Appendix A

Data Dictionaries

1 Introduction

This chapter contains information on the data used in this course and how to load it into PostgreSQL and Pandas. To begin loading the data, clone the git repo that can be found at `https://github.com/NickRoss/sql-data`.

The repo itself contains a script (`load_data.py`) which will load the data into a PostgreSQL compatible database as well as a Docker image for running PostgreSQL via containers. If one wishes to load the data themselves, this appendix contains a basic framework for loading each table.

This section also contains information on how to load some of the datasets into Pandas. In both cases (Pandas and SQL), the <FILEPATH> parameter needs to be changed to the location of the file on your local machine.

2 Iowa Fleet data

This table contains automobile registration information and annual fees for the state of Iowa. Note that a few changes were made to the file. In particular, O'brien county was miscoded at times and NULL counties were removed. The final table contains 41,202 rows of data.

CREATE TABLE and COPY commands which load the data into a PostgreSQL compatible database can be found below:

```
create table cls.cars (
    year int
    , countyname varchar(20)
    , motorvehicle varchar(3)
    , vehiclecat varchar(15)
    , vehicletype varchar(55)
    , tonnage  varchar(30)
    , registrations int
    , annualfee float
    , completecategory varchar(90)
);

COPY cls.cars FROM  '<FILEPATH>/iowa_cars.tdf'
        CSV DELIMITER AS E'\t'
```

To load the data into Pandas, the following command can be used:

```
dfCars = pd.read_csv('<FILEPATH>/iowa_cars.tdf'
                ,sep='\t', engine='python', names=['year', 'countyname'
                         , 'motorvehicle' ,'vehiclecat', 'vehicletype'
                         , 'tonnage', 'registrations', 'annualfee',
                         , 'completecategory'])
```

Table A.1: Data Dictionary for Iowa Cars Data

Column	Example Values	Data Type	Description
Year	2011	Int	Calendar year vehicle was registered
CountyName	"Adair"	Varchar(20)	County vehicle was registered. Those without a county listed were registered/titled by the State
MotorVehicle	"Yes"	VarChar(3)	Indicates whether motor vehicle (Yes) or trailer (No).
VehicleCat	"Trailer"	VarChar(15)	Broad category for vehicle types.
Vehicletype	"Bus"	VarcChar(25)	Type of vehicle registered.
tonnage	"4 tons"	charchar(30)	Tonnage category for truck and truck tractor vehicle types.
registrations	397	int	Number of vehicle registrations.
annualfee	1470	float	Annual fee associated with vehicle registrations.
completecategory	"Truck – 3 Tons"	varchar(90)	Combination of VehicleType and tonnage.

3 NY MTA Data

The data in this table represents hourly traffic on NY's MTA system.[1] Information in Table A.1 contains the map between toll plaza ID and the name of the toll plaza. The final table contains 1,165,728 rows of data.

In order to load the data use the following set of commands:

```
create table cls.mta (
                plaza int
                , mtadt date
                , hr int
                , direction varchar(1)
                , vehiclesEZ int
                , vehiclesCASH int
);

COPY cls.mta from '<FILEPATH>/MTA_Hourly.tdf'
        CSV DELIMITER AS E'\t'
```

Loading the data into Pandas can be accomplished with the following command:

[1]Information was downloaded from this location: https://catalog.data.gov/dataset/hourly-traffic-on-metropolitan-transportation-authority-mta-bridges-and-tunnels-beginning-

```
dfMTA = pd.read_csv('<FILEPATH>/MTA_Hourly.tdf'
                    , sep='\t', engine='python', names=['plaza', 'mtadt'
                    , 'hr', 'direction', 'vehiclesez', 'vehiclescash'])

dfMTA.mtadt = pd.to_datetime(dfMTA.mtadt)
```

Column	Example Values	Data Type	Description
plaza	1,2,3	Int	Plaza Number (more information below)
mtadt	1/1/2012	Date	Observation Date
hr	0 - 23	Int	Hour of observation
direction	I, O	varchar(1)	Direction of traffic (Inbound vs. Outbound)
vehiclesez	1254	int	The number of vehicles that pass through each bridge and pay with EZ pass
vehiclescash	1254	int	The number of vehicles that pass through each bridge and pay with cash

Figure A.1: Information on Plaza number for MTA Hourly data

Plaza ID	Name
1	Robert F. Kennedy Bridge Bronx Plaza (TBX)
2	Robert F. Kennedy Bridge Manhattan Plaza (TBM)
3	Bronx-Whitestone Bridge (BWB)
4	Henry Hudson Bridge (HHB)
5	Marine Parkway-Gil Hodges Memorial Bridge (MPB)
6	Cross Bay Veterans Memorial Bridge (CBB)
7	Queens Midtown Tunnel (QMT)
8	Hugh L. Carey Tunnel (HLC) formally known as Brooklyn-Battery Tunnel (BBT)
9	Throgs Neck Bridge (TNB)
11	Verrazano-Narrows Bridge (VNB)

4 Daily Stock Data: s2010 and s2011

The tables s2010 and s2011 contain information on daily prices for stocks that appear on the NYSE or NASDAQ. The table s2010 has 816,066 rows while the table s2011 has 864,110 rows.

The columns **symb and retdate** define a unique row for each table.

The commands below will generate two tables, s2010 and s2011 in the schema "stocks" and then load the data into those two tables. Note that "<FILEPATH>" has to be changed to the path of where the data lies in on the machine which is loading the data.

```
create table stocks.s2010 (
        symb varchar(6)
        , retdate date
        , opn float
        , high float
        , low  float
        , cls float
        , vol int
        , exch varchar(8));

COPY stocks.s2010 FROM
         '<FILEPATH>/s2010.tdf'
        CSV DELIMITER E'\t';

reate table stocks.s2011 (
        symb varchar(6)
        , retdate date
        , opn float
        , high float
        , low  float
        , cls float
        , vol int
        , exch varchar(8));

COPY stocks.s2011 FROM '<FILEPATH>/s2011.tdf'
        CSV DELIMITER E'\t';
```

To load the data into Pandas DataFrames, use the following command:

```
df2010 = pd.read_csv('<FILEPATH>/s2010.tdf'
                ,sep='\t', engine='python', names=['symb', 'retdate',
                                                   'opn', 'high', 'low', 'cls', 'vol', 'exch'])



df2011 = pd.read_csv('<FILEPATH>/s2011.tdf'
                ,sep='\t', engine='python', names=['symb', 'retdate',
                                                   'opn', 'high', 'low', 'cls', 'vol', 'exch'])
```

If you wish to have the dates be converted to dates you can use the following commands to update the DataFrame.

```
df2010[:, 'retdate'] = pd.to_datetime(df2010.retdate)
df2011[:, 'retdate'] = pd.to_datetime(df2011.retdate)
```

Alternatively, you can load `retdate` as a date using the following:

```
df2010D = pd.read_csv('../sql-data/raw_data/s2010.tdf'
                ,sep='\t', engine='python', names=['symb', 'retdate',
                                                   'opn', 'high', 'low', 'cls', 'vol', 'exch'],
                     parse_dates = ['retdate']
                    )

df2011D = pd.read_csv('../sql-data/raw_data/s2011.tdf'
                ,sep='\t', engine='python', names=['symb', 'retdate',
                                                   'opn', 'high', 'low', 'cls', 'vol', 'exch'],
                     parse_dates = ['retdate']
                    )
```

Column	Type	Description
symb	Varchar	Code for the stock being traded.
retdate	Date	Date for the stock being traded.
opn	float	The open price of the stock.
high	float	The high price of the stock that day.
low	float	the low price of the stock that day.
cls	float	the closing price of the stock that day.
vol	int	the number of share traded that day.
exch	varchar	what exchange the stock is traded on.

5 Annual Fundamental Financial information: fnd

The tables fnd contains information taken from annual reports for stocks. The key to these tables are the columns datadate and gvkey. The table has 33,817 rows of data and spans most of 2010 and 2011.

The commands below will load the fnd data in the schema "stocks". Note that "<FILEPATH>" has to be changed to the path of where the data lies in on the machine which is loading the data.

```
create table stocks.fnd (
        gvkey varchar(8)
        , datadate date
        , fyear int
        , indfmr varchar(4)
        , consol varchar(1)
        , popsrc varchar(1)
        , datafmt varchar(3)
        , tic varchar(8)
        , cusip varchar(11)
        , conm varchar(30)
        , fyr int
        , cash float
        , dp float
        , ebitda float
        , emp float
        , invt float
        , netinc float
        , ppent float
        , rev float
        , ui float
        , cik varchar(10)
    );

COPY stocks.fnd FROM  '<FILPATH>/fnd.tdf
        CSV DELIMITER E'\t';
```

To load the data into Pandas, use the following command:

```
dffnd = pd.read_csv('<FILEPATH>/fnd.tdf'
                ,sep='\t', engine='python', names=['gvkey', 'datadate',
                      'fyear', 'indfmr', 'consol', 'popsrc', 'datafmt', 'tic'
                      , 'cusip', 'conm', 'fyr', 'cash', 'dp', 'ebitda', 'emp'
                      , 'invt' , 'netinc', 'ppent', 'rev', 'ui', 'cik'])
```

Column	Min. Val/Len	Max. Val/Len	Description
cash	-0.01	168896.51	The amount of cash on the balance sheet. Measured in millions of dollars.
cik	10	10	SEC identifier for corporations.
conm	3	30	Company Name
consol	1	1	If the information is consolidated with subsidiaries or kept separate.
cusip	9	9	Another identifier, this one maintained by the CUSIP bureau.
datafmt	3	3	Represents how the data was collected.
dp	-0.24	23713.56	GAAP depreciation and Amortization from the income statement. Measured in Millions of dollars.
ebitda	-45026.00	124840.00	Earnings Before Interest Taxes and Depreciation, measured in millions of dollars
emp	0.00	2100.00	Number of employees, measured in thousands.
fyear	2008	2011	Fiscal year. Note that a fiscal year is defined as the year with the most months of the calendar year with June falling forward.
fyr	1	12	Month in which the fiscal year ends.

Column	Min. Val/Len	Max. Val/Len	Description
gvkey	6	6	Unique Company Identifier used in the Fundamental Data
indfmr	4	4	Represents how the information is presented in the database.
invt	0.00	373176.43	Inventory from the balance sheet. Measured in Millions of Dollars.
netinc	-71969.00	104821.00	Net Income, in millions of dollars from the Income Statement.
popsrc	1	1	Source of the data. D means Domestic.
ppent	0.00	218567.00	Total Property Plants and Equipment from the Balance Sheet, measured in Millions of Dollars.
retdate	8	8	Data Date: Date which the information becomes available to the public. Represents the date of the fiscal year-end.
rev	-6749.63	470171.00	Total Sales from the Income Statement, measured in Millions of Dollars
tic	1	8	Ticker Symbol. Note that this is modified under certain circumstances.
ui	0.00	0.00	Unearned Income, measured in millions of dollars.

6 Soap Transaction Data

This table consists of information relating to a subscription soap service. There are two ways that customers can order: either via subscription or by a one-off ("unit") purchase. There are two different order types: single bars or double bars, though an order can have multiple of a single type in it. For example, if a row is double bars and there are "2" in the units column, this means that there were four total bars in the order associated with that row. The table has 1,047,381 rows of data.

The following commands define a table for the soap data as well as populate that table.

```
create table cls.trans (
        orderid int
        , userid int
        , trans varchar(15)
        , type varchar(15)
        , local varchar(10)
        , trans_dt date
        , units int
        , coupon float
        , months int
        , amt float );

COPY cls.trans from '<FILEPATH>/soapData.tdