Below is a draft software specification that defines a set of Python API functions. Each function is intended to support one or more of the instructions from the semiconductor chip test data analytics domain. You can adjust the details (e.g., parameter names, input types, or internal helper calls) as needed during further design and implementation.

**Function 1: Apply Die Clustering**  
**Function Name**: apply\_die\_clustering  
**ID**: 67\_5, 73\_4  
**Purpose**: Use a spatial clustering algorithm (for example, DBSCAN) to group failing die coordinates and identify clusters on a wafer.  
**Signature**:

def apply\_die\_clustering(die\_coordinates: list, algorithm: str = "DBSCAN") -> dict:

"""

Clusters the given list of die coordinates using a spatial clustering technique.

Parameters:

die\_coordinates (list): A list of coordinates for failing dies.

algorithm (str): The clustering algorithm to use (default is 'DBSCAN').

Returns:

dict: A dictionary mapping cluster identifiers to lists of die coordinates.

"""

**Used For**:

* Workflow 67, Instruction 5: “Apply Clustering Algorithm: Use a spatial clustering technique … on the failing die coordinates …”
* Workflow 73, Instruction 4: “Apply Clustering Algorithm: Use a spatial clustering technique … on the fail die coordinates …”

**Function 2: Detect Failure Patterns**  
**Function Name**: detect\_failure\_patterns  
**ID**: 13\_4  
**Purpose**: Apply image processing and pattern recognition techniques to identify common failure patterns (e.g., grid, edge-ring, or center clusters) on wafer maps.  
**Signature**:

def detect\_failure\_patterns(wafer\_map\_image, techniques: list = ["image\_processing", "pattern\_recognition"]) -> list:

"""

Detects common failure patterns from a wafer map using specified image processing techniques.

Parameters:

wafer\_map\_image: An image or array representing the wafer map.

techniques (list): A list of techniques to apply.

Returns:

list: A list of detected patterns.

"""

**Used For**:

* Workflow 13, Instruction 4: “Visual Pattern Recognition - Apply image processing or pattern recognition techniques …”

**Function 3: Cluster Wafers by Soft Bin Profile**  
**Function Name**: cluster\_wafers\_by\_soft\_bin  
**ID**: 22\_3  
**Purpose**: Group wafers by applying a clustering algorithm (e.g., k-means or hierarchical clustering) based on similar soft bin profiles.  
**Signature**:

def cluster\_wafers\_by\_soft\_bin(soft\_bin\_profiles: list, method: str = "k-means") -> dict:

"""

Clusters wafers based on similar soft bin profiles using a specified clustering method.

Parameters:

soft\_bin\_profiles (list): List of soft bin profile data for wafers.

method (str): The clustering method to use (e.g., 'k-means' or 'hierarchical').

Returns:

dict: Clusters of wafers keyed by cluster labels.

"""

**Used For**:

* Workflow 22, Instruction 3: “Cluster Wafers by Soft Bin Profile …”

**Function 4: Count E-test Measurements**  
**Function Name**: count\_etest\_measurements  
**ID**: 36\_4  
**Purpose**: For each grouped wafer, count the total number of E-test measurement records.  
**Signature**:

def count\_etest\_measurements(grouped\_wafers: dict) -> dict:

"""

Counts the E-test measurement records for each wafer group.

Parameters:

grouped\_wafers (dict): A dictionary where keys are wafer IDs and values are lists of E-test records.

Returns:

dict: A mapping from wafer IDs to the count of E-test measurement records.

"""

**Used For**:

* Workflow 36, Instruction 4: “Count Measurements: For each grouped wafer …”

**Function 5: Count Test Bins**  
**Function Name**: count\_test\_bins  
**ID**: 55\_4  
**Purpose**: Group records by test bin and count the number of occurrences for each bin.  
**Signature**:

def count\_test\_bins(test\_records: list) -> dict:

"""

Groups test records by test bin and counts the occurrences in each bin.

Parameters:

test\_records (list): A list of test record dictionaries including a test bin field.

Returns:

dict: A dictionary with test bin as keys and their counts as values.

"""

**Used For**:

* Workflow 55, Instruction 4: “Count Test Bins - Group the records by test bin …”

**Function 6: Count Tested Dies**  
**Function Name**: count\_tested\_dies  
**ID**: 72\_3  
**Purpose**: Count the number of unique dies tested on a wafer.  
**Signature**:

def count\_tested\_dies(test\_data: list) -> int:

"""

Counts the number of unique dies tested on a wafer.

Parameters:

test\_data (list): A list of test records containing die identifiers.

