**Data Requirements for Liquidity Forecasting System**

**Core Required Data (Minimum Viable Product)**

**1. Daily Cash Flow Data (REQUIRED)**

**File Format**: CSV, Excel (.xlsx), or Parquet **Update Frequency**: Daily **Historical Requirement**: Minimum 3 years, preferably 5+ years

**File Structure: daily\_cash\_flows.csv**

date,account\_id,account\_name,inflows,outflows,end\_of\_day\_balance,currency

2019-01-01,ACC001,Operating Account 1,1250000.00,1100000.00,45000000.00,USD

2019-01-01,ACC002,Operating Account 2,850000.00,920000.00,12000000.00,USD

2019-01-02,ACC001,Operating Account 1,1180000.00,1350000.00,44830000.00,USD

2019-01-02,ACC002,Operating Account 2,920000.00,880000.00,12040000.00,USD

**Column Specifications:**

* date: YYYY-MM-DD format, no missing dates
* account\_id: Unique identifier for each bank account
* account\_name: Human-readable account name
* inflows: Positive cash receipts (USD, no negatives)
* outflows: Positive cash payments (USD, no negatives)
* end\_of\_day\_balance: Account balance at day end
* currency: ISO currency code (USD, EUR, etc.)

**Data Quality Requirements:**

* No missing dates (weekends/holidays = 0 flows, not missing rows)
* Inflows/outflows always ≥ 0
* Balance reconciliation: Previous balance + inflows - outflows = current balance
* Maximum 5% missing values per column
* Outliers flagged if >3 standard deviations from rolling 30-day mean

**2. Historical Liquidity Usage (REQUIRED)**

**File Format**: CSV or Excel **Update Frequency**: Daily **Historical Requirement**: Same period as cash flow data

**File Structure: liquidity\_usage.csv**

date,short\_term\_cash\_used,commercial\_paper\_issued,credit\_line\_drawn,long\_term\_redemptions,total\_liquidity\_need,trigger\_reason

2019-01-01,50000.00,0.00,0.00,0.00,50000.00,routine

2019-01-15,150000.00,500000.00,0.00,0.00,650000.00,seasonal\_outflow

2019-03-31,200000.00,1000000.00,2000000.00,0.00,3200000.00,quarter\_end

2019-07-15,100000.00,0.00,5000000.00,0.00,5100000.00,unexpected\_outflow

**Column Specifications:**

* date: YYYY-MM-DD format
* short\_term\_cash\_used: Amount drawn from cash reserves
* commercial\_paper\_issued: CP issuance amount
* credit\_line\_drawn: Credit facility usage
* long\_term\_redemptions: Early investment liquidations
* total\_liquidity\_need: Sum of all liquidity sources used
* trigger\_reason: Categorical (routine, seasonal\_outflow, quarter\_end, stress\_event, etc.)

**3. Cost Structure Data (REQUIRED)**

**File Format**: JSON or YAML (preferred) or CSV **Update Frequency**: Weekly or when rates change

**File Structure: cost\_structure.yaml**

liquidity\_tiers:

short\_term\_cash:

annual\_rate: 0.05 # 5% opportunity cost

fixed\_cost: 0 # No transaction fees

min\_balance: 1000000 # $1M minimum cash balance

max\_capacity: .inf # No limit

access\_time\_hours: 0 # Immediate

reliability\_score: 1.0 # 100% reliable

currency: USD

commercial\_paper:

annual\_rate: 0.055 # Current CP rate

fixed\_cost: 5000 # Issuance costs

min\_balance: 0

max\_capacity: 500000000 # $500M program limit

access\_time\_hours: 24 # 1 day to issue

reliability\_score: 0.95 # 95% reliable

currency: USD

credit\_lines:

annual\_rate: 0.08 # All-in drawn rate

commitment\_fee\_rate: 0.004 # 40bps commitment fee

fixed\_cost: 2500 # Draw fee

min\_balance: 0

max\_capacity: 1000000000 # $1B facility

access\_time\_hours: 4 # 4 hours to draw

reliability\_score: 0.98 # 98% reliable

covenant\_buffer: 0.2 # Keep 20% covenant buffer

currency: USD

long\_term\_redemptions:

annual\_rate: 0.12 # Early redemption penalty

fixed\_cost: 25000 # High transaction costs

min\_balance: 0

max\_capacity: 2000000000 # $2B available

access\_time\_hours: 72 # 3 days settlement

reliability\_score: 0.90 # Market dependent

currency: USD

rate\_update\_frequency: weekly

last\_updated: "2025-06-13"

data\_source: "Treasury Department"

**Enhanced Data (Recommended for Better Performance)**

**4. Transaction-Level Detail (RECOMMENDED)**

**File Format**: CSV or Parquet (for large files) **Update Frequency**: Daily

**File Structure: transaction\_detail.csv**

date,transaction\_id,account\_id,amount,transaction\_type,counterparty,description,category,subcategory

2019-01-01,TXN001,ACC001,250000.00,inflow,CUSTOMER\_A,Wire transfer,customer\_payment,accounts\_receivable

2019-01-01,TXN002,ACC001,-150000.00,outflow,VENDOR\_B,ACH payment,vendor\_payment,accounts\_payable

2019-01-01,TXN003,ACC001,500000.00,inflow,BANK\_C,Loan drawdown,financing,credit\_facility

**Transaction Categories:**

inflow\_categories:

- customer\_payments

- loan\_proceeds

- investment\_income

- asset\_sales

- other\_income

outflow\_categories:

