Obtaining our Data

12 STUDENTS COMPLETED +6

Ask a Question

0 Questions Need Help

Finish Reading I'M DONE

NEXT LESSON

CREATE A STUDY GROUP

Obtaining Our Data

Introduction

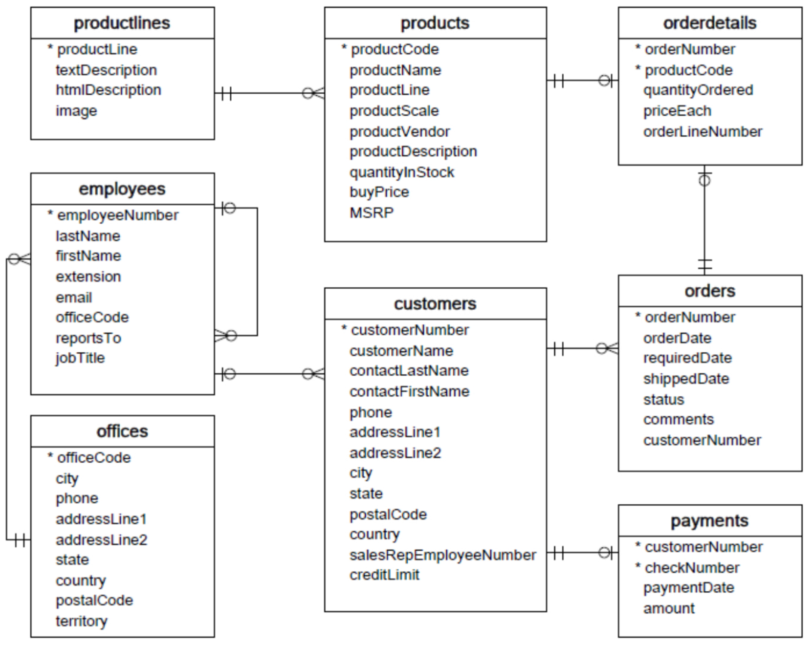
In this lesson, we'll sythesize many of our data loading skills to date in order to merge multiple datasets from various sources.

Objectives

You will be able to:

- Understand the ETL process and the steps it consists of
- Understand the challenges of working with data from multiple sources

Loading SQL DB to DataFrames



In [1]:
```
import sqlite3
import pandas as pd

#Create a connection
con = sqlite3.connect('data.sqlite')
#Create a cursor
cur = con.cursor()
#Select some data
cur.execute("""select * from orders join orderdetails using(orderNumber);""")
df = pd.DataFrame(cur.fetchall())
df.columns = [i[0] for i in cur.description]
print(df.shape)
df.head()
```

```
(2996, 11)
```

Out[1]:

	orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	productCode
0	10100	2003-01-06	2003-01-13	2003-01-10	Shipped		363	S18_1749
1	10100	2003-01-06	2003-01-13	2003-01-10	Shipped		363	S18_2248
2	10100	2003-01-06	2003-01-13	2003-01-10	Shipped		363	S18_4409
3	10100	2003-01-06	2003-01-13	2003-01-10	Shipped		363	S24_3969
4	10101	2003-01-09	2003-01-18	2003-01-11	Shipped	Check on availability.	128	S18_2325

In [2]:
```
import sqlite3
import pandas as pd
```

In [3]:
```
#Create a connection
con = sqlite3.connect('data.sqlite')
#Create a cursor
cur = con.cursor()
#Select some data
cur.execute("""select * from products;""")
df = pd.DataFrame(cur.fetchall())
df.columns = [i[0] for i in cur.description]
print(df.shape)
df.head()
```

```
(110, 9)
```

Out[3]:

	productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock
0	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front...	7933
1	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels; steering function; deta...	7305
2	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi logos and insignias, saddl...	6625
3	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, official Harley Davidson logos...	5582
4	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Turnable front wheels; steer...	3252

Merging Data

Recall that we can also join data from multiple tables in sql.

In [4]:
```
#Create a connection
con = sqlite3.connect('data.sqlite')
#Create a cursor
cur = con.cursor()
#Select some data
cur.execute("""select * from products
                        join orderdetails
                        using (productCode);""")
df = pd.DataFrame(cur.fetchall())
df.columns = [i[0] for i in cur.description]
print(df.shape)
df.head()
```

```
(2996, 13)
```

Out[4]:

	productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock
0	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front...	7933
1	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front...	7933
2	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front...	7933
3	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front...	7933
4	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front...	7933

We can also merge data from a seperate csv file. For example, say we take a seperate data source regarding daily sales data for our various branches. We might first generate a view from our database:

In [5]:
```
#Create a connection
con = sqlite3.connect('data.sqlite')
#Create a cursor
cur = con.cursor()
#Select some data
cur.execute("""select * from customers
                        join orders
                        using(customerNumber);""")
df = pd.DataFrame(cur.fetchall())
df.columns = [i[0] for i in cur.description]
print(df.shape)
```

```
df.head()
```

```
(326, 19)
```

Out[5]:

/	salesRepEmployeeNumber	creditLimit	orderNumber	orderDate	requiredDate	shippedDate	status	commen
e	1370	21000.00	10123	2003-05-20	2003-05-29	2003-05-22	Shipped	
e	1370	21000.00	10298	2004-09-27	2004-10-05	2004-10-01	Shipped	
e	1370	21000.00	10345	2004-11-25	2004-12-01	2004-11-26	Shipped	
\	1166	71800.00	10124	2003-05-21	2003-05-29	2003-05-25	Shipped	Custom ve concern about tl exact col
\	1166	71800.00	10278	2004-08-06	2004-08-16	2004-08-09	Shipped	

And then load the seperate datefile:

In [6]:
```
daily_sums = pd.read_csv('Daily_Sales_Summaries.csv')
daily_sums.head()
```

Out[6]:

	orderDate	min	max	sum	mean	std
0	2003-01-06	1660.12	4080.00	10223.83	2555.957500	1132.572429
1	2003-01-09	1463.85	4343.56	10549.01	2637.252500	1244.866467
2	2003-01-10	1768.33	3726.45	5494.78	2747.390000	1384.599930
3	2003-01-29	1283.48	5571.80	50218.95	3138.684375	1168.280303
4	2003-01-31	1338.04	4566.99	40206.20	3092.784615	1148.570425

In [7]:
```
merged = pd.merge(df, daily_sums)
```

Checking Merged Data

It's always good practice to check assumptions and preview transformed data views throughout your process. Let's take a look:

In [8]:
```
merged.head()
```

Out[8]:

1	addressLine2	city	state	postalCode	...	orderDate	requiredDate	shippedDate	status	comments	
e e		Nantes		44000	...	2003-05-20	2003-05-29	2003-05-22	Shipped		21
e e		Nantes		44000	...	2004-09-27	2004-10-05	2004-10-01	Shipped		19
e e		Nantes		44000	...	2004-11-25	2004-12-01	2004-11-26	Shipped		5
s s		Marseille		13008	...	2004-11-25	2004-12-02	2004-11-29	Shipped		5
g t.		Las Vegas	NV	83030	...	2003-05-21	2003-05-29	2003-05-25	Shipped	Customer very concerned about the exact color ...	7

Pandas merge method conveniently uses common column names between the dataframes. You can always specifically specify what columns to join on by using the `on` clause as in `pd.merge(df1, df2, on=[col1, col2])` . Unfortunately, columns that are not identically named beforehand will not work with this convenience method. Additionally, it is imperitive to check the formatting of the join keys between the tables. A number formatted as a string can often ruin joins, and seperate formatting conventions such as 'U.S.' versus 'USA' are also important preprocessing considerations before merging data files from various sources. In this case, everything worked smoothly, but it's good to keep in mind what problems may occur.

Saving Transformed Data to File

Finally, we can save our transformed dataset.

In [9]:
```
merged.to_csv('Merged_Dataset.csv', index=False)
```

Summary

Well done! In this lesson we review merges, as well as potential pitfalls in merging datasets from different sources. In the next lab, you'll get some practice doing this as an initial step to a regression task.

If the notebook is having issues loading, click here to open in a new tab. (Don't close this Learn tab or the notebook will exit.)