Returns:

int: The count of unique dies.

"""

**Used For**:

* Workflow 72, Instruction 3: “Count Tested Dies: Process the test data …”

**Function 7: Summarize Die Outcomes**  
**Function Name**: summarize\_die\_outcomes  
**ID**: 60\_4  
**Purpose**: Summarize the outcomes by counting passing and failing dies on the most recent wafer.  
**Signature**:

def summarize\_die\_outcomes(die\_results: list) -> dict:

"""

Summarizes die outcomes by counting how many dies passed versus failed.

Parameters:

die\_results (list): A list of die test outcome records.

Returns:

dict: A summary with counts of 'passing' and 'failing' dies.

"""

**Used For**:

* Workflow 60, Instruction 4: “Summarize Passing vs. Failing Dies …”

**Function 8: Map Failing Regions to Test Programs**  
**Function Name**: map\_failing\_regions\_to\_test\_programs  
**ID**: 3\_4  
**Purpose**: Align spatial data from wafer maps with test program records to identify which test programs are failing in specific regions.  
**Signature**:

def map\_failing\_regions\_to\_test\_programs(wafer\_map: dict, test\_programs: list) -> dict:

"""

Maps regions from the wafer map to corresponding test programs based on spatial information.

Parameters:

wafer\_map (dict): A mapping of die locations and failure regions.

test\_programs (list): A list of test program records.

Returns:

dict: A mapping of regions to associated test programs.

"""

**Used For**:

* Workflow 3, Instruction 4: “Map Failing Regions to Test Programs …”

**Function 9: Map Failures to Regions**  
**Function Name**: map\_failures\_to\_regions  
**ID**: 40\_4  
**Purpose**: Assign each failing die to its corresponding region based on defined coordinate criteria.  
**Signature**:

def map\_failures\_to\_regions(failing\_dies: list, coordinate\_criteria: dict) -> dict:

"""

Maps each failing die to a region based on provided coordinate criteria.

Parameters:

failing\_dies (list): A list of failing die coordinate records.

coordinate\_criteria (dict): Criteria that defines regions on a wafer.

Returns:

dict: A mapping from region identifiers to lists of failing dies.

"""

**Used For**:

* Workflow 40, Instruction 4: “Map Failures to Regions …”

**Function 10: Map E-test Values to Die Locations**  
**Function Name**: map\_etest\_values\_to\_die\_locations  
**ID**: 58\_4  
**Purpose**: Overlay E-test edge site data onto wafer maps to correlate edge measurement shifts with failing die locations.  
**Signature**:

def map\_etest\_values\_to\_die\_locations(etest\_data: dict, wafer\_map: dict) -> dict:

"""

Overlays E-test edge site data on wafer maps to correlate edge metric shifts with failing die locations.

Parameters:

etest\_data (dict): E-test measurement data, particularly from edge sites.

wafer\_map (dict): Spatial mapping of die locations.

Returns:

dict: A mapping showing correlation between edge metrics and failure locations.

"""

**Used For**:

* Workflow 58, Instruction 4: “Map E-test Values to Die Locations …”

**Function 11: Map Test Bins to Metadata**  
**Function Name**: map\_test\_bins\_to\_metadata  
**ID**: 15\_3  
**Purpose**: Cross-reference frequent test bins with metadata to determine common test program or hardware setups.  
**Signature**:

def map\_test\_bins\_to\_metadata(test\_bins: list, metadata: dict) -> dict:

"""

Maps test bins to associated metadata such as test program or hardware setup.

Parameters:

test\_bins (list): A list of test bin identifiers.

metadata (dict): Metadata mapping that includes test program and hardware details.

Returns:

dict: A mapping of test bins to metadata attributes.

"""

**Used For**:

* Workflow 15, Instruction 3: “Map Test Bins to Test Programs/Hardware …”

**Function 12: Detect Recurring Patterns in Wafer Maps**  
**Function Name**: detect\_recurring\_patterns\_in\_wafer\_maps  
**ID**: 25\_4, 37\_5  
**Purpose**: Use image processing and pattern recognition to detect recurring patterns (e.g., edge-ring, grid) in wafer maps.  
**Signature**:

def detect\_recurring\_patterns\_in\_wafer\_maps(wafer\_map\_image, patterns: list = None) -> list:

"""

Detects recurring patterns in wafer maps using image processing techniques.

Parameters:

wafer\_map\_image: An image or matrix representing the wafer map.

patterns (list, optional): Specific patterns to look for (e.g., ['edge-ring', 'grid']).

Returns:

list: A list of detected pattern descriptors.

