**Medical inventory optimization**

**Exploratory Data Analysis (SQL) by Pratiksha Saheb**

**Software: MySQL Workbench**

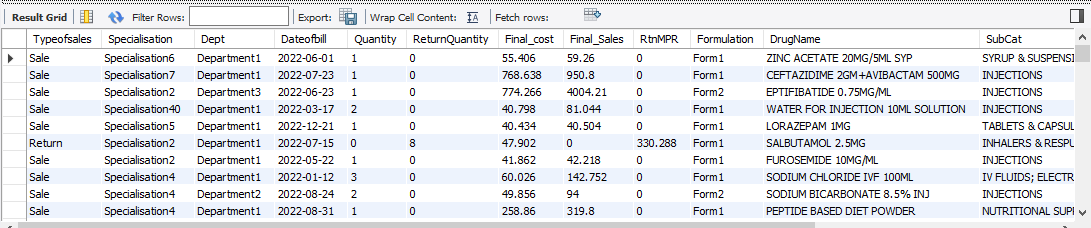
**Business decisions based on the unclean ‘medicines\_db’ data**

**1. Set the current database to "medicines\_db".**

use medicines\_db;

**2. Displaying the Table**

select \* from medicine\_detail limit 10;



**3. Calculating the first moment (measures of central tendency such as mean, median, mode) for the dataset.**

**Mean:**

select

round(avg(Quantity),2) as mean\_Quantity,

round(avg(ReturnQuantity),2) as mean\_ReturnQuantity,

round(avg(Final\_cost),2) as mean\_Final\_cost,

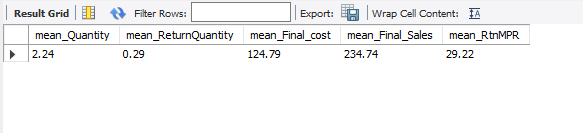
round(avg(Final\_Sales),2) as mean\_Final\_Sales,

round(avg(RtnMPR),2) as mean\_RtnMPR

from

medicine\_detail;

**Output:**



**Median:**

with ranked as

(

select Final\_cost,

Final\_Sales,

Quantity,

ReturnQuantity,

RtnMPR,

row\_number() over (order by Final\_cost) as r,

count(\*) over () as c

from medicine\_detail),

median as

(

select Final\_cost,

Final\_Sales,

Quantity,

ReturnQuantity,

RtnMPR

from ranked

where r in (floor((c+1)/2),ceiling((c+1)/2))

)

select round(avg(Final\_cost), 2),

round(avg(Final\_Sales),2),

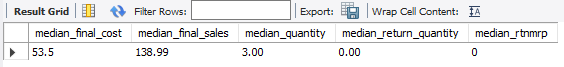
round(avg(Quantity),2),

round(avg(ReturnQuantity),2),

round(avg(RtnMPR),2)

from median;

**Output:**



**Mode:**

select

mode\_Quantity.mode\_value as mode\_Quantity,

mode\_Quantity.mode\_count as mode\_Quantity\_count,

mode\_ReturnQuantity.mode\_value as mode\_ReturnQuantity,

mode\_ReturnQuantity.mode\_count as mode\_ReturnQuantity\_count,

mode\_Final\_Sales.mode\_value as mode\_Final\_Sales,

mode\_Final\_Sales.mode\_count as mode\_Final\_Sales\_count,

mode\_Final\_cost.mode\_value as mode\_Final\_cost,

mode\_Final\_cost.mode\_count as mode\_Final\_cost\_count,

mode\_RtnMPR.mode\_value as mode\_RtnMPR,

mode\_RtnMPR.mode\_count as mode\_RtnMPR\_count

from(

SELECT Quantity AS mode\_value, COUNT(\*) AS mode\_count

FROM medicine\_detail

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1 ) as mode\_Quantity,

( SELECT ReturnQuantity AS mode\_value, COUNT(\*) AS mode\_count

FROM medicine\_detail

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_ReturnQuantity,

(SELECT final\_cost AS mode\_value, COUNT(\*) AS mode\_count

FROM medicine\_detail

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_Final\_cost,

(SELECT Final\_Sales AS mode\_value, COUNT(\*) AS mode\_count

FROM medicine\_detail

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_Final\_Sales,

( select RtnMPR AS mode\_value, COUNT(\*) AS mode\_count

FROM medicine\_detail

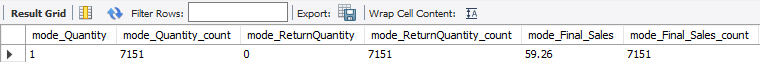
GROUP BY Quantity

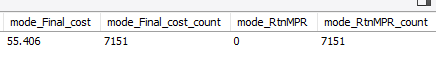
ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_RtnMPR;

