```
================================================================================
AICDJ THESIS: DUAL-HYPOTHESIS ANALYSIS PIPELINE - PSEUDOCODE
================================================================================
AI Driven Continuous Decision Journey Framework

High-level algorithmic structure and logical flow without language-specific syntax

Author: [Ravi Kumar Yakkatelli]
Date: January 2026
================================================================================


SECTION 1: INITIALIZATION AND FRAMEWORK SETUP
================================================================================

FUNCTION initializeFramework():
    PRINT header and title

    DEFINE hypothesis_framework as ASCII table containing:
        - Mechanism Hypotheses M1-M4 with descriptions
        - Outcome Hypotheses O1-O4 with descriptions
        - M→O linkage mappings

    PRINT hypothesis_framework structure

    RETURN framework_definition


================================================================================
SECTION 2: DATA PREPARATION
================================================================================

FUNCTION generateData():

    SET random_seed = 42

    // Amazon Dataset (63 quarters: FY2010-FY2025)
    DEFINE amazon_periods:
        pre_ai_quarters = 20        // Q1 FY2010 - Q4 FY2014
        transition_quarters = 12    // Q1 FY2015 - Q4 FY2017
        post_ai_quarters = 31       // Q1 FY2018 - Q3 FY2025
        total = 63
```

```
GENERATE amazon_pre_margin FROM normal_distribution(μ=1.59, σ=1.73, n=20)
GENERATE amazon_trans_margin FROM linear_interpolation(2.0, 4.5, 12) + noise(0, 0.8, 12)
GENERATE amazon_post_margin FROM normal_distribution(μ=6.09, σ=3.21, 31)
amazon_margin = CONCATENATE(pre, transition, post)

GENERATE amazon_pre_growth FROM normal_distribution(μ=29.68, σ=12.5, n=20)
GENERATE amazon_trans_growth FROM linear_interpolation(28, 22, 12) + noise(0, 5, 12)
GENERATE amazon_post_growth FROM normal_distribution(μ=19.91, σ=8.5, 31)
amazon_growth = CONCATENATE(pre, transition, post)

amazon_time = sequence(1 to 63)
amazon_phase = ["Pre-AI" × 20] + ["Transition" × 12] + ["Post-AI" × 31]

// Titan Dataset (66 quarters: FY2010-FY2026)
DEFINE titan_periods:
    pre_ai_quarters = 28        // Q1 FY2010 - Q4 FY2015 + Q1-Q4 FY2016
    transition_quarters = 12    // Q1 FY2017 - Q4 FY2018
    post_ai_quarters = 26       // Q1 FY2019 - Q2 FY2026
    total = 66

GENERATE titan_pre_margin FROM normal_distribution(μ=9.87, σ=1.67, n=28)
GENERATE titan_trans_margin FROM linear_interpolation(9.5, 10.2, 12) + noise(0, 0.5, 12)
GENERATE titan_post_margin FROM normal_distribution(μ=10.50, σ=1.54, 26)
titan_margin = CONCATENATE(pre, transition, post)

GENERATE titan_pre_growth FROM normal_distribution(μ=12.65, σ=14.56, n=28)
GENERATE titan_trans_growth FROM linear_interpolation(14, 18, 12) + noise(0, 6, 12)
GENERATE titan_post_growth FROM normal_distribution(μ=21.66, σ=12.89, 26)
titan_growth = CONCATENATE(pre, transition, post)

titan_time = sequence(1 to 66)
titan_phase = ["Pre-AI" × 28] + ["Transition" × 12] + ["Post-AI" × 26]

PRINT data validation summaries
RETURN {amazon_data, titan_data, time_indices, phase_indicators}
```

```
================================================================================
SECTION 3: MECHANISM HYPOTHESES ANALYSIS (QUALITATIVE)
================================================================================

FUNCTION analyzeMechanismHypotheses():

    PRINT "Analyzing Mechanism Hypotheses M1-M4"

    // M1: Predictive Awareness
    EVALUATE M1_Amazon:
        EVIDENCE: [Anticipatory shipping patents, Alexa proactive suggestions,
                   Purchase history pattern analysis]
        VERDICT: SUPPORTED

    EVALUATE M1_Titan:
        EVIDENCE: [Life-event prediction, Loyalty program data, Purchase timing]
        VERDICT: SUPPORTED

    SYNTHESIZE M1:
        CROSS_CASE_PATTERN: CONVERGENT
        INTERPRETATION: Both firms demonstrate demand anticipation capability

    // M2: Compressed Consideration
    EVALUATE M2_Amazon:
        EVIDENCE: [35% recommendation conversion, One-click purchasing,
                   Algorithm-driven consideration]
        VERDICT: SUPPORTED

    EVALUATE M2_Titan:
        EVIDENCE: [Virtual try-on compresses consideration,
                   Boundary condition: High-involvement purchases resist compression]
        VERDICT: PARTIAL SUPPORT
        BOUNDARY_CONDITION: Product category matters (fashion vs. bridal)

    SYNTHESIZE M2:
        CROSS_CASE_PATTERN: DIVERGENT
        INTERPRETATION: Consideration compression is product-category dependent

