```
/*               */
/* ***************************************************************** */

/* ================================================================= */
/* SECTION 1: LIBRARY SETUP AND SYSTEM CONFIGURATION                 */
/* ================================================================= */

/* Define libraries for production and backup */
LIBNAME arblib "C:\CryptoArb\Data" COMPRESS=YES;
LIBNAME backup "C:\CryptoArb\Backup" COMPRESS=YES;

/* Set system options for optimal performance */
OPTIONS
    FULLSTIMER          /* Display detailed performance metrics */
    MPRINT              /* Show macro execution in log */
    OBS=MAX             /* Process all observations */
    THREADS             /* Enable multi-threading */
    CPUCOUNT=4;         /* Utilize 4 CPU cores */

/* Create a timestamp for data versioning */
%LET run_datetime = %SYSFUNC(DATETIME(), DATETIME20.);
%PUT &=run_datetime;

/* ================================================================= */
/* SECTION 2: EXCHANGE MASTER DATA CREATION                          */
/* ================================================================= */

DATA arblib.exchanges (LABEL="Cryptocurrency Exchange Master Data");
    LENGTH
        ExchangeID      $3
        ExchangeName    $20
        Country         $20
        FeeTier         $10;

    FORMAT
        LaunchDate      DATE9.
        LastUpdated     DATETIME20.;

    INFORMAT
        LaunchDate      DATE9.
        LastUpdated     DATETIME20.;

    INPUT
        ExchangeID     $
        ExchangeName   $
        FeeRate
        Country        $
        LaunchDate     :DATE9.
        HasFiat        $
        FeeTier        $;

    /* Calculate effective fee based on volume tier */
    EffectiveFee = FeeRate;
    IF FeeTier = 'VIP' THEN EffectiveFee = FeeRate * 0.7;
    IF FeeTier = 'PRO' THEN EffectiveFee = FeeRate * 0.85;

    /* Set last updated timestamp */
    LastUpdated = DATETIME();

    /* Add calculated fields */
    Region = UPCASE(Country);
```

```
    ExchangeType = IFC(HasFiat='Y','Fiat','Crypto');

    /* Data validation checks */
    IF missing(ExchangeID) THEN
        PUT "WARNING: Missing ExchangeID for record " _N_;

    DATALINES;
BIN Binance 0.10 Malta 14JUL2017 Y PRO
COI Coinbase 0.50 USA 20JUN2012 Y VIP
KRA Kraken 0.25 USA 28SEP2011 Y STANDARD
FTX FTX 0.20 Bahamas 01MAY2019 Y PRO
BYB Bybit 0.15 Singapore 11MAR2018 N STANDARD
;
RUN;

PROC DATASETS LIBRARY=arblib;
    MODIFY exchanges;
    INDEX CREATE ExchangeID;
    INDEX CREATE Country;
QUIT;

/* ========================================================================== */
/* SECTION 3: CRYPTO ASSETS MASTER DATA                                       */
/* ========================================================================== */

DATA arblib.assets (LABEL="Cryptocurrency Asset Master Data");
    LENGTH
        AssetID       $5
        AssetName     $20
        Blockchain    $15;

    FORMAT
        MarketCap     DOLLAR20.
        CirculatingSupply COMMA20.;

    INFORMAT
        LaunchDate    DATE9.;

    INPUT
        AssetID       $
        AssetName     $
        MarketCap
        CirculatingSupply
        LaunchDate    :DATE9.
        Blockchain    $;

    /* Calculate market dominance */
    TotalMarketCap = 1000000000000; /* Placeholder for total crypto market cap */
    MarketDominance = (MarketCap / TotalMarketCap) * 100;

    /* Age calculation */
    AssetAge = INTCK('YEAR', LaunchDate, TODAY());

    /* Risk classification */
    IF MarketCap > 10000000000 THEN RiskCategory = 'Low';
    ELSE IF MarketCap > 1000000000 THEN RiskCategory = 'Medium';
    ELSE RiskCategory = 'High';

    DATALINES;
BTC   Bitcoin      500000000000  18700000  03JAN2009  Bitcoin
ETH   Ethereum     200000000000  120000000 30JUL2015  Ethereum
SOL   Solana       10000000000   350000000 16MAR2020  Solana
```

```
ADA   Cardano      15000000000   340000000 01OCT2017  Cardano
XRP   XRP          25000000000   480000000 03AUG2013  Ripple
;
RUN;

/* ========================================================================== */
/* SECTION 4: REAL-TIME PRICE FEED PROCESSING                                 */
/* ========================================================================== */

/* Simulate streaming price data */
DATA arblib.pricefeeds (LABEL="Real-Time Cryptocurrency Price Data");
    LENGTH
        ExchangeID    $3
        AssetID       $5;

    FORMAT
        Timestamp     DATETIME20.