"""

**Used For**:

* Workflow 25, Instruction 4: “Detect Recurring Patterns …”
* Workflow 37, Instruction 5: “Apply Pattern Detection …”

**Function 13: Detect Checkerboard Patterns**  
**Function Name**: detect\_checkerboard\_patterns  
**ID**: 53\_3  
**Purpose**: Scan wafer maps for checkerboard patterns in the distribution of failing dies using image processing algorithms.  
**Signature**:

def detect\_checkerboard\_patterns(wafer\_map\_image) -> list:

"""

Detects checkerboard patterns in wafer maps.

Parameters:

wafer\_map\_image: An image or data representation of a wafer map.

Returns:

list: A list of detected checkerboard pattern occurrences.

"""

**Used For**:

* Workflow 53, Instruction 3: “Pattern Detection …”

**Function 14: Recognize Wafer Map Patterns**  
**Function Name**: recognize\_wafer\_map\_patterns  
**ID**: 3\_2  
**Purpose**: Automatically detect and classify unusual wafer map patterns using image processing and clustering algorithms.  
**Signature**:

def recognize\_wafer\_map\_patterns(wafer\_map\_image) -> dict:

"""

Recognizes and classifies unusual patterns in a wafer map.

Parameters:

wafer\_map\_image: The input wafer map image or data.

Returns:

dict: A classification of detected patterns.

"""

**Used For**:

* Workflow 3, Instruction 2: “Pattern Recognition Analysis …”

**Function 15: Rank Test Programs**  
**Function Name**: rank\_test\_programs  
**ID**: 10\_5  
**Purpose**: Rank test programs based on their contribution to yield drop by sorting them according to failure metrics.  
**Signature**:

def rank\_test\_programs(test\_programs: list, failure\_metrics: dict) -> list:

"""

Ranks test programs based on their failure contributions.

Parameters:

test\_programs (list): A list of test program identifiers.

failure\_metrics (dict): A mapping of test programs to failure contribution scores.

Returns:

list: A sorted list of test programs, highest impact first.

"""

**Used For**:

* Workflow 10, Instruction 5: “Rank Test Programs …”

**Function 16: Rank Test Measurements**  
**Function Name**: rank\_test\_measurements  
**ID**: 76\_5, 11\_4  
**Purpose**: Sort and rank test measurements based on failure contribution and frequency to determine which ones most frequently lead to failing dies.  
**Signature**:

def rank\_test\_measurements(test\_measurements: list, failure\_counts: dict) -> list:

"""

Ranks test measurements based on failure contribution and frequency.

Parameters:

test\_measurements (list): A list of test measurement identifiers.

failure\_counts (dict): A mapping from test measurement to count of failures.

Returns:

list: Sorted test measurements in descending order of failure impact.

"""

**Used For**:

* Workflow 76, Instruction 5: “Rank Test Measurements …”
* Workflow 11, Instruction 4: “Rank Test Measurements …”

**Function 17: Rank Parametric Measurements**  
**Function Name**: rank\_parametric\_measurements  
**ID**: 32\_6  
**Purpose**: Rank parametric test measurements by their correlation strength or impact on yield variation.  
**Signature**:

def rank\_parametric\_measurements(parametric\_data: dict, correlation\_scores: dict) -> list:

"""

Ranks parametric measurements based on correlation strength or impact on yield variation.

Parameters:

parametric\_data (dict): A dictionary of parametric measurements.

correlation\_scores (dict): A mapping of measurements to correlation or impact scores.

Returns:

list: Sorted measurements in order of significance.

"""

**Used For**:

* Workflow 32, Instruction 6: “Rank Test Measurements …”

**Function 18: Assess Hardware Issues**  
**Function Name**: assess\_hardware\_issues  
**ID**: 2\_8  
**Purpose**: Evaluate if repeated usage of the same equipment chain combined with voltage-related failures indicates a hardware issue.  
**Signature**:

def assess\_hardware\_issues(equipment\_chain: dict, failure\_records: list) -> bool:

"""

Assesses whether repeated equipment chain usage with voltage-related failures suggests a hardware issue.

Parameters:

equipment\_chain (dict): Information about the equipment chain (tester, load board, etc.).

failure\_records (list): A list of failure records.

Returns:

bool: True if a hardware issue is suspected, False otherwise.

"""

**Used For**:

* Workflow 2, Instruction 8: “Assess Hardware Issue Implications …”

**Function 19: Assess Process Impact**  
**Function Name**: assess\_process\_impact  
**ID**: 58\_6  
**Purpose**: Evaluate whether a shift in E-test values significantly contributes to low yield.  
**Signature**:

def assess\_process\_impact(etest\_values: list, yield\_data: list, threshold: float) -> bool:

"""

Evaluates the impact of E-test value shifts on yield.

Parameters:

etest\_values (list): A list of E-test measurement values.

yield\_data (list): Corresponding yield data.

threshold (float): A significance threshold for correlation.

Returns:

bool: True if the impact is significant, False otherwise.

"""

**Used For**:

* Workflow 58, Instruction 6: “Assess Process Impact …”

**Function 20: Assess Statistical Significance**  
**Function Name**: assess\_statistical\_significance  
**ID**: 3\_7  
**Purpose**: Apply statistical tests to determine whether wafer-to-wafer variation is statistically significant.  
**Signature**:

def assess\_statistical\_significance(data: list, test: str = "t-test") -> dict:

"""

Performs statistical significance testing on the provided data.

Parameters:

data (list): Data points across wafers.

test (str): The statistical test to use (default is 't-test').

Returns:

dict: Results of the statistical significance test.

"""

**Used For**:

* Workflow 3, Instruction 7: “Statistical Significance Assessment …”

**Function 21: Interpret Statistical Findings**  
**Function Name**: interpret\_statistical\_findings  
**ID**: 22\_7  
**Purpose**: Interpret statistical analysis to assess whether observed declines are significant and persistent.  
**Signature**:

def interpret\_statistical\_findings(stat\_results: dict) -> str:

"""

Interprets statistical analysis results to determine significance and persistence of declines.

Parameters:

stat\_results (dict): A dictionary of statistical test outcomes.

Returns:

str: An interpretation summary.

"""

**Used For**:

* Workflow 22, Instruction 7: “Interpretation of Findings …”

**Function 22: Validate Consistency**  
**Function Name**: validate\_consistency  
**ID**: 16\_7, 26\_7, 5\_7  
**Purpose**: Check that correlations, measurement impacts, or failure findings are consistent across different wafer batches or lots.  
**Signature**:

def validate\_consistency(data\_groups: list) -> bool:

"""

Validates whether observed patterns or correlations are consistent across data groups.

Parameters:

data\_groups (list): A list of data subsets (e.g., different lots or batches).

Returns:

bool: True if consistency is observed, False otherwise.

"""

**Used For**:

* Workflow 16, Instruction 7: “Validate Consistency …”
* Workflow 26, Instruction 7: “Validate Consistency Across Lots …”
* Workflow 5, Instruction 7: “Validate Findings …”

**Function 23: Evaluate Test Program Status**  
**Function Name**: evaluate\_test\_program\_status  
**ID**: 8\_4  
**Purpose**: Review test program measurement outcomes and assign a pass/fail status based on failure criteria.  
**Signature**:

def evaluate\_test\_program\_status(test\_program\_results: dict) -> dict:

"""

Evaluates each test program and assigns a 'Pass' or 'Fail' status based on measurement outcomes.

Parameters:

test\_program\_results (dict): Mapping of test programs to their measurement outcomes.

Returns:

dict: A mapping of test program IDs to pass/fail statuses.

"""

**Used For**:

* Workflow 8, Instruction 4: “Evaluate Each Test Program …”

**Function 24: Evaluate Die Outcomes**  
**Function Name**: evaluate\_die\_outcomes  
**ID**: 60\_3  
**Purpose**: Determine for each die whether it passed all tests or failed any, based on test limits.  
**Signature**:

def evaluate\_die\_outcomes(die\_tests: dict) -> dict:

"""

Determines the outcome for each die by checking all its test measurements against limits.

Parameters:

die\_tests (dict): A mapping of die identifiers to their test measurement results.

Returns:

dict: A mapping of die IDs to outcome statuses ('passing' or 'failing').

"""

**Used For**:

* Workflow 60, Instruction 3: “Evaluate Die Outcomes …”

**Function 25: Evaluate Software Change Impact**  
**Function Name**: evaluate\_software\_change\_impact  
**ID**: 12\_7  
**Purpose**: Investigate whether recent software changes (e.g., in probing parameters) have affected spatial yield patterns.  
**Signature**:

def evaluate\_software\_change\_impact(software\_logs: list, yield\_patterns: dict) -> dict:

"""

Evaluates the impact of recent software changes on spatial yield patterns.

Parameters:

software\_logs (list): A list of software change logs or update records.

yield\_patterns (dict): Spatial yield pattern data before and after changes.

Returns:

dict: Analysis results indicating potential impacts.

