**SQL Query spotify\_2023**

select \* from spotify\_2023;

---# Measure of central tendency # --

---# 1) artist\_count

---# FIRST MOMENT BUSINESS DECISION

select

AVG(artist\_count) AS avg\_artist,

min(artist\_count) as min\_artist\_count,

max(artist\_count) as max\_artist\_count,

count(artist\_count) as count\_artist\_count,

count(distinct(artist\_count))as unique\_artist\_count,

sum(artist\_count) as sum\_artist\_count ,

max(artist\_count) - min(artist\_count) as range\_artist\_count from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(artist\_count) from

(select(artist\_count),

row\_number()over(order by (artist\_count)) as row\_num,

count(\*) over() as total\_artist\_count

from spotify\_2023

) as median\_artist\_count

where row\_num = (total\_artist\_count+1)/2 or row\_num =(total\_artist\_count+2)/2;

-- # MODE # --

SELECT (artist\_count), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (artist\_count) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(artist\_count) AS non\_null\_rows,

COUNT(\*) - COUNT(artist\_count) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(artist\_count) from spotify\_2023; -- # VERIANCE # --

Select stddev(artist\_count) as std\_artist\_count from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(artist\_count -

(SELECT AVG(artist\_count) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(artist\_count) FROM spotify\_2023),3))

) AS skewness\_artist\_count FROM spotify\_2023;

#After cleaning prepeocessing

-- #Detected Null values #

Select \* from spotify\_2023 where artist\_count is null;

select count(\*) as artist\_count\_null from spotify\_2023 where artist\_count is null;

------#######

---# released\_year

select

AVG(released\_year) AS avg\_released\_year,

min(released\_year)as min\_released\_year,

max(released\_year)as max\_released\_year,

count(released\_year)as count\_released\_year,

count(distinct(released\_year))as unique\_released\_year,

sum(released\_year)as sum\_released\_year,

max(released\_year) - min(released\_year) as range\_released\_year from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(released\_year) from

(select(released\_year),

row\_number()over(order by (released\_year)) as row\_num,

count(\*)over() as total\_released\_year

from spotify\_2023

) as median\_released\_year

where row\_num = (total\_released\_year+1)/2 or row\_num =(total\_released\_year +2)/2;

-- # MODE # --

SELECT (released\_year), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (released\_year) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(released\_year) AS non\_null\_rows,

COUNT(\*) - COUNT(released\_year) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(released\_year) from spotify\_2023; -- # VERIANCE # --

Select stddev(released\_year) as std\_released\_year from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(released\_year -

(SELECT AVG(released\_year) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(released\_year) FROM spotify\_2023),3))

) AS skewness\_released\_year FROM spotify\_2023;

#################### After cleaning prepeocessing #######################

-- #### Detected Null values #####

Select \* from spotify\_2023 where released\_year is null;

select count(\*) as released\_year\_null from spotify\_2023 where released\_year is null;

----------------- #### -----

---# released\_month

---# FIRST MOMENT BUSINESS DECISION

select

AVG(released\_month) AS avg\_released\_month,

min(released\_month)as min\_released\_month,

max(released\_month)as max\_released\_month,

count(released\_month)as count\_released\_month,

count(distinct(released\_month))as unique\_released\_month,

sum(released\_month)as sum\_released\_month,

max(released\_month) - min(released\_month) as range\_released\_month from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(released\_month) from

(select(released\_month),

row\_number()over(order by (released\_month)) as row\_num,

count(\*)over() as total\_released\_month

from spotify\_2023

) as median\_released\_month

where row\_num = (total\_released\_month + 1) / 2 or row\_num =(total\_released\_month +2 ) /2;

-- # MODE # --

SELECT (released\_month), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (released\_month) ORDER BY COUNT(\*)

DESC limit 1;

# SQL query to count the total number of rows, non-null rows for the key column,

# and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(released\_month) AS non\_null\_rows,

COUNT(\*) - COUNT(released\_month) AS missing\_values

FROM spotify\_2023;

# After cleaning prepeocessing

-- #Detected Null values

Select \* from spotify\_2023 where released\_month is null;

select count(\*) as released\_month\_null from spotify\_2023 where released\_month is null;

