1. Created a table using the query editor

USE Spotify

CREATE TABLE tbl\_spotify\_most\_played\_23 (

track\_name nvarchar(255),

artist\_name nvarchar(255),

artist\_count int,

released\_year int,

released\_month int,

released\_day int,

in\_spotify\_playlists int,

in\_spotify\_charts int,

streams int,

in\_apple\_playlists int,

in\_apple\_charts int,

in\_deezer\_playlists int,

in\_deezer\_charts int,

in\_shazam\_charts int,

bpm int,

key\_ varchar(10),

mode varchar(10),

danceability int,

valence int,

energy int,

acousticness int,

instrumentalness int,

liveness int,

speechiness int

);

1. Since the dataset used has only 954 rows, used the GUI for SQL Server to insert the data in all the columns. Right click table\_name -> Edit top 200 rows -> Copy & Paste the data.
2. Removed whitespaces for all the text columns using TRIM()

UPDATE tbl\_spotify\_most\_played\_23

SET

[track\_name] = TRIM(track\_name),

[artist\_name] = TRIM(artist\_name),

[key\_] = TRIM(key\_),

[mode] = TRIM(mode)

1. Because of Arithmetic Overflow Error, altered the column for ‘streams’ and changed the datatype to bigint instead.

ALTER TABLE tbl\_spotify\_most\_played\_23

ALTER COLUMN streams bigint

1. Checked duplicated values in track\_name column to see if UNIQUE constraint would be viable.

SELECT track\_name, COUNT(\*) AS total\_count

FROM [dbo].[tbl\_spotify\_most\_played\_23]

GROUP BY track\_name

HAVING COUNT(\*) > 1

1. Added constraints for Data Integrity. Added CHECK constraint to ensure streams are positive.

ALTER TABLE [dbo].[tbl\_spotify\_most\_played\_23]

ADD CONSTRAINT ck\_streams\_pos CHECK (streams > 0)

GO

Also added a default constraint for year columns. If no year is provided, default to 2023.

ALTER TABLE [dbo].[tbl\_spotify\_most\_played\_23]

ADD CONSTRAINT df\_release\_year DEFAULT 2023 FOR released year

1. Used GROUPBY clause to do aggregation and find out few insights:
   * *Top 3 artitsts by streams total:*  1.Taylor Swift, 2.The Weeknd, 3.Bad Bunny

SELECT TOP(3) artist\_name, SUM(streams) AS TotalStreams

FROM tbl\_spotify\_most\_played\_23

GROUP BY artist\_name

ORDER BY TotalStreams DESC;

* + Average danceabilty by each mode: Major = 65%, Minor = 69%

SELECT mode, AVG(danceability) AS AVGDanceability

FROM tbl\_spotify\_most\_played\_23

GROUP BY mode

* + *Top 3 monthly stream trends:*1. May = 24605957716, 2. September = 11309054926, 3. October = 11248081403

1. Used the % wildcard with LIKE clause to find out highest Valance for track\_name with the word Love in it.

SELECT track\_name, valence

FROM tbl\_spotify\_most\_played\_23

WHERE track\_name LIKE '%Love%'

ORDER BY valence DESC

Findings:

- 2 Be Loved (Am I Ready) – 92 Valence

1. Used INTO clause to create temporary tables with # and then INNER joined them to find out high danceability and energy by track\_name

SELECT track\_name, danceability

INTO #high\_dance\_songs

FROM tbl\_spotify\_most\_played\_23

WHERE danceability > 70

GO

SELECT track\_name, energy

INTO #high\_energy\_songs

FROM tbl\_spotify\_most\_played\_23

WHERE energy > 80

GO

SELECT hd.track\_name ,hd.danceability, he.energy

FROM #high\_dance\_songs AS hd

JOIN #high\_energy\_songs AS he

ON hd.track\_name = he.track\_name

GO

Findings:

- Highest danceability and track name is Seven (feat. Latto) (Explicit Ver)

1. Used UNION to return Unique track names that appear in either apple and spotify charts.

SELECT track\_name

FROM tbl\_spotify\_most\_played\_23

WHERE in\_spotify\_charts = 1

INTERSECT

SELECT track\_name

FROM tbl\_spotify\_most\_played\_23

WHERE in\_apple\_charts = 1

Findings:

Total of 54 tracks that appear in either apple or spotify charts.

Used INTERESECT to return Common track names that appear in both apple and spotify charts.

SELECT track\_name

FROM tbl\_spotify\_most\_played\_23

WHERE in\_spotify\_charts = 1

INTERSECT

SELECT track\_name

FROM tbl\_spotify\_most\_played\_23

WHERE in\_apple\_charts = 1

Findings:

Only 2 tracks that appear in both apple and spotify charts. Daydreaming & Keep driving by Harry Styles.

1. Used a CASE statement with a derived table to find out avg energy for each danceability group.

SELECT

Danceability\_grp,

AVG(energy) AS avg\_energy

FROM (

SELECT

CASE

WHEN danceability >= 70 THEN 'High Danceability'

WHEN danceability BETWEEN 40 AND 69 THEN 'Medium Danceability'

ELSE 'Low Danceability'

END AS Danceability\_grp,

energy

FROM tbl\_spotify\_most\_played\_23

) AS sb\_query

GROUP BY Danceability\_grp

ORDER BY avg\_energy

Findings:

Low danceability – 51 avg\_energy, Moderate Danceability – 63, High Danceablity

1. Used OVER clause with RANK window function to figure out the ranks of the tracks by streams within each month.

SELECT released\_month, track\_name, streams,

RANK() OVER (PARTITION BY released\_month ORDER BY streams DESC) AS monthly\_rank

FROM tbl\_spotify\_most\_played\_23

1. Categorized beats per minute using a Scalar UDF.

CREATE FUNCTION dbo.fn\_bpm\_category (@bpm int)

RETURNS NVARCHAR(20)

AS

BEGIN

DECLARE @category NVARCHAR(20)

IF @bpm < 100

SET @category = 'Slow'

ELSE IF @bpm >= 100 AND @bpm <= 120

SET @category = 'Moderate'

ELSE

SET @category = 'Fast'

RETURN @category

END

GO

SELECT track\_name, bpm, dbo.fn\_bpm\_category(bpm) AS BPM\_Category

FROM tbl\_spotify\_most\_played\_23