        Price         DOLLAR10.2
        Volume        COMMA15.;

    /* Generate synthetic price data */
    DO i = 1 TO 100;
        /* Base values */
        Timestamp = DATETIME() - (100 - i) * 60;
        ExchangeID = SCAN('BIN|COI|KRA|FTX', CEIL(RANUNI(0)*4), '|');
        AssetID = SCAN('BTC|ETH|SOL|ADA|XRP', CEIL(RANUNI(0)*5), '|');

        /* Simulate realistic price movements */
        BasePrice = CHOOSE(
            FIND(AssetID, 'BTC|ETH|SOL|ADA|XRP'),
            30000, 1800, 25, 0.35, 0.50
        );

        /* Add random fluctuations */
        Price = BasePrice * (1 + (RANUNI(0) - 0.5) * 0.02;
        Volume = ROUND(1000000 * RANUNI(0), 100);

        OUTPUT;
    END;

    DROP i BasePrice;
RUN;

/* ========================================================================== */
/* SECTION 5: ARBITRAGE OPPORTUNITY DETECTION                                 */
/* ========================================================================== */

/* Macro for arbitrage calculation */
%MACRO calculate_arbitrage(min_profit=0.5, max_age_minutes=5);

    /* Get latest prices from each exchange */
    PROC SQL;
        CREATE TABLE latest_prices AS
        SELECT
            a.AssetName,
            p.ExchangeID,
            p.AssetID,
            p.Price,
            p.Volume,
            p.Timestamp
        FROM
            arblib.pricefeeds p
```

```
        JOIN
            arblib.assets a ON p.AssetID = a.AssetID
        WHERE
            p.Timestamp >= (SELECT MAX(Timestamp) - &max_age_minutes*60
                            FROM arblib.pricefeeds)
        ORDER BY
            p.AssetID,
            p.ExchangeID;
    QUIT;

    /* Calculate arbitrage opportunities */
    PROC SQL;
        CREATE TABLE arblib.arbitrage_opportunities AS
        SELECT
            buy.AssetName,
            buy.ExchangeID AS BuyExchange,
            sell.ExchangeID AS SellExchange,
            buy.AssetID,
            buy.Price AS BuyPrice,
            sell.Price AS SellPrice,
            buy.Volume AS BuyVolume,
            sell.Volume AS SellVolume,
            (sell.Price - buy.Price) AS GrossProfit,
            ((sell.Price - buy.Price)/buy.Price)*100 AS ProfitPct,
            buy.Timestamp AS PriceTimestamp,
            "&run_datetime" AS AnalysisTime FORMAT=$20.,
            &min_profit AS MinProfitThreshold
        FROM
            latest_prices buy,
            latest_prices sell,
            arblib.exchanges e_buy,
            arblib.exchanges e_sell
        WHERE
            buy.AssetID = sell.AssetID AND
            buy.ExchangeID NE sell.ExchangeID AND
            buy.ExchangeID = e_buy.ExchangeID AND
            sell.ExchangeID = e_sell.ExchangeID AND
            sell.Price > buy.Price*(1 + &min_profit/100) AND
            buy.Timestamp = sell.Timestamp
        ORDER BY
            ProfitPct DESC;
    QUIT;

    /* Calculate net profit after fees */
    DATA arblib.arbitrage_net;
        MERGE
            arblib.arbitrage_opportunities (IN=a)
            arblib.exchanges (RENAME=(ExchangeID=BuyExchange FeeRate=BuyFee))
            arblib.exchanges (RENAME=(ExchangeID=SellExchange FeeRate=SellFee));
        BY BuyExchange SellExchange;

        IF a;
        NetProfit = GrossProfit - (BuyPrice*BuyFee/100) - (SellPrice*SellFee/100);
        NetProfitPct = (NetProfit / BuyPrice) * 100;

        /* Flag high-confidence opportunities */
        IF NetProfitPct >= 1 THEN Confidence = 'High';
        ELSE IF NetProfitPct >= 0.5 THEN Confidence = 'Medium';
        ELSE Confidence = 'Low';
    RUN;

%MEND calculate_arbitrage;
```

```
/* Execute arbitrage calculation */
%calculate_arbitrage(min_profit=0.75, max_age_minutes=10);

/* ========================================================================== */
/* SECTION 6: REPORTING AND OUTPUT                                            */
/* ========================================================================== */

/* Create summary report */
PROC TABULATE DATA=arblib.arbitrage_net;
    CLASS AssetName Confidence;
    VAR NetProfitPct;
    TABLE
        AssetName='Asset',
        Confidence='Confidence Level' ALL,
        NetProfitPct='Net Profit %' * (MEAN MIN MAX);
    TITLE 'Cryptocurrency Arbitrage Opportunity Summary';
RUN;

/* Export results to CSV */
PROC EXPORT DATA=arblib.arbitrage_net
    OUTFILE="C:\CryptoArb\Reports\arbitrage_opportunities_%SYSFUNC(TRANWRD(&run_datetime,:,_)).csv"
    DBMS=CSV REPLACE;
RUN;

/* Create backup */
PROC DATASETS LIBRARY=arblib NOLIST;
    COPY OUT=backup;
    SELECT exchanges assets pricefeeds arbitrage_opportunities arbitrage_net;
QUIT;

/* ========================================================================== */
/* SECTION 7: DATA QUALITY CHECKS                                             */
/* ========================================================================== */

/* Check for data anomalies */
PROC MEANS DATA=arblib.pricefeeds N NMISS MIN MAX MEAN;
    VAR Price Volume;
    CLASS AssetID ExchangeID;
RUN;

/* Verify referential integrity */
PROC SQL;
```