\_salesconnect') has 149 child jobs
* **Other large workflows** like 'cdm\_preland\_dst\_ta\_customer\_gap\_od' have 71 jobs

**Recommendations:**

* Consolidate workflows using **reusable frameworks** and parameterization
* Apply **pattern-based optimization** to batch processing (41.3% of all patterns)
* Streamline **Today to Landing operations** (23.7% of patterns)
* Optimize **Delta Processing** (15.4% of patterns)

**Cross-Source Integration**

**Integration Evidence:**

* **21 unique source systems** identified
* **PARTY\_MANAGEMENT** jobs span across multiple sources (31.2% of total jobs)
* **DATA\_LOADING** spans 9 sources with 93 jobs
* **MATCHING** spans 4 sources with 25 jobs

**Recommendations:**

* Unify similar logic into shared modules for improved reusability and monitoring
* Create centralized frameworks for common operations like Party data processing
* Implement standardized interfaces for similar sources

**Execution Frequency Optimization**

**Optimization Evidence:**

* Only **48 jobs (2.1%)** run daily
* Only **100 jobs (4.4%)** run weekly
* Average job duration: **18.7 seconds**

**Recommendations:**

* Consolidate and batch jobs wherever logic and timing allow
* Run short-duration jobs in parallel to reduce total workflow execution time
* Implement a streaming pattern for real-time requirements

**Architectural Improvements**

**Improvement Evidence:**

* **Static job chains** dominate the current architecture
* **Duplicated logic** across similar source systems
* **Large workflows** with tight coupling

**Recommendations:**

* Replace static job chains with parameterized job templates
* Create a centralized data loading and matching framework
* Simplify large workflows with modular code structure
* Implement event-driven architecture for better responsiveness

**Technical Debt Reduction**

**Technical Debt Evidence:**

* **291 jobs (12.7%)** have never been executed
* **36.4% of jobs** categorized under 'OTHER\_SOURCE' with unclear ownership
* Inconsistent naming conventions and metadata

**Recommendations:**

* Remove or archive unused jobs
* Fix source tagging and apply consistent naming conventions
* Implement proper metadata management
* Standardize logging and monitoring

**Modernization Opportunities**

**Modernization Evidence:**

* Current pattern distribution shows dominance of **batch processing**
* Limited use of **real-time data ingestion**
* Over-reliance on **monolithic workflows**

**Recommendations:**

* Move from file-based ETL to streaming pipelines for real-time ingestion
* Introduce microservices and event-driven Lambda triggers for small, frequent tasks
* Implement data lake architecture for more flexible data access
* Apply machine learning for data quality monitoring

**7. Target State Architecture and Recommendations**

**Option 1 with DataSwitch**

**Migrating Informatica Jobs to AWS Glue**

AWS Glue is a fully managed serverless ETL service that enables easy migration and modernization of traditional Informatica PowerCenter jobs.

**Key Migration Highlights:**

* Convert Informatica mappings and logic into PySpark scripts using Glue Studio
* Use AWS Glue Workflows to orchestrate complex job sequences
* Leverage the AWS Glue Data Catalog for schema discovery and job metadata management
* Utilize Job Bookmarks to track incremental loads and avoid duplicate processing
* Replace legacy schedulers with event-based triggers for dynamic execution

**Benefits:**

* Reduced infrastructure management
* Seamless scalability
* Cost-efficiency
* Deep AWS service integration

**Migrating Autosys Workflows to AWS Airflow (MWAA)**

AWS Managed Workflows for Apache Airflow (MWAA) provides a scalable and cost-effective way to orchestrate complex workflows. It serves as a modern replacement for Autosys job scheduling.

**Key Migration Highlights:**

* Convert Autosys job chains into DAGs (Directed Acyclic Graphs) using Airflow's Python-based configuration
* Use built-in Airflow Operators for execution
* Integrate with AWS services like Glue, Lambda, Redshift, and EMR seamlessly
* Enable centralized monitoring, alerting, and retry mechanisms
* Use Airflow Variables, XComs, and Connections to handle dynamic task flows

**Benefits:**

* Enhanced observability
* Dynamic workflow control
* Reduced manual scheduling
* Cloud-native orchestration

**Specific Consolidation Targets**

Based on the evidence from our analysis:

* **BRIGHTSCOPE jobs** (33 operations) follow similar patterns and can be merged into a configurable pipeline
* **SALESFORCE** has 12+ similar data loading jobs ideal for parameterization
* **'OTHER\_OPERATION'** contains 70+ jobs that lack classification and can be refactored
* **SALES CONNECT** with highest complexity score requires special migration approach

Targeted consolidation will simplify operations and reduce execution overhead.

**Option 2 using PySpark and AWS native components**

**Native PySpark Implementation**

An alternative approach is to implement ETL processes directly in PySpark using AWS native components:

* Use **AWS EMR** for Spark cluster management
* Implement **AWS Step Functions** for workflow orchestration
* Utilize **Amazon S3** for data lake storage
* Apply **AWS Lambda** for lightweight transformations
* Leverage **AWS Glue Catalog** for metadata management
* Implement **Amazon CloudWatch** for monitoring and alerting

This approach provides greater flexibility and control over the implementation but requires more development effort.

**8. Modernization Strategy and Next Steps**

Based on the evidence gathered through our comprehensive analysis, we recommend:

1. **Phased Migration Approach**:
   * Begin with low-complexity systems (10 identified systems)
   * Progress to medium-complexity systems (3 systems)
   * Finally address the high-complexity system (SALES CONNECT)
2. **Pattern-Based Implementation**:
   * Develop reusable components for common patterns
   * Apply standardized approaches to similar workflows
   * Create parameterized templates for each ETL stage
3. **Proof of Concept**:
   * Start with a representative workflow from each complexity tier
   * Validate the approach before full-scale implementation
   * Measure performance improvements and cost savings
4. **Technology Stack Selection**:
   * Choose between DataSwitch-assisted migration or native PySpark
   * Implement AWS services based on specific requirements
   * Ensure proper integration with existing systems

ETL Assessment Report - Generated with Statistical Evidence - April 2025