Project Information:

`Project Statement:`

`Client:`

One of the leading vehicle fuel pump manufacturers. These pumps are used to take fuel as input and push fuel as output at a high velocity. More the velocity, more is the speed at which vehicle will move.

`Business Problem:`

Machines which manufacture the pumps. Unplanned machine downtime which is leading to loss of productivity.

`Business objective:` Minimize unplanned machine downtime.

`Business constraint:` Minimize maintenance cost.

`Business success criteria:` Reduce the unplanned downtime by at least 10%

`Economic success criteria:` Achieve a cost saving of at least $1M"

About This `Report:

`Prepared by :`Muhammad Ali Talha
`Submitted on:` 2023-10-31
`Software Used:` PostgreSQL

EDA ON SQL TO GET INSDES FROM DB

1 : Creating a Databse and Table in SQL and declaring `Data Types

1. Created a new `Database` named it `intern1` .
2. In the `intern` , i created a table with the name of `machine_downtime`.
3. The code for creating the table is :

```
CREATE TABLE (Date (date), Machine_ID (varchar 255),Assembly_Line_No (varchar 255),Hydraulic_Pressure(bar)
(float),Coolant_Pressure (float),Air_System_Pressure(bar) (float),Coolant_Temperature (float),Hydraulic_Oil_Temperature(Â°C)
(float),Spindle_Bearing_Temperature(Â°C) (float),Spindle_Vibration(Âµm) (float),Tool_Vibration(Âµm) (float),Spindle_Speed(RPM)
(float),Voltage(volts) (float),Torque(Nm) (float),Cutting(kN) (float),Downtime (varchar 255)
```

2: Change the column name `Date` formatting in `Excel`.

1. In our CSV file the column contain name `Date` the date format in the CSV was `dd-mm-yyy` .
2. This date format is not accepted in `SQL`.
3. Hence, it was required to reformat it and change it into `yyy-mm-dd`.
4. Save the file after changes being made.

3: Importing data into the Table `machine_downtime`

1. Right-Click on the database and select the option `Import/Export Data.
2. Select the `import` tab on the top. Provided the path of the `CSV` file in the dialogue box, Select the file format `CSV`. and click ok.
3. Import was succesfull.

4 : Checking the count of databse in `SQL` and `EXCEL`

1. After importing data in the sql. It is important to chheck the exact number of rows are imported in SQL from EXCEL.
2. The total number of rows in the `Excel` is `2500`.
3. We use the query in `SQL` to chech that

```
SELECT
    count(*)
FROM
    machine_downtime
```

4. Total number of `columns` in `Excel` is `14` . To check in `sql' we use following:

```
SELECT count(*)
FROM information_schema.columns
WHERE table_name = 'machine_downtime';
```

5: Summary of the data types of the dataset

1. Cheching the datatype of the columns is important.
2. It can be done by using this query:

```
SELECT count(*)
FROM information_schema.columns
WHERE table_name = 'machine_downtime';
```

6 : Checking Duplicate in whole dataset.

1. Duplicates always create issues. It is import to figure them out earlier.
2. The following query will help in getting the duplicate rows from the dataset.
3. It turned out there is no duplicates ass we run the `Distinct` query.

```
SELECT
   DISTINCT COUNT (*)
FROM
   machine_downtime
```

7: Checking data by `year`, `month` , and `days`.

1. Check how many `years` of data we have :

```
SELECT
    DISTINCT COUNT (*)
FROM
    machine_downtime
```

It is 2 years

2. Check how many `Months` of data we have :

```
SELECT COUNT(DISTINCT EXTRACT(MONTH FROM Date)) AS total_months
FROM machine_downtime;
```

```
FROM (
    SELECT EXTRACT(YEAR FROM Date) AS year,
           COUNT(DISTINCT EXTRACT(MONTH FROM Date)) AS months_in_year
    FROM machine_downtime
    GROUP BY EXTRACT(YEAR FROM Date)
) AS subquery;
```

Its turned out same.

3. Calculating the total months across all years:

```
SELECT SUM(months_in_year) AS total_months
FROM (
    SELECT EXTRACT(YEAR FROM Date) AS year,
           COUNT(DISTINCT EXTRACT(MONTH FROM Date)) AS months_in_year
    FROM machine_downtime
    GROUP BY EXTRACT(YEAR FROM Date)
) AS subquery;
```

8: Statistical analysis for all the column which are `Numerical

- 1 . Min Value
- 2 . Max Value
- 3 . Average /Mean
- 4 . Sum
- 5 . Standard Deviation
- 6 . Variance
- 7 . Median
- 8 . Mode `

8.1 Column Analysis for `Hydrolic Pressure`

```
SELECT
    COUNT(*) AS count,
    MIN(Hydraulic_Pressure) AS min_Hydraulic_Pressure,
    MAX(Hydraulic_Pressure) AS max_Hydraulic_Pressure,
    AVG(Hydraulic_Pressure) AS avg_Hydraulic_Pressure,
    SUM(Hydraulic_Pressure) AS sum_Hydraulic_Pressure,
    STDDEV(Hydraulic_Pressure) AS stddev_Hydraulic_Pressure,
    VARIANCE(Hydraulic_Pressure) AS variance_Hydraulic_Pressure,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Hydraulic_Pressure) AS median_Hydraulic_Pressure,
    MODE() WITHIN GROUP (ORDER BY Hydraulic_Pressure) AS mode_Hydraulic_Pressure
FROM machine_downtime;
```

"count"	"min_hydraulic_pressure"	"max_hydraulic_pressure"	"avg_hydraulic_pressure"	"sum_hydraulic_pressure"	"stddev_hydraulic_pressure"	"va
2500	-14.32645418	191	101.40908377755412	252508.61860610975	30.28930063257439	917

> The negative minimum hydraulic pressure value is unusual and should be further investigated. It may be necessary to identify the root cause of this anomaly, such as sensor calibration issues or system malfunctions.

> The maximum hydraulic pressure value of 191 bar represents the upper extreme of system operation. It's essential to ensure that the system can safely handle and manage such high-pressure conditions.

> The standard deviation and variance values indicate substantial variability in hydraulic pressure data. This suggests that the system may operate under a wide range of conditions. Investigate the causes of this variability and take corrective actions if needed.

> The right-skewed distribution, as indicated by the median and average, suggests that the system frequently operates at lower pressure levels but occasionally experiences higher-pressure spikes.

> Monitoring the mode value of 88.28 bar can help identify the most common operational pressure level and enable adjustments if necessary.

Outliers of `hydrolic pressure` values .

```
SELECT
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Hydraulic_Pressure) AS q1,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Hydraulic_Pressure) AS median,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Hydraulic_Pressure) AS q3,
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Hydraulic_Pressure) - 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Hydraulic_Pressure) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Hydraulic_Pressure)) AS lower_bound,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Hydraulic_Pressure) + 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Hydraulic_Pressure) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Hydraulic_Pressure)) AS upper_bound
FROM machine_downtime;
```

"q1"	"median"	"q3"	"lower_bound"	"upper_bound"
76.355	96.7634864	126.41572687499999	1.263909687500032	201.50681718749996

- Q1 (25th Percentile): 76.35
- Median (50th Percentile): 96.76
- Q3 (75th Percentile): 126.42
- Lower Bound for Outliers: 1.26
- Upper Bound for Outliers: 201.51
  These statistics represent the quartiles and bounds for identifying potential outliers in the hydraulic pressure data. Data points below 1.26 or above 201.51 can be considered outliers. The median value of 96.76 provides a measure of the central tendency of the data, and quartiles help assess the data distribution.

Detecting the outlier

```
SELECT Hydraulic_Pressure
FROM machine_downtime
WHERE Hydraulic_Pressure < 1.26 OR Hydraulic_Pressure > 201.51;
```

"hydraulic_pressure" -14.32645418

8.2: EDA analysis for the column name 'coolant_pressure

```
SELECT
    COUNT(*) AS count,
    MIN(Coolant_Pressure) AS min_Coolant_Pressure,
    MAX(Coolant_Pressure) AS max_Coolant_Pressure,
    AVG(Coolant_Pressure) AS avg_Coolant_Pressure,
    SUM(Coolant_Pressure) AS sum_Coolant_Pressure,
    STDDEV(Coolant_Pressure) AS stddev_Coolant_Pressure,
    VARIANCE(Coolant_Pressure) AS variance_Coolant_Pressure,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Coolant_Pressure) AS median_Coolant_Pressure,
    MODE() WITHIN GROUP (ORDER BY Coolant_Pressure) AS mode_Coolant_Pressure
FROM machine_downtime;
```

