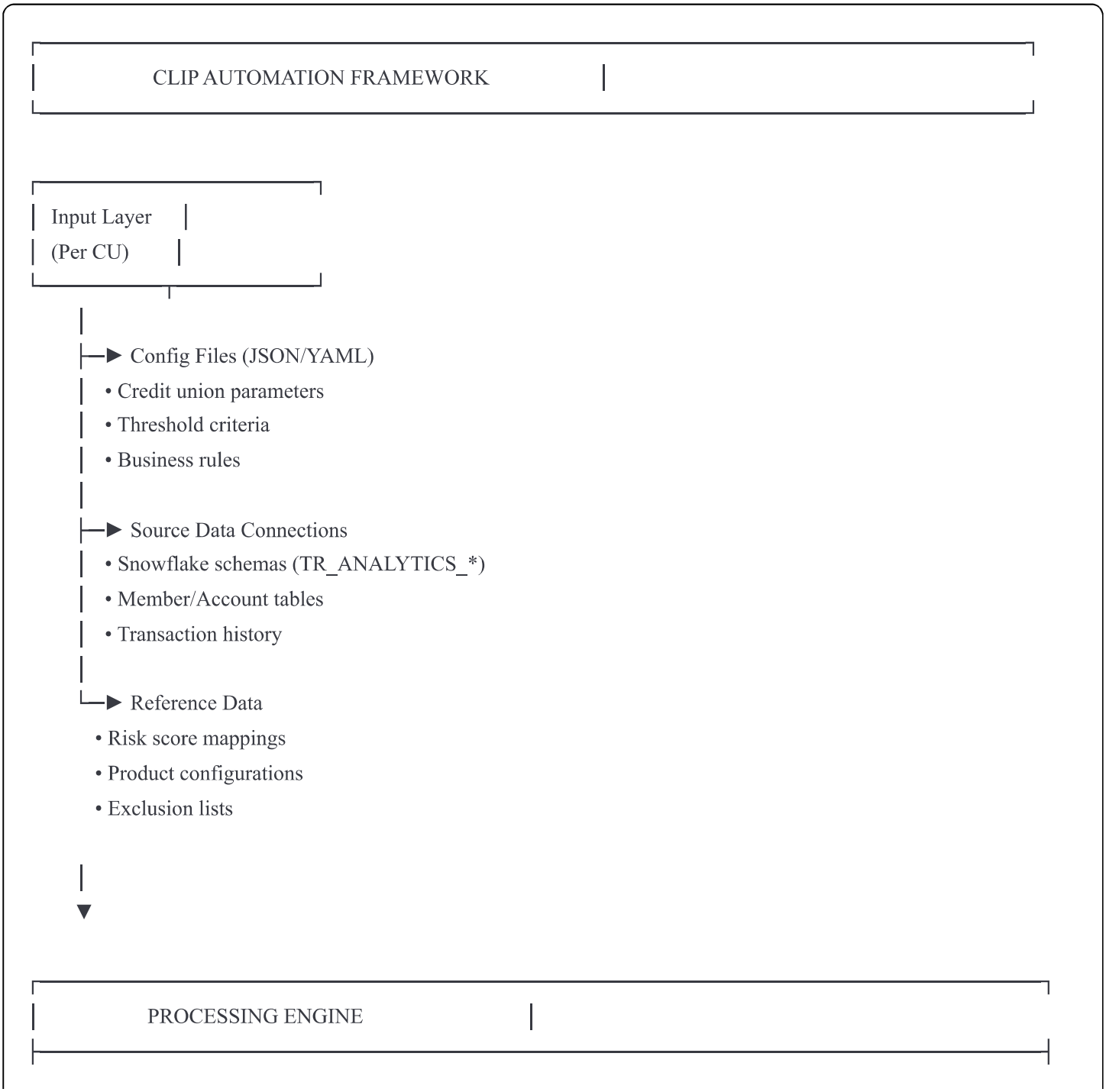
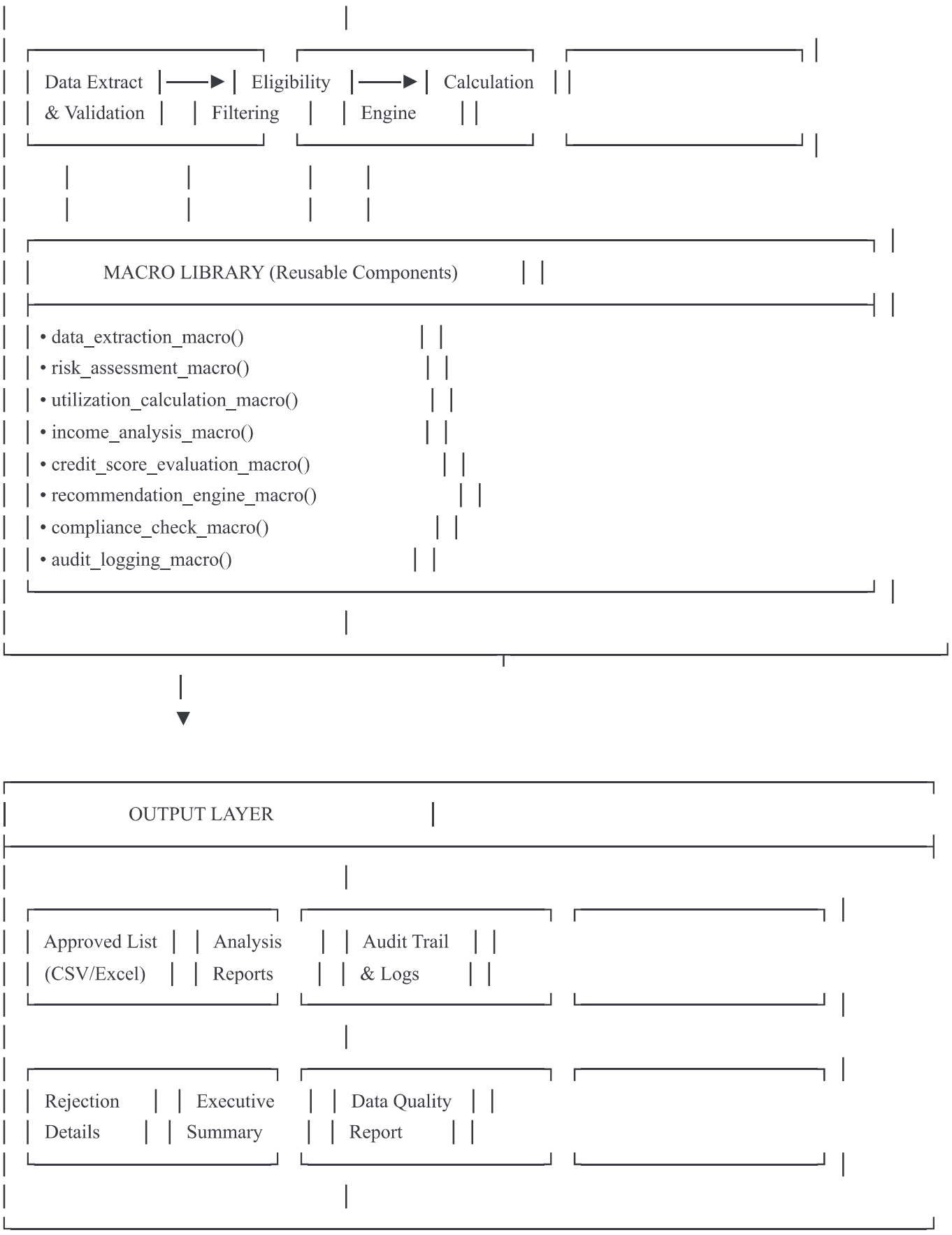
CLIP Credit Line Increase Process - Automation Design