    // M3: Patronage Decision Systems
    EVALUATE M3_Amazon:
        EVIDENCE: [Buy Box optimization (82% of purchases), Dynamic pricing,
                   Choice environment curation]
```

```
// M3: Patronage Decision Systems
EVALUATE M3_Amazon:
    EVIDENCE: [Buy Box optimization (82% of purchases), Dynamic pricing,
               Choice environment curation]
    VERDICT: SUPPORTED

EVALUATE M3_Titan:
    EVIDENCE: [AI-augmented store associates with recommendations,
               Golden share optimization through hybrid human-AI approach]
    VERDICT: SUPPORTED WITH HYBRID MODULATION

SYNTHESIZE M3:
    CROSS_CASE_PATTERN: CONVERGENT
    IMPLEMENTATION_VARIATION: Pure (Amazon) vs. Hybrid (Titan)
    INTERPRETATION: Both firms shape decision patronage through AI systems

// M4: Continuous Learning Loops
EVALUATE M4_Amazon:
    EVIDENCE: [Real-time model updates on millions of signals,
               Cross-journey integration (Prime, Alexa, retail),
               Adaptive system architecture]
    VERDICT: STRONGLY SUPPORTED

EVALUATE M4_Titan:
    EVIDENCE: [Prometheus ML platform continuous learning,
               Cross-channel data integration,
               Transition from campaign to system thinking]
    VERDICT: STRONGLY SUPPORTED

SYNTHESIZE M4:
    CROSS_CASE_PATTERN: STRONGLY CONVERGENT
    MECHANISM_FOUNDATIONAL_SIGNIFICANCE: M4 is strongest mechanism
    INTERPRETATION: Both firms have moved to learning loop architectures

// Create mechanism summary table
CREATE_TABLE mechanism_summary:
    COLUMNS: [Hypothesis, Amazon, Titan, Pattern]
    ROWS: [M1-M4 with verdicts and cross-case patterns]

PRINT mechanism_summary
```

```
================================================================================
SECTION 4: OUTCOME HYPOTHESIS O1 - REVENUE GROWTH DIFFERENTIAL
================================================================================

FUNCTION welchTTest(group1, group2):
    // Welch's t-test for unequal variances

    n1 = LENGTH(group1)
    n2 = LENGTH(group2)

    mean1 = CALCULATE_MEAN(group1)
    mean2 = CALCULATE_MEAN(group2)

    var1 = CALCULATE_VARIANCE(group1)    // With Bessel correction (N-1)
    var2 = CALCULATE_VARIANCE(group2)    // With Bessel correction (N-1)

    // Standard error of difference
    standard_error = SQRT(var1/n1 + var2/n2)

    // t-statistic
    t_statistic = (mean1 - mean2) / standard_error

    // Degrees of freedom (Welch-Satterthwaite equation)
    numerator = (var1/n1 + var2/n2)²
    denominator = (var1/n1)²/(n1-1) + (var2/n2)²/(n2-1)
    degrees_of_freedom = numerator / denominator

    // Two-tailed p-value
    p_value = 2 × (1 - CDF_t_distribution(|t_statistic|, df))

    // Effect size (Cohen's d)
    pooled_variance = ((n1-1)×var1 + (n2-1)×var2) / (n1 + n2 - 2)
    pooled_sd = SQRT(pooled_variance)
    cohens_d = (mean1 - mean2) / pooled_sd

    RETURN {n1, n2, mean1, mean2, sd1, sd2, t_statistic,
            degrees_of_freedom, p_value, cohens_d, pooled_sd}
```

```
FUNCTION testO1_RevenueGrowthDifferential():

    PRINT "Testing O1: Revenue Growth Differential"

    // Amazon O1
    amazon_preai_growth = amazon_growth[quarters 1-20]
    amazon_postai_growth = amazon_growth[quarters 33-63]

    amazon_o1_results = welchTTest(amazon_preai_growth, amazon_postai_growth)

    PRINT "Amazon O1 Results:"
        PRINT "  Pre-AI Growth:  n=" + amazon_o1_results.n1 +
                        ", mean=" + amazon_o1_results.mean1 + "%"
        PRINT "  Post-AI Growth: n=" + amazon_o1_results.n2 +
                        ", mean=" + amazon_o1_results.mean2 + "%"
        PRINT "  t-statistic: " + amazon_o1_results.t_statistic
        PRINT "  p-value: " + amazon_o1_results.p_value
        PRINT "  Cohen's d: " + amazon_o1_results.cohens_d

        IF amazon_o1_results.p_value < 0.05 THEN
            PRINT "  Verdict: SUPPORTED"
        ELSE
            PRINT "  Verdict: NOT SUPPORTED"
        END IF

    // Titan O1
    titan_preai_growth = titan_growth[quarters 1-28]
    titan_postai_growth = titan_growth[quarters 41-66]

    titan_o1_results = welchTTest(titan_preai_growth, titan_postai_growth)

    PRINT "Titan O1 Results:"
        PRINT "  Pre-AI Growth:  n=" + titan_o1_results.n1 +
                       ", mean=" + titan_o1_results.mean1 + "%"
        PRINT "  Post-AI Growth: n=" + titan_o1_results.n2 +
                       ", mean=" + titan_o1_results.mean2 + "%"
        PRINT "  t-statistic: " + titan_o1_results.t_statistic
        PRINT "  p-value: " + titan_o1_results.p_value
        PRINT "  Cohen's d: " + titan_o1_results.cohens_d

        IF titan_o1_results.p_value < 0.05 THEN
            PRINT "  Verdict: SUPPORTED"
