Based on the question **- What are the reasons why some customers received corrupt tracks when they placed their orders?** and studying the Chinook database, here are the three questions I have decided to go with:

**1. Which tracks are affected by corrupt files, and how many customers downloaded each of these tracks?**

**2. Are there any patterns in the corrupt tracks by genre, artist, or album that suggest a common source of corruption?**

**3. Are the corrupt tracks associated with certain devices, operating systems, or network conditions on the customer’s side?**

To proceed with the answers, I have created a table called 'CorruptTracks' by running the following code:

--Create the table 'CorruptTracks'

CREATE TABLE public."CorruptTracks"

(

"CorruptTrackId" SERIAL PRIMARY KEY,

"TrackId" INT REFERENCES public."Track"("TrackId"),

"CustomerId" INT REFERENCES public."Customer"("CustomerId"),

"DownloadDate" TIMESTAMP,

"DeviceType" VARCHAR(50),

"OperatingSystem" VARCHAR(50),

"NetworkCondition" VARCHAR(50),

"CountrySpecific" BOOLEAN,

"IssueDescription" TEXT

);

Then I inserted some hypothetical values into 'CorruptTracks' to be able to answer the questions:

--Insert hypothetical values in 'CorruptTracks'

INSERT INTO public."CorruptTracks"

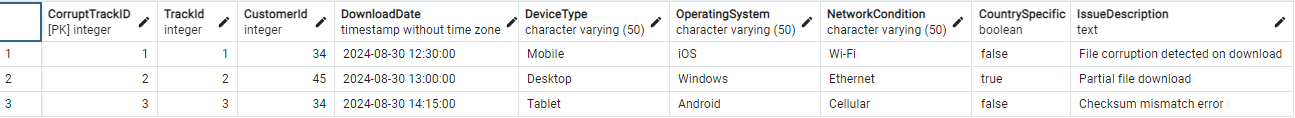
("TrackId", "CustomerId", "DownloadDate", "DeviceType", "OperatingSystem", "NetworkCondition", "CountrySpecific", "IssueDescription")

VALUES (1, 34, '2024-08-30 12:30:00', 'Mobile', 'iOS', 'Wi-Fi', FALSE, 'File corruption detected on download'),

(2, 45, '2024-08-30 13:00:00', 'Desktop', 'Windows', 'Ethernet', TRUE, 'Partial file download'),

(3, 34, '2024-08-30 14:15:00', 'Tablet', 'Android', 'Cellular', FALSE, 'Checksum mismatch error');

This is the output I got after running SELECT \* FROM public."CorruptTracks";



Now that we have the new table ready, let’s answer the questions.  
  
**1. Which tracks are affected by corrupt files, and how many customers downloaded each of these tracks?**

I joined ‘CorruptTracks’ table with the ‘Track’ table to get the name of the tracks and counts the number of customers who downloaded each corrupt track, using this query:

--Fetching name and count of corrupt tracks

SELECT

t."Name" AS "TrackName",

COUNT(ct."CustomerId") AS "TotalCustomersAffected"

FROM

public."CorruptTracks" ct

JOIN

public."Track" t ON ct."TrackId" = t."TrackId"

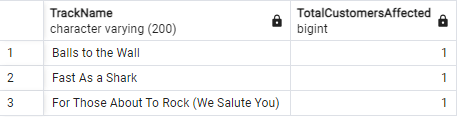
GROUP BY

t."Name"

ORDER BY

"TotalCustomersAffected" DESC;

Based on the hypothetical values in ‘CorruptTracks’ and the above query, this is the output I got:



**2. Are there any patterns in the corrupt tracks by genre, artist, or album that suggest a common source of corruption?**

I checked for patterns by genre, artist, and album by joining multiple related tables and counting occurrences of corrupt tracks.

--Finding patterns by joining multiple tables

SELECT

g."Name" AS "Genre",

ar."Name" AS "Artist",

al."Title" AS "Album",

COUNT(ct."TrackId") AS "CorruptTrackCount"

FROM

public."CorruptTracks" ct

JOIN

public."Track" t ON ct."TrackId" = t."TrackId"

JOIN

public."Album" al ON t."AlbumId" = al."AlbumId"

JOIN

public."Artist" ar ON al."ArtistId" = ar."ArtistId"

JOIN

public."Genre" g ON t."GenreId" = g."GenreId"

GROUP BY

g."Name", ar."Name", al."Title"

ORDER BY

"CorruptTrackCount" DESC;

If certain genres, artists, or albums have a disproportionately high number of corrupt tracks, this could indicate a systemic issue. By understanding which parts of the catalogue are affected, I can narrow down potential root causes.

This is the output of the above query:  
  
A screenshot of a computer

Description automatically generated

**3. Are the corrupt tracks associated with certain devices, operating systems, or network conditions on the customer’s side?**

I grouped the data by device type, operating system, and network condition to see if any of these factors are associated with the corrupt tracks.

--Checking customer data

SELECT

"DeviceType",

"OperatingSystem",

"NetworkCondition",

COUNT("TrackId") AS "CorruptTrackCount"

FROM

public."CorruptTracks"

GROUP BY

"DeviceType", "OperatingSystem", "NetworkCondition"

ORDER BY

"CorruptTrackCount" DESC;

This is the output of the above query:

A screenshot of a phone

Description automatically generated

**Best practices for writing SQL queries:**

Here are some best practices I observed while writing SQL queries:

1. **Use descriptive table and column names**: Ensure that table and column names are descriptive and follow a consistent naming convention (e.g., CorruptTracks, CustomerId).
2. **Set proper relationships**: Always establish foreign keys where relationships between tables exist to maintain data integrity (e.g., TrackId, CustomerId).
3. **Limit query to necessary columns**: Only select the columns you need in your queries to optimize performance and readability.
4. **Use aggregation carefully**: When aggregating data (e.g., counting corrupt tracks), ensure that the grouping is logical and accurate to avoid misleading results.
5. **Check for null values**: Be aware of potential null values in your data and handle them appropriately in your queries.

**Observations for writing better SQL scripts:**

1. **Understand the data Structure**: Before writing queries, ensure you fully understand the structure of the database and the relationships between tables.
2. **Use joins judiciously**: Always confirm that your joins are accurate, particularly when joining multiple tables, to avoid cartesian products and incorrect data.
3. **Test queries with small datasets**: When developing complex queries, start with a small dataset or use LIMIT to test and debug.
4. **Indexing for performance**: Use indexing on columns that are frequently used in JOIN conditions or as filters to enhance query performance.
5. **Optimize Group By clauses**: When using GROUP BY, ensure that the columns used are necessary and well-indexed to avoid performance issues.