Executive Summary

This design proposes a modular, macro-driven automation framework to replace Alteryx workflows for credit line increase analysis across multiple credit unions. The solution emphasizes reusability, maintainability, and scalability.

High-Level Architecture Flow




Data Extract & Validation
Eligibility Filtering
Calculation Engine
MACRO LIBRARY (Reusable Components)
• data_extraction_macro()
• risk_assessment_macro()
• utilization_calculation_macro()
• income_analysis_macro()
• credit_score_evaluation_macro()
• recommendation_engine_macro()
• compliance_check_macro()
• audit_logging_macro()
OUTPUT LAYER
Approved List (CSV/Excel)
Analysis Reports
Audit Trail & Logs
Rejection Details
Executive Summary
Data Quality Report

```
┌──────────────────────────────────────────────────────────────┐
│                    MONITORING & ALERTING                     │
├──────────────────────────────────────────────────────────────┤
│ • Email notifications on completion                          │
│ • Error alerts and failure notifications                     │
│ • Performance metrics and runtime stats                      │
│ • Data quality threshold alerts                              │
└──────────────────────────────────────────────────────────────┘
```

Detailed Process Flow (Per Credit Union)

```
START: Credit Union Analysis Run
  │
  ├─► STEP 1: INITIALIZATION
  │   ├─ Load CU-specific config file
  │   ├─ Validate required parameters
  │   ├─ Establish database connections
  │   └─ Initialize logging
  │
  ├─► STEP 2: DATA EXTRACTION
  │   ├─ Extract member demographics
  │   ├─ Pull account balances & history
  │   ├─ Get transaction patterns (6-12 months)
  │   ├─ Retrieve credit scores
  │   ├─ Load payment history
  │   └─ Apply exclusion lists
  │
  │  [MACRO: data_extraction_macro(tenant_id, lookback_period)]
  │
  ├─► STEP 3: ELIGIBILITY SCREENING
  │   ├─ Account age check (>= minimum months)
  │   ├─ Payment history validation (no 30+ day lates)
  │   ├─ Minimum balance requirements
  │   ├─ Credit score threshold
  │   ├─ Debt-to-income ratio check
  │   └─ Exclude charged-off/collections
  │
  │  [MACRO: eligibility_filter_macro(config_rules)]
  │
  ├─► STEP 4: UTILIZATION ANALYSIS
  │   ├─ Calculate current utilization %
  │   ├─ Analyze utilization trends
```

```
    ├─ Identify high utilizers (>70%)
    ├─ Calculate available credit headroom
    └─ Segment by utilization bands
    
    [MACRO: utilization_calculation_macro()]
    
├─► STEP 5: RISK ASSESSMENT
    ├─ Credit score banding
    ├─ Income verification check
    ├─ Employment stability
    ├─ Recent inquiry analysis
    ├─ Bankruptcy/foreclosure screening
    └─ Calculate risk score
    
    [MACRO: risk_assessment_macro()]
    
├─► STEP 6: RECOMMENDATION ENGINE
    ├─ Determine increase amount
    │   • Percentage-based (e.g., 20-50%)
    │   • Fixed amount tiers
    │   • Income-based limits
    ├─ Apply business rule caps
    ├─ Calculate new utilization post-increase
    └─ Generate confidence score
    
    [MACRO: recommendation_engine_macro(increase_strategy)]
    
├─► STEP 7: COMPLIANCE & VALIDATION
    ├─ Regulatory limit checks
    ├─ Internal policy validation
    ├─ Cross-account exposure check
    ├─ Documentation requirements
    └─ Approval threshold routing
    
    [MACRO: compliance_check_macro()]
    
├─► STEP 8: OUTPUT GENERATION
    ├─ Create approved members list
    ├─ Generate detailed analysis report
    ├─ Produce rejection report with reasons
    ├─ Executive summary dashboard
    ├─ Data quality metrics
    └─ Audit trail documentation
```

```
│   [MACRO: output_generator_macro(output_format)]
│
├─► STEP 9: POST-PROCESSING
│   ├─ Send email notifications
│   ├─ Upload to shared drive/S3
│   ├─ Update tracking database
│   ├─ Archive input parameters
│   └─ Generate performance metrics
│
└─► END: Analysis Complete
```

