Step 1: Database setup

```
import sqlite3
import pandas as pd
connection = sqlite3.connect("chinook.db")
tables_query = """
SELECT name
FROM sqlite_master
WHERE type='table';
"""
df_tables = pd.read_sql_query(tables_query, connection)
print("List of tables in chinook.db:")
display(df_tables)
```

```
List of tables in chinook.db:
```

	name
0	albums
1	sqlite_sequence
2	artists
3	customers
4	employees
5	genres
6	invoices
7	invoice_items
8	media_types
9	playlists
10	playlist_track
11	tracks
12	sqlite_stat1

Next steps: Generate code with df_tables View recommended plots New interactive sheet

Step 2: Information extraction from SQL database

```
#Avikalp Karrahe
initial = 'K'
# Query customers with LastName starting with 'K'
query_customers = f"""
SELECT CustomerId, FirstName, LastName, Email, Country
FROM customers
WHERE LastName LIKE '{initial}%'
;
"""
df_customers_k = pd.read_sql_query(query_customers, connection)
if df_customers_k.empty:
    print(f"No customers found with last name starting '{initial}'.")
    # If none found, we move to the nearest letter logic here,'J%' or 'L%'
    # choose 'L' next:
    nearest_letter = 'L'
    query_customers = f"""
    SELECT CustomerId, FirstName, LastName, Email, Country
    FROM customers
    WHERE LastName LIKE '{nearest_letter}%'
    ;
    """
    df_customers_k = pd.read_sql_query(query_customers, connection)
    if df_customers_k.empty:
        print(f"No customers found with last name starting '{nearest_letter}' either.")
    else:
        print(f"Using nearest letter '{nearest_letter}' instead.\n")
print("Customers matching LastName initial 'K' (or nearest):")
display(df_customers_k)
```

```
Customers matching LastName initial 'K' (or nearest):
```

	CustomerId	FirstName	LastName	Email	Country
0	2	Leonie	Köhler	leonekohler@surfeu.de	Germany
1	45	Ladislav	Kovács	ladislav_kovacs@apple.hu	Hungary

Next steps: Generate code with df_customers_k | View recommended plots | New interactive sheet

```
if not df_customers_k.empty:
    # Get the list of customer IDs
    customer_ids = tuple(df_customers_k['CustomerId'].tolist())
    query_tracks = f"""
    SELECT DISTINCT t.TrackId,
           t.Name as TrackName,
           t.AlbumId,
           t.Milliseconds,
           t.UnitPrice as TrackUnitPrice,
           i.InvoiceId,
           c.CustomerId
    FROM customers c
    JOIN invoices i
      ON c.CustomerId = i.CustomerId
    JOIN invoice_items ii
      ON i.InvoiceId = ii.InvoiceId
    JOIN tracks t
      ON ii.TrackId = t.TrackId
    WHERE c.CustomerId IN {customer_ids}
    ORDER BY t.Name
    ;
    """
    df_tracks = pd.read_sql_query(query_tracks, connection)
    print("Tracks purchased by these customers:")
    display(df_tracks)
else:
    df_tracks = pd.DataFrame()
    print("No matching customers to retrieve tracks for.")
```

```
Tracks purchased by these customers:
```

	TrackId	TrackName	AlbumId	Milliseconds	TrackUnitPrice	InvoiceId	CustomerId
0	918	Alberta	73	222406	0.99	241	2
1	2274	All Dead, All Dead	186	190119	0.99	280	45
2	385	All Star	33	176326	0.99	12	2
3	2	Balls to the Wall	2	342562	0.99	1	2
4	2130	Beach Sequence	176	212297	0.99	67	2
...	...	...	...	...	...	...	...
71	2154	Untitled	178	122801	0.99	67	2
72	376	Vôo Sobre o Horizonte	33	225097	0.99	12	2
73	198	When My Left Eye Jumps	20	235311	0.99	219	2
74	2166	World Wide Suicide	179	209188	0.99	67	2
75	349	You Shook Me(2)	30	619467	0.99	12	2

76 rows × 7 columns

Next steps: Generate code with df_tracks | View recommended plots | New interactive sheet

```
if not df_tracks.empty:
    # Extract unique track IDs
    track_ids = tuple(df_tracks['TrackId'].unique().tolist())
    query_artists = f"""
    SELECT DISTINCT ar.ArtistId,
           ar.Name as ArtistName
    FROM tracks t
    JOIN albums al
      ON t.AlbumId = al.AlbumId
    JOIN artists ar
      ON al.ArtistId = ar.ArtistId
    WHERE t.TrackId IN {track_ids}
    ORDER BY ar.Name ASC
    ;
    """
    df_artists = pd.read_sql_query(query_artists, connection)
    print("Unique Artists associated with those purchased tracks (sorted):")
    display(df_artists)
else:
    df_artists = pd.DataFrame()
    print("No tracks found, so no artists to list.")
```

```
Unique Artists associated with those purchased tracks (sorted):
    ArtistId                     ArtistName
 0         2                         Accept
 1         6            Antônio Carlos Jobim
 2       158   Battlestar Galactica (Classic)
 3        13                     Body Count
 4        14                 Bruce Dickinson
 5        15                       Buddy Guy
 6        81                    Eric Clapton
 7        23 Frank Zappa & Captain Beefheart
 8       148                          Heroes
 9        90                     Iron Maiden
10        91                     James Brown
11        92                      Jamiroquai
12        52                            Kiss
13        22                     Led Zeppelin
14       100                    Lenny Kravitz
15       149                            Lost
16        24                    Marcos Valle
17        50                       Metallica
18       111                         O Terço
19       116                      Passengers
20       117                    Paul D'Ianno
21       118                       Pearl Jam
22        51                           Queen
23       156                      The Office
24       144                         The Who
25       146                           Titãs
26       150                              U2
27        21                  Various Artists
28       153                  Velvet Revolver
29        72               Vinícius De Moraes
30       155                  Zeca Pagodinho
```

Next steps: Generate code with df_artists | View recommended plots | New interactive sheet