**Output:**





**4. Calculating the second moment (measures of dispersion such as variance, standard deviation, range) for the dataset.**

**Variance:**

SELECT

ROUND(VARIANCE(Quantity), 2) AS variance\_quantity,

ROUND(VARIANCE(ReturnQuantity), 2) AS variance\_return\_quantity,

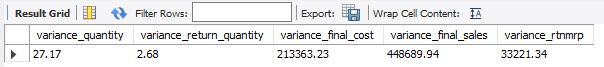
ROUND(VARIANCE(Final\_Cost), 2) AS variance\_final\_cost,

ROUND(VARIANCE(Final\_Sales), 2) AS variance\_final\_sales,

ROUND(VARIANCE(RtnMPR), 2) AS variance\_rtnmrp

FROM medicine\_detail;

**Output:**



**Standard Deviation:**

SELECT

ROUND(STDDEV(Quantity), 2) AS stddev\_quantity,

ROUND(STDDEV(ReturnQuantity), 2) AS stddev\_return\_quantity,

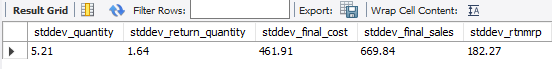
ROUND(STDDEV(Final\_Cost), 2) AS stddev\_final\_cost,

ROUND(STDDEV(Final\_Sales), 2) AS stddev\_final\_sales,

ROUND(STDDEV(RtnMPR), 2) AS stddev\_rtnmrp

FROM medicine\_detail;

**Output:**



**Range:**

SELECT

MAX(Quantity) - MIN(Quantity) AS range\_quantity,

MAX(ReturnQuantity) - MIN(ReturnQuantity) AS range\_return\_quantity,

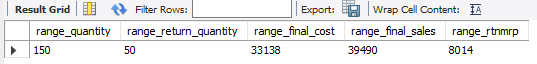
MAX(Final\_Cost) - MIN(Final\_Cost) AS range\_final\_cost,

MAX(Final\_Sales) - MIN(Final\_Sales) AS range\_final\_sales,

MAX(RtnMPR) - MIN(RtnMPR) AS range\_rtnmrp

FROM medicine\_detail;

**Output:**



**5. Calculating the third moment (skewness) for the dataset.**

**Skewness:**

select Quantity\_skewness1.Quantity\_skewness,

ReturnQuantity\_skewness1.ReturnQuantity\_skewness,

Final\_cost\_skewness1.Final\_cost\_skewness,

Final\_Sales\_skewness1.Final\_Sales\_skewness,

RtnMPR\_skewness1.RtnMPR\_skewness

from

(SELECT

ROUND((SUM(POW(Quantity - (SELECT AVG(Quantity) FROM medicine\_detail), 3)) / (COUNT(\*) \*

POW(STDDEV(Quantity), 3))), 2) AS Quantity\_skewness

FROM medicine\_detail) Quantity\_skewness1,

(select

round((sum(pow(ReturnQuantity - (select avg(ReturnQuantity) from medicine\_detail),3))/(count(\*) \*

pow(stddev(ReturnQuantity),3))),2) as ReturnQuantity\_skewness from medicine\_detail)

ReturnQuantity\_skewness1,

(select

round((sum(pow(Final\_cost - (select avg(Final\_cost) from medicine\_detail),3))/(count(\*) \*

pow(stddev(Final\_cost),3))),2) as Final\_cost\_skewness from medicine\_detail)

Final\_cost\_skewness1,

(

select

round((sum(pow(Final\_Sales - (select avg(Final\_Sales) from medicine\_detail),3))/(count(\*) \*

pow(stddev(Final\_Sales),3))),2) as Final\_Sales\_skewness from medicine\_detail)

Final\_Sales\_skewness1,

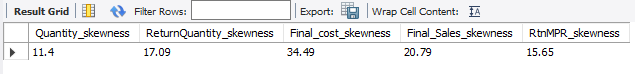
(select

round((sum(pow(RtnMPR - (select avg(RtnMPR) from medicine\_detail),3))/(count(\*) \*

pow(stddev(RtnMPR),3))),2) as RtnMPR\_skewness from medicine\_detail)

RtnMPR\_skewness1;

**Output:**



**6. Calculating the fourth moment (kurtosis) for the dataset.**

**Kurtosis:**

select

kurtosis\_quantity1.kurtosis\_quantity,

kurtosis\_Returnquantity1.kurtosis\_Returnquantity,

kurtosis\_Final\_cost1.kurtosis\_Final\_cost,

kurtosis\_Final\_Sales1.kurtosis\_Final\_Sales,

kurtosis\_RtnMPR1.kurtosis\_RtnMPR

from

(SELECT

ROUND((SUM(POWER(Quantity - (SELECT AVG(Quantity) FROM medicine\_detail), 4)) / (COUNT(Quantity) \* POWER(STDDEV(Quantity),