Folder Structure

```
CLIP_AUTOMATION/
│
├── config/
│   ├── global_config.yaml            # System-wide settings
│   ├── credit_unions/
│   │   ├── CU_001_config.json        # Example: First Tech FCU
│   │   ├── CU_002_config.json        # Example: Navy FCU
│   │   ├── CU_003_config.json        # Example: State Employees
│   │   └── config_template.json      # Template for new CUs
│   │
│   └── business_rules/
│       ├── risk_scoring_rules.yaml
│       ├── increase_strategies.yaml
│       └── compliance_rules.yaml
│
├── inputs/
│   ├── reference_data/
│   │   ├── exclusion_lists/
│   │   │   ├── CU_001_exclusions.csv
│   │   │   └── CU_002_exclusions.csv
│   │   ├── product_configs/
│   │   └── risk_mappings/
│   │
│   └── manual_overrides/
│       └── override_template.csv
│
├── src/
│   ├── macros/
```

```
│   │   ├── __init__.py
│   │   ├── data_extraction.py
│   │   ├── eligibility_filter.py
│   │   ├── utilization_calc.py
│   │   ├── risk_assessment.py
│   │   ├── recommendation_engine.py
│   │   ├── compliance_check.py
│   │   ├── output_generator.py
│   │   └── audit_logging.py
│   │
│   ├── core/
│   │   ├── database_connector.py
│   │   ├── config_loader.py
│   │   ├── validation_engine.py
│   │   └── error_handler.py
│   │
│   ├── models/                    # dbt models (if using dbt)
│   │   ├── staging/
│   │   ├── intermediate/
│   │   └── marts/
│   │
│   └── orchestration/
│       ├── main_pipeline.py       # Master orchestrator
│       ├── cu_processor.py        # Per-CU execution
│       └── scheduler.py           # Automated scheduling
│
├── outputs/
│   ├── YYYY-MM/                   # Monthly runs
│   │   ├── CU_001/
│   │   │   ├── approved_list.csv
│   │   │   ├── analysis_report.xlsx
│   │   │   ├── rejection_details.csv
│   │   │   ├── executive_summary.pdf
│   │   │   └── audit_trail.log
│   │   │
│   │   ├── CU_002/
│   │   └── CU_003/
│   │
│   └── consolidated/
│       └── all_cus_summary_YYYY-MM.xlsx
│
├── logs/
│   ├── execution_logs/
│   │   └── YYYY-MM-DD_HH-MM-SS.log
```

```
│   ├── error_logs/
│   └── performance_metrics/
│
├── tests/
│   ├── unit_tests/
│   ├── integration_tests/
│   └── test_data/
│
├── docs/
│   ├── user_guide.md
│   ├── configuration_guide.md
│   ├── macro_documentation.md
│   └── troubleshooting.md
│
├── sql/
│   ├── base_queries/
│   │   ├── member_extraction.sql
│   │   ├── transaction_analysis.sql
│   │   └── utilization_calc.sql
│   │
│   └── validation_queries/
│
├── requirements.txt
├── setup.py
├── README.md
└── run_clip_analysis.py              # Main entry point
```

Credit Union Specific Configuration Example

```json
```

```
{
 "credit_union_id": "CU_001",
 "credit_union_name": "Example Federal Credit Union",
 "tenant_id": "TR_ANALYTICS_EXAMPLEFCU",

 "eligibility_criteria": {
  "minimum_account_age_months": 12,
  "minimum_credit_score": 650,
  "maximum_delinquency_30_days": 0,
  "maximum_delinquency_60_days": 0,
  "minimum_current_limit": 500,
  "maximum_current_utilization": 95,
  "exclude_charged_off": true,
  "exclude_bankruptcy": true,
  "bankruptcy_lookback_years": 2
 },

 "increase_strategy": {
  "method": "tiered_percentage",
  "tiers": [
   {
    "credit_score_min": 750,
    "increase_percentage": 50,
    "max_increase_amount": 5000
   },
   {
    "credit_score_min": 700,
    "increase_percentage": 35,
    "max_increase_amount": 3000
   },
   {
    "credit_score_min": 650,
    "increase_percentage": 25,
    "max_increase_amount": 2000
   }
  ],
  "absolute_maximum_limit": 25000
 },

 "risk_parameters": {
  "high_utilization_threshold": 75,
  "analysis_period_months": 6,
  "minimum_payment_rate": 1.0,
```

```
    "income_verification_required": true,
    "debt_to_income_max": 0.43
  },

  "output_preferences": {
    "include_rejection_reasons": true,
    "generate_executive_summary": true,
    "export_formats": ["csv", "xlsx", "pdf"],
    "email_recipients": [
      "analyst@examplefcu.com",
      "manager@examplefcu.com"
    ]
  },

  "data_sources": {
    "member_table": "MEMBER",
    "account_table": "ACCOUNT",
    "transaction_table": "TRANSACTIONS",
    "credit_score_table": "CREDIT_SCORES"
  }
}
```

Macro Library Overview

1. Data Extraction Macro

Purpose: Pull all required data from Snowflake for a specific credit union

Inputs:

- Tenant ID (TR_ANALYTICS_*)
- Lookback period
- Date range

Outputs:

- DataFrame with member/account data
- Transaction history
- Credit scores

Key Logic:

- Multi-schema query handling
- Data type validation
- Null handling
- Deduplication

---

2. Eligibility Filter Macro

Purpose: Apply configurable business rules to filter eligible accounts

Inputs:

- Raw member/account data
- Config rules dictionary

Outputs:

- Filtered DataFrame (eligible members)
- Rejection log with reasons

Key Logic:

- Account age calculation
- Payment history validation
- Balance/limit checks
- Status exclusions

---

3. Utilization Calculation Macro

Purpose: Calculate current and historical utilization patterns

Inputs:

- Account balance data

- Credit limit data
- Historical snapshots

Outputs:

- Utilization metrics per account
- Trend indicators

Key Logic:

- Current utilization %
- Average utilization (6-month)
- Volatility score
- High balance identification

---

4. Risk Assessment Macro

Purpose: Calculate composite risk scores for credit line increases

Inputs:

- Credit scores
- Payment history
- Income data
- Derogatory records

Outputs:

- Risk score (0-100)
- Risk band (Low/Medium/High)
- Contributing factors

Key Logic:

- Weighted scoring model
- Recent inquiry impact

- Bankruptcy/foreclosure flags
- Income stability

---

5. Recommendation Engine Macro

Purpose: Determine optimal credit line increase amounts

Inputs:

- Eligible accounts
- Risk scores
- Current limits
- Strategy config

Outputs:

- Recommended new limit
- Increase amount
- Confidence score
- Approval routing

Key Logic:

- Strategy selection (percentage/fixed/tiered)
- Cap application
- Post-increase utilization projection
- ROI estimation

---

6. Compliance Check Macro

Purpose: Validate recommendations against regulatory and policy rules

Inputs:

- Recommended increases

- Compliance rules
- Member aggregated exposure

Outputs:

- Compliance pass/fail flag
- Required documentation list
- Approval level routing

Key Logic:

- Regulatory limit checks (e.g., TILA)
- Internal policy validation
- Cross-product exposure
- Documentation requirements

---

7. Output Generator Macro

Purpose: Create standardized output files in multiple formats

Inputs:

- Approved list
- Analysis data
- Config settings

Outputs:

- CSV/Excel files
- PDF reports
- Summary dashboards

Key Logic:

- Template-based report generation
- Multi-format export

- Data quality metrics
- Visualization creation

---

8. Audit Logging Macro

Purpose: Track all processing steps for compliance and debugging

Inputs:

- Processing events
- Config used
- Results summary

Outputs:

- Detailed audit trail
- Performance metrics
- Error logs

Key Logic:

- Timestamp all events
- Parameter logging
- Record counts at each stage
- Execution time tracking

Execution Flow by Credit Union

```
FOR EACH Credit Union in Configuration:
  │
  ├─► Load CU-specific config
  ├─► Initialize logging
  ├─► Connect to tenant schema (TR_ANALYTICS_*)
  │
  ├─► Run Data Extraction Macro
```

Key Benefits of This Design

1. **Modularity**: Each macro is independent and testable
2. **Reusability**: Same macros work across all credit unions
3. **Configurability**: Business rules in config files, not code
4. **Scalability**: Easy to add new credit unions
5. **Maintainability**: Changes to logic in one place
6. **Auditability**: Comprehensive logging and tracking
7. **Transparency**: Clear data lineage
8. **Flexibility**: Multiple output formats and delivery methods

---

Technology Stack Recommendation

Primary Implementation: Python + dbt + Snowflake

Alternative: SQL-based (Stored Procedures) if Python not preferred

Orchestration:

- Airflow (for complex scheduling)
- SQL Server Agent Jobs (for simple scheduled runs)
- Manual execution script

Testing Framework: pytest for unit and integration tests

Documentation: Sphinx or MkDocs for auto-generated docs

Next Steps for Implementation

1. **Phase 1**: Build core macro library with sample CU
2. **Phase 2**: Develop configuration management system
3. **Phase 3**: Create output generation templates
4. **Phase 4**: Implement logging and monitoring
5. **Phase 5**: Add remaining CUs and validate
6. **Phase 6**: Schedule and automate
7. **Phase 7**: Build monitoring dashboards

Sample Execution Command

```bash
```

```
# Run for single credit union
python run_clip_analysis.py --cu CU_001 --run-date 2025-01-01

# Run for all credit unions
python run_clip_analysis.py --all --run-date 2025-01-01

# Run with custom config override
python run_clip_analysis.py --cu CU_001 --config custom_config.json

# Dry run (validation only)
python run_clip_analysis.py --cu CU_001 --dry-run
```

End of Design Document