Step 3: Scraping wikipedia

```
import requests
from bs4 import BeautifulSoup
def scrape_wiki_info(url: str, is_band: bool = True) -> dict:
    # Initialize default return structure
    if is_band:
        result = {
            'date_of_formation': 'N/A',
            'place_of_origin': 'N/A',
            'number_of_members': 'N/A',
            'labels': 'N/A'
        }
    else:
        result = {
            'date_of_birth': 'N/A',
            'place_of_birth': 'N/A',
            'number_of_children': 'N/A',
            'labels': 'N/A'
        }
    # Download the page
    response = requests.get(url)
    if response.status_code != 200:
        print(f"Could not retrieve page: {url}")
        return result
    soup = BeautifulSoup(response.text, 'html.parser')
    infobox = soup.find('table', {'class': 'infobox'})
    if not infobox:
        print(f"No infobox found on the Wikipedia page: {url}")
        return result
    rows = infobox.find_all('tr')
    # Helper function to clean text
    def clean_text(txt):
        return txt.replace('\xa0', ' ').strip()
    for row in rows:
        header = row.find('th')
        cell = row.find('td')
        if not header or not cell:
            continue
        header_text = clean_text(header.get_text()).lower()
        cell_text = clean_text(cell.get_text())
        # ----- If it's a band -----
        if is_band:
            if 'years active' in header_text or 'formed' in header_text:
                result['date_of_formation'] = cell_text.split('–')[0].strip()
            if 'origin' in header_text:
                result['place_of_origin'] = cell_text
            if 'members' in header_text:
                members_list = cell.find_all('li')
                if members_list:
                    result['number_of_members'] = str(len(members_list))
                else:
                    # fallback to a simple count of line breaks
                    splitted = cell_text.split('\n')
                    if len(splitted) > 1:
                        result['number_of_members'] = str(len(splitted))
                    else:
                        result['number_of_members'] = "N/A"
            if 'labels' in header_text:
                labels_list = [clean_text(label.get_text()) for label in cell.find_all('li')]
                if not labels_list:
                    labels_list = [x.strip() for x in cell_text.split('\n') if x.strip() != '']
                result['labels'] = labels_list if labels_list else 'N/A'
        # ----- If it's a solo artist -----
        else:
            if 'born' in header_text:
                result['date_of_birth'] = cell_text.split('(')[0].replace('Born', '').strip()
                parentheses = row.find('span', {'class': 'birthplace'})
                if parentheses:
                    result['place_of_birth'] = clean_text(parentheses.get_text())
            if 'children' in header_text:
                result['number_of_children'] = cell_text
            if 'labels' in header_text:
                labels_list = [clean_text(label.get_text()) for label in cell.find_all('li')]
                if not labels_list:
                    labels_list = [x.strip() for x in cell_text.split('\n') if x.strip() != '']
                result['labels'] = labels_list if labels_list else 'N/A'
    return result

accept_url = "https://en.wikipedia.org/wiki/Accept_(band)"
accept_info = scrape_wiki_info(accept_url, is_band=True)

print("Accept (Band) Info")
for k, v in accept_info.items():
    print(f"{k}: {v}")

travis_url = "https://en.wikipedia.org/wiki/Travis_Scott"
travis_info = scrape_wiki_info(travis_url, is_band=False)

print("\nTravis Scott (Solo Artist) Info")
for k, v in travis_info.items():
    print(f"{k}: {v}")
```

```
Accept (Band) Info
date_of_formation: 1976
place_of_origin: Solingen, West Germany
number_of_members: 13
labels: ['Nuclear Blast', 'Portrait/Epic', 'RCA Germany', 'PolyGram', 'Passport', 'Napalm']

Travis Scott (Solo Artist) Info
date_of_birth: Jacques Bermon Webster II
place_of_birth: N/A
number_of_children: 2
labels: ['Grand Hustle', 'Epic', 'Very GOOD Beats', 'Cactus Jack[7][8]']
```

Step 4: API call to extract more info

```
def itunes_search(artist_name: str) -> pd.DataFrame:
    # Construct the query
    base_url = "https://itunes.apple.com/search"
    params = {
        "term": artist_name,
        "entity": "musicTrack",
        "limit": 50
    }
    response = requests.get(base_url, params=params)
    if response.status_code != 200:
        print(f"Error: iTunes API returned status code {response.status_code}")
        return pd.DataFrame()
    data = response.json()
    results = data.get("results", [])
    # Collect relevant fields
    records = []
    for r in results:
        track_name = r.get("trackName", None)
        release_date = r.get("releaseDate", None)
        track_price = r.get("trackPrice", None)
        records.append({
            "trackName": track_name,
            "releaseDate": release_date,
            "trackPrice": track_price
        })
    df = pd.DataFrame(records)
    return df
print("iTunes Search Results for 'Accept'") # From Step 2
df_accept = itunes_search("Accept")
display(df_accept.head(10))
print("\niTunes Search Results for 'Travis Scott'")
df_travis = itunes_search("Travis Scott")
display(df_travis.head(10))
```

```
iTunes Search Results for 'Accept'
                     trackName           releaseDate  trackPrice
0                       Accept  2007-02-23T12:00:00Z        0.99
1                       Accept  2019-03-01T12:00:00Z       -1.00
2               I Am Your Future  2007-02-23T12:00:00Z        0.99
3       I'm Your Future (screwed)  2007-02-23T12:00:00Z        0.99
4            Strength In Numbers  2007-02-23T12:00:00Z        0.99
5                       Accept  2020-06-20T12:00:00Z        0.99
6  Accept (Remixed & Remastered)  2024-12-13T12:00:00Z        1.29
7                          9mm  2007-02-23T12:00:00Z        0.99
8                 Packin Da Gat  2007-02-23T12:00:00Z        0.99
9                       Accept  2014-03-11T12:00:00Z        0.99

iTunes Search Results for 'Travis Scott'
                                           trackName           releaseDate  trackPrice
0  Take What You Want (feat. Ozzy Osbourne & Trav...  2019-09-06T12:00:00Z        1.29
1                 Love Galore (feat. Travis Scott)  2017-04-28T12:00:00Z        1.29
2                  Sky Walker (feat. Travis Scott)  2017-08-24T12:00:00Z        1.29
3                   Bake Sale (feat. Travis Scott)  2016-01-21T08:00:00Z        1.29
4                    Let It Fly (feat. Travis Scott)  2018-09-28T12:00:00Z        1.29
5      On Everything (feat. Travis Scott, Rick Ross &...  2017-06-23T07:00:00Z        1.29
6           Tourist (feat. Travis Scott & Lil Wayne)  2016-07-29T07:00:00Z        1.29
7           It's Secured (feat. Nas & Travis Scott)  2017-06-23T07:00:00Z        1.29
8             Ghostface Killers (feat. Travis Scott)  2017-12-23T12:00:00Z        1.29
9           Don't Quit (feat. Travis Scott & Jeremih)  2017-06-23T07:00:00Z        1.29
```

Step 5: Combining information

```python
def combine_artist_info(artists_list, wiki_info_list, is_band_list):
    master_records = []
    # Ensure all lists align
    for idx, artist_name in enumerate(artists_list):
        wiki_url = wiki_info_list[idx]
        band_flag = is_band_list[idx]
        # 1) Scrape Wikipedia
        wiki_data = scrape_wiki_info(wiki_url, is_band=band_flag)
        # 2) iTunes search
        df_itunes = itunes_search(artist_name)
        # If iTunes had no records, we'll still build at least one row
        if df_itunes.empty:
            combined_row = {**wiki_data}
            combined_row["ArtistName"] = artist_name
            combined_row["trackName"] = None
            combined_row["releaseDate"] = None
            combined_row["trackPrice"] = None
            master_records.append(combined_row)
        else:
            # For each track row, combine with wiki data
            for _, row in df_itunes.iterrows():
                combined_row = {**wiki_data}
                combined_row["ArtistName"] = artist_name
                combined_row["trackName"] = row["trackName"]
                combined_row["releaseDate"] = row["releaseDate"]
                combined_row["trackPrice"] = row["trackPrice"]
                master_records.append(combined_row)
    # Final DataFrame
    df_final = pd.DataFrame(master_records)
    # Reorder columns in a more logical sequence
    column_order = [
        "ArtistName",
        "date_of_birth",
        "place_of_birth",
        "number_of_children",
        "date_of_formation",
        "place_of_origin",
        "number_of_members",
        "labels",
        "trackName",
        "releaseDate",
        "trackPrice"
    ]
    df_final = df_final[[c for c in column_order if c in df_final.columns]]
    return df_final

artists_example = ["Accept", "Travis Scott"]
wiki_urls_example = [
    "https://en.wikipedia.org/wiki/Accept_(band)",  # band
    "https://en.wikipedia.org/wiki/Travis_Scott"    # solo artist
]
is_band_example = [True, False]

df_final_report = combine_artist_info(artists_example, wiki_urls_example,
is_band_example)

print("FINAL COMBINED REPORT")
display(df_final_report.head(15))  # First 15 rows
```

```
FINAL COMBINED REPORT
```

1 to 15 of 15 entries Filter

index	ArtistName	date_of_birth	place_of_birth	number_of_children	date_of_formation	place_of_origin	number_of_members	labels
0	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
1	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
2	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
3	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
4	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
5	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
6	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
7	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
8	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
9	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
10	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
11	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
12	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo
13	Accept	NaN	NaN	NaN	1976	Solingen, West Germany	13	Nuclear Blast,Portrait/Epic, Germany,PolyGram,Passpo