"""

**Used For**:

* Workflow 12, Instruction 7: “Evaluate Software Change Impact …”

**Function 26: Evaluate Retest Impact**  
**Function Name**: evaluate\_retest\_impact  
**ID**: 15\_5  
**Purpose**: Analyze historical retest data or simulate retest scenarios to determine if retesting can improve yield.  
**Signature**:

def evaluate\_retest\_impact(retest\_data: list, simulation: bool = False) -> dict:

"""

Evaluates the impact of retesting on yield improvement.

Parameters:

retest\_data (list): Historical retest records.

simulation (bool): Whether to simulate retest scenarios.

Returns:

dict: An analysis of potential yield improvement from retesting.

"""

**Used For**:

* Workflow 15, Instruction 5: “Evaluate Retest Impact …”

**Function 27: Define Trend Metrics**  
**Function Name**: define\_trend\_metrics  
**ID**: 24\_1  
**Purpose**: Determine the aggregation method (e.g., average, median, standard deviation) for E-test measurements to capture lot-level trends.  
**Signature**:

def define\_trend\_metrics(etest\_measurements: list, method: str = "average") -> dict:

"""

Defines trend metrics for E-test measurements using the specified aggregation method.

Parameters:

etest\_measurements (list): A list of E-test measurements.

method (str): The aggregation method to use.

Returns:

dict: Aggregated trend metrics.

"""

**Used For**:

* Workflow 24, Instruction 1: “Define Trend Metrics …”

**Function 28: Define Analysis Time Period**  
**Function Name**: define\_analysis\_time\_period  
**ID**: 34\_1  
**Purpose**: Specify the temporal boundaries (e.g., weekly, monthly) over which correlations will be evaluated.  
**Signature**:

def define\_analysis\_time\_period(time\_series: list, period: str = "monthly") -> dict:

"""

Defines the analysis time period for correlational studies.

Parameters:

time\_series (list): A time series data set.

period (str): The temporal period to use (e.g., 'weekly', 'monthly').

Returns:

dict: Time period boundaries and segmented data.

"""

**Used For**:

* Workflow 34, Instruction 1: “Define Analysis Time Period …”

**Function 29: Sort Test Bins**  
**Function Name**: sort\_test\_bins  
**ID**: 52\_3  
**Purpose**: Sort the test bins in descending order based on their failure counts.  
**Signature**:

def sort\_test\_bins(test\_bins: dict) -> list:

"""

Sorts test bins based on failure counts in descending order.

Parameters:

test\_bins (dict): A mapping of test bin IDs to failure counts.

Returns:

list: A sorted list of test bin IDs.

"""

**Used For**:

* Workflow 52, Instruction 3: “Sort Test Bins …”

**Function 30: Sort and Select Top Test Bins**  
**Function Name**: sort\_and\_select\_top\_test\_bins  
**ID**: 38\_4  
**Purpose**: Sort test bins by failure counts and select the top three bins.  
**Signature**:

def sort\_and\_select\_top\_test\_bins(test\_bins: dict, top\_n: int = 3) -> list:

"""

Sorts test bins by failure counts and selects the top N bins.

Parameters:

test\_bins (dict): A dictionary of test bins with their failure counts.

top\_n (int): Number of top bins to select (default is 3).

Returns:

list: A list of the top N test bin IDs.

"""

**Used For**:

* Workflow 38, Instruction 4: “Sort and Select Top 3 …”

**Function 31: Perform Statistical Testing on E-test Metrics**  
**Function Name**: perform\_statistical\_testing\_etest  
**ID**: 12\_6  
**Purpose**: Conduct statistical tests (e.g., t-test, Mann–Whitney U test) on site-level E-test metrics to evaluate differences between groups.  
**Signature**:

def perform\_statistical\_testing\_etest(high\_yield: list, low\_yield: list, test: str = "t-test") -> dict:

"""

Performs statistical tests on E-test metrics to assess differences between high-yield and low-yield groups.

Parameters:

high\_yield (list): E-test metrics for high-yield lots.

low\_yield (list): E-test metrics for low-yield lots.

test (str): The statistical test to perform.

Returns:

dict: Results of the statistical test.

"""

**Used For**:

* Workflow 12, Instruction 6: “Statistical Testing …”

**Function 32: Visualize Yield Trends**  
**Function Name**: visualize\_yield\_trends  
**ID**: 33\_7  
**Purpose**: Generate visualizations (e.g., time-series, box plots) to illustrate yield distributions and highlight underperformance by specific test houses.  
**Signature**:

def visualize\_yield\_trends(yield\_data: dict, visualization\_type: str = "time\_series") -> None:

"""

Generates visualizations for yield trends.

