**Exploratory Data Analysis (EDA)**

*Summary Statistics (from python script 1.2.2)*

Specific sensors are selected for the summary statistics to study the relationship between

A table with numbers and letters

Description automatically generated

Results:

All the sensors have been standardized with a mean close to 0 and a standard deviation of approximately 1.00024.

* For "OunhHslCRwIRilo", it has negative means which indicate that there are potential signed values or different scales. It also shows the mean is approximately to zero, indicating that the data is approximately normally distributed around zero. The IQR for 1.6094 is quite large relative to its median 0.458, suggesting potential outliers or a skewed distribution. Further analysis can be seen from distribution plot.
* For "BmpcKiosIw," the maximum value of 14.32624 stands out significantly compared to a mean of almost 0 and a standard deviation of 1. This large max value indicates the presence of a potential outlier or extreme value far from the other data points.
* For "SwpYipezsdueC," it has the largest range of values of 5.0526 among the variables shown which indicate high variability in its measurements. The minimum value of -4.11842 appears to be significantly lower than the rest of the data and could potentially be an outlier.
* For "ArsbiQzICA", it seems to show unusual percentiles. Given that the 75th percentile and the median are equal and both are the same as the maximum value. It might suggest a significant skew in the data or an issue with the sensor where a number of readings are maxing out at a particular value.
* For "ETcatZBXS", it has a small mean, the large range and IQR imply this sensor is picking up a wide spread of values with potential outliers skewing the distribution. This could signal an irregularity worth investigating further for this particular sensor measurement.

Given these observations, "BmpcKiosIw", "SwpYipezsdueC" and "ETcatZBXS" would be the most interesting sensors to investigate further for anomalies or outliers.

Histograms *(from python script 1.2.3)*

A graph with blue and white bars

Description automatically generated

* The shape of the histogram is difficult to determine precisely due to the limited bins and it also seems to have heavy right tail with some extremely high values on the right side and it has different scales which indicate the negative means. The data also spread out widely which has confirmed the observation that the IQR is quite large relative to the median.

A graph with blue bars

Description automatically generated

* The histogram seems to have slight positive skew where the values tend to cluster toward left side between 0 to 20. The standard deviation also relatively small, which is visually evident in the histogram where most of the data concentrates in a narrow range around zero. This reinforces the notion that the outlier is a significant departure from the rest of the data.

A graph with blue squares

Description automatically generated

* The histogram looks unimodal and approximately normal, though slightly right-skewed. The histogram also confirms the observation that the data has a large range of values which can be seen a wide spread from 125 to 140. This confirms the observation that there is high variability in the measurements. However, the extremely low min value of -4.11 does not appear as an obvious outlier in this histogram.

A graph with blue and black bars

Description automatically generated

* The histogram confirms the observation about unusual percentiles. In a normal distribution, the 75th percentile would be higher than the median. In this case, the entire right side of the histogram is compressed into a single bin which indicate that the 75th percentile, median and maximum value are all the same. This aligns with the observation that the 75th percentile and median are equal to the maximum value. Besides, the histogram supports the possibility of a sensor malfunction. The fact that a large number of readings are concentrated at the maximum value could indicate that the sensor is saturated or limited at that particular value.

A graph with blue bars

Description automatically generated

* The histogram confirms that the data has a small mean where the center of the distribution is slightly shifted to the right. The histogram visually confirms a large range of values. The x-axis spans from 0 to 5800, indicating a wider spread of values compared to the other histograms.

*Boxplot (from python script 1.2.4)*

A diagram of a box plot

Description automatically generated A diagram of a box plot

Description automatically generated A diagram with a blue line

Description automatically generated with medium confidence A blue rectangular object with black lines

Description automatically generated

A diagram of a box plot

Description automatically generated

**"BmpcKiosIw" sensor:** The boxplot confirms the presence of outliers in the data. There is a single data point far above the upper whisker, which is consistent with the observation from the summary statistics that the maximum value stands out significantly compared to the rest of the data.

**"SwpYipezsdueC"** **sensor:** The boxplot confirms the presence of outliers in the data. There is data points below the lower whisker which is consistent with the observation from the summary statistics that the data has a large range and a minimum value significantly lower than the rest of the data.

**"ETcatZBXS" sensor:** The boxplot does not show any outliers, there are no data points beyond the whiskers. This may due to large range is confirmed. The whiskers extend to values much higher than the upper quartile, indicating a wider spread of data points.

Based on the boxplot for each of the sensors, we can say that "BmpcKiosIw" and "SwpYipezsdueC" exist outliers and "ETcatZBXS" do not have any outliers.

Further study of these outliers needed to ensure whether the outliers are just the extreme values or anomalies.

*Correlation* *(from python script 1.2.5)*

A table with numbers and letters

Description automatically generated

*Correlation between sensors*

A diagram of a matrix

Description automatically generated with medium confidence

*Heatmap* *(from python script 1.2.6)*

The correlation results provide insight into how sensor readings are related to one another. A correlation coefficient close to -1 or 1 indicates a strong relationship, with -1 being a perfect negative correlation, 1 a perfect positive correlation and values near 0 suggesting no correlation. Based on the heatmap, when the colors go to darker blue, it represents a stronger negative correlation whereas when the colors go dark red, it represents a stronger positive correlation.

1. **"OunhHslCRwIRilo"** **correlation with:**

" SwpYipezsdueC " sensor: There is a slight negative correlation of -0.16 between these two sensors which represented by light blue cell. This means that as the values from the 'OunhHslCRwIRilo' sensor increases, the values from the 'SwpYipezsdueC' sensor tends to decrease, but the relationship is weak.

"ArsbiQzICA" sensor: This sensor signaling a weak to moderate negative correlation of -0.27 which represented by darker blue cell. There is a slightly more inverse relationship between the sensors as 'OunhHslCRwIRilo' increases, 'ArsbiQzICA' is likely to decrease.

1. **" BmpcKiosIw " correlation with:**

" SwpYipezsdueC " sensor: This would be the darkest blue cells on the heatmap, indicating a moderate negative correlation of -0.47. It is significant enough to imply that these two sensors have an inverse relationship such that when the value of one increases, the other tends to decrease.

"ArsbiQ2ICA " sensor: They have a moderate positive correlation of 0.25 which represented by the white cell. This means that there seems to be a moderate tendency that if 'BmpcKiosIw' values go up, 'ArsbiQzICA' values may also increase.

1. **" SwpYipezsdueC "** **correlation with:**

" ArsbiQzICA "sensor: This is a weak positive correlation of 0.18 which can be represented by light blue color cell. If 'SwpYipezsdueC' values increase, 'ArsbiQzICA' might see a small increase.

" ETcatZBXS " sensor: A weak positive correlation exists for 0.24 and it is white color cell in the heatmap which indicate that when values from 'SwpYipezsdueC' increase, so might values from 'ETcatZBXS' but not strongly.

1. **" ArsbiQzICA " correlation with:**

" ETcatZBXS " sensor: They have weak positive correlation of 0.13 which can be represented by light blue cell. This suggests much independence between the two sensors values with a slight inclination to move together.

Some sensors do not provide any correlation relationship as very close to zero which shows no significance impact on the results of the sensors and they are actually independent.

*Time-series plot (Line graph from python script 1.2.7)*

The following line graphs are plotted using the average sensor readings for each date to see the pattern and trend.

A graph with a line and a dotted line

Description automatically generated

A line graph with numbers and a line

Description automatically generatedA line graph with blue dots

Description automatically generatedA line graph with blue lines

Description automatically generatedA line graph with blue dots

Description automatically generated

A graph of multiple sensors

Description automatically generated

By looking at each of the individual graph for each sensor, they seem do not have much fluctuation within the 4 days. Further looking at the multiple line graph, we can validate the facts that the sensor for "OunhHslCRwIRilo", "BmpcKiosIw", "SwpYipezsdueC" and "ArsbiQzICA" have slight fluctuation which show the manufacturing progress is under control. Most sensor readings show a slight increasing trend over time except "BmpcKiosIw" and "ETcatZBXS". The slightly decrease in sensor "BmpcKiosIw" might due to a parameter that is gradually being reduced during the manufacturing process.

However, the sensor "ETcatZBXS" exhibits the highest values compared to other sensors and it shows a decreasing trend over time. The high values in sensor "ETcatZBXS" might because of different measurement scales or fault. We can also observe that the readings show a sudden increase and decrease trend. This may due to some specific steps in the manufacturing process that have cause a sudden increase and decrease in the parameter.

**Database Design and Implementation**

**Database selection: Google BigQuery**

Google BigQuery is used since we need to deal with time-series based that will recorded values regularly and large datasets with multiple columns in the tool sensor data. To accommodate the sensor data from various tools, we require a database that is not only robust but also provides the agility to handle complex queries and large volumes of data. Most importantly, we need to make sure users able to perform complex queries efficiently under large volumes of data, Google BigQuery is suitable to design the database as it is a fully-managed, serverless data warehouse that enables fast SQL queries using the processing power of Google's infrastructure.

Here are the main reasons why we prefer using Google BigQuery:

* **Scalability:** It is highly scalable which means it can automatically adjust to handle growing amounts of tool sensor data without requiring manual intervention. This feature is particularly valuable considering the exponential growth of sensor-generated data.
* **Speed:** BigQuery offers exceptional processing speed for massive datasets, enabling users to gain real-time insights into tool performance and make data-driven decisions promptly. This speed is crucial for effective tool performance monitoring and maintenance plan evaluation.
* **Integration with Data Analytics and Visualization Tools:** BigQuery integrates with popular data analytics and visualization tools like Google Data Studio, Looker and various third-party tools. This feature enables users to access and analyze data efficiently, allowing for easy EDA and visualization.
* **Data Sharing and Collaboration:** BigQuery fosters collaboration by enabling effortless data insight sharing within and beyond the organization. Users can securely share data and insights with team members, clients and partners which it enables informed decision-making based on shared data knowledge.

**Database design**

A screenshot of a computer

Description automatically generated

Entities and each attribute are identified through a simple Entity Relationship (ER) diagram. Each would have relationships as below:

**One-to-One Relationships:**

WAFER\_ID - Run: Each wafer run may have a unique identifier associated with it.

**One-to-Many Relationships:**

ToolName - Sensor Data: Each tool may produce multiple sets of sensor data over different runs.

Lot\_ID - WAFER\_ID: Each lot may contain multiple wafers, each identified by a unique WAFER\_ID.

Recipe\_ID - Run: Each recipe may be used in multiple runs.

**Many-to-Many Relationships:**

Sensor Data - Run: Multiple sets of sensor data may be associated with each run and each set of sensor data may be associated with multiple runs.

*Initial Setup*

*Step 1:* Log in to Google Cloud Platform and navigate to Google Cloud Console and enable Google BigQuery and BigQuery API to access BigQuery services and resources.

*Step 2:* Create a new project to create a dataset.

*Step 3:* Select create table from dataset created in Step 2 and upload the tool sensor data in ‘csv’ file format. If there is no data to upload, we could have few options to choose to load the data from various sources including Google Cloud Storage, Google Drive or Cloud Storage. It also supports a variety of data formats, including CSV, JSON, Avro and Parquet.

A screenshot of a computer

Description automatically generated

*Step 4:*

**Automatic Schema Detection:** Select auto detect the schema where Google BigQuery able to detect the schema of the data automatically. This means that we do not need to manually specify the schema fields. It is one of the advantages of using Google BigQuery as it analyzes the data and infers the schema based on its structure.

**Automatic Partitioning:** We can whether choose not to partition or partition by day, by hour, by month or by year. However, we choose not to partition the tables for this case since it is more suitable for users to have control with specific requirements.

Here is the schema that Google BigQuery has detected:

A screenshot of a computer

Description automatically generated A screenshot of a checklist

Description automatically generated A screenshot of a checklist

Description automatically generated

A screenshot of a computer

Description automatically generated

*Google BigQuery Execution (User Guide)*

As a user, we are interested in extracting data for data processing pipelines, Exploratory Data Analysis (EDA) and visualizations in the context of semiconductor manufacturing.

**Step 1: Data Extraction**

|  |
| --- |
| SELECT \*  FROM `project.dataset.tool sensor data`  LIMIT 10 |

We would like to check whether the data is correctly extracted or not. We extract all columns (\*) by selecting from the table created previously and limit the data that only 10 rows will be returned.

Here is the breakdown of the source of data:

*project:* This is the name of the project.

*dataset*: This is the name of the specific dataset within the project that contains the table we are interested in.

*tool sensor data:* This is the name of the table you want to extract data from.

**Step 2: Querying & Filtering**

We start to write queries by extracting specific columns with specific conditions based on requirements using querying and filtering methods.

Example 1:

|  |
| --- |
| SELECT \*  FROM `project.dataset.tool sensor data`  WHERE column\_name = 'specific\_name'  ORDER BY time\_column |

This query retrieves time-series data from tool sensors data for a specific wafer run which means that the data is ordered with date from the earliest to the latest date by default using ‘ORDER BY’ clause. This allows users to track sensor readings over time by filtering the column name that we want to filter with specific condition using ‘WHERE’ clause and analyze how the tool performs during the process.

Example 2:

|  |
| --- |
| SELECT    Wafer\_ID,    Lot\_ID,    ToolName,    Time\_column,    sensor\_column  FROM `project.dataset.tool sensor data`  ORDER BY    Wafer\_ID,    Lot\_ID,    Time\_column |

This query allows users to understand the relationship between different columns such that how sensor data will correspond to specific wafers, lots, tools and timestamp.

We would like to retrieve the ‘Wafer\_ID’, ‘Lot\_ID’, ‘ToolName’, ‘Time\_column’ and certain sensor column from the table. The ‘ORDER BY’ clause will then sort the results based on the columns chosen in ascending order specifically from ‘Wafer\_ID’, ‘Lot\_ID’ then ‘Time\_column’.

Example 3:

|  |
| --- |
| SELECT  Time\_column,  ToolName,  AVG(sensor\_column) as avg\_sensor\_column  FROM `project.dataset.tool sensor data`  GROUP BY  ToolName,  Time\_column  ORDER BY  Time\_column |

The purpose of using this query is to analyze and summarize the average sensor readings over time. By grouping the data by tool name and timestamp, it provides insights into how sensor readings vary across different tools and how they change over time. This analysis can help identify trends, patterns or anomalies in the sensor data by plotting line graph.

We will need to calculate the average reading of the sensor for each combination of timestamp and tool name as the ‘GROUP BY’ clause will arrange the average sensor readings for each unique combination of tool name and timestamp.

**Step 3: Data Processing Pipeline**

*Data Transformation:*

|  |
| --- |
| SELECT    Selected\_column\_name,    AVG(Sensor\_column) AS Average\_sensor\_reading,    MIN(Sensor\_column) AS Min\_sensor\_reading,    MAX(Sensor\_column) AS Max\_sensor\_reading  FROM `project.dataset.tool sensor data`  GROUP BY column\_name |

Users can use queries to calculate the average, minimum, maximum values and etc. before creating summary statistics by using aggregations such as ‘AVG’, ‘MIN’ and ‘MAX’. ‘AS’ means to assign the results to a new column name. For example, the average of sensor column is calculated and named it to a new column name ‘Average\_sensor\_reading’ while it would not replace the original column name, it would instead create a new table. The ‘GROUP BY’ clause should always use when the aggregation clause is used.

*Data Enrichment*:

|  |
| --- |
| SELECT      t1.column1, t2.column2  FROM      table1 AS t1  JOIN      table2 AS t2  ON      t1.common\_key = t2.common\_key |

|  |
| --- |
| SELECT \*  FROM table1  UNION ALL  SELECT \*  FROM table2 |

Users can combine different sources using ‘JOIN’ or ‘UNION’ operation. For example, it can join two tables together and combine the values using ‘JOIN’ and ‘UNION’ will combine both data and exclude outlier data.

**Step 4: Exploratory Data Analysis (EDA)**

EDA is a critical step that involves summarizing main characteristics of the data to understand what the data can tell us beyond hypothesis testing.

Now we have a basic understanding on how the aggregation clause works at Step 3, now we are using it to create summary statistics.

*Standardization*

|  |
| --- |
| WITH sensor\_data AS (    SELECT      Sensor\_column1,      Sensor\_column2    FROM `project.dataset.tool sensor data`  ),  statistics AS (    SELECT      AVG(Sensor\_column1) AS avg\_Sensor\_column1,      STDDEV(Sensor\_column1) AS std\_Sensor\_column1,      AVG(Sensor\_column2) AS avg\_Sensor\_column2,      STDDEV(Sensor\_column2) AS std\_Sensor\_column2    FROM sensor\_data  ),  standardized\_data AS (    SELECT      (Sensor\_column1- avg\_Sensor\_column1) / std\_Sensor\_column1 AS z\_ Sensor\_column1,      (Sensor\_column2- avg\_Sensor\_column2) / std\_Sensor\_column2 AS z\_ Sensor\_column2    FROM sensor\_data, statistics  ) |

This query is designed to standardize the sensor data by calculating z-scores for each sensor column which can make a comparison of sensor readings across different scales. By standardizing the data, anomalies, trends and patterns across different sensors can be easily identified.

We use ‘WITH’ clause to create Common Table Expression (CTE) and name it as ‘sensor data’, ‘statistics’ and ‘standardized\_data’ that act as the base data for subsequent calculation.

*‘sensor data’*: consists of the sensor column that need to be examine.

*‘statistics’*: consists of calculation of summary statistics using ‘sensor data’

*‘standardized\_data’*: consists of formula that calculate z-score for each sensor column from ‘sensor data’ and results in ‘statistics’.

|  |
| --- |
| z-score= |

*Summary Statistics*

|  |
| --- |
| SELECT    'min' AS statistic,    MIN(z\_Sensor\_column1) AS Sensor\_column1,    MIN(z\_Sensor\_column2) AS Sensor\_column2  FROM standardized\_data  UNION ALL  SELECT    'max' AS statistic,    MAX(z\_Sensor\_column1) AS Sensor\_column1,    MAX(z\_Sensor\_column2) AS Sensor\_column2  FROM standardized\_data  UNION ALL  SELECT    'median' AS statistic,    APPROX\_QUANTILES(z\_Sensor\_column1, 2)[OFFSET(1)] AS Sensor\_column1,    APPROX\_QUANTILES(z\_Sensor\_column2, 2)[OFFSET(1)] AS Sensor\_column2  FROM standardized\_data  UNION ALL  SELECT    'count' AS statistic,    COUNT(z\_Sensor\_column1) AS Sensor\_column1,    COUNT(z\_Sensor\_column2) AS Sensor\_column2  FROM standardized\_data |

The summary statistics calculated is based on the standardized data, we will obtain the minimum, maximum, median and count for analysis used.

*Correlation*

|  |
| --- |
| SELECT    CORR(Sensor1, Sensor2) AS Correlation\_Coefficient  FROM `project.dataset.tool sensor data` |

Here is an example of a simple query to calculate correlation. Users can calculate the correlation between sensors to determine whether they have positive or negative relationship with each other by inputting ‘CORR’ of any 2 sensors that users would like to explore the relationship and assign a new name as ‘Correlation\_Coefficient’.

**Step 5: Visualization**

Users can now create visualizations based on the results in step 4 to observe the relationships for each sensor. Google BigQuery able to export the results generated to spreadsheets for visualization.

A screenshot of a computer

Description automatically generated

After we have run the queries and results will be generated below the box, then we selected ‘Explore Data’ and choose ‘Explore with Sheets’ and will connect to Google spreadsheet. Users may also choose ‘Explore with Looker Studio’ option or ‘Explore with Python notebook’ to create visualizations. Users are free to choose any of the options to create visualizations.

It is advisable for users to create visualizations such as histograms and heatmap as:

*Histograms:* It shows the distribution of each sensor to determine the patterns.

*Line Graph:* It shows the trend or pattern over time.

*Heatmap:* It shows the correlation relationship between different sensors and the correlation strength is represented by the intensity of the color.

*Histogram*

Before creating the histogram, we would like to calculate the bins and count of values within each bin first such that:

|  |
| --- |
| SELECT    FLOOR(column1 / num\_bins) \* num\_bins AS bin\_column1,    COUNT(\*) AS count\_column1  FROM   `project.dataset.tool sensor data`  GROUP BY    bin\_column1  ORDER BY    bin\_column1 |

Use ‘FLOOR’ to calculate the bin number for each value in column1. It divides each value in column1 by the specified number of bins and take the integer part and then multiplies by the number of bins again. Use ‘COUNT’ to count the number of rows of each bin.

Then use ‘GROUP BY’ to group the data by the calculated bin number by ensuring that rows with similar values falling into the same bin are aggregated together. Lastly, we arrange the bins in ascending order for the ease in interpreting the histogram using ‘ODER BY’.

Creation of histogram using spreadsheet: Run the queries and generate the results and connect to spreadsheet. Then, click the icon ‘Chart’ to create a new sheet and choose the chart type. If ‘Histogram Chart’ not able to select, use ‘Column Chart’ instead. Then drag the ‘bins’ calculated for the specific column to x-axis and drag the ‘count’ to series and select as sum as we want to know the total count for each bin to see the pattern.

L*ine Graph*

|  |
| --- |
| SELECT  Time\_column,  ToolName,  AVG(sensor\_column) as avg\_sensor\_column  FROM `project.dataset.tool sensor data`  GROUP BY  ToolName,  Time\_column  ORDER BY  Time\_column |

The above queries can be obtained from Step 2: Querying & Filtering Example 3. Use the queries to plot a line graph.

Creation of Line Graph using spreadsheet: Run the queries and generate results and connect to spreadsheet. Select ‘Line Chart’, drag the time column extracted to x-axis and the average sensor column that need to be examine as series.

*Heatmap*

Calculate the correlation between selected column to see the relationship. Users can calculate multiple columns to explore more relationship.

|  |
| --- |
| SELECT    CORR(column1, column2) as column1\_column2\_corr,    CORR(column1, column3) as column1\_column3\_corr,    CORR(column2, column3) AS column2\_column3\_corr  FROM `database.tool\_sensor\_data\_cleaned.tool sensor data` |

Creation of heatmap using spreadsheet: Run the queries and generate results and connect to spreadsheet. There is no direct heatmap to select, use pivot table instead. Select ‘Pivot Table’, drag the correlation data to rows. Select the table generated, find format at the top and select ‘conditional formatting’. Go to color scale section and choose a desired color format. The darkest color means the stronger relationship whereas the lightest color means the weaker relationship.

**Testing**

Extract the data to see whether the dataset is functioning or not.

A screenshot of a computer

Description automatically generated

Here comes with a few examples of querying and filtering data based on conditions such as:

Example 1:

We filter the tool name named ‘A’ in ToolName column and order the sequence from earliest date to latest date. As the results shown below, the naming of ‘A’ has been extracted together with other rows that also contain ‘A’ in ToolName.

A screenshot of a computer

Description automatically generated

Example 2:

We select the required columns only such as Wafer\_ID, Lot\_ID, ToolName and new\_timestamp to see the results in sensors such as ‘OunhHslCRwIRilo’ and ‘BmpcKiosIw’ where the data has ordered by the information in Wafer\_ID, Lot\_ID and new\_timestamp. From the results shown below, the Wafer\_ID is listed in ascending order with the date in sequence and sensor columns have generated the readings based on the sequence accordingly.

A screenshot of a computer

Description automatically generated

*Data Processing & Exploratory Data Analysis (EDA):*

Standardization & Summary Statistics:

A screenshot of a computer

Description automatically generated

Histogram:

We use sensor column named ‘OunhHslCRwIRilo’ and ‘BmpcKiosIw’ as example and the number of bins set as 10.

A screenshot of a table

Description automatically generated A screenshot of a table

Description automatically generated A screenshot of a computer

Description automatically generated

A graph with blue bars

Description automatically generated

A graph with a bar

Description automatically generated

Since we have the results shown in Section 3 that using python, the results showed that both histogram using python and Google BigQuery have the same pattern which verified the process.

Line Graph:

We use sensor column named ‘OunhHslCRwIRilo’ as example to plot the line graph.

A screenshot of a computer

Description automatically generated

A graph with a line

Description automatically generated

We can see the line graph generated for sensor column ‘OunhHslCRwIRilo’ is same as the line graph plot using python which verified the process.

Correlation:

We select sensor column named ‘OunhHslCRwIRilo’, ‘BmpcKiosIw’ and ‘SwpYipezsdueC’ as example to explore the correlation and create heatmap using Pivot Table.

A screenshot of a computer program

Description automatically generated

Correlation Heatmap using Pivot Table:

