Exercise1

Creating a database engine

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: 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,

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 [ ]:

```
# Import necessary module
from ____ import ____

# Create engine: engine


#_____________#
#Solutions

# Import necessary module
from sqlalchemy import create_engine

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

Exercise2

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 [ ]:

```
# Import necessary module


# Create engine: engine


# Save the table names to a list: table_names


# Print the table names to the shell
print(____)



#____________#
#Solutions

# 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
table_names=engine.table_names()

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

Exercise3

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 [ ]:

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

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

# Open engine connection: con


# Perform query: rs
rs = con.execute(____)

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

# Close connection


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



#____________#
#Solutions

# Import packages
from sqlalchemy import create_engine
import pandas as pd

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

# Open engine connection: con
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())
```

Exercise4

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 [ ]:

```
# Open engine in context manager
# Perform query and save results to DataFrame: df
with engine.connect() as con:
    rs = ____
    df = pd.DataFrame(____)
    df.columns = ____

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

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


#_____________#
#Solutions

# 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())
```

Exercise5

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.

In this interactive exercise, you'll select all records of the Employee table for which 'EmployeeId' is greater than or equal to 6.

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 [ ]:

```
# Create engine: engine
engine = create_engine(____)

# Open engine in context manager
# Perform query and save results to DataFrame: df
with engine.connect() as con:
    rs = con.execute(____)
    df = pd.DataFrame(____)
    df.columns = ____

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


#_____________#
#Solutions

# 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())
```

Exercise6

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.

In this interactive exercise, you'll select all records of the Employee table and order them in increasing order by the column BirthDate.

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 [ ]:

```
# Create engine: engine


# Open engine in context manager
with engine.connect() as con:
    rs = ____
    df = ____

    # Set the DataFrame's column names


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



#____________#
#Solutions

# 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())
```

Exercise7

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.
- The remainder of the code is included to confirm that the DataFrame created by this method is equal to that created by the previous method that you learned.

In [ ]:

```
# Import packages
from sqlalchemy import create_engine
import ____ as ____

# Create engine: engine


# Execute query and store records in DataFrame: df
df = pd.read_sql_query(____, ____)

# 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))


#_____________#
#Solutions

# 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 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))
```

Exercise8

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 [ ]:

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

# Create engine: engine

# Execute query and store records in DataFrame: df

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



#_____________#
#Solutions

# 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",
    engine
)

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

Exercise9

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 [ ]:

```
# Open engine in context manager
# Perform query and save results to DataFrame: df
with engine.connect() as con:
    ____
    ____
    ____

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



#_____________#
#Solutions

# Open engine in context manager
# 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 = A
    df = pd.DataFrame(rs.fetchall())
    df.columns = rs.keys()

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

Exercise10

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

- 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 [ ]:

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


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


#_____________#
#Solutions

# Execute query and store records in DataFrame: df
df = pd.read_sql_query(
    "SELECT * FROM PlaylistTrack INNER JOIN Track ON PlaylistTrack.TrackId = Track.TrackId
    engine
)

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