"count"	"min_coolant_pressure"	"max_coolant_pressure"	"avg_coolant_pressure"	"sum_coolant_pressure"	"stddev_coolant_pressure"	"variance_co
2500	0.325	11.35	4.94705847351793	12273.652072797984	0.9973571809758427	0.994721346

> The statistics reveal a wide range of pressure values within the dataset. The minimum recorded coolant pressure, at 0.325 bar, suggests that the system occasionally operates at relatively low pressure levels. This could be indicative of specific conditions or instances where lower pressures are required. On the other end of the spectrum, the maximum coolant pressure recorded is 11.35 bar, representing the upper limit of the system's pressure capacity. This maximum value offers insights into the system's ability to handle high-pressure conditions, potentially during peak operational demands.

> The average coolant pressure, calculated at approximately 4.95 bar, serves as a vital indicator of the system's typical operating pressure. This value offers an understanding of the central tendency within the dataset. Alongside the average, the median coolant pressure, approximately 4.94 bar, further emphasizes the symmetrical distribution of the data.

> Further insights into the data's dispersion are provided through the standard deviation and variance calculations. The standard deviation, at approximately 0.997, suggests some degree of variability in the coolant pressure data. This variability may arise from different operating conditions or system responses. The variance, approximately 0.995, provides a measure of how spread out the data is, representing the square of the standard deviation.

> The mode of the coolant pressure data, approximately 4.57 bar, signifies the most frequently occurring pressure level within the dataset. Understanding the mode can help identify typical operational states and potential areas of focus for system optimization.

The IQR data

```
SELECT
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Pressure) AS q1,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Coolant_Pressure) AS median,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Coolant_Pressure) AS q3,
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Pressure) - 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Coolant_Pressure) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Pressure)) AS lower_bound,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Coolant_Pressure) + 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Coolant_Pressure) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Pressure)) AS upper_bound
FROM machine_downtime;
```

"q1"	"median"	"q3"	"lower_bound"	"upper_bound"
4.462463542	4.939960216	5.518401201	2.8785570535000007	7.102307689499999

> With the calculated lower and upper bounds for outliers, at 2.88 and 7.10, respectively, we can effectively identify data points that fall outside the expected range. Outliers, in this context, are those data points that either significantly deviate from the central data distribution by being too low or too high.

> Handling outliers is crucial for maintaining data integrity and making informed decisions. Investigating these unusual data points can lead to valuable insights about the system's behavior or potential data quality issues.

The outlier distribution

Coolant Pressure
11.35
11.3
0.325

> These data points have values either significantly higher (11.35 and 11.3) or significantly lower (0.325) than the central data distribution of the coolant pressure. It's important to further investigate these outliers to determine the underlying causes. These outliers may represent exceptional conditions, measurement errors, or other unique events that require special attention in your analysis or operations.

Column EDA analysis for `Air_System_Pressure` analysis

```
SELECT
    COUNT(*) AS count,
    MIN(Air_System_Pressure) AS min_Air_System_Pressure,
    MAX(Air_System_Pressure) AS max_Air_System_Pressure,
    AVG(Air_System_Pressure) AS avg_Air_System_Pressure,
    SUM(Air_System_Pressure) AS sum_Air_System_Pressure,
    STDDEV(Air_System_Pressure) AS stddev_Air_System_Pressure,
    VARIANCE(Air_System_Pressure) AS variance_Air_System_Pressure,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Air_System_Pressure) AS median_Air_System_Pressure,
    MODE() WITHIN GROUP (ORDER BY Air_System_Pressure) AS mode_Air_System_Pressure
FROM machine_downtime;
```

> he dataset contains 2,500 observations of air system pressure measurements, offering insights into the operational behavior of the system. The statistics computed for the "Air_System_Pressure" data reveal key characteristics. The minimum air system pressure recorded is 5.063 bar, indicating the lower end of pressure levels within the system. On the other hand, the maximum air system pressure observed is 7.974 bar, showcasing the system's ability to handle higher-pressure conditions when necessary. The average air system pressure, approximately 6.499 bar, serves as a central indicator of the typical operating pressure, emphasizing the stability of the system's performance. The calculated standard deviation, around 0.407, provides insight into the data's variability, which suggests a relatively consistent operating range. The variance, approximately 0.166, highlights the data's spread. The median air system pressure, at approximately 6.505 bar, aligns closely with the average, indicating a symmetric distribution. The mode, around 5.629 bar, represents the most frequently occurring air system pressure level within the dataset.

8.1: Calculate the iqr data

```
SELECT
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Air_System_Pressure) AS q1,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Air_System_Pressure) AS median,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Air_System_Pressure) AS q3,
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Air_System_Pressure) - 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Air_System_Pressure) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Air_System_Pressure)) AS lower_bound,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Air_System_Pressure) + 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Air_System_Pressure) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Air_System_Pressure)) AS upper_bound
FROM machine_downtime;
```

"q1"	"median"	"q3"	"lower_bound"	"upper_bound"
6.2179865485	6.5051914	6.780550364	5.3741408252500005	7.62439608725

> The first quartile (Q1) is calculated at approximately 6.218, representing the 25th percentile of the data. This indicates the lower range of air system pressure levels, defining the bottom 25% of observations. The median, approximately 6.505, serves as a measure of central tendency, reflecting the point where 50% of the data falls below and 50% above. Finally, the third quartile (Q3), approximately 6.781, represents the 75th percentile and signifies the upper range of air system pressure levels, encompassing the top 25% of observations.

> To identify potential outliers, the Interquartile Range (IQR) method was applied. The calculated lower bound for outliers is 5.374, and the upper bound is 7.624. Data points falling below 5.374 or exceeding 7.624 are considered potential outliers, indicating air system pressure values that significantly deviate from the central distribution.

8.3: The outliers values are

SELECT Air_System_Pressure FROM machine_downtime WHERE Air_System_Pressure < 5.374 OR Air_System_Pressure > 7.624;

air_system_pressure
5.091411159
5.063480035
7.971606819
7.804750131
5.366874865
7.712440729
7.973991528
7.781150882

air_system_pressure
5.329776651
5.151876018
5.305505788
7.644785584
5.283833449

> The "air_system_pressure" data consists of thirteen measurements taken over a certain period, with values ranging from approximately 5.06 to 7.97. These readings likely pertain to air pressure within a system or environment, and their variations may be indicative of changes or fluctuations within that system, which could be of interest for monitoring and analysis.

Column EDA analysis for `Coolant_Temperature` analysis

```
SELECT
    COUNT(*) AS count,
    MIN(Coolant_Temperature) AS min_Coolant_Temperature,
    MAX(Coolant_Temperature) AS max_Coolant_Temperature,
    AVG(Coolant_Temperature) AS avg_Coolant_Temperature,
    SUM(Coolant_Temperature) AS sum_Coolant_Temperature,
    STDDEV(Coolant_Pressure) AS stddev_Coolant_Temperature,
    VARIANCE(Coolant_Pressure) AS variance_Coolant_Temperature,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Coolant_Temperature) AS median_Coolant_Temperature,
    MODE() WITHIN GROUP (ORDER BY Coolant_Temperature) AS mode_Coolant_Temperature
FROM machine_downtime;
```

output: "count"	"min_coolant_temperature"	"max_coolant_temperature"	"avg_coolant_temperature"	"sum_coolant_temperature"	"stddev_coolant_tempera
2500	4.1	98.2	18.559887459807154	46177.0000000002	0.9973571809758427

> The statistical analysis of the "Coolant_Temperature" data provides essential insights into the temperature characteristics. With 2,500 observations, the dataset reveals a wide temperature range. The minimum recorded temperature is 4.1°C, indicating the lower extreme of the operating conditions. In contrast, the maximum temperature observed is 98.2°C, signifying the upper limit of the system's temperature capacity. The average coolant temperature, approximately 18.56°C, represents the central tendency of the data, highlighting the typical temperature range for the system. The standard deviation, about 0.997, indicates the data's variability, suggesting fluctuations in the operating temperature. The variance, approximately 0.995, reflects the spread of the data. The median temperature, at around 21.2°C, aligns closely with the average, indicating a symmetric distribution. The mode, approximately 26.4°C, represents the most frequently occurring temperature within the dataset.

