Below is one example software specification that groups similar instructions into a few API functions. In this specification, most instructions require a type of correlation analysis, while others need statistical comparison, trend analysis, or specific significance testing. Each function’s “Used For” section lists the corresponding Workflow/Instruction IDs and a brief description of the related query.

**1. Perform Correlation Analysis API**

**Function Name**: perform\_correlation\_analysis  
**ID**: 7\_7, 24\_7, 49\_6, 56\_6, 57\_8, 58\_5, 59\_7, 67\_8, 69\_6, 75\_6, 2\_7, 8\_6, 30\_6, 34\_8  
**Purpose**:  
This function is designed to quantify the relationship between two metrics by applying various correlation methods (e.g., Pearson, Spearman, or regression analysis). It supports use cases such as analyzing the relationship between yield and test failures, E-test measurement trends versus yield fluctuations, and other similar correlations.

**Signature**:

def perform\_correlation\_analysis(data\_x, data\_y, method='pearson'):

"""

Perform correlation analysis between two datasets using the specified statistical method.

Parameters:

data\_x (list or np.array): The first dataset.

data\_y (list or np.array): The second dataset.

method (str): The correlation method to use (e.g., 'pearson', 'spearman', 'regression').

Returns:

float or dict: The computed correlation coefficient or a summary of the regression analysis.

"""

pass

**Used For**:

* **7\_7**: Correlate yield performance with various test failures.
* **24\_7**: Quantify the relationship between E-test measurement trends and yield fluctuations.
* **49\_6**: Analyze the correlation between test measurement statistics and E-test fluctuations.
* **56\_6**: Determine the relationship between foundry setting changes and yield drops.
* **57\_8**: Evaluate if higher parametric test variance is associated with yield drops.
* **58\_5**: Quantify the relationship between edge E-test metrics and failing die occurrences.
* **59\_7**: Assess the correlation between specific failure patterns and manufacturing anomalies.
* **67\_8**: Evaluate relationships between die failure clusters and process adjustments across lots.
* **69\_6**: Assess the strength of the relationship between yield drops and process change events.
* **75\_6**: Assess the relationship between test bin failure frequency and lot yield.
* **2\_7**: Evaluate the relationship between yield dips and failure metrics in specific test programs.
* **8\_6**: Determine if variations in voltage measurements correlate with soft bin increases.
* **30\_6**: Assess the relationship between E-test site variation and yield drop.
* **34\_8**: Evaluate the temporal correlation between two aligned metrics.

**2. Perform Statistical Comparison API**

**Function Name**: perform\_statistical\_comparison  
**ID**: 17\_6, 51\_6, 62\_6, 31\_7, 4\_7  
**Purpose**:  
This function executes statistical tests (e.g., t-test, ANOVA, or Mann–Whitney U test) to compare differences between two or more groups. It is useful for determining if observed differences (e.g., in lot yield, testers’ yields, or yield distributions) are statistically significant.

**Signature**:

def perform\_statistical\_comparison(group1, group2, test\_type='t-test', alpha=0.05):

"""

Perform statistical comparison between two groups using the specified test.

Parameters:

group1 (list or np.array): Data for the first group.

group2 (list or np.array): Data for the second group.

test\_type (str): The type of statistical test to perform (e.g., 't-test', 'anova', 'mannwhitney').

alpha (float): The significance level for the test.

Returns:

dict: A dictionary containing test statistics, p-value, and a conclusion on significance.

"""

pass

**Used For**:

* **17\_6**: Compare lot yield differences between two groups (e.g., after a change in probe configuration).
* **51\_6**: Determine if differences between testers’ yields are statistically significant.
* **62\_6**: Assess yield differences across testers with comparative statistical tests.
* **31\_7**: Compare yield distributions between two testers to identify consistent underperformance.
* **4\_7**: Determine if low-yield patterns are linked to a specific test house or tester.

**3. Perform Trend Analysis API**

**Function Name**: perform\_trend\_analysis  
**ID**: 19\_5  
**Purpose**:  
This function analyzes time-series data to detect and quantify trends. It can apply techniques such as moving averages or regression analysis, making it ideal for identifying trends in wafer yields over time.

**Signature**:

def perform\_trend\_analysis(time\_series\_data, method='moving\_average'):

"""

Perform trend analysis on time-series data using the specified method.

Parameters:

time\_series\_data (list or np.array): The time-series dataset to be analyzed.

method (str): The trend analysis method (e.g., 'moving\_average', 'regression').

Returns:

dict: A summary of the trend analysis, including trend direction and metrics.

"""

pass

**Used For**:

* **19\_5**: Detect trends in wafer-level yield over the last quarter.

**4. Perform Significance Testing API**

**Function Name**: perform\_significance\_testing  
**ID**: 38\_7  
**Purpose**:  
This function conducts significance testing on paired performance data to determine if process changes (such as retesting the worst wafer) result in statistically significant improvements in yield and reductions in test bin counts.

**Signature**:

def perform\_significance\_testing(before\_test, after\_test, metric='yield'):

"""

Perform significance testing to evaluate the effect of process changes such as retesting.

Parameters:

before\_test (list or np.array): Metrics recorded before the process change.

after\_test (list or np.array): Metrics recorded after the process change.

metric (str): The performance metric to analyze (e.g., 'yield', 'test\_bins').

Returns:

dict: A dictionary with statistical test results including test statistic, p-value, and effect size.

"""

pass

**Used For**:

* **38\_7**: Evaluate if retesting improves lot yield significantly and determine the reduction in test bins for lots with low yield.

This specification outlines a set of API functions with clear names, purposes, and parameter signatures that support the execution of the provided semiconductor chip test data analytics instructions.