Introduction_to_Relational_Databases

Pop quiz: The relational model

Which of the following is not part of the relational model?

Options:

1. Each row or record in a table represents an instance of an entity type.
2. Each column in a table represents an attribute or feature of an instance.
3. Every table contains a primary key column, which has a unique entry for each row.
4. A database consists of at least 3 tables.
5. There are relations between tables.

Answer: 4

1.1 Creating a database engine in Python

Here, you're going to fire up your very first SQL engine. You'll create an engine to connect to the SQLite database `'Chinook.sqlite'`, which is in your working directory. Remember that to create an engine to connect to `'Northwind.sqlite'`, Hugo executed the command

> engine = create_engine('sqlite:///Northwind.sqlite')

Here, `'sqlite:///Northwind.sqlite'` is called the connection string to the SQLite database `Northwind.sqlite`. A little bit of background on the Chinook database (https://github.com/lerocha/chinook-database): the Chinook database contains information about a semi-fictional digital media store in which media data is real and customer, employee and sales data has been manually created.

Why the name Chinook, you ask? According to their website (https://github.com/lerocha/chinook-database),

> The name of this sample database was based on the Northwind database. Chinooks are winds in the interior West of North America, where the Canadian Prairies and Great Plains meet various mountain ranges. Chinooks are most prevalent over southern Alberta in Canada. Chinook is a good name choice for a database that intends to be an alternative to Northwind.

Instructions

- Import the function `create_engine` from the module `sqlalchemy`.
- Create an engine to connect to the SQLite database `'Chinook.sqlite'` and assign it to `engine`.

In [1]:

```
# Import necessary module
from sqlalchemy import create_engine

# Create engine: engine
engine = create_engine('sqlite:///Chinook.sqlite')
```

1.2 What are the tables in the database?

In this exercise, you'll once again create an engine to connect to `'Chinook.sqlite'`. Before you can get any data out of the database, however, you'll need to know what tables it contains!

To this end, you'll save the table names to a list using the method `table_names()` on the engine and then you will print the list.

Instructions:

- Import the function `create_engine` from the module `sqlalchemy`.
- Create an engine to connect to the SQLite database `'Chinook.sqlite'` and assign it to `engine`.
- Using the method `table_names()` on the engine engine, assign the table names of `'Chinook.sqlite'` to the variable `table_names`.
- Print the object `table_names` to the shell.

In [2]:

```
# Import necessary module
from sqlalchemy import create_engine

# Create engine: engine
engine = create_engine('sqlite:///Chinook.sqlite')

# Save the table names to a list:

table_names = engine.table_names()
# Print the table names to the shell
print(table_names)
```

```
['Album', 'Artist', 'Customer', 'Employee', 'Genre', 'Invoice', 'InvoiceLin
e', 'MediaType', 'Playlist', 'PlaylistTrack', 'Track']
```

2. Querying relational databases in Python

2.1 The Hello World of SQL Queries!

Now, it's time for liftoff! In this exercise, you'll perform the Hello World of SQL queries, `SELECT`, in order to retrieve all columns of the table `Album` in the Chinook database. Recall that the query `SELECT *` selects all columns.

Instructions

- Open the engine connection as con using the method `connect()` on the engine.
- Execute the query that selects ALL columns from the `Album` table. Store the results in `rs`.
- Store all of your query results in the DataFrame `df` by applying the `fetchall()` method to the results `rs`.
- Close the connection!

In [3]:

```
# Import packages
from sqlalchemy import create_engine
import pandas as pd

# Create engine: engine
engine = create_engine('sqlite:///Chinook.sqlite')

# Open engine connection:
con = engine.connect()

# Perform query: rs
rs = con.execute('SELECT * FROM Album')

# Save results of the query to DataFrame: df
df = pd.DataFrame(rs.fetchall())

# Close connection
con.close()

# Print head of DataFrame df
print(df.head())
```

```
   0                                     1  2
0  1  For Those About To Rock We Salute You  1
1  2                      Balls to the Wall  2
2  3                      Restless and Wild  2
3  4                      Let There Be Rock  1
4  5                               Big Ones  3
```

2.2 Customizing the Hello World of SQL Queries

Congratulations on executing your first SQL query! Now you're going to figure out how to customize your query in order to:

- Select specified columns from a table;
- Select a specified number of rows;
- Import column names from the database table.
- Recall that Hugo performed a very similar query customization in the video:

> engine = create_engine('sqlite:///Northwind.sqlite')

> with engine.connect() as con: rs = con.execute("SELECT OrderID, OrderDate, ShipName FROM Orders") df = pd.DataFrame(rs.fetchmany(size=5)) df.columns = rs.keys()

Packages have already been imported as follows:

> from sqlalchemy import create_engine import pandas as pd The engine has also already been created:

> engine = create_engine('sqlite:///Chinook.sqlite')

The engine connection is already open with the statement

```
with engine.connect() as con:
```

All the code you need to complete is within this context.

Instructions

- Execute the SQL query that selects the columns `LastName` and `Title` from the `Employee` table. Store the results in the variable rs.
- Apply the method `fetchmany()` to `rs` in order to retrieve 3 of the records. Store them in the DataFrame `df`.
- Using the `rs` object, set the DataFrame's column names to the corresponding names of the table columns.

In [4]:

```
# Open engine in context manager
# Perform query and save results to DataFrame: df
with engine.connect() as con:
    rs = con.execute("SELECT LastName, Title FROM Employee")
    df = pd.DataFrame(rs.fetchmany(size=3))
    df.columns = rs.keys()

# Print the length of the DataFrame df
print(len(df))

# Print the head of the DataFrame df
print(df.head())
```

```
3
  LastName                Title
0    Adams      General Manager
1  Edwards        Sales Manager
2  Peacock  Sales Support Agent
```

3.2 Filtering your database records using SQL's WHERE

You can now execute a basic SQL query to select records from any table in your database and you can also perform simple query customizations to select particular columns and numbers of rows.

There are a couple more standard SQL query chops that will aid you in your journey to becoming an SQL ninja.

Let's say, for example that you wanted to get all records from the Customer table of the Chinook database for which the Country is `'Canada'`. You can do this very easily in SQL using a SELECT statement followed by a `WHERE` clause as follows:

```
SELECT * FROM Customer WHERE Country = 'Canada'
```

In fact, you can filter any `SELECT` statement by any condition using a `WHERE` clause. This is called filtering your records.

Packages are already imported as follows:

import pandas as pd from sqlalchemy import create_engine Query away!

Instructions

- Complete the argument of `create_engine()` so that the engine for the SQLite database `'Chinook.sqlite'` is created.
- Execute the query that selects all records from the `Employee` table where `'EmployeeId'` is greater than or equal to `6`. Use the `>=` operator and assign the results to `rs`.
- Apply the method `fetchall()` to `rs` in order to fetch all records in `rs`. Store them in the DataFrame `df`.
- Using the `rs` object, set the DataFrame's column names to the corresponding names of the table columns.

In [5]:

```
# Create engine: engine
engine = create_engine('sqlite:///Chinook.sqlite')

# Open engine in context manager
# Perform query and save results to DataFrame: df
with engine.connect() as con:
    rs = con.execute('SELECT * FROM Employee WHERE EmployeeId>= 6' )
    df = pd.DataFrame(rs.fetchall())
    df.columns = rs.keys()

# Print the head of the DataFrame df
print(df.head())
```

```
   EmployeeId  LastName FirstName       Title  ReportsTo            BirthDat
e  \
0           6  Mitchell   Michael  IT Manager           1  1973-07-01 00:00:0
0
1           7      King    Robert    IT Staff           6  1970-05-29 00:00:0
0
2           8  Callahan     Laura    IT Staff           6  1968-01-09 00:00:0
0

              HireDate                     Address        City State Countr
y  \
0  2003-10-17 00:00:00         5827 Bowness Road NW     Calgary    AB  Canad
a
1  2004-01-02 00:00:00  590 Columbia Boulevard West  Lethbridge    AB  Canad
a
2  2004-03-04 00:00:00                 923 7 ST NW  Lethbridge    AB  Canad
a

  PostalCode              Phone                Fax                    Email
0    T3B 0C5  +1 (403) 246-9887  +1 (403) 246-9899  michael@chinookcorp.com
1    T1K 5N8  +1 (403) 456-9986  +1 (403) 456-8485   robert@chinookcorp.com
2    T1H 1Y8  +1 (403) 467-3351  +1 (403) 467-8772    laura@chinookcorp.com
```

In [6]:

```
with engine.connect() as con:
    rs = con.execute('SELECT * FROM Employee WHERE EmployeeId>= 6' )
    print(rs.keys())
```

```
['EmployeeId', 'LastName', 'FirstName', 'Title', 'ReportsTo', 'BirthDate',
'HireDate', 'Address', 'City', 'State', 'Country', 'PostalCode', 'Phone', 'F
ax', 'Email']
```

3.3 Ordering your SQL records with ORDER BY

You can also order your SQL query results. For example, if you wanted to get all records from the Customer table of the Chinook database and order them in increasing order by the column SupportRepId, you could do so with the following query:

> "SELECT * FROM Customer ORDER BY SupportRepId"

In fact, you can order any `SELECT` statement by any column.

Packages are already imported as follows:

> import pandas as pd from sqlalchemy import create_engine

Get querying!

Instructions

- Using the function `create_engine()`, create an engine for the SQLite database Chinook.sqlite and assign it to the variable engine.
- In the context manager, execute the query that selects all records from the Employee table and orders them in increasing order by the column `BirthDate`. Assign the result to `rs`.
- In a call to `pd.DataFrame()`, apply the method `fetchall()` to `rs` in order to fetch all records in `rs`. Store them in the DataFrame `df`.
- Set the DataFrame's column names to the corresponding names of the table columns.

In [7]:

```
# Create engine: engine

engine = create_engine('sqlite:///Chinook.sqlite')
# Open engine in context manager
with engine.connect() as con:
    rs = con.execute("SELECT * From Employee ORDER BY BirthDate ")
    df = pd.DataFrame(rs.fetchall())

    # Set the DataFrame's column names

    df.columns = rs.keys()
# Print head of DataFrame
print(df.head())
```

```
   EmployeeId  LastName FirstName                Title  ReportsTo  \
0           4      Park  Margaret  Sales Support Agent        2.0
1           2   Edwards     Nancy        Sales Manager        1.0
2           1     Adams    Andrew      General Manager        NaN
3           5   Johnson     Steve  Sales Support Agent        2.0
4           8  Callahan     Laura             IT Staff        6.0

             BirthDate              HireDate              Address        City
\
0  1947-09-19 00:00:00  2003-05-03 00:00:00     683 10 Street SW     Calgary
1  1958-12-08 00:00:00  2002-05-01 00:00:00         825 8 Ave SW     Calgary
2  1962-02-18 00:00:00  2002-08-14 00:00:00  11120 Jasper Ave NW    Edmonton
3  1965-03-03 00:00:00  2003-10-17 00:00:00         7727B 41 Ave     Calgary
4  1968-01-09 00:00:00  2004-03-04 00:00:00          923 7 ST NW  Lethbridge

  State Country PostalCode              Phone                Fax  \
0    AB  Canada    T2P 5G3  +1 (403) 263-4423  +1 (403) 263-4289
1    AB  Canada    T2P 2T3  +1 (403) 262-3443  +1 (403) 262-3322
2    AB  Canada    T5K 2N1  +1 (780) 428-9482  +1 (780) 428-3457
3    AB  Canada    T3B 1Y7   1 (780) 836-9987   1 (780) 836-9543
4    AB  Canada    T1H 1Y8  +1 (403) 467-3351  +1 (403) 467-8772

                     Email
0  margaret@chinookcorp.com
1     nancy@chinookcorp.com
2    andrew@chinookcorp.com
3     steve@chinookcorp.com
4     laura@chinookcorp.com
```

4. Querying relational databases directly with pandas

4.1 Pandas and The Hello World of SQL Queries!

Here, you'll take advantage of the power of `pandas` to write the results of your SQL query to a DataFrame in one swift line of Python code!

You'll first import `pandas` and create the SQLite `'Chinook.sqlite'` engine. Then you'll query the database to select all records from the Album table.

Recall that to select all records from the Orders table in the Northwind database, Hugo executed the following command:

```
df = pd.read_sql_query("SELECT * FROM Orders", engine)
```

Instructions

- Import the pandas package using the alias pd.
- Using the function `create_engine()`, create an engine for the SQLite database `Chinook.sqlite` and assign it to the variable engine.
- Use the pandas function `read_sql_query()` to assign to the variable df the DataFrame of results from the following query: select all records from the table Album.

In [8]:

```
# Import packages
from sqlalchemy import create_engine
import pandas as pd

# Create engine:
engine = create_engine("sqlite:///Chinook.sqlite")

# Execute query and store records in DataFrame: df
df = pd.read_sql_query("SELECT * FROM Album",engine)

# Print head of DataFrame
print(df.head())

# Open engine in context manager and store query result in df1
with engine.connect() as con:
    rs = con.execute("SELECT * FROM Album")
    df1 = pd.DataFrame(rs.fetchall())
    df1.columns = rs.keys()

# Confirm that both methods yield the same result
print(df.equals(df1))
```

```
   AlbumId                                 Title  ArtistId
0        1  For Those About To Rock We Salute You         1
1        2                      Balls to the Wall         2
2        3                      Restless and Wild         2
3        4                      Let There Be Rock         1
4        5                            Big Ones         3
True
```

Pandas for more complex querying

Here, you'll become more familiar with the pandas function `read_sql_query()` by using it to execute a more complex query: a `SELECT` statement followed by both a `WHERE` clause AND an `ORDER BY` clause.

You'll build a DataFrame that contains the rows of the Employee table for which the EmployeeId is greater than or equal to `6` and you'll order these entries by `BirthDate`.

Instructions

- Using the function create_engine(), create an engine for the SQLite database Chinook.sqlite and assign it to the variable engine.
- Use the `pandas` function `read_sql_query()` to assign to the variable `df` the DataFrame of results from the following query: **select** all records from the `Employee` table **where** the `EmployeeId` is greater than or equal to `6` and **ordered by** `BirthDate` (make sure to use `WHERE` and `ORDER BY` in this precise order).

In [9]:

```
# Import packages
from sqlalchemy import create_engine
import pandas as pd

# Create engine: engine

engine = create_engine("sqlite:///Chinook.sqlite")
# Execute query and store records in DataFrame: df

df=pd.read_sql_query("SELECT * FROM Employee WHERE EmployeeId>=6 ORDER BY BirthDate",e

# Print head of DataFrame
print(df.head())

```

```
   EmployeeId  LastName FirstName       Title  ReportsTo            BirthDat
e  \
0           8  Callahan     Laura    IT Staff           6  1968-01-09 00:00:0
0
1           7      King    Robert    IT Staff           6  1970-05-29 00:00:0
0
2           6  Mitchell   Michael  IT Manager           1  1973-07-01 00:00:0
0

              HireDate                     Address         City State Countr
y  \
0  2004-03-04 00:00:00                   923 7 ST NW  Lethbridge    AB  Canad
a
1  2004-01-02 00:00:00  590 Columbia Boulevard West  Lethbridge    AB  Canad
a
2  2003-10-17 00:00:00         5827 Bowness Road NW     Calgary    AB  Canad
a

  PostalCode              Phone                Fax                    Email
0    T1H 1Y8  +1 (403) 467-3351  +1 (403) 467-8772    laura@chinookcorp.com
1    T1K 5N8  +1 (403) 456-9986  +1 (403) 456-8485   robert@chinookcorp.com
2    T3B 0C5  +1 (403) 246-9887  +1 (403) 246-9899  michael@chinookcorp.com
```

In [10]:

```
# Import packages
from sqlalchemy import create_engine
import pandas as pd

# Create engine: engine

engine = create_engine("sqlite:///Chinook.sqlite")
# Execute query and store records in DataFrame: df

df=pd.read_sql_query("SELECT * From Artist",engine)
df1=pd.read_sql_query("SELECT * From Album",engine)
# Print head of DataFrame
print(df.head())
print(df1.head())
```

```
   ArtistId              Name
0         1             AC/DC
1         2            Accept
2         3          Aerosmith
3         4  Alanis Morissette
4         5    Alice In Chains
   AlbumId                                 Title  ArtistId
0        1  For Those About To Rock We Salute You         1
1        2                      Balls to the Wall         2
2        3                      Restless and Wild         2
3        4                      Let There Be Rock         1
4        5                            Big Ones            3
```

The power of SQL lies in relationships between tables: INNER JOIN

Here, you'll perform your first INNER JOIN! You'll be working with your favourite SQLite database, Chinook.sqlite. For each record in the Album table, you'll extract the Title along with the Name of the Artist. The latter will come from the Artist table and so you will need to INNER JOIN these two tables on the ArtistID column of both.

Recall that to INNER JOIN the Orders and Customers tables from the Northwind database, Hugo executed the following SQL query:

> "SELECT OrderID, CompanyName FROM Orders INNER JOIN Customers on Orders.CustomerID = Customers.CustomerID"

The following code has already been executed to import the necessary packages and to create the engine:

> import pandas as pd from sqlalchemy import create_engine engine = create_engine('sqlite:///Chinook.sqlite')

Instructions

- Assign to rs the results from the following query: select all the records, extracting the `Title` of the record and `Name` of the artist of each record from the `Album` table and the `Artist` table, respectively. To do so, `INNER JOIN` these two tables on the `ArtistID` column of both.

- In a call to `pd.DataFrame()`, apply the method `fetchall()` to `rs` in order to fetch all records in `rs`. Store them in the DataFrame `df`.
- Set the DataFrame's column names to the corresponding names of the table columns.

In [11]:

```
# Open engine in context manager
engine = create_engine("sqlite:///Chinook.sqlite")

# Perform query and save results to DataFrame: df
with engine.connect() as con:
    rs = con.execute("SELECT Title,Name FROM Album INNER JOIN Artist on Album.ArtistId
    df = pd.DataFrame(rs.fetchall())
    df.columns = rs.keys()
    df.head()

# Print head of DataFrame df
print(df.head())

```

```
                                Title       Name
0  For Those About To Rock We Salute You      AC/DC
1                      Balls to the Wall     Accept
2                      Restless and Wild     Accept
3                      Let There Be Rock      AC/DC
4                               Big Ones  Aerosmith
```

Filtering your INNER JOIN

Congrats on performing your first INNER JOIN! You're now going to finish this chapter with one final exercise in which you perform an INNER JOIN and filter the result using a WHERE clause.

Recall that to INNER JOIN the Orders and Customers tables from the Northwind database, Hugo executed the following SQL query:

"SELECT OrderID, CompanyName FROM Orders INNER JOIN Customers on Orders.CustomerID = Customers.CustomerID" The following code has already been executed to import the necessary packages and to create the engine:

import pandas as pd from sqlalchemy import create_engine engine = create_engine('sqlite:///Chinook.sqlite') Instructions 100 XP Use the pandas function read_sql_query() to assign to the variable df the DataFrame of results from the following query: select all records from PlaylistTrack INNER JOIN Track on PlaylistTrack.TrackId = Track.TrackId that satisfy the condition Milliseconds < 250000.

In [12]:

```
# Execute query and store records in DataFrame: df

df = pd.read_sql_query('SELECT * FROM PlaylistTrack INNER JOIN Track on PlaylistTrack.T

# Print head of DataFrame
print(df.head())
```

```
   PlaylistId  TrackId  TrackId             Name  AlbumId  MediaTypeId  \
0           1     3390     3390  One and the Same      271            2
1           1     3392     3392     Until We Fall      271            2
2           1     3393     3393     Original Fire      271            2
3           1     3394     3394       Broken City      271            2
4           1     3395     3395          Somedays      271            2

   GenreId Composer  Milliseconds    Bytes  UnitPrice
0       23     None        217732  3559040       0.99
1       23     None        230758  3766605       0.99
2       23     None        218916  3577821       0.99
3       23     None        228366  3728955       0.99
4       23     None        213831  3497176       0.99
```