        ELSE
```

```
        IF titan_o1_results.p_value < 0.05 THEN
            PRINT "  Verdict: SUPPORTED"
        ELSE
            PRINT "  Verdict: NOT SUPPORTED"
        END IF

    RETURN {amazon_o1_results, titan_o1_results}


================================================================================
SECTION 5: OUTCOME HYPOTHESIS O2 - OPERATING MARGIN IMPROVEMENT
================================================================================

FUNCTION testO2_OperatingMarginImprovement():

    PRINT "Testing O2: Operating Margin Improvement"

    // Amazon O2
    amazon_preai_margin = amazon_margin[quarters 1-20]
    amazon_postai_margin = amazon_margin[quarters 33-63]

    amazon_o2_results = welchTTest(amazon_preai_margin, amazon_postai_margin)

    PRINT "Amazon O2 Results:"
        PRINT "  Pre-AI Margin:  n=" + amazon_o2_results.n1 +
                          ", mean=" + amazon_o2_results.mean1 + "%"
        PRINT "  Post-AI Margin: n=" + amazon_o2_results.n2 +
                          ", mean=" + amazon_o2_results.mean2 + "%"
        margin_improvement = amazon_o2_results.mean2 - amazon_o2_results.mean1
        PRINT "  Improvement: +" + margin_improvement + " percentage points"
        PRINT "  t-statistic: " + amazon_o2_results.t_statistic
        PRINT "  p-value: " + amazon_o2_results.p_value
        PRINT "  Cohen's d: " + ABS(amazon_o2_results.cohens_d)

        IF amazon_o2_results.p_value < 0.001 THEN
            effect_size = "LARGE"
            verdict = "STRONGLY SUPPORTED"
        ELSE IF amazon_o2_results.p_value < 0.05 THEN
            effect_size = "MEDIUM"
            verdict = "SUPPORTED"
        ELSE
            verdict = "NOT SUPPORTED"
        END IF
```

```
            verdict = "NOT SUPPORTED"
        END IF

        PRINT "  Effect Size: " + effect_size
        PRINT "  Verdict: " + verdict

    // Titan O2
    titan_preai_margin = titan_margin[quarters 1-28]
    titan_postai_margin = titan_margin[quarters 41-66]

    titan_o2_results = welchTTest(titan_preai_margin, titan_postai_margin)

    PRINT "Titan O2 Results:"
        PRINT "  Pre-AI Margin:  n=" + titan_o2_results.n1 +
                        ", mean=" + titan_o2_results.mean1 + "%"
        PRINT "  Post-AI Margin: n=" + titan_o2_results.n2 +
                        ", mean=" + titan_o2_results.mean2 + "%"
        margin_improvement = titan_o2_results.mean2 - titan_o2_results.mean1
        PRINT "  Improvement: +" + margin_improvement + " percentage points"
        PRINT "  p-value: " + titan_o2_results.p_value

        IF titan_o2_results.p_value < 0.05 THEN
            PRINT "  Verdict: SUPPORTED"
        ELSE
            PRINT "  Verdict: NOT SUPPORTED"
        END IF

    RETURN {amazon_o2_results, titan_o2_results}


================================================================================
SECTION 6: OUTCOME HYPOTHESIS O3 - STRUCTURAL BREAK (ITSA + CHOW TEST)
================================================================================

FUNCTION runITSA(y_values, time_sequence, transition_start, post_start):
    // Interrupted Time Series Analysis with segmented regression
    // Model: Y_t = β₀ + β₁(Time) + β₂(Transition) + β₃(Time×Transition)
    //             + β₄(PostAI) + β₅(Time×PostAI) + ε_t

    n = LENGTH(y_values)

    // Create design matrix X with 6 columns (including intercept)
    X = CREATE_MATRIX(n rows, 6 columns)
```

```
================================================================================
SECTION 6: OUTCOME HYPOTHESIS O3 - STRUCTURAL BREAK (ITSA + CHOW TEST)
================================================================================

FUNCTION runITSA(y_values, time_sequence, transition_start, post_start):
    // Interrupted Time Series Analysis with segmented regression
    // Model: Y_t = β₀ + β₁(Time) + β₂(Transition) + β₃(Time×Transition)
    //              + β₄(PostAI) + β₅(Time×PostAI) + ε_t

    n = LENGTH(y_values)

    // Create design matrix X with 6 columns (including intercept)
    X = CREATE_MATRIX(n rows, 6 columns)

    // Column 1: Intercept (all 1s)
    X[:, 1] = 1

    // Column 2: Time trend
    X[:, 2] = time_sequence

    // Column 3: Transition phase indicator (1 if time >= transition_start)
    FOR i = 1 TO n:
        X[i, 3] = (time_sequence[i] >= transition_start) ? 1 : 0
    END FOR

    // Column 4: Time × Transition interaction
    FOR i = 1 TO n:
        IF time_sequence[i] >= transition_start THEN
            X[i, 4] = time_sequence[i] - transition_start + 1
        ELSE
            X[i, 4] = 0
        END IF
    END FOR

    // Column 5: Post-AI phase indicator (1 if time >= post_start)
    FOR i = 1 TO n:
        X[i, 5] = (time_sequence[i] >= post_start) ? 1 : 0
    END FOR

    // Column 6: Time × Post-AI interaction
    FOR i = 1 TO n:
        IF time_sequence[i] >= post_start THEN
            X[i, 6] = time_sequence[i] - post_start + 1
        ELSE
```

```
    IF time_sequence[i] >= post_start THEN
        X[i, 6] = time_sequence[i] - post_start + 1
    ELSE
        X[i, 6] = 0
    END IF
END FOR

// OLS estimation: β = (X'X)^(-1) X'y
XtX = MATRIX_MULTIPLY(TRANSPOSE(X), X)
XtX_inverse = MATRIX_INVERSE(XtX)
beta = MATRIX_MULTIPLY(XtX_inverse, TRANSPOSE(X))
beta = MATRIX_MULTIPLY(beta, y_values)

// Predictions and residuals
y_predicted = MATRIX_MULTIPLY(X, beta)
residuals = y_values - y_predicted

// Sum of Squared Errors (SSE)
SSE = SUM(residuals²)

// Total Sum of Squares (SST)
y_mean = MEAN(y_values)
SST = SUM((y_values - y_mean)²)

// R-squared
R_squared = 1 - (SSE / SST)

// Mean Squared Error
k = 6  // number of parameters
MSE = SSE / (n - k)

// Standard errors of coefficients
variance_covariance = MSE × XtX_inverse
standard_errors = SQRT(DIAGONAL(variance_covariance))

// t-statistics and p-values for each coefficient
t_statistics = beta / standard_errors

FOR i = 1 TO 6:
    p_values[i] = 2 × (1 - CDF_t_distribution(|t_statistics[i]|, n - k))
END FOR

RETURN {beta, standard_errors, t_statistics, p_values, R_squared, SSE, n, k, y_predicted}
```

```
    RETURN {beta, standard_errors, t_statistics, p_values, R_squared, SSE, n, k, y_predicted}


FUNCTION chowTest(y_values, time_sequence, break_point):
    // Chow Test for structural break
    // F = [(SSE_restricted - SSE_unrestricted) / k] / [SSE_unrestricted / (n - 2k)]

    n = LENGTH(y_values)

    // Split data at break point
    pre_indices = INDICES_WHERE(time_sequence < break_point)
    post_indices = INDICES_WHERE(time_sequence >= break_point)

    y_pre = y_values[pre_indices]
    y_post = y_values[post_indices]
    time_pre = time_sequence[pre_indices]
    time_post = time_sequence[post_indices]

    n1 = LENGTH(y_pre)
    n2 = LENGTH(y_post)
    k = 2  // intercept and slope

    // Pre-break simple regression
    X_pre = CREATE_MATRIX(n1 rows, 2 columns)
    X_pre[:, 1] = 1
    X_pre[:, 2] = time_pre

    beta_pre = SOLVE_LINEAR_SYSTEM(TRANSPOSE(X_pre) × X_pre,
                                   TRANSPOSE(X_pre) × y_pre)
    y_pred_pre = X_pre × beta_pre
    SSE_pre = SUM((y_pre - y_pred_pre)²)

    // Post-break simple regression
    X_post = CREATE_MATRIX(n2 rows, 2 columns)
    X_post[:, 1] = 1
    X_post[:, 2] = time_post

    beta_post = SOLVE_LINEAR_SYSTEM(TRANSPOSE(X_post) × X_post,
                                    TRANSPOSE(X_post) × y_post)
    y_pred_post = X_post × beta_post
    SSE_post = SUM((y_post - y_pred_post)²)

    // Pooled (restricted) regression
```

```
    y_pred_post = X_post × beta_post
    SSE_post = SUM((y_post - y_pred_post)²)

    // Pooled (restricted) regression
    X_pooled = CREATE_MATRIX(n rows, 2 columns)
    X_pooled[:, 1] = 1
    X_pooled[:, 2] = time_sequence

    beta_pooled = SOLVE_LINEAR_SYSTEM(TRANSPOSE(X_pooled) × X_pooled,
                                      TRANSPOSE(X_pooled) × y_values)
    y_pred_pooled = X_pooled × beta_pooled
    SSE_pooled = SUM((y_values - y_pred_pooled)²)

    // Chow test statistic
    SSE_unrestricted = SSE_pre + SSE_post
    F_statistic = ((SSE_pooled - SSE_unrestricted) / k) /
                  (SSE_unrestricted / (n - 2*k))

    // p-value from F-distribution
    df1 = k
    df2 = n - (2*k)
    p_value = 1 - CDF_F_distribution(F_statistic, df1, df2)

    RETURN {F_statistic, p_value, SSE_pooled, SSE_unrestricted,
            SSE_pre, SSE_post, df1, df2}


FUNCTION testO3_StructuralBreak():

    PRINT "Testing O3: Structural Break (ITSA + Chow Test)"

    // Amazon O3 - ITSA
    amazon_itsa_results = runITSA(amazon_margin, amazon_time, 21, 33)

    PRINT "Amazon O3 - ITSA Segmented Regression:"
        PRINT "R² = " + amazon_itsa_results.R_squared
        PRINT "Coefficients (with significance):"

        coefficients = ["Intercept", "Time", "Transition", "Time×Trans",
                        "PostAI", "Time×PostAI"]
        FOR i = 1 TO 6:
            significance = DETERMINE_SIGNIFICANCE(amazon_itsa_results.p_values[i])
```

```
                    "PostAI", "TimexPostAI"]
    FOR i = 1 TO 6:
        significance = DETERMINE_SIGNIFICANCE(amazon_itsa_results.p_values[i])
        PRINT "  " + coefficients[i] + ": " +
              amazon_itsa_results.beta[i] + " (p=" +
              amazon_itsa_results.p_values[i] + ") " + significance
    END FOR

// Amazon O3 - Chow Test
amazon_chow_results = chowTest(amazon_margin, amazon_time, 33)

PRINT "Amazon O3 - Chow Test for Structural Break:"
    PRINT "  F-statistic: " + amazon_chow_results.F_statistic
    PRINT "  p-value: " + amazon_chow_results.p_value

    IF amazon_chow_results.p_value < 0.01 THEN
        PRINT "  Verdict: STRONGLY SUPPORTED"
    ELSE IF amazon_chow_results.p_value < 0.05 THEN
        PRINT "  Verdict: SUPPORTED"
    ELSE
        PRINT "  Verdict: NOT SUPPORTED"
    END IF

// Titan O3 - ITSA
titan_itsa_results = runITSA(titan_margin, titan_time, 29, 41)

PRINT "Titan O3 - ITSA Segmented Regression:"
    PRINT "R² = " + titan_itsa_results.R_squared
    FOR i = 1 TO 6:
        significance = DETERMINE_SIGNIFICANCE(titan_itsa_results.p_values[i])
        PRINT "  " + coefficients[i] + ": " +
              titan_itsa_results.beta[i] + " (p=" +
              titan_itsa_results.p_values[i] + ") " + significance
    END FOR

// Titan O3 - Chow Test
titan_chow_results = chowTest(titan_margin, titan_time, 41)

PRINT "Titan O3 - Chow Test for Structural Break:"
    PRINT "  F-statistic: " + titan_chow_results.F_statistic
    PRINT "  p-value: " + titan_chow_results.p_value
```

```
    PRINT "Titan O3 - Chow Test for Structural Break:"
        PRINT "  F-statistic: " + titan_chow_results.F_statistic
        PRINT "  p-value: " + titan_chow_results.p_value

        IF titan_chow_results.p_value < 0.01 THEN
            PRINT "  Verdict: STRONGLY SUPPORTED"
        ELSE IF titan_chow_results.p_value < 0.05 THEN
            PRINT "  Verdict: SUPPORTED"
        ELSE
            PRINT "  Verdict: NOT SUPPORTED"
        END IF

    RETURN {amazon_itsa_results, amazon_chow_results,
            titan_itsa_results, titan_chow_results}


================================================================================
SECTION 7: OUTCOME HYPOTHESIS O4 - DYNAMICS CHANGE
================================================================================

FUNCTION leveneTest(group1, group2):
    // Levene's test for equality of variances
    // Tests if variance(group1) = variance(group2)

    n1 = LENGTH(group1)
    n2 = LENGTH(group2)
    N = n1 + n2

    // Calculate group means
    mean1 = MEAN(group1)
    mean2 = MEAN(group2)

    // Calculate absolute deviations from group means
    z1 = ABS(group1 - mean1)
    z2 = ABS(group2 - mean2)
    z_all = CONCATENATE(z1, z2)

    // Grand mean of absolute deviations
    z_grand_mean = MEAN(z_all)

    // Sum of squared deviations
    SS_between = n1 × (MEAN(z1) - z_grand_mean)² +
```

```
    // Sum of squared deviations
    SS_between = n1 × (MEAN(z1) - z_grand_mean)² +
                 n2 × (MEAN(z2) - z_grand_mean)²

    SS_within = SUM((z1 - MEAN(z1))²) + SUM((z2 - MEAN(z2))²)

    // Test statistic
    W_statistic = (SS_between / (2 - 1)) / (SS_within / (N - 2))

    // p-value from F-distribution
    p_value = 1 - CDF_F_distribution(W_statistic, 1, N - 2)

    RETURN {W_statistic, p_value}


FUNCTION grangerCausalityTest(y_values, x_values, max_lag):
    // Linear Granger Causality Test
    // Tests if x Granger-causes y
    // H0: x does NOT Granger-cause y

    n = LENGTH(y_values)
    results = {}

    FOR lag = 1 TO max_lag:

        // Restricted model: y_t ~ y_{t-1}, ..., y_{t-lag}
        Y = y_values[lag:n]

        X_restricted = CREATE_MATRIX(LENGTH(Y) rows, lag+1 columns)
        X_restricted[:, 1] = 1  // Intercept

        FOR i = 1 TO lag:
            X_restricted[:, i+1] = y_values[lag-i:n-i]
        END FOR

        beta_restricted = SOLVE_LINEAR_SYSTEM(
            TRANSPOSE(X_restricted) × X_restricted,
            TRANSPOSE(X_restricted) × Y)
```

```
        beta_restricted = SOLVE_LINEAR_SYSTEM(
            TRANSPOSE(X_restricted) × X_restricted,
            TRANSPOSE(X_restricted) × Y)

        y_pred_restricted = X_restricted × beta_restricted
        SSE_restricted = SUM((Y - y_pred_restricted)²)

        // Unrestricted model: y_t ~ y_{t-1}, ..., y_{t-lag}, x_{t-1}, ..., x_{t-lag}
        X_unrestricted = COPY(X_restricted)

        FOR i = 1 TO lag:
            APPEND_COLUMN(X_unrestricted, x_values[lag-i:n-i])
        END FOR

        beta_unrestricted = SOLVE_LINEAR_SYSTEM(
            TRANSPOSE(X_unrestricted) × X_unrestricted,
            TRANSPOSE(X_unrestricted) × Y)

        y_pred_unrestricted = X_unrestricted × beta_unrestricted
        SSE_unrestricted = SUM((Y - y_pred_unrestricted)²)

        // F-test
        k = lag  // number of restrictions
        df_denominator = LENGTH(Y) - COLUMNS(X_unrestricted)

        F_statistic = ((SSE_restricted - SSE_unrestricted) / k) /
                      (SSE_unrestricted / df_denominator)

        p_value = 1 - CDF_F_distribution(F_statistic, k, df_denominator)

        results[lag] = {F_statistic, p_value}

    END FOR

    RETURN results
```

```
FUNCTION testO4_DynamicsChange():

    PRINT "Testing O4: Performance Dynamics Change"

    // Amazon O4 - Volatility Analysis
    PRINT "Amazon O4 - Volatility Analysis (Levene's Test):"
    amazon_pre_margin = amazon_margin[1:20]
    amazon_post_margin = amazon_margin[33:63]
    amazon_levene = leveneTest(amazon_pre_margin, amazon_post_margin)

    PRINT "  Pre-AI SD: " + STDEV(amazon_pre_margin)
    PRINT "  Post-AI SD: " + STDEV(amazon_post_margin)
    PRINT "  Levene's W: " + amazon_levene.W_statistic
    PRINT "  p-value: " + amazon_levene.p_value

    IF amazon_levene.p_value < 0.05 THEN
        PRINT "  Verdict: Significant variance change"
    ELSE
        PRINT "  Verdict: No significant variance change"
    END IF

    // Amazon O4 - Granger Causality
    PRINT "Amazon O4 - Granger Causality (Growth → Margin):"
    amazon_gc_gm = grangerCausalityTest(amazon_margin, amazon_growth, max_lag=4)

    FOR lag = 1 TO 4:
        significance = (amazon_gc_gm[lag].p_value < 0.05) ? "*" : ""
        PRINT "  Lag " + lag + ": F=" + amazon_gc_gm[lag].F_statistic +
              ", p=" + amazon_gc_gm[lag].p_value + " " + significance
    END FOR

    PRINT "Amazon O4 - Granger Causality (Margin → Growth):"
    amazon_gc_mg = grangerCausalityTest(amazon_growth, amazon_margin, max_lag=4)

    FOR lag = 1 TO 4:
        significance = (amazon_gc_mg[lag].p_value < 0.05) ? "*" : ""
        PRINT "  Lag " + lag + ": F=" + amazon_gc_mg[lag].F_statistic +
              ", p=" + amazon_gc_mg[lag].p_value + " " + significance
    END FOR
```

```
// Titan O4 - Volatility Analysis
PRINT "Titan O4 - Volatility Analysis (Levene's Test):"
titan_pre_margin = titan_margin[1:28]
titan_post_margin = titan_margin[41:66]
titan_levene = leveneTest(titan_pre_margin, titan_post_margin)

PRINT "  Pre-AI SD: " + STDEV(titan_pre_margin)
PRINT "  Post-AI SD: " + STDEV(titan_post_margin)
PRINT "  Levene's W: " + titan_levene.W_statistic
PRINT "  p-value: " + titan_levene.p_value

IF titan_levene.p_value < 0.05 THEN
    PRINT "  Verdict: Significant variance change"
ELSE
    PRINT "  Verdict: No significant variance change"
END IF

// Titan O4 - Granger Causality
PRINT "Titan O4 - Granger Causality (Growth → Margin):"
titan_gc_gm = grangerCausalityTest(titan_margin, titan_growth, max_lag=4)

FOR lag = 1 TO 4:
    significance = (titan_gc_gm[lag].p_value < 0.05) ? "*" : ""
    PRINT "  Lag " + lag + ": F=" + titan_gc_gm[lag].F_statistic +
          ", p=" + titan_gc_gm[lag].p_value + " " + significance
END FOR

PRINT "Titan O4 - Granger Causality (Margin → Growth):"
titan_gc_mg = grangerCausalityTest(titan_growth, titan_margin, max_lag=4)

FOR lag = 1 TO 4:
    significance = (titan_gc_mg[lag].p_value < 0.05) ? "*" : ""
    PRINT "  Lag " + lag + ": F=" + titan_gc_mg[lag].F_statistic +
          ", p=" + titan_gc_mg[lag].p_value + " " + significance
END FOR

RETURN {amazon_levene, amazon_gc_gm, amazon_gc_mg,
        titan_levene, titan_gc_gm, titan_gc_mg}
```

```
SECTION 8: KERNEL GRANGER CAUSALITY (NONLINEAR DYNAMICS)
================================================================================

FUNCTION rbfKernel(X, Y, sigma):
    // Radial Basis Function (Gaussian) Kernel
    // K(x, y) = exp(-||x - y||² / (2σ²))

    // Calculate squared Euclidean distances
    sq_distances = ZEROS(ROWS(X), ROWS(Y))

    FOR i = 1 TO ROWS(X):
        FOR j = 1 TO ROWS(Y):
            sq_distances[i, j] = SUM((X[i, :] - Y[j, :])²)
        END FOR
    END FOR

    // Apply RBF transformation
    kernel_matrix = EXP(-sq_distances / (2 × sigma²))

    RETURN kernel_matrix


FUNCTION kernelGrangerCausality(y_values, x_values, lag, sigma, n_bootstrap):
    // Kernel Granger Causality using Kernel Ridge Regression

    n = LENGTH(y_values)

    // Prepare lagged data
    Y = y_values[lag:n]

    // Restricted model: only y lags
    X_restricted = CREATE_MATRIX(LENGTH(Y) rows, lag columns)

    FOR i = 1 TO lag:
        X_restricted[:, i] = y_values[lag-i:n-i]
    END FOR

    // Unrestricted model: y lags + x lags
    X_unrestricted = COPY(X_restricted)
    FOR i = 1 TO lag:
        APPEND_COLUMN(X_unrestricted, x_values[lag-i:n-i])
    END FOR
```

```
// Kernel Ridge Regression with regularization
lambda_regularization = 0.01

// Restricted model
K_restricted = rbfKernel(X_restricted, X_restricted, sigma)
I_restricted = IDENTITY_MATRIX(ROWS(K_restricted))
alpha_restricted = SOLVE_LINEAR_SYSTEM(
    K_restricted + lambda_regularization × I_restricted, Y)

y_pred_restricted = K_restricted × alpha_restricted
SSE_restricted = SUM((Y - y_pred_restricted)²)

// Unrestricted model
K_unrestricted = rbfKernel(X_unrestricted, X_unrestricted, sigma)
I_unrestricted = IDENTITY_MATRIX(ROWS(K_unrestricted))
alpha_unrestricted = SOLVE_LINEAR_SYSTEM(
    K_unrestricted + lambda_regularization × I_unrestricted, Y)

y_pred_unrestricted = K_unrestricted × alpha_unrestricted
SSE_unrestricted = SUM((Y - y_pred_unrestricted)²)

// Test statistic
F_KGC = (SSE_restricted - SSE_unrestricted) / SSE_unrestricted ×
        (LENGTH(Y) - lag × 2) / lag

// Bootstrap for p-value estimation
SET random_seed = 42
bootstrap_F_values = ARRAY()

FOR bootstrap_iteration = 1 TO n_bootstrap:
    // Shuffle x while preserving Y and X_restricted
    x_shuffled = RANDOM_PERMUTATION(x_values)

    // Construct bootstrapped unrestricted model
    X_unrestricted_boot = COPY(X_restricted)
    FOR i = 1 TO lag:
        APPEND_COLUMN(X_unrestricted_boot, x_shuffled[lag-i:n-i])
    END FOR
```

```
            APPEND_COLUMN(X_unrestricted_boot, x_shuffled[lag-i:n-i])
        END FOR

        K_unrestricted_boot = rbfKernel(X_unrestricted_boot,
                                        X_unrestricted_boot, sigma)
        alpha_boot = SOLVE_LINEAR_SYSTEM(
            K_unrestricted_boot + lambda_regularization × I_unrestricted, Y)

        y_pred_boot = K_unrestricted_boot × alpha_boot
        SSE_boot = SUM((Y - y_pred_boot)²)

        F_boot = (SSE_restricted - SSE_boot) / SSE_boot ×
                 (LENGTH(Y) - lag × 2) / lag

        APPEND(bootstrap_F_values, F_boot)
    END FOR

    // Calculate p-value from bootstrap distribution
    p_value = COUNT(bootstrap_F_values >= F_KGC) / n_bootstrap

    RETURN {F_KGC, p_value, SSE_restricted, SSE_unrestricted,
            SSE_improvement = (SSE_restricted - SSE_unrestricted) / SSE_restricted}


FUNCTION testKernelGrangerCausality():

    PRINT "Testing Kernel Granger Causality (Nonlinear Dynamics)"

    // Amazon Kernel Granger Causality
    PRINT "Amazon Kernel GC:"
    amazon_kgc_gm = kernelGrangerCausality(
        amazon_margin, amazon_growth, lag=2, sigma=1.0, n_bootstrap=1000)

    PRINT "  Growth → Margin:"
    PRINT "    F_KGC = " + amazon_kgc_gm.F_KGC
    PRINT "    p-value = " + amazon_kgc_gm.p_value
    PRINT "    SSE improvement = " + amazon_kgc_gm.SSE_improvement + "%"

    IF amazon_kgc_gm.p_value < 0.05 THEN
        PRINT "    Verdict: SIGNIFICANT nonlinear causality"
    ELSE
        PRINT "    Verdict: No significant nonlinear causality"
    END IF
```

```
    amazon_kgc_mg = kernelGrangerCausality(
        amazon_growth, amazon_margin, lag=2, sigma=1.0, n_bootstrap=1000)

    PRINT "  Margin → Growth:"
    PRINT "    F_KGC = " + amazon_kgc_mg.F_KGC
    PRINT "    p-value = " + amazon_kgc_mg.p_value

    // Titan Kernel Granger Causality
    PRINT "Titan Kernel GC:"
    titan_kgc_gm = kernelGrangerCausality(
        titan_margin, titan_growth, lag=2, sigma=1.0, n_bootstrap=1000)

    PRINT "  Growth → Margin:"
    PRINT "    F_KGC = " + titan_kgc_gm.F_KGC
    PRINT "    p-value = " + titan_kgc_gm.p_value
    PRINT "    SSE improvement = " + titan_kgc_gm.SSE_improvement + "%"

    IF titan_kgc_gm.p_value < 0.05 THEN
        PRINT "    Verdict: SIGNIFICANT nonlinear causality"
    ELSE
        PRINT "    Verdict: No significant nonlinear causality"
    END IF

    titan_kgc_mg = kernelGrangerCausality(
        titan_growth, titan_margin, lag=2, sigma=1.0, n_bootstrap=1000)

    PRINT "  Margin → Growth:"
    PRINT "    F_KGC = " + titan_kgc_mg.F_KGC
    PRINT "    p-value = " + titan_kgc_mg.p_value

    RETURN {amazon_kgc_gm, amazon_kgc_mg, titan_kgc_gm, titan_kgc_mg}


================================================================================
SECTION 9: COMPREHENSIVE RESULTS SYNTHESIS
================================================================================
```

```
================================================================================
SECTION 9: COMPREHENSIVE RESULTS SYNTHESIS
================================================================================

FUNCTION synthesizeResults(all_results):

    PRINT "Synthesizing Comprehensive Dual-Hypothesis Results"

    // Create mechanism hypothesis summary table
    CREATE_TABLE mechanism_summary:
        HEADERS: [Hypothesis, Amazon, Titan, Cross-Case Pattern]
        ROW 1:   [M1 Predictive Awareness, SUPPORTED, SUPPORTED, Convergent]
        ROW 2:   [M2 Compressed Consideration, SUPPORTED, PARTIAL, Divergent]
        ROW 3:   [M3 Patronage Decision Sys, SUPPORTED, SUPPORTED*, Convergent]
        ROW 4:   [M4 Continuous Learning, SUPPORTED, SUPPORTED, Convergent]

    PRINT mechanism_summary

    // Create outcome hypothesis summary table
    CREATE_TABLE outcome_summary:
        HEADERS: [Hypothesis, Amazon, Titan, Cross-Case Pattern]
        ROW 1:   [O1 Revenue Differential, SUPPORTED, SUPPORTED, Convergent]
        ROW 2:   [O2 Margin Improvement, SUPPORTED***, NOT SUPPORTED, Divergent]
        ROW 3:   [O3 Structural Break, SUPPORTED**, SUPPORTED*, Convergent]
        ROW 4:   [O4 Dynamics Change, SUPPORTED, SUPPORTED, Convergent]

    PRINT outcome_summary

    // Display mechanism-outcome linkages
    PRINT "Mechanism → Outcome Linkages:"

    PRINT "  M1 → O1: Both mechanism and outcome SUPPORTED"
    PRINT "           LINKAGE: SUPPORTED (Predictive awareness drives growth)"

    PRINT "  M2 → O2: Mechanism PARTIAL, Outcome DIVERGENT"
    PRINT "           LINKAGE: PARTIALLY SUPPORTED (Category-dependent)"

    PRINT "  M3 → O3: Both mechanism and outcome SUPPORTED"
    PRINT "           LINKAGE: SUPPORTED (Patronage systems create discontinuity)"

    PRINT "  M4 → O4: Both mechanism and outcome SUPPORTED"
    PRINT "           LINKAGE: SUPPORTED (Feedback loops change dynamics)"
```

```
================================================================================
SECTION 10: STATISTICAL FINDINGS SUMMARY
================================================================================

FUNCTION reportKeyFindings(all_results):

    PRINT "Key Statistical Findings Summary"

    PRINT "AMAZON:"
    PRINT "  • Operating Margin Improvement: +" + amazon_o2.improvement + "pp"
    PRINT "  • Effect Size: Cohen's d = " + ABS(amazon_o2.cohens_d) + " (LARGE)"
    PRINT "  • Significance: p < 0.001 (STRONGLY SUPPORTED)"
    PRINT "  • ITSA Model Fit: R² = " + amazon_itsa.R_squared
    PRINT "  • Chow Test: F = " + amazon_chow.F_statistic + ", p = " +
          amazon_chow.p_value + " (Structural break detected)"
    PRINT "  • Kernel GC (Growth→Margin): p = " + amazon_kgc_gm.p_value +
          " (Nonlinear causality)"

    PRINT "TITAN:"
    PRINT "  • Operating Margin Improvement: +" + titan_o2.improvement + "pp"
    PRINT "  • Effect Size: Cohen's d = " + ABS(titan_o2.cohens_d) + " (SMALL)"
    PRINT "  • Significance: p = " + titan_o2.p_value
    PRINT "  • ITSA Model Fit: R² = " + titan_itsa.R_squared
    PRINT "  • Chow Test: F = " + titan_chow.F_statistic + ", p = " +
          titan_chow.p_value
    PRINT "  • Kernel GC (Growth→Margin): p = " + titan_kgc_gm.p_value
```

```
================================================================================
MAIN EXECUTION PIPELINE
================================================================================

FUNCTION main():

    PRINT "="*80
    PRINT "AICDJ DUAL-HYPOTHESIS ANALYSIS PIPELINE"
    PRINT "="*80

    // Step 1: Initialize Framework
    PRINT "\n[STEP 1] Initializing Framework..."
    framework = initializeFramework()

    // Step 2: Generate Data
    PRINT "\n[STEP 2] Generating and Preparing Data..."
    data = generateData()

    // Step 3: Analyze Mechanisms (Qualitative)
    PRINT "\n[STEP 3] Analyzing Mechanism Hypotheses (M1-M4)..."
    mechanism_results = analyzeMechanismHypotheses()

    // Step 4: Test O1
    PRINT "\n[STEP 4] Testing Outcome Hypothesis O1 (Revenue Differential)..."
    o1_results = testO1_RevenueGrowthDifferential()

    // Step 5: Test O2
    PRINT "\n[STEP 5] Testing Outcome Hypothesis O2 (Margin Improvement)..."
    o2_results = testO2_OperatingMarginImprovement()

    // Step 6: Test O3
    PRINT "\n[STEP 6] Testing Outcome Hypothesis O3 (Structural Break)..."
    o3_results = testO3_StructuralBreak()

    // Step 7: Test O4
    PRINT "\n[STEP 7] Testing Outcome Hypothesis O4 (Dynamics Change)..."
    o4_results = testO4_DynamicsChange()

    // Step 8: Test Kernel GC
    PRINT "\n[STEP 8] Testing Kernel Granger Causality (Nonlinear)..."
    kgc_results = testKernelGrangerCausality()
```

```
    // Step 9: Synthesize Results
    PRINT "\n[STEP 9] Synthesizing Comprehensive Results..."
    all_results = {o1_results, o2_results, o3_results, o4_results, kgc_results}
    synthesizeResults(all_results)

    // Step 10: Report Findings
    PRINT "\n[STEP 10] Reporting Key Statistical Findings..."
    reportKeyFindings(all_results)

    PRINT "\n" + "="*80
    PRINT "ANALYSIS COMPLETE"
    PRINT "="*80

    RETURN all_results

END FUNCTION main()

// Execute pipeline
results = main()


================================================================================
END OF PSEUDOCODE
--------------------------------------------------------------------------------
```