#released\_day

---# FIRST MOMENT BUSINESS DECISION

select

AVG(released\_day) AS avg\_released\_day,

min(released\_day)as min\_released\_day,

max(released\_day)as max\_released\_day,

count(released\_day)as count\_released\_day,

count(distinct(released\_day))as unique\_released\_day,

sum(released\_day)as sum\_released\_day,

max(released\_day) - min(released\_day) as range\_released\_day from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(released\_day) from

(select(released\_day),

row\_number()over(order by (released\_day)) as row\_num,

count(\*)over() as total\_released\_day

from spotify\_2023

) as median\_released\_day

where row\_num = (total\_released\_day+ 1) / 2 or row\_num =(total\_released\_day+2 ) /2;

-- # MODE # --

SELECT (released\_day), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (released\_day) ORDER BY COUNT(\*)

DESC limit 1;

# SQL query to count the total number of rows, non-null rows for the key column,

# and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(released\_day) AS non\_null\_rows,

COUNT(\*) - COUNT(released\_day) AS missing\_values

FROM spotify\_2023;

#After cleaning prepeocessing

-- #Detected Null values

Select \* from spotify\_2023 where released\_day is null;

select count(\*) as released\_day\_null from spotify\_2023 where released\_day is null;

---# in\_spotify\_playlists,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(in\_spotify\_playlists) AS avg\_in\_spotify\_playlists,

min(in\_spotify\_playlists)as min\_in\_spotify\_playlists,

max(in\_spotify\_playlists)as max\_in\_spotify\_playlists,

count(in\_spotify\_playlists)as count\_in\_spotify\_playlists,

count(distinct(in\_spotify\_playlists))as unique\_in\_spotify\_playlists,

sum(in\_spotify\_playlists)as sum\_in\_spotify\_playlistst ,

max(in\_spotify\_playlists) - min(in\_spotify\_playlists) as range\_in\_spotify\_playlists from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(in\_spotify\_playlists) from

(select(in\_spotify\_playlists),

row\_number()over(order by (in\_spotify\_playlists)) as row\_num,

count(\*)over() as total\_in\_spotify\_playlists

from spotify\_2023

) as median\_in\_spotify\_playlists

where row\_num = (total\_in\_spotify\_playlists +1)/2 or row\_num =(total\_in\_spotify\_playlists +2)/2;

-- # MODE # --

SELECT (in\_spotify\_playlists), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (in\_spotify\_playlists) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(in\_spotify\_playlists) AS non\_null\_rows,

COUNT(\*) - COUNT(in\_spotify\_playlists) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(in\_spotify\_playlists) from spotify\_2023; -- # VERIANCE # --

Select stddev(in\_spotify\_playlists) as std\_in\_spotify\_playlists from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(in\_spotify\_playlists -

(SELECT AVG(in\_spotify\_playlists) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(in\_spotify\_playlists) FROM spotify\_2023),3))

) AS skewness\_in\_spotify\_playlists FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where in\_spotify\_playlists is null;

select count(\*) as in\_spotify\_playlists\_null from spotify\_2023 where in\_spotify\_playlists is null;

# in\_spotify\_charts,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(in\_spotify\_charts) AS avg\_in\_spotify\_charts,

min(in\_spotify\_charts)as min\_in\_spotify\_charts,

max(in\_spotify\_charts)as max\_in\_spotify\_charts,

count(in\_spotify\_charts)as count\_in\_spotify\_charts,

count(distinct(in\_spotify\_charts))as unique\_in\_spotify\_charts,

sum(in\_spotify\_charts)as sum\_in\_spotify\_charts,

max(in\_spotify\_charts) - min(in\_spotify\_charts) as range\_in\_spotify\_charts

from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(in\_spotify\_charts) from

(select(in\_spotify\_charts),

row\_number()over(order by (in\_spotify\_charts)) as row\_num,

count(\*)over() as total\_in\_spotify\_charts

from spotify\_2023

) as median\_in\_spotify\_charts

where row\_num = (total\_in\_spotify\_charts +1)/2 or row\_num =(total\_in\_spotify\_charts +2)/2;