4))), 2) AS kurtosis\_quantity

from medicine\_detail) kurtosis\_quantity1,

(SELECT

ROUND((SUM(POWER(ReturnQuantity - (SELECT AVG(ReturnQuantity) FROM medicine\_detail), 4)) / (COUNT(ReturnQuantity) \* POWER(STDDEV(ReturnQuantity),

4))), 2) AS kurtosis\_Returnquantity

from medicine\_detail) kurtosis\_Returnquantity1,

(SELECT

ROUND((SUM(POWER(Final\_cost - (SELECT AVG(Final\_cost) FROM medicine\_detail), 4)) / (COUNT(Final\_cost) \* POWER(STDDEV(Final\_cost),

4))), 2) AS kurtosis\_Final\_cost

from medicine\_detail) kurtosis\_Final\_cost1,

(SELECT

ROUND((SUM(POWER(Final\_Sales - (SELECT AVG(Final\_Sales) FROM medicine\_detail), 4)) / (COUNT(Final\_Sales) \* POWER(STDDEV(Final\_Sales),

4))), 2) AS kurtosis\_Final\_Sales

from medicine\_detail) kurtosis\_Final\_Sales1,

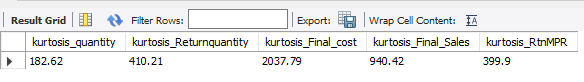
(SELECT

ROUND((SUM(POWER(RtnMPR - (SELECT AVG(RtnMPR) FROM medicine\_detail), 4)) / (COUNT(RtnMPR) \* POWER(STDDEV(RtnMPR),

4))), 2) AS kurtosis\_RtnMPR

from medicine\_detail) kurtosis\_RtnMPR1;

**Output:**



**Pre-processing Code (SQL)**

**Software: MySQL Workbench**

**1. Creating the database.**

CREATE DATABASE medicines\_db;

**2. Set the current database to " medicines\_db ".**

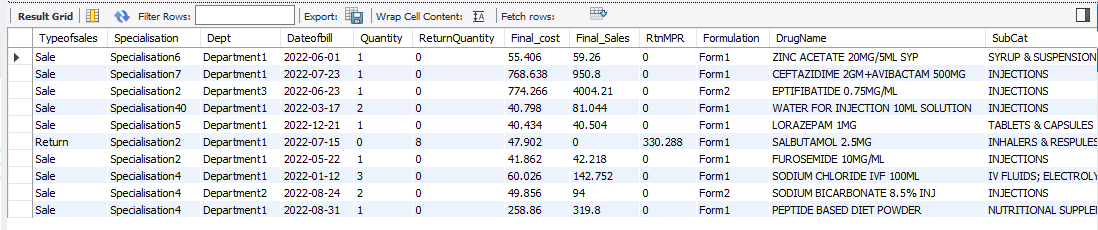
USE med\_inventory;

**3. Importing the table in CSV format into MySQL.**

Right-click on tables -> table data import wizard -> browse to select the table ‘medicine\_detail’.

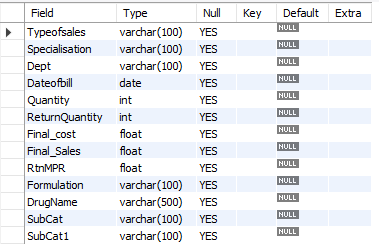
**4. Displaying the table.**

SELECT \* FROM medicine\_detail LIMIT 10



**5. Checking the schema of the dataset to ensure that all columns have the correct format.**

DESCRIBE medicine\_detail;



**6. Counting the missing and non-missing values for each column and the total number of rows in the 'medicine\_detail' table.**

SELECT

COUNT(CASE WHEN TRIM(Typeofsales) = '' OR Typeofsales IS NULL THEN 1 END) AS

typeofsales\_missing,

COUNT(CASE WHEN TRIM(Typeofsales) <> '' AND Typeofsales IS NOT NULL THEN 1 END) AS

typeofsales\_non\_missing,

count(case when trim(Typeofsales) = '' or Typeofsales is null then 1 end) as Typeofsales\_missing,

count(case when trim(Typeofsales) <> '' or Typeofsales is not null then 1 end) as Typeofsales\_non\_missing,

COUNT(CASE WHEN TRIM(Specialisation) = '' OR Specialisation IS NULL THEN 1 END) AS

specialisation\_missing,

COUNT(CASE WHEN TRIM(Specialisation) <> '' AND Specialisation IS NOT NULL THEN 1 END) AS

specialisation\_non\_missing ,

COUNT(CASE WHEN TRIM(Dept) = '' OR Dept IS NULL THEN 1 END) AS dept\_missing,

COUNT(CASE WHEN TRIM(Dept) <> '' AND Dept IS NOT NULL THEN 1 END) AS dept\_non\_missing,

COUNT(CASE WHEN TRIM(Dateofbill) = '' OR Dateofbill IS NULL THEN 1 END) AS dateofbill\_missing,

COUNT(CASE WHEN TRIM(Dateofbill) <> '' AND Dateofbill IS NOT NULL THEN 1 END) AS

dateofbill\_non\_missing,

COUNT(CASE WHEN Quantity IS NULL THEN 1 END) AS quantity\_missing,

COUNT(CASE WHEN Quantity IS NOT NULL THEN 1 END) AS quantity\_non\_missing,

COUNT(CASE WHEN ReturnQuantity IS NULL THEN 1 END) AS returnquantity\_missing,

COUNT(CASE WHEN ReturnQuantity IS NOT NULL THEN 1 END) AS returnquantity\_non\_missing,

COUNT(CASE WHEN Final\_Cost IS NULL THEN 1 END) AS final\_cost\_missing,

COUNT(CASE WHEN Final\_Cost IS NOT NULL THEN 1 END) AS final\_cost\_non\_missing,

COUNT(CASE WHEN Final\_Sales IS NULL THEN 1 END) AS final\_sales\_missing,

COUNT(CASE WHEN Final\_Sales IS NOT NULL THEN 1 END) AS final\_sales\_non\_missing,

COUNT(CASE WHEN RtnMPR IS NULL THEN 1 END) AS rtnmrp\_missing,

COUNT(CASE WHEN RtnMPR IS NOT NULL THEN 1 END) AS rtnmrp\_non\_missing,

COUNT(CASE WHEN TRIM(Formulation) = '' OR Formulation IS NULL THEN 1 END) AS

formulation\_missing,

COUNT(CASE WHEN TRIM(Formulation) <> '' AND Formulation IS NOT NULL THEN 1 END) AS

formulation\_non\_missing,

COUNT(CASE WHEN TRIM(DrugName) = '' OR DrugName IS NULL THEN 1 END) AS

drugname\_missing,

COUNT(CASE WHEN TRIM(DrugName) <> '' AND DrugName IS NOT NULL THEN 1 END) AS

drugname\_non\_missing,

COUNT(CASE WHEN TRIM(SubCat) = '' OR SubCat IS NULL THEN 1 END) AS subcat\_missing,

COUNT(CASE WHEN TRIM(SubCat) <> '' AND SubCat IS NOT NULL THEN 1 END) AS

subcat\_non\_missing,

COUNT(CASE WHEN TRIM(SubCat1) = '' OR SubCat1 IS NULL THEN 1 END) AS subcat1\_missing,

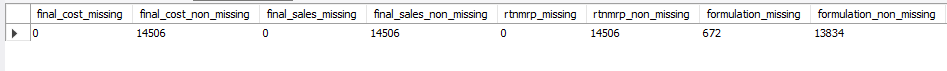
COUNT(CASE WHEN TRIM(SubCat1) <> '' AND SubCat1 IS NOT NULL THEN 1 END) AS

subcat1\_non\_missing,

COUNT(\*) AS total\_rows

FROM medicine\_detail;

**Output:**

**Observation: Missing values have been identified in the following columns: Formulation(672), DrugName(1705), SubCat(1705), and SubCat1(1729).**

**8. Replacing the missing values with ‘unknown’ in the columns Formulation, DrugName, SubCat and SubCat1.**

UPDATE medicine\_detail

SET Formulation = CASE WHEN Formulation = '' THEN 'unknown' ELSE Formulation END;

UPDATE medicine\_detail

SET DrugName = CASE WHEN DrugName = '' THEN 'unknown' ELSE DrugName END;

UPDATE medicine\_detail

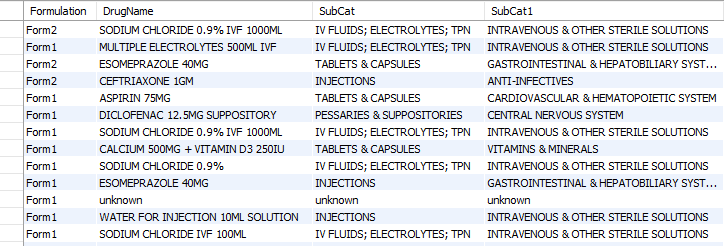
SET SubCat = CASE WHEN SubCat = '' THEN 'unknown' ELSE SubCat END;

UPDATE medicine\_detail

SET SubCat1 = CASE WHEN SubCat1 = '' THEN 'unknown' ELSE SubCat1 END;

**Showing the columns after replacing the missing values with ‘unknown’:**

select Formulation, DrugName,SubCat,SubCat1 from medicine\_detail;



**9. Creating a new table called `missing\_values` by selecting rows from `medicine\_detail` where any of the columns (`Formulation`, `DrugName`, `SubCat`, or `SubCat1`) has the value 'unknown'.**

CREATE TABLE missing\_values AS

SELECT \*

FROM medicine\_detail

WHERE Formulation = 'unknown'

OR DrugName = 'unknown'

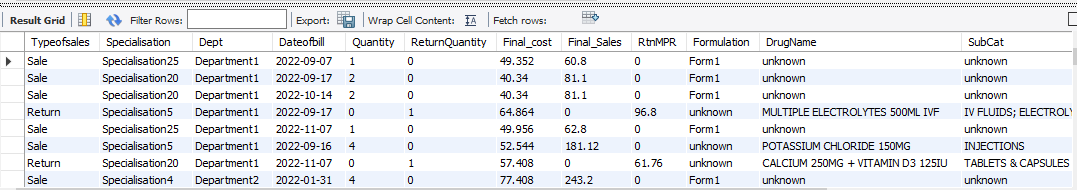
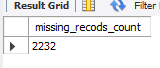
OR SubCat = 'unknown'

OR SubCat1 = 'unknown';

**Showing missing\_values table and count of records with at least one or more missing values:**

select \* from missing\_values;

select count(\*) as missing\_recods\_count from missing\_values;

**10. Identifying duplicate rows based on Typeofsales, Specialisation, Dept, Dateofbill, Quantity, ReturnQuantity, Final\_cost, Final\_Sales, RtnMPR, Formulation, DrugName, SubCat, SubCat1, Dateofbill, where the DrugName is 'unknown',** **SubCat is 'unknown',** **subcat1 is 'unknown' and Formulation is 'unknown'.**

select Typeofsales, Specialisation, Dept, Dateofbill, Quantity, ReturnQuantity,

Final\_cost, Final\_Sales, RtnMPR, Formulation, DrugName, SubCat, SubCat1, count(\*)

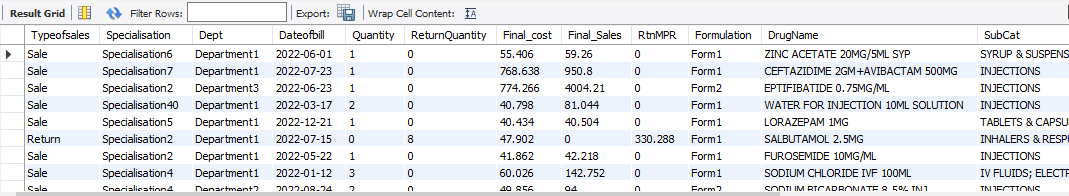
from medicine\_detail

WHERE Formulation<>'unknown'and DrugName <> 'unknown'and SubCat <> 'unknown' and subcat1 <>'unknown'

GROUP BY Typeofsales, Specialisation, Dept, Dateofbill, Quantity, ReturnQuantity,

Final\_cost, Final\_Sales, RtnMPR, Formulation, DrugName, SubCat, SubCat1

HAVING COUNT(\*) > 1;



**11. Removing the duplicate rows from medicine\_detail table and counting the remaining rows.**

DELETE FROM medicine\_detail

WHERE (Typeofsales, Specialisation, Dept, Dateofbill, Quantity, ReturnQuantity,

Final\_cost, Final\_Sales, RtnMPR, Formulation, DrugName, SubCat, SubCat1) IN (

SELECT t.Typeofsales, t.Specialisation, t.Dept,t.Dateofbill,t.Quantity,

t.ReturnQuantity,t.Final\_cost,t.Final\_Sales,t.RtnMPR,t.Formulation,t.DrugName,t.SubCat,t.SubCat1

FROM (

SELECT Typeofsales, Specialisation, Dept, Dateofbill, Quantity, ReturnQuantity,

Final\_cost, Final\_Sales, RtnMPR, Formulation, DrugName, SubCat, SubCat1

FROM medicine\_detail

WHERE Formulation<>'unknown'and DrugName <> 'unknown'and SubCat <> 'unknown' and subcat1 <>'unknown'

GROUP BY Typeofsales, Specialisation, Dept, Dateofbill, Quantity, ReturnQuantity,

Final\_cost, Final\_Sales, RtnMPR, Formulation, DrugName, SubCat, SubCat1

HAVING COUNT(\*) > 1

) AS t

);

SELECT COUNT(\*) AS total\_rows FROM medicine\_detail;



**12. In a normal distribution, approximately 68%, 95%, and 99.7% of the data falls within one, two, and three standard deviations of the mean respectively. By using three standard deviations as the threshold for removing outliers, we are effectively removing data points that are more than three standard deviations away from the mean.**

CREATE TABLE new\_table AS

SELECT \*

FROM medicine\_detail

WHERE Quantity BETWEEN

(SELECT AVG(Quantity) - 3 \* STDDEV(Quantity) FROM medicine\_detail)

AND

(SELECT AVG(Quantity) + 3 \* STDDEV(Quantity) FROM medicine\_detail)

AND ReturnQuantity BETWEEN

(SELECT AVG(ReturnQuantity) - 3 \* STDDEV(ReturnQuantity) FROM medicine\_detail)

AND

(SELECT AVG(ReturnQuantity) + 3 \* STDDEV(ReturnQuantity) FROM medicine\_detail)

AND Final\_Cost BETWEEN

(SELECT AVG(Final\_Cost) - 3 \* STDDEV(Final\_Cost) FROM medicine\_detail)

AND

(SELECT AVG(Final\_Cost) + 3 \* STDDEV(Final\_Cost) FROM medicine\_detail)

AND Final\_Sales BETWEEN

(SELECT AVG(Final\_Sales) - 3 \* STDDEV(Final\_Sales) FROM medicine\_detail)

AND

(SELECT AVG(Final\_Sales) + 3 \* STDDEV(Final\_Sales) FROM medicine\_detail)

AND RtnMPR BETWEEN

(SELECT AVG(RtnMPR) - 3 \* STDDEV(RtnMPR) FROM medicine\_detail)

AND

(SELECT AVG(RtnMPR) + 3 \* STDDEV(RtnMPR) FROM medicine\_detail);

select count(\*) as total\_rows from new\_table;



**Exploratory Data Analysis (SQL)**

**Software: MySQL Workbench**

**Business decisions based on the clean**

**‘medicines\_db’ data**

**1. Calculating the first moment (measures of central tendency such as mean, median, mode) for the dataset.**

**Mean :**

select

round(avg(Quantity),2) as mean\_Quantity,

round(avg(ReturnQuantity),2) as mean\_ReturnQuantity,

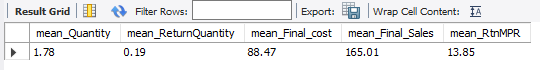
round(avg(Final\_cost),2) as mean\_Final\_cost,

round(avg(Final\_Sales),2) as mean\_Final\_Sales,

round(avg(RtnMPR),2) as mean\_RtnMPR

from

new\_table;



**Median:**

with ranked as

(

select Final\_cost,

Final\_Sales,

Quantity,

ReturnQuantity,

RtnMPR,

row\_number() over (order by Final\_cost) as r,

count(\*) over () as c

from new\_table),

median as

(

select Final\_cost,

Final\_Sales,

Quantity,

ReturnQuantity,

RtnMPR

from ranked

where r in (floor((c+1)/2),ceiling((c+1)/2))

)

select round(avg(Final\_cost), 2),

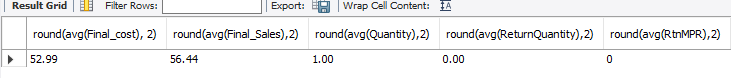
round(avg(Final\_Sales),2),

round(avg(Quantity),2),

round(avg(ReturnQuantity),2),

round(avg(RtnMPR),2)

from median;



**Mode:**

select

mode\_Quantity.mode\_value as mode\_Quantity,

mode\_Quantity.mode\_count as mode\_Quantity\_count,

mode\_ReturnQuantity.mode\_value as mode\_ReturnQuantity,

mode\_ReturnQuantity.mode\_count as mode\_ReturnQuantity\_count,

mode\_Final\_Sales.mode\_value as mode\_Final\_Sales,

mode\_Final\_Sales.mode\_count as mode\_Final\_Sales\_count,

mode\_Final\_cost.mode\_value as mode\_Final\_cost,

mode\_Final\_cost.mode\_count as mode\_Final\_cost\_count,

mode\_RtnMPR.mode\_value as mode\_RtnMPR,

mode\_RtnMPR.mode\_count as mode\_RtnMPR\_count

from(

SELECT Quantity AS mode\_value, COUNT(\*) AS mode\_count

FROM new\_table

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1 ) as mode\_Quantity,

( SELECT ReturnQuantity AS mode\_value, COUNT(\*) AS mode\_count

FROM new\_table

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_ReturnQuantity,

(SELECT final\_cost AS mode\_value, COUNT(\*) AS mode\_count

FROM new\_table

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_Final\_cost,

(SELECT Final\_Sales AS mode\_value, COUNT(\*) AS mode\_count

FROM new\_table

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_Final\_Sales,

( select RtnMPR AS mode\_value, COUNT(\*) AS mode\_count

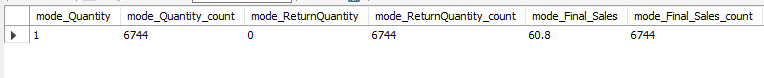
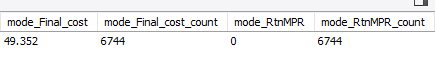
FROM new\_table

GROUP BY Quantity

ORDER BY COUNT(\*) DESC

LIMIT 1

) as mode\_RtnMPR;

**2. Calculating the second moment (measures of dispersion such as variance, standard deviation, range) for the dataset.**

**Variance:**

SELECT

ROUND(VARIANCE(Quantity), 2) AS variance\_quantity,

ROUND(VARIANCE(ReturnQuantity), 2) AS variance\_return\_quantity,

ROUND(VARIANCE(Final\_Cost), 2) AS variance\_final\_cost,

ROUND(VARIANCE(Final\_Sales), 2) AS variance\_final\_sales,

ROUND(VARIANCE(RtnMPR), 2) AS variance\_rtnmrp

FROM new\_table;



SELECT

ROUND(STDDEV(Quantity), 2) AS stddev\_quantity,

ROUND(STDDEV(ReturnQuantity), 2) AS stddev\_return\_quantity,

ROUND(STDDEV(Final\_Cost), 2) AS stddev\_final\_cost,

ROUND(STDDEV(Final\_Sales), 2) AS stddev\_final\_sales,

ROUND(STDDEV(RtnMPR), 2) AS stddev\_rtnmrp

FROM new\_table;

****

**Range:**

SELECT

MAX(Quantity) - MIN(Quantity) AS range\_quantity,

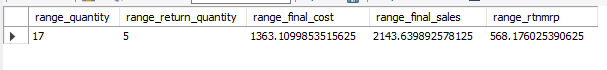
MAX(ReturnQuantity) - MIN(ReturnQuantity) AS range\_return\_quantity,

MAX(Final\_Cost) - MIN(Final\_Cost) AS range\_final\_cost,

MAX(Final\_Sales) - MIN(Final\_Sales) AS range\_final\_sales,

MAX(RtnMPR) - MIN(RtnMPR) AS range\_rtnmrp

FROM new\_table;



**3. Calculating the third moment (skewness) for the dataset.**

**Skewness:**

select Quantity\_skewness1.Quantity\_skewness,

ReturnQuantity\_skewness1.ReturnQuantity\_skewness,

Final\_cost\_skewness1.Final\_cost\_skewness,

Final\_Sales\_skewness1.Final\_Sales\_skewness,

RtnMPR\_skewness1.RtnMPR\_skewness

from

(SELECT

ROUND((SUM(POW(Quantity - (SELECT AVG(Quantity) FROM new\_table), 3)) / (COUNT(\*) \*

POW(STDDEV(Quantity), 3))), 2) AS Quantity\_skewness

FROM new\_table) Quantity\_skewness1,

(select

round((sum(pow(ReturnQuantity - (select avg(ReturnQuantity) from new\_table),3))/(count(\*) \*

pow(stddev(ReturnQuantity),3))),2) as ReturnQuantity\_skewness from new\_table)

ReturnQuantity\_skewness1,

(select

round((sum(pow(Final\_cost - (select avg(Final\_cost) from new\_table),3))/(count(\*) \*

pow(stddev(Final\_cost),3))),2) as Final\_cost\_skewness from new\_table)

Final\_cost\_skewness1,

(

select

round((sum(pow(Final\_Sales - (select avg(Final\_Sales) from new\_table),3))/(count(\*) \*

pow(stddev(Final\_Sales),3))),2) as Final\_Sales\_skewness from new\_table)

Final\_Sales\_skewness1,

(select

round((sum(pow(RtnMPR - (select avg(RtnMPR) from new\_table),3))/(count(\*) \*

pow(stddev(RtnMPR),3))),2) as RtnMPR\_skewness from new\_table)

RtnMPR\_skewness1;



**4. Calculating the fourth moment (kurtosis) for the dataset.**

**Kurtosis:**

select

kurtosis\_quantity1.kurtosis\_quantity,

kurtosis\_Returnquantity1.kurtosis\_Returnquantity,

kurtosis\_Final\_cost1.kurtosis\_Final\_cost,

kurtosis\_Final\_Sales1.kurtosis\_Final\_Sales,

kurtosis\_RtnMPR1.kurtosis\_RtnMPR

from

(SELECT

ROUND((SUM(POWER(Quantity - (SELECT AVG(Quantity) FROM new\_table), 4)) / (COUNT(Quantity) \* POWER(STDDEV(Quantity),

4))), 2) AS kurtosis\_quantity

from new\_table) kurtosis\_quantity1,

(SELECT

ROUND((SUM(POWER(ReturnQuantity - (SELECT AVG(ReturnQuantity) FROM new\_table), 4)) / (COUNT(ReturnQuantity) \* POWER(STDDEV(ReturnQuantity),

4))), 2) AS kurtosis\_Returnquantity

from new\_table) kurtosis\_Returnquantity1,

(SELECT

ROUND((SUM(POWER(Final\_cost - (SELECT AVG(Final\_cost) FROM new\_table), 4)) / (COUNT(Final\_cost) \* POWER(STDDEV(Final\_cost),

4))), 2) AS kurtosis\_Final\_cost

from new\_table) kurtosis\_Final\_cost1,

(SELECT

ROUND((SUM(POWER(Final\_Sales - (SELECT AVG(Final\_Sales) FROM new\_table), 4)) / (COUNT(Final\_Sales) \* POWER(STDDEV(Final\_Sales),

4))), 2) AS kurtosis\_Final\_Sales

from new\_table) kurtosis\_Final\_Sales1,

(SELECT

ROUND((SUM(POWER(RtnMPR - (SELECT AVG(RtnMPR) FROM new\_table), 4)) / (COUNT(RtnMPR) \* POWER(STDDEV(RtnMPR),

4))), 2) AS kurtosis\_RtnMPR

from new\_table) kurtosis\_RtnMPR1;



**Comparison table showing the business decisions results for the unclean and clean data:**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Uncleaned Data** | **Cleaned\_Data** |
| **Mean** | **Quantity -2.25** | **Quantity-1.78** |
|  | **ReturnQuantity -0.30** | **ReturnQuantity -0.19** |
|  | **Final\_cost-124.43** | **Final\_cost-88.47** |
|  | **Final\_Sales-232.79** | **Final\_Sales-165.01** |
|  | **RtnMPR-29.95** | **RtnMPR-13.85** |
| **Median** | **Quantity-3.00** | **Quantity-1.00** |
|  | **ReturnQuantity -0.00** | **ReturnQuantity -0.00** |
|  | **Final\_cost-141** | **Final\_cost-52.99** |
|  | **Final\_Sales-53.57** | **Final\_Sales-56.44** |
|  | **RtnMPR-0-00** | **RtnMPR-0.00** |
| **Mode** | **Quantity-1** | **Quantity-1** |
|  | **ReturnQuantity -0** | **ReturnQuantity -0** |
|  | **Final\_cost-49.352** | **Final\_cost-49.352** |
|  | **Final\_Sales-60.8** | **Final\_Sales-60.8** |
|  | **RtnMPR-0** | **RtnMPR-0** |
| **Variance** | **Quantity-27.14** | **Quantity-3.32** |
|  | **ReturnQuantity 2.17** | **ReturnQuantity -0.41** |
|  | **Final\_cost-221685.41** | **Final\_cost-17736.69** |
|  | **Final\_Sales-458223.23** | **Final\_Sales-71134.19** |
|  | **RtnMPR-34178.39** | **RtnMPR-2596.22** |
| **StandardDeviation** | **Quantity-5.21** | **Quantity-1.82** |
|  | **ReturnQuantity -1.67** | **ReturnQuantity -0.64** |
|  | **Final\_cost-470.83** | **Final\_cost-133.18** |
|  | **Final\_Sales-676.93** | **Final\_Sales-266.71** |
|  | **RtnMPR-184.87** | **RtnMPR-49.96** |
| **Range** | **Quantity-150** | **Quantity-17** |
|  | **ReturnQuantity -50** | **ReturnQuantity -5** |
|  | **Final\_cost-33138** | **Final\_cost-1363.11** |
|  | **Final\_Sales-39490** | **Final\_Sales-2143.64** |
|  | **RtnMPR-8014** | **RtnMPR-568.18** |
| **Skewness** | **Quantity-11.27** | **Quantity-2.91** |
|  | **ReturnQuantity -16.88** | **ReturnQuantity -4.23** |
|  | **Final\_cost-34.22** | **Final\_cost-5.86** |
|  | **Final\_Sales-21.09** | **Final\_Sales-4.3** |
|  | **RtnMPR-15.58** | **RtnMPR5.11** |
| **Kurtosis** | **Quantity-180.5**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | **Quantity-14.7** |
|  | **ReturnQuantity -398.03** | **ReturnQuantity -23.43** |
|  | **Final\_cost-1984.44** | **Final\_cost-41.6** |
|  | **Final\_Sales-947.71** | **Final\_Sales-24.76** |
|  | **RtnMPR-395.28** | **RtnMPR-35.42** |

**Observation: Overall, the results indicate that the unclean data exhibits higher mean, variance, standard deviation, range, skewness, and kurtosis values compared to the clean data. This suggests greater inconsistencies, variability, and potential outliers in the unclean data. Cleaning the data has resulted in more stable and normalized distributions with reduced variability and potential biases, making it more reliable for business decision-making.**