Parameters:

yield\_data (dict): Yield data keyed by time or other grouping.

visualization\_type (str): Type of visualization (e.g., 'time\_series', 'box\_plot').

Returns:

None

"""

**Used For**:

* Workflow 33, Instruction 7: “Visualize Trends …”

**Function 33: Visualize Voltage and Soft Bin Trends**  
**Function Name**: visualize\_voltage\_and\_softbin\_trends  
**ID**: 8\_7  
**Purpose**: Create visualizations that overlay voltage measurement trends with soft bin increases over time to reveal potential causations.  
**Signature**:

def visualize\_voltage\_and\_softbin\_trends(voltage\_data: list, softbin\_data: list) -> None:

"""

Creates visualizations to overlay voltage measurement trends with soft bin increases.

Parameters:

voltage\_data (list): Voltage measurement trends.

softbin\_data (list): Soft bin increase data over time.

Returns:

None

"""

**Used For**:

* Workflow 8, Instruction 7: “Visualize Data Trends …”

**Function 34: Corroborate with Additional Metrics**  
**Function Name**: corroborate\_with\_additional\_metrics  
**ID**: 54\_7  
**Purpose**: Cross-check yield data trends with additional parameters (e.g., tester, probe type) to reinforce findings on key test bins and their failure modes.  
**Signature**:

def corroborate\_with\_additional\_metrics(yield\_trends: dict, additional\_params: dict) -> dict:

"""

Cross-checks yield trends with additional parameters to reinforce findings.

Parameters:

yield\_trends (dict): Data trends for yield.

additional\_params (dict): Additional parameters (tester, probe type, etc.).

Returns:

dict: A report summarizing corroborative evidence.

"""

**Used For**:

* Workflow 54, Instruction 7: “Corroborate with Additional Metrics …”

**Function 35: Review Variability Report**  
**Function Name**: review\_variability\_report  
**ID**: 66\_7  
**Purpose**: Confirm that the variability report accurately reflects the data and meets analysis requirements.  
**Signature**:

def review\_variability\_report(report: dict, requirements: dict) -> bool:

"""

Reviews the variability report against analysis requirements.

Parameters:

report (dict): The variability report.

requirements (dict): Analysis requirements to validate.

Returns:

bool: True if the report meets requirements, otherwise False.

"""

**Used For**:

* Workflow 66, Instruction 7: “Review and Validate …”

**Function 36: Flag Low-Yield Lots**  
**Function Name**: flag\_low\_yield\_lots  
**ID**: 31\_3  
**Purpose**: Identify and flag lots that are below the target yield based on specified criteria.  
**Signature**:

def flag\_low\_yield\_lots(lot\_data: list, target\_yield: float) -> list:

"""

Flags lots whose yield is below the target yield.

Parameters:

lot\_data (list): A list of lot yield records.

target\_yield (float): The target yield threshold.

Returns:

list: A list of flagged lot identifiers.

"""

**Used For**:

* Workflow 31, Instruction 3: “Flag Low-Yield Lots …”

**Function 37: Compile Failing Measurements**  
**Function Name**: compile\_failing\_measurements  
**ID**: 11\_4  
**Purpose**: Organize and list the identifiers of test measurements that resulted in failing outcomes.  
**Signature**:

def compile\_failing\_measurements(test\_results: dict) -> list:

"""

Compiles a list of test measurement IDs that have failing outcomes.

Parameters:

test\_results (dict): A mapping of measurement IDs to their pass/fail status.

Returns:

list: A list of measurement IDs with failing outcomes.

"""

**Used For**:

* Workflow 11, Instruction 4: “Compile List of Failing Measurements …”

**Function 38: Perform Statistical Tests on Test Bins**  
**Function Name**: perform\_statistical\_tests\_on\_test\_bins  
**ID**: 78\_6  
**Purpose**: Apply statistical tests (e.g., chi-square, logistic regression) to evaluate the significance of the relationship between test bin failures and yield drops.  
**Signature**:

def perform\_statistical\_tests\_on\_test\_bins(test\_bin\_data: dict, yield\_data: list, test: str = "chi-square") -> dict:

"""

Performs statistical tests on the relationship between test bin failures and yield drops.

Parameters:

test\_bin\_data (dict): Failure counts or distribution data for test bins.

yield\_data (list): Lot-level yield data.

test (str): Statistical test to use.

Returns:

dict: Results of the statistical analysis.