-- # MODE # --

SELECT (in\_spotify\_charts), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (in\_spotify\_charts) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(in\_spotify\_charts) AS non\_null\_rows,

COUNT(\*) - COUNT(in\_spotify\_charts) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(in\_spotify\_charts) from spotify\_2023; -- # VERIANCE # --

Select stddev(in\_spotify\_charts) as std\_artist\_count from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(in\_spotify\_charts -

(SELECT AVG(in\_spotify\_charts) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(in\_spotify\_charts) FROM spotify\_2023),3))

) AS skewness\_in\_spotify\_charts FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where in\_spotify\_charts is null;

select count(\*) as in\_spotify\_charts\_null from spotify\_2023 where in\_spotify\_charts is null;

# streams,

----

---# FIRST MOMENT BUSINESS DECISION

select

AVG(streams) AS avg\_streams,

min(streams)as min\_streams,

max(streams)as max\_streams,

count(streams)as count\_streams,

count(distinct(streams))as unique\_streams,

sum(streams)as sum\_streams ,

max(streams) - min(streams) as range\_streams from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(streams) from

(select(streams),

row\_number()over(order by (streams)) as row\_num,

count(\*)over() as total\_streams

from spotify\_2023

) as median\_streams

where row\_num = (total\_streams +1)/2 or row\_num =(total\_streams +2)/2;

-- # MODE # --

SELECT (streams), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (streams) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(streams) AS non\_null\_rows,

COUNT(\*) - COUNT(streams) AS missing\_values

FROM spotify\_2023;

-- # SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(streams) from spotify\_2023; -- # VERIANCE # --

Select stddev(streams) as std\_streams from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(streams -

(SELECT AVG(streams) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(streams) FROM spotify\_2023),3))

) AS skewness\_artist\_count FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where streams is null;

select count(\*) as artist\_count\_null from spotify\_2023 where streams is null;

# in\_apple\_playlists,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(in\_apple\_playlists) AS avg\_in\_apple\_playlists,

min(in\_apple\_playlists)as min\_in\_apple\_playlists,

max(in\_apple\_playlists)as max\_in\_apple\_playlists,

count(in\_apple\_playlists)as count\_in\_apple\_playlists,

count(distinct(in\_apple\_playlists))as unique\_in\_apple\_playlists,

sum(in\_apple\_playlists)as sum\_in\_apple\_playlists ,

max(in\_apple\_playlists) - min(in\_apple\_playlists) as range\_in\_apple\_playlists

from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(in\_apple\_playlists) from

(select(in\_apple\_playlists),

row\_number()over(order by (in\_apple\_playlists)) as row\_num,

count(\*)over() as total\_in\_apple\_playlists

from spotify\_2023

) as median\_in\_apple\_playlists

where row\_num = (total\_in\_apple\_playlists +1)/2 or row\_num =(total\_in\_apple\_playlists +2)/2;

-- # MODE # --

SELECT (in\_apple\_playlists), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (in\_apple\_playlists) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(in\_apple\_playlists) AS non\_null\_rows,

COUNT(\*) - COUNT(in\_apple\_playlists) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(in\_apple\_playlists) from spotify\_2023; -- # VERIANCE # --

Select stddev(in\_apple\_playlists) as std\_in\_apple\_playlists from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(in\_apple\_playlists -

(SELECT AVG(in\_apple\_playlists) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(in\_apple\_playlists) FROM spotify\_2023),3))

) AS skewness\_in\_apple\_playlists FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where in\_apple\_playlists is null;

select count(\*) as in\_apple\_playlists\_null from spotify\_2023 where in\_apple\_playlists is null;

# in\_apple\_charts,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(in\_apple\_charts) AS avg\_in\_apple\_charts,

min(in\_apple\_charts)as min\_in\_apple\_charts,

max(in\_apple\_charts)as max\_in\_apple\_charts,

count(in\_apple\_charts)as count\_in\_apple\_charts,

count(distinct(in\_apple\_charts))as unique\_in\_apple\_charts,

sum(in\_apple\_charts)as sum\_in\_apple\_charts ,

max(in\_apple\_charts) - min(in\_apple\_charts) as range\_in\_apple\_charts from spotify\_2023;

# SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(in\_apple\_charts) from

(select(in\_apple\_charts),

row\_number()over(order by (in\_apple\_charts)) as row\_num,

count(\*)over() as total\_in\_apple\_charts

from spotify\_2023

) as median\_in\_apple\_charts

where row\_num = (total\_in\_apple\_charts +1)/2 or row\_num =(total\_in\_apple\_charts +2)/2;

-- # MODE # --

SELECT (in\_apple\_charts), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (in\_apple\_charts) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(in\_apple\_charts) AS non\_null\_rows,

COUNT(\*) - COUNT(in\_apple\_charts) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(in\_apple\_charts) from spotify\_2023; -- # VERIANCE # --

Select stddev(in\_apple\_charts) as std\_in\_apple\_charts from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(in\_apple\_charts -

(SELECT AVG(in\_apple\_charts) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(in\_apple\_charts) FROM spotify\_2023),3))

) AS skewness\_in\_apple\_charts FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where in\_apple\_charts is null;

select count(\*) as in\_apple\_charts\_null from spotify\_2023 where in\_apple\_charts is null;

---# in\_deezer\_playlists,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(in\_deezer\_playlists) AS avg\_in\_deezer\_playlists,

min(in\_deezer\_playlists)as min\_in\_deezer\_playlists,

max(in\_deezer\_playlists)as max\_in\_deezer\_playlists,

count(in\_deezer\_playlists)as count\_in\_deezer\_playlists,

count(distinct(in\_deezer\_playlists))as unique\_in\_deezer\_playlists,

sum(in\_deezer\_playlists)as sum\_in\_deezer\_playlists ,

max(in\_deezer\_playlists) - min(in\_deezer\_playlists) as range\_in\_deezer\_playlists from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(in\_deezer\_playlists) from

(select(in\_deezer\_playlists),

row\_number()over(order by (in\_deezer\_playlists)) as row\_num,

count(\*)over() as total\_in\_deezer\_playlists

from spotify\_2023

) as median\_in\_deezer\_playlists

where row\_num = (total\_in\_deezer\_playlists +1)/2 or row\_num =(total\_in\_deezer\_playlists +2)/2;

-- # MODE # --

SELECT (in\_deezer\_playlists), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (in\_deezer\_playlists) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(in\_deezer\_playlists) AS non\_null\_rows,

COUNT(\*) - COUNT(in\_deezer\_playlists) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(in\_deezer\_playlists) from spotify\_2023; -- # VERIANCE # --

Select stddev(in\_deezer\_playlists) as std\_in\_deezer\_playlists from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(in\_deezer\_playlists -

(SELECT AVG(in\_deezer\_playlists) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(in\_deezer\_playlists) FROM spotify\_2023),3))

) AS skewness\_in\_deezer\_playlists FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where in\_deezer\_playlists is null;

select count(\*) as in\_deezer\_playlists\_null from spotify\_2023 where in\_deezer\_playlists is null;

# in\_deezer\_charts,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(in\_deezer\_charts) AS avg\_in\_deezer\_charts,

min(in\_deezer\_charts)as min\_in\_deezer\_charts,

max(in\_deezer\_charts)as max\_in\_deezer\_charts,

count(in\_deezer\_charts)as count\_in\_deezer\_charts,

count(distinct(in\_deezer\_charts))as unique\_in\_deezer\_charts,

sum(in\_deezer\_charts)as sum\_in\_deezer\_charts ,

max(in\_deezer\_charts) - min(in\_deezer\_charts) as range\_in\_deezer\_charts from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(in\_deezer\_charts) from

(select(in\_deezer\_charts),

row\_number()over(order by (in\_deezer\_charts)) as row\_num,

count(\*)over() as total\_in\_deezer\_charts

from spotify\_2023

) as median\_in\_deezer\_charts

where row\_num = (total\_in\_deezer\_charts +1)/2 or row\_num =(total\_in\_deezer\_charts +2)/2;