The IQR data

```
SELECT
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Temperature) AS q1,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Coolant_Temperature) AS median,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Coolant_Temperature) AS q3,
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Temperature) - 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Coolant_Temperature) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Temperature)) AS lower_bound,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY Coolant_Temperature) + 1.5 * (PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
Coolant_Temperature) - PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY Coolant_Temperature)) AS upper_bound
FROM machine_downtime;
```

> The quartile analysis of the "Coolant_Temperature" data highlights key temperature characteristics. The first quartile (Q1) is calculated at approximately 10.4°C, signifying the 25th percentile of the data. This represents the lower range of temperature values, covering the bottom 25% of observations. The median, approximately 21.2°C, serves as a central measure, representing the point where 50% of the data falls below and 50% above. The third quartile (Q3), at about 25.6°C, represents the 75th percentile and captures the upper range of temperature values, encompassing the top 25% of observations.

> To identify potential outliers, the Interquartile Range (IQR) method was applied. The lower bound for outliers is -12.4°C, and the upper bound is 48.4°C. Data points falling below -12.4°C or exceeding 48.4°C are considered potential outliers, indicating temperature values significantly deviating from the central distribution.

Detecting outliers

```
SELECT Coolant_Temperature
FROM machine_downtime
WHERE Coolant_Temperature < -12.4 OR Coolant_Temperature > 48.4;
```

"coolant_temperature"

98.2

> The value of 98.2°C for "Coolant_Temperature" stands out as a significant outlier, significantly exceeding the upper bound for potential outliers calculated through the Interquartile Range (IQR) method. This exceptional temperature reading may indicate a unique event, measurement error, or a condition requiring special attention. Investigating and addressing such outliers is vital for maintaining data integrity and ensuring accurate system analysis and decision-making.

9: EDA on the catagorical or text columns

1. We have 3 column which are varchar datatype.

Frequency distribution in "Machine_ID" column

```
SELECT machine_id, COUNT(*) AS Frequency
FROM machine_downtime
GROUP BY machine_id
ORDER BY Frequency DESC;
```

```
--Unique Values in "machine_id" column

SELECT COUNT(DISTINCT machine_id) AS unique_category_count
FROM machine_downtime;
```

```
-- Mode in the "machine_id" column

SELECT MODE() WITHIN GROUP (ORDER BY machine_id) AS mode_category
FROM machine_downtime;
```

```
--Checking NULL VALUES in "machine_id"

SELECT COUNT(CASE WHEN machine_id IS NULL THEN 1 ELSE NULL END) AS missing_category_count
FROM machine_downtime;
```

10: Running differend EDA combination to get insides.

1. Try to found number of downtime , machine id and dates when ;machine failure happens

```
SELECT   date, machine_id, count(downtime)
FROM
machine_downtime
WHERE downtime = 'Machine_Failure'
GROUP BY 1, 2
ORDER BY 1, 3 DESC
```

2.Finding on which t=year which machine has failure count the number of its downtime

```
SELECT
    EXTRACT (YEAR from DATE), machine_id, count(downtime)
FROM
    machine_downtime
WHERE
    downtime = 'Machine_Failure'
GROUP BY 1, 2
ORDER BY 1, 3 DESC
```

3. The dates and machine_id when machine failed at the lowest "Hydraulic Pressure"

```
SELECT
date, machine_id, MIN (Hydraulic_Pressure) AS Min_Hyd_Presure
FROM
```

```
machine_downtime
WHERE
    downtime = 'Machine_Failure'
GROUP BY
    1, machine_id
```

4. The dates and machine_id when machine failed at the Highest "Hydraulic Pressure"

```
SELECT
date, machine_id, MAX (Hydraulic_Pressure) AS Max_Hyd_Presure
FROM
machine_downtime
WHERE
    downtime = 'Machine_Failure'
GROUP BY
    1, machine_id
LIMIT 10
```