**SQL EDA DOCUMENT**

**Created a database named samliver.** Create database samliver; use samliver;

**Created a table feature-**

 create table feature(N int,LTx\_date date,reLTx\_date date,   Date\_of\_enrollment\_to\_the\_study date, Gender int,Donar\_Age int,BMI double,Diabetes int, Arterial\_hypertension int,Posreperfusion\_biopsy\_steatosis double,Posreperfusion\_biopsy\_steatosis2 double,MRS double,Creatinin double,MDRD\_GFR int, WBC double,Erytrocyte double,Hemoglobin int,Trombo int,CRP double,

Steatosis int, Inflammation int,Balooning int,Fibrosis int,NASH int, NAS int, Age double, Sex char(10), Diagnosis int, Time\_from\_LTx int,BMI\_1 double, Waist\_circumference int, Hypertension int,Diabetes\_2 int, FK int,CyA int,MMF int,AZA int,Sirolimus int,Everolimus int,Prednison int,Medrol int,Statins int,bilirubin double,AST double, ALT double,glycaemia\_fasting double, HbA1c int,C\_peptid double,Insulinemia\_fasting double,HOMA\_IR int,QUICKI double,TAG double,total\_cholesterol double,LDL double, HDL double, GraftfailureTarget char(10));

**Metadata of the table** is retrieved using,

Describe sample\_liverdata;

**Shape of the data**

**column-56**

select count(\*) as numberofcolumns from information\_schema.columns where table\_name = 'sample\_liverdata';

By using select query we consider all columns from information\_schema.columns as numberofcolumns in a table which we created.

**row-108**

select count(\*) as numberofrows from sample\_liverdata;

All the rows in a column are counted from the table by using select query.

Note: 8 rows are missed while importing in mysql but that will not affect the analysis as they represent the target(no) which is similar to the108  rows present. In simple words they are the duplicate rows.

**Distinct rows,**

select distinct \* from sample\_liverdata;

To retrieve all the distinct rows from the table without any duplications.

**Missing values**

**Method1**

 SELECT Posreperfusion\_biopsy\_steatosis, COUNT(\*) AS missing\_count

FROM sample\_liverdata

WHERE Posreperfusion\_biopsy\_steatosis IS NULL OR Posreperfusion\_biopsy\_steatosis= 'N/A' OR N = 'N/A'''

GROUP BY Posreperfusion\_biopsy\_steatosis;

Using select query to select the column and count of all missing occurrences is given as missing\_count from the table using from clause, where clause filters the rows by the condition mentioned above and is grouped by column name.

**Method2**

SELECT

    COUNT(\*) AS total\_rows,

    COUNT(NULLIF(reLTx\_date, '')) AS non\_missing\_count\_reLTx\_date,

    COUNT(\*) - COUNT(NULLIF(reLTx\_date, '')) AS missing\_count\_reLTx\_date,

    COUNT(NULLIF(Posreperfusion\_biopsy\_steatosis, '')) AS non\_missing\_count\_Posreperfusion\_biopsy\_steatosis,

    SUM(CASE WHEN Posreperfusion\_biopsy\_steatosis = '' OR Posreperfusion\_biopsy\_steatosis = 'N/A' THEN 1 ELSE 0 END) AS missing\_count\_Posreperfusion\_biopsy\_steatosis,

    COUNT(NULLIF(Posreperfusion\_biopsy\_steatosis2, '')) AS non\_missing\_count\_Posreperfusion\_biopsy\_steatosis2,

    SUM(CASE WHEN Posreperfusion\_biopsy\_steatosis2 = '' OR Posreperfusion\_biopsy\_steatosis2 = 'N/A' THEN 1 ELSE 0 END) AS missing\_count\_Posreperfusion\_biopsy\_steatosis2,

From sample\_liverdata;

Nullif function is used to count the missing values, and the non-missing values are subtracted from the total number of rows to get the number of missing values.

Like if else conditional statement case and as is used to give missing values as 1 and non-missing as 0. And this is executed for all the columns.

**Posreperfusion\_biopsy\_steatosis2 - this column has range of values with percentage symbol.**

  UPDATE sample\_liverdata

SET Posreperfusion\_biopsy\_steatosis2 = REPLACE(Posreperfusion\_biopsy\_steatosis, '%', '') \* 0.01

WHERE Posreperfusion\_biopsy\_steatosis2 = '5%';

We are using dml command update to the table and setting the values of the column by replacing % by null and multiplying it with 0.01 for decimal representation.

UPDATE sample\_liverdata

SET Posreperfusion\_biopsy\_steatosis2  = 0.01

WHERE Posreperfusion\_biopsy\_steatosis2  = 0.05;

Now, the decimal value is replaced with the actual value by the use of set and where clause.

**Missing value imputation by mean**

UPDATE sample\_liverdata

SET CRP = (

    SELECT mean\_value

    FROM (

        SELECT AVG(CRP) AS mean\_value

        FROM sample\_liverdata

        WHERE CRP IS NOT NULL

    ) AS subquery

)

Updating the table by setting value of crp by selecting the mean\_value from,now using sub-query again using select average of crp as an alias mean value from table and using where clause, where the values of crp not null is considered for calculating average and finally the missing value is imputed using mean.

This method is followed for all the columns which has missing values.

**Identifying the outlier values**

SELECT

  'Arterial\_hypertension' AS column\_name,

  AVG(Arterial\_hypertension) - 1.5 \* STDDEV(Arterial\_hypertension) AS lower\_threshold,

  AVG(Arterial\_hypertension) + 1.5 \* STDDEV(Arterial\_hypertension) AS upper\_threshold

FROM sample\_liverdata;

This code is used to calculate the lower and upper threshold values for each column.

**Finding the outliers,**

SELECT

    `BMI\_[0]`,

    CASE

        WHEN `BMI\_[0]`> 37.12551653 OR `BMI\_[0]` < 17.87481247 THEN 'Yes'

        ELSE 'No'

    END AS has\_outliers

FROM sample\_liverdata;

This code is used to check what all values are are outliers.

Note:

We can replace the outlier value with lower and upper values as we did for missing values.

**EDA**

**MEAN,MEDIAN,STD,VARIANCE,SKEWNESS,KURTOSIS**

 SELECT 'Gender' AS column\_name,

    AVG(Gender) AS mean,

    STDDEV(Gender) AS standard\_deviation,

    (

        SUM(POWER(Gender - (SELECT AVG(Gender) FROM sample\_liverdata), 3)) / COUNT(Gender)

    ) / POWER(STDDEV(Gender), 3) AS skewness,

    (

        SUM(POWER(Gender - (SELECT AVG(Gender) FROM sample\_liverdata), 4)) / COUNT(Gender)

    ) / POWER(STDDEV(Gender), 4) - 3 AS kurtosis

FROM sample\_liverdata;

By using this code we obtain 1st,2nd,3rd and 4th moment business decision.

**Mode**

SELECT

    (

        SELECT BMI

        FROM sample\_liverdata

        GROUP BY BMI

        ORDER BY COUNT(\*) DESC

        LIMIT 1

    )

This code select bmi column from the table and group by BMI and orders the value in descending order and limit by 1 to get the first value which can be performed for all the columns.