- vendor\_payments

- payroll

- debt\_service

- capital\_expenditures

- tax\_payments

- other\_expenses

**5. Market Data (RECOMMENDED)**

**File Format**: CSV or API feed **Update Frequency**: Daily

**File Structure: market\_data.csv**

date,fed\_funds\_rate,3m\_treasury,1y\_treasury,commercial\_paper\_rate,credit\_spread,vix,dollar\_index

2019-01-01,2.40,2.45,2.55,2.60,1.20,20.5,96.2

2019-01-02,2.40,2.47,2.56,2.62,1.18,19.8,96.4

**6. Economic Calendar (RECOMMENDED)**

**File Format**: CSV **Update Frequency**: Monthly

**File Structure: economic\_calendar.csv**

date,event\_type,importance,description,impact\_direction

2019-01-15,earnings,high,Quarterly earnings release,outflow

2019-03-31,quarter\_end,high,Q1 quarter end,outflow

2019-04-15,tax\_payment,high,Corporate tax payment,outflow

2019-06-30,dividend,medium,Dividend payment,outflow

2019-12-25,holiday,low,Christmas holiday,minimal

**7. Business Context Data (OPTIONAL BUT VALUABLE)**

**File Format**: CSV **Update Frequency**: Weekly

**File Structure: business\_context.csv**

date,revenue\_forecast,capex\_planned,acquisition\_activity,regulatory\_changes,business\_seasonality

2019-01-01,45000000,2000000,0,0,low

2019-03-31,52000000,5000000,0,1,high

**Data Validation Requirements**

**Pre-Processing Validation Checklist**

**Completeness Checks:**

# Required validations

assert data['date'].isna().sum() == 0, "No missing dates allowed"

assert data['inflows'].isna().sum() < len(data) \* 0.05, "Max 5% missing inflows"

assert data['outflows'].isna().sum() < len(data) \* 0.05, "Max 5% missing outflows"

**Consistency Checks:**

# Balance reconciliation

balance\_check = (

data['end\_of\_day\_balance'].shift(1) +

data['inflows'] -

data['outflows'] -

data['end\_of\_day\_balance']

).abs() < 1000 # Allow $1K rounding differences

assert balance\_check.mean() > 0.95, "95% of balances must reconcile"

**Range Checks:**

# Reasonable value ranges

assert data['inflows'].min() >= 0, "No negative inflows"

assert data['outflows'].min() >= 0, "No negative outflows"

assert data['inflows'].max() < 100\_000\_000, "Flag inflows >$100M"

assert data['outflows'].max() < 100\_000\_000, "Flag outflows >$100M"

**Data Delivery Methods**

**Option 1: File Upload (Simplest)**

Acceptable formats:

- CSV files (UTF-8 encoded)

- Excel files (.xlsx, not .xls)

- Parquet files (for large datasets)

Delivery method:

- Secure file transfer (SFTP)

- Cloud storage (AWS S3, Azure Blob)

- Direct upload to application

**Option 2: Database Connection (Preferred)**

-- Example database schema

CREATE TABLE daily\_cash\_flows (

date DATE NOT NULL,

account\_id VARCHAR(20) NOT NULL,

account\_name VARCHAR(100),

inflows DECIMAL(15,2) NOT NULL,

outflows DECIMAL(15,2) NOT NULL,

end\_of\_day\_balance DECIMAL(15,2),

currency CHAR(3) DEFAULT 'USD',

PRIMARY KEY (date, account\_id)

);

-- Required database access:

- Read-only access to production tables

- Daily incremental data refresh

- Historical data backfill capability

**Option 3: API Integration (Enterprise)**

{

"endpoint": "https://your-system.com/api/v1/cash-flows",

"authentication": "Bearer token or API key",

"response\_format": {

"date": "YYYY-MM-DD",

"accounts": [

{

"account\_id": "string",

"inflows": "number",

"outflows": "number",

"balance": "number"

}

]

}

}

**Sample Data Generation**

If you don't have historical data, we can generate realistic sample data using:

**Data Simulation Parameters:**

simulation\_config = {

"start\_date": "2019-01-01",

"end\_date": "2024-12-31",

"base\_daily\_inflows": 1\_000\_000,

"base\_daily\_outflows": 900\_000,

"seasonality\_amplitude": 0.2, # 20% seasonal variation

"volatility": 0.15, # 15% daily volatility

"trend": 0.03, # 3% annual growth

"crisis\_periods": [

("2020-03-01", "2020-05-31"), # COVID impact

("2022-03-01", "2022-04-30") # Geopolitical stress

]

}

**Data Quality Scoring**

We'll implement automated data quality scoring:

def calculate\_data\_quality\_score(df):

scores = {

'completeness': 1 - df.isna().mean().mean(),

'consistency': balance\_reconciliation\_rate(df),

'validity': valid\_value\_percentage(df),

'timeliness': data\_freshness\_score(df),

'accuracy': outlier\_detection\_score(df)

}

return np.mean(list(scores.values()))

# Target: >0.95 (95% data quality score)

**Critical Data Preparation Steps**

**Before Project Start:**

1. **Data Inventory**: Catalog all available data sources
2. **Access Permissions**: Ensure read access to all required systems
3. **Data Dictionary**: Document all field definitions and business rules
4. **Historical Backfill**: Ensure at least 3 years of clean historical data
5. **Update Procedures**: Establish daily data refresh processes

**Week 1 Priorities:**

1. **Data Validation**: Run all validation checks on historical data
2. **Gap Analysis**: Identify and document any missing data
3. **Quality Baseline**: Establish current data quality metrics
4. **Sample Generation**: Create sample datasets for development if needed
5. **Access Testing**: Verify all data connections work reliably

This comprehensive data specification ensures the ML system will have high-quality inputs for accurate predictions and cost optimization. The modular approach allows starting with minimum required data and enhancing with additional sources over time.