"""

**Used For**:

* Workflow 78, Instruction 6: “Select and Perform Statistical Tests …”

**Function 39: Correlate and Rank Measurements**  
**Function Name**: correlate\_and\_rank\_measurements  
**ID**: 26\_6  
**Purpose**: Perform correlation or regression analysis to determine which test measurements most strongly correlate with lower yields, and rank them by impact.  
**Signature**:

def correlate\_and\_rank\_measurements(measurement\_data: dict, yield\_data: list) -> list:

"""

Correlates test measurements with yield data and ranks them by impact.

Parameters:

measurement\_data (dict): Test measurement values.

yield\_data (list): Yield data corresponding to the measurements.

Returns:

list: A sorted list of measurements ranked by their correlation with yield loss.

"""

**Used For**:

* Workflow 26, Instruction 6: “Statistical Correlation and Ranking …”

**Function 40: Trend Analysis and Correlation**  
**Function Name**: trend\_analysis\_correlation  
**ID**: 27\_6  
**Purpose**: Generate trend line plots for failing die counts and equipment usage, and perform statistical correlation analysis to identify coinciding changes over time.  
**Signature**:

def trend\_analysis\_correlation(die\_counts: list, equipment\_usage: list) -> dict:

"""

Analyzes trends and computes statistical correlations between failing die counts and equipment usage.

Parameters:

die\_counts (list): Time-series data of failing die counts.

equipment\_usage (list): Time-series data of equipment usage metrics.

Returns:

dict: Correlation results and trend analysis.

"""

**Used For**:

* Workflow 27, Instruction 6: “Trend Analysis and Correlation …”

**Function 41: Perform Correlation Analysis**  
**Function Name**: perform\_correlation\_analysis  
**ID**: 16\_8  
**Purpose**: Quantify the relationship between process changes and abnormal test measurement trends using statistical correlation methods.  
**Signature**:

def perform\_correlation\_analysis(process\_changes: list, test\_trends: list, method: str = "pearson") -> dict:

"""

Performs correlation analysis to quantify the relationship between process changes and test measurement trends.

Parameters:

process\_changes (list): A list of process change events.

test\_trends (list): A time-series of test measurement trends.

method (str): Correlation method to use (default 'pearson').

Returns:

dict: Correlation analysis results.

"""

**Used For**:

* Workflow 16, Instruction 8: “Correlation Analysis …”

**Function 42: Simulate Relaxed Limits**  
**Function Name**: simulate\_relaxed\_limits  
**ID**: 11\_5  
**Purpose**: For high-impact test measurements, simulate relaxed parametric limits and recalculate pass/fail outcomes under the new thresholds.  
**Signature**:

def simulate\_relaxed\_limits(test\_measurements: dict, relaxed\_thresholds: dict) -> dict:

"""

Simulates the effect of relaxing parametric limits on test outcomes.

Parameters:

test\_measurements (dict): Original test measurement data with current limits.

relaxed\_thresholds (dict): New relaxed threshold values.

Returns:

dict: Recomputed pass/fail outcomes under relaxed limits.

"""

**Used For**:

* Workflow 11, Instruction 5: “Simulate Relaxed Limits …”

**Function 43: Recognize Grid and Checkerboard Patterns**  
**Function Name**: recognize\_grid\_checkerboard\_patterns  
**ID**: 17\_3  
**Purpose**: Automatically detect grid and checkerboard patterns in wafer maps using image processing techniques.  
**Signature**:

def recognize\_grid\_checkerboard\_patterns(wafer\_map\_image) -> dict:

"""

Detects grid and checkerboard patterns in wafer maps.

Parameters:

wafer\_map\_image: An image or matrix representing the wafer map.

Returns:

dict: Detected pattern details.

"""

**Used For**:

* Workflow 17, Instruction 3: “Pattern Recognition …”

**Function 44: Estimate Yield Recovery**  
**Function Name**: estimate\_yield\_recovery  
**ID**: 11\_6  
**Purpose**: Recompute yield metrics under the simulated relaxed limits scenario to estimate potential yield improvement or recovery.  
**Signature**:

def estimate\_yield\_recovery(original\_yield: float, simulated\_outcomes: dict) -> float:

"""

Estimates potential yield recovery under a simulated relaxed limits scenario.

Parameters:

original\_yield (float): The original yield percentage.

simulated\_outcomes (dict): Recomputed pass/fail outcomes under relaxed limits.

Returns:

float: The estimated yield recovery percentage.

"""

**Used For**:

* Workflow 11, Instruction 6: “Estimate Yield Recovery …”

**Function 45: Visualize Correlation Matrices**  
**Function Name**: visualize\_correlation\_matrices  
**ID**: 20\_7  
**Purpose**: Generate visual correlation matrices and scatter plots to represent relationships between test bin distributions and tester configuration variables.  
**Signature**:

def visualize\_correlation\_matrices(data\_matrix: list, labels: list) -> None:

"""

Generates visual correlation matrices and scatter plots.

Parameters:

data\_matrix (list): A 2D data matrix for correlation analysis.

labels (list): Labels for the variables in the data matrix.

Returns:

None

"""

**Used For**:

* Workflow 20, Instruction 7: “Visualization …”

**Function 46: Visualize Grid Failures**  
**Function Name**: visualize\_grid\_failures  
**ID**: 10\_8  
**Purpose**: Create visual representations (e.g., heat maps, scatter plots) to illustrate the correlation between grid failures and tester-prober configurations.  
**Signature**:

def visualize\_grid\_failures(failure\_data: dict, config\_data: dict) -> None:

"""

Creates visual representations to show the correlation between grid failures and tester-prober configurations.

Parameters:

failure\_data (dict): Data on grid failures.

config\_data (dict): Tester-prober configuration information.

Returns:

None

"""

**Used For**:

* Workflow 10, Instruction 8: “Visualization …”

**Function 47: Group and Compare Yields**  
**Function Name**: group\_and\_compare\_yields  
**ID**: 14\_4  
**Purpose**: Group lot data by tester (e.g., Tester A vs. Tester B) and compute statistical summaries to identify yield differences.  
**Signature**:

def group\_and\_compare\_yields(lot\_data: list, group\_by: str = "tester") -> dict:

"""

Groups lot data by a specified attribute and computes statistical summaries.

Parameters:

lot\_data (list): A list of lot records with yield and tester information.

group\_by (str): The attribute to group by (default 'tester').

Returns:

dict: Statistical summaries (mean, median, std) for each group.

"""

**Used For**:

* Workflow 14, Instruction 4: “Group and Compare Yields …”

**Function 48: Group Failure by Probe Cycle**  
**Function Name**: group\_failure\_by\_probe\_cycle  
**ID**: 25\_5  
**Purpose**: Group failing dies by probe cycle (e.g., early vs. late) and count the number of failures in each category.  
**Signature**:

def group\_failure\_by\_probe\_cycle(failing\_dies: list, probe\_cycle\_info: dict) -> dict:

"""

Groups failing dies by probe cycle and counts failures.

Parameters:

failing\_dies (list): A list of failing die records.

probe\_cycle\_info (dict): Information mapping dies to probe cycles.

Returns:

dict: A dictionary with probe cycle categories and their failure counts.

"""

**Used For**:

* Workflow 25, Instruction 5: “Group and Aggregate Failure Data …”

**Function 49: Group and Analyze Wafers**  
**Function Name**: group\_and\_analyze\_wafers  
**ID**: 26\_4  
**Purpose**: Group wafers by foundry and manufacturing batch, then compute the frequency and statistical significance of center-fail patterns within each group.  
**Signature**:

def group\_and\_analyze\_wafers(wafer\_data: list, group\_by: list = ["foundry", "manufacturing\_batch"]) -> dict:

"""

Groups wafers by specified attributes and analyzes the frequency and significance of center-fail patterns.

Parameters:

wafer\_data (list): A list of wafer records with pattern and batch information.

group\_by (list): Attributes to group by (default ['foundry', 'manufacturing\_batch']).

Returns:

dict: Analysis results for each group.

"""

**Used For**:

* Workflow 26, Instruction 4: “Group and Analyze Data …”

**Function 50: Determine Front-End vs. Back-End Impact**  
**Function Name**: determine\_frontend\_backend\_impact  
**ID**: 6\_6  
**Purpose**: Evaluate whether yield loss is primarily due to front-end (fabrication) or back-end (packaging/scribing) issues based on correlation trends and process characteristics.  
**Signature**:

def determine\_frontend\_backend\_impact(yield\_loss\_data: dict, process\_characteristics: dict) -> str:

"""

Determines if the yield loss is primarily front-end or back-end related.

Parameters:

yield\_loss\_data (dict): Data capturing yield loss and related trends.

process\_characteristics (dict): Known process attributes for front-end and back-end.

Returns:

str: 'front-end' or 'back-end' indicating the likely source of yield loss.

"""

**Used For**:

* Workflow 6, Instruction 6: “Determine Front-End vs. Back-End Impact …”

This specification provides a high-level design for the API functions. Each function’s header (signature and docstring) outlines its purpose, parameters, and return type, and the “Used For” section indicates which instructions it is meant to support. You can now proceed with detailed implementation based on this design.