-- # MODE # --

SELECT (in\_deezer\_charts), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (in\_deezer\_charts) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(in\_deezer\_charts) AS non\_null\_rows,

COUNT(\*) - COUNT(in\_deezer\_charts) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(in\_deezer\_charts) from spotify\_2023; -- # VERIANCE # --

Select stddev(in\_deezer\_charts) as std\_in\_deezer\_charts from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(in\_deezer\_charts -

(SELECT AVG(in\_deezer\_charts) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(in\_deezer\_charts) FROM spotify\_2023),3))

) AS skewness\_in\_deezer\_charts FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where in\_deezer\_charts is null;

select count(\*) as in\_deezer\_charts\_null from spotify\_2023 where in\_deezer\_charts is null;

# in\_shazam\_charts,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(in\_shazam\_charts) AS avg\_in\_shazam\_charts,

min(in\_shazam\_charts)as min\_in\_shazam\_charts,

max(in\_shazam\_charts)as max\_in\_shazam\_charts,

count(in\_shazam\_charts)as count\_in\_shazam\_charts,

count(distinct(in\_shazam\_charts))as unique\_in\_shazam\_charts,

sum(in\_shazam\_charts)as sum\_in\_shazam\_charts ,

max(in\_shazam\_charts) - min(in\_shazam\_charts) as range\_in\_shazam\_charts

from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(in\_shazam\_charts) from

(select(in\_shazam\_charts),

row\_number()over(order by (in\_shazam\_charts)) as row\_num,

count(\*)over() as total\_in\_shazam\_charts

from spotify\_2023

) as median\_in\_shazam\_charts

where row\_num = (total\_in\_shazam\_charts +1)/2 or row\_num =(total\_in\_shazam\_charts+2)/2;

-- # MODE # --

SELECT (in\_shazam\_charts), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (in\_shazam\_charts) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(in\_shazam\_charts) AS non\_null\_rows,

COUNT(\*) - COUNT(in\_shazam\_charts) AS missing\_values

FROM spotify\_2023;

-- # SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(in\_shazam\_charts) from spotify\_2023; -- # VERIANCE # --

Select stddev(in\_shazam\_charts) as std\_in\_shazam\_charts from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(in\_shazam\_charts -

(SELECT AVG(in\_shazam\_charts) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(in\_shazam\_charts) FROM spotify\_2023),3))

) AS skewness\_in\_shazam\_charts FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where in\_shazam\_charts is null;

select count(\*) as in\_shazam\_charts\_null from spotify\_2023 where in\_shazam\_charts is null;

---- #bpm,

---# FIRST MOMENT BUSINESS DECISION

select

AVG(bpm) AS avg\_bpm,

min(bpm)as min\_bpm,

max(bpm)as max\_bpm,

count(bpm)as count\_bpm,

count(distinct(bpm))as unique\_bpm,

sum(bpm)as sum\_bpm ,

max(bpm) - min(bpm) as range\_bpm from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(bpm) from

(select(bpm),

row\_number()over(order by (bpm)) as row\_num,

count(\*)over() as total\_bpm

from spotify\_2023

) as median\_bpm

where row\_num = (total\_bpm+1)/2 or row\_num =(total\_bpm+2)/2;

-- # MODE # --

SELECT (bpm), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (bpm) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(bpm) AS non\_null\_rows,

COUNT(\*) - COUNT(bpm) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(bpm) from spotify\_2023; -- # VERIANCE # --

Select stddev(bpm) as std\_bpm from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(bpm -

(SELECT AVG(bpm) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(bpm) FROM spotify\_2023),3))

) AS skewness\_bpm FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where bpm is null;

select count(\*) as bpm\_null from spotify\_2023 where bpm is null;

#danceability\_%,

use wysa\_data;

ALTER TABLE spotify\_2023

CHANGE COLUMN `danceability\_%` danceability\_percent int;

select \* from spotify\_2023;

---# FIRST MOMENT BUSINESS DECISION

select

AVG (danceability\_percent) AS avg\_danceability\_percent,

min(danceability\_percent)as min\_danceability\_percent,

max(danceability\_percent)as max\_danceability\_percent,

count(danceability\_percent) as count\_danceability\_percent,

count(distinct(danceability\_percent))as unique\_danceability\_percent,

sum(danceability\_percent)as sum\_danceability\_percent ,

max(danceability\_percent) - min(danceability\_percent) as range\_danceability\_percent

from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(danceability\_percent) from

(select(danceability\_percent),

row\_number()over(order by (danceability\_percent)) as row\_num,

count(\*)over() as total\_danceability\_percent

from spotify\_2023

) as median\_danceability\_percent

where row\_num = (total\_danceability\_percent +1)/2 or row\_num =(total\_danceability\_percent +2)/2;

-- # MODE # --

SELECT (danceability\_percent), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (danceability\_percent) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(danceability\_percent) AS non\_null\_rows,

COUNT(\*) - COUNT(danceability\_percent) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(danceability\_percent) from spotify\_2023; -- # VERIANCE # --

Select stddev(danceability\_percent) as std\_danceability\_percent from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(danceability\_percent -

(SELECT AVG(danceability\_percent) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(danceability\_percent) FROM spotify\_2023),3))

) AS skewness\_danceability\_percent FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where danceability\_percent is null;

select count(\*) as danceability\_percent\_null from spotify\_2023 where danceability\_percent is null;

---# valence\_%,

ALTER TABLE spotify\_2023

CHANGE COLUMN `valence\_%` valence\_percent int;

---# FIRST MOMENT BUSINESS DECISION

select

AVG(valence\_percent) AS avg\_valence\_percent,

min(valence\_percent)as min\_valence\_percent,

max(valence\_percent)as max\_valence\_percent,

count(valence\_percent)as count\_valence\_percent,

count(distinct(valence\_percent))as unique\_valence\_percent,

sum(valence\_percent)as sum\_valence\_percent,

max(valence\_percent) - min(valence\_percent) as range\_valence\_percent

from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(valence\_percent) from

(select(valence\_percent),

row\_number()over(order by (valence\_percent)) as row\_num,

count(\*)over() as total\_valence\_percent

from spotify\_2023

) as median\_valence\_percent

where row\_num = (total\_valence\_percent +1)/2 or row\_num =(total\_valence\_percent +2)/2;

-- # MODE # --

SELECT (valence\_percent), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (valence\_percent) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(valence\_percent) AS non\_null\_rows,

COUNT(\*) - COUNT(valence\_percent) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(valence\_percent) from spotify\_2023; -- # VERIANCE # --

Select stddev(valence\_percent) as std\_valence\_percent from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(valence\_percent -

(SELECT AVG(valence\_percent) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(valence\_percent) FROM spotify\_2023),3))

) AS skewness\_valence\_percent FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where valence\_percent is null;

select count(\*) as valence\_percent\_null from spotify\_2023 where valence\_percent is null;

# energy\_%,

ALTER TABLE spotify\_2023

CHANGE COLUMN `energy\_%` energy\_percent int;

---# FIRST MOMENT BUSINESS DECISION

select

AVG(energy\_percent) AS avg\_energy\_percent,

min(energy\_percent)as min\_energy\_percent,

max(energy\_percent)as max\_energy\_percent,

count(energy\_percent)as count\_energy\_percent,

count(distinct(energy\_percent))as unique\_energy\_percent,

sum(energy\_percent)as sum\_energy\_percent,

max(energy\_percent) - min(energy\_percent) as range\_energy\_percent from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(energy\_percent) from

(select(energy\_percent),

row\_number()over(order by (energy\_percent)) as row\_num,

count(\*)over() as total\_energy\_percent

from spotify\_2023

) as median\_energy\_percent

where row\_num = (total\_energy\_percent +1)/2 or row\_num =(total\_energy\_percent +2)/2;

-- # MODE # --

SELECT (energy\_percent), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (energy\_percent) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(energy\_percent) AS non\_null\_rows,

COUNT(\*) - COUNT(energy\_percent) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(energy\_percent) from spotify\_2023; -- # VERIANCE # --

Select stddev(energy\_percent) as std\_energy\_percent from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(energy\_percent -

(SELECT AVG(energy\_percent) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(energy\_percent) FROM spotify\_2023),3))

) AS skewness\_energy\_percent FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where energy\_percent is null;

select count(\*) as energy\_percent\_null from spotify\_2023 where energy\_percent is null;

# acousticness\_% ,

ALTER TABLE spotify\_2023

CHANGE COLUMN `acousticness\_%` acousticness\_percent int;

---# FIRST MOMENT BUSINESS DECISION

select

AVG(acousticness\_percent) AS avg\_acousticness\_percent,

min(acousticness\_percent)as min\_acousticness\_percent,

max(acousticness\_percent)as max\_acousticness\_percent,

count(acousticness\_percent)as count\_acousticness\_percent,

count(distinct(acousticness\_percent))as unique\_acousticness\_percent,

sum(acousticness\_percent)as sum\_acousticness\_percent ,

max(acousticness\_percent) - min(acousticness\_percent) as range\_acousticness\_percent

from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(acousticness\_percent) from

(select(acousticness\_percent),

row\_number()over(order by (acousticness\_percent)) as row\_num,

count(\*)over() as total\_acousticness\_percent

from spotify\_2023

) as median\_acousticness\_percent

where row\_num = (total\_acousticness\_percent +1)/2 or row\_num =(total\_acousticness\_percent +2)/2;

-- # MODE # --

SELECT (acousticness\_percent), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (acousticness\_percent) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(acousticness\_percent) AS non\_null\_rows,

COUNT(\*) - COUNT(acousticness\_percent) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(acousticness\_percent) from spotify\_2023; -- # VERIANCE # --

Select stddev(acousticness\_percent) as std\_acousticness\_percent from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(acousticness\_percent -

(SELECT AVG(acousticness\_percent) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(acousticness\_percent) FROM spotify\_2023),3))

) AS skewness\_acousticness\_percent FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where acousticness\_percent is null;

select count(\*) as acousticness\_percent\_null from spotify\_2023 where acousticness\_percent is null;

# instrumentalness\_%,

ALTER TABLE spotify\_2023

CHANGE COLUMN `instrumentalness\_%` instrumentalness\_percent int;

---# FIRST MOMENT BUSINESS DECISION

select

AVG(instrumentalness\_percent) AS avg\_instrumentalness\_percent,

min(instrumentalness\_percent)as min\_instrumentalness\_percent,

max(instrumentalness\_percent)as max\_instrumentalness\_percent,

count(instrumentalness\_percent)as count\_instrumentalness\_percent,

count(distinct(instrumentalness\_percent))as unique\_instrumentalness\_percent,

sum(instrumentalness\_percent)as sum\_instrumentalness\_percent,

max(instrumentalness\_percent) - min(instrumentalness\_percent) as range\_instrumentalness\_percent from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(instrumentalness\_percent) from

(select(instrumentalness\_percent),

row\_number()over(order by (instrumentalness\_percent)) as row\_num,

count(\*)over() as total\_instrumentalness\_percent

from spotify\_2023

) as median\_instrumentalness\_percent

where row\_num = (total\_instrumentalness\_percent +1)/2 or row\_num =(total\_instrumentalness\_percent +2)/2;

-- # MODE # --

SELECT (instrumentalness\_percent), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (instrumentalness\_percent) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(instrumentalness\_percent) AS non\_null\_rows,

COUNT(\*) - COUNT(instrumentalness\_percent) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(instrumentalness\_percent) from spotify\_2023; -- # VERIANCE # --

Select stddev(instrumentalness\_percent) as std\_instrumentalness\_percent from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(instrumentalness\_percent -

(SELECT AVG(instrumentalness\_percent) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(instrumentalness\_percent) FROM spotify\_2023),3))

) AS skewness\_instrumentalness\_percent FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where instrumentalness\_percent is null;

select count(\*) as instrumentalness\_percent\_null from spotify\_2023 where instrumentalness\_percent is null;

# liveness\_%,

ALTER TABLE spotify\_2023

CHANGE COLUMN `liveness\_%` liveness\_percent int;

---# FIRST MOMENT BUSINESS DECISION

select

AVG(liveness\_percent) AS avg\_liveness\_percent,

min(liveness\_percent)as min\_liveness\_percent,

max(liveness\_percent)as max\_liveness\_percent,

count(liveness\_percent)as count\_liveness\_percent,

count(distinct(liveness\_percent))as unique\_liveness\_percent,

sum(liveness\_percent)as sum\_liveness\_percent,

max(liveness\_percent) - min(liveness\_percent) as range\_liveness\_percent from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(liveness\_percent) from

(select(liveness\_percent),

row\_number()over(order by (liveness\_percent)) as row\_num,

count(\*)over() as total\_liveness\_percent

from spotify\_2023

) as median\_liveness\_percent

where row\_num = (total\_liveness\_percent +1)/2 or row\_num =(total\_liveness\_percent +2)/2;

-- # MODE # --

SELECT (liveness\_percent), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (liveness\_percent) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(liveness\_percent) AS non\_null\_rows,

COUNT(\*) - COUNT(liveness\_percent) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(liveness\_percent) from spotify\_2023; -- # VERIANCE # --

Select stddev(liveness\_percent) as std\_liveness\_percent from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(liveness\_percent -

(SELECT AVG(liveness\_percent) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(liveness\_percent) FROM spotify\_2023),3))

) AS skewness\_liveness\_percent FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where liveness\_percent is null;

select count(\*) as liveness\_percent\_null from spotify\_2023 where liveness\_percent is null;

# speechiness\_%,

ALTER TABLE spotify\_2023

CHANGE COLUMN `speechiness\_%` speechiness\_percent int;

---# FIRST MOMENT BUSINESS DECISION

select

AVG(speechiness\_percent) AS avg\_speechiness\_percent,

min(speechiness\_percent)as min\_speechiness\_percent,

max(speechiness\_percent)as max\_speechiness\_percent,

count(speechiness\_percent)as count\_speechiness\_percent,

count(distinct(speechiness\_percent))as unique\_speechiness\_percent,

sum(speechiness\_percent)as sum\_speechiness\_percent ,

max(speechiness\_percent) - min(speechiness\_percent) as range\_speechiness\_percent

from spotify\_2023;

#SQL query to median, mode from the Spotify database

---# MEDIAN # --

select(speechiness\_percent) from

(select(speechiness\_percent),

row\_number()over(order by (speechiness\_percent)) as row\_num,

count(\*)over() as total\_speechiness\_percent

from spotify\_2023

) as median\_speechiness\_percent

where row\_num = (total\_speechiness\_percent +1)/2 or row\_num =(total\_speechiness\_percent +2)/2;

-- # MODE # --

SELECT (speechiness\_percent), COUNT(\*) AS frequency

FROM spotify\_2023

GROUP BY (speechiness\_percent) ORDER BY COUNT(\*)

DESC limit 1;

#SQL query to count the total number of rows, non-null rows for the key column, and calculate the number of missing values in the dataset

SELECT

COUNT(\*) AS total\_rows,

COUNT(speechiness\_percent) AS non\_null\_rows,

COUNT(\*) - COUNT(speechiness\_percent) AS missing\_values

FROM spotify\_2023;

-- ## SECOND MOMENT BUSINESS DECISION ## --LIMIT 1

Select variance(speechiness\_percent) from spotify\_2023; -- # VERIANCE # --

Select stddev(speechiness\_percent) as std\_speechiness\_percent from spotify\_2023; -- # STANDARD DEVAITION # --

-- # THIRD MOMENT BUSINESS DECISION # --

-- SKEWNESS --

SELECT (SUM(POWER(speechiness\_percent -

(SELECT AVG(speechiness\_percent) FROM spotify\_2023),3)) /

(COUNT(\*) \* POWER ((SELECT STDDEV(speechiness\_percent) FROM spotify\_2023),3))

) AS skewness\_speechiness\_percent FROM spotify\_2023;

# After cleaning prepeocessing

-- # Detected Null values

Select \* from spotify\_2023 where speechiness\_percent is null;

select count(\*) as speechiness\_percent\_null from spotify\_2023 where speechiness\_percent is null;

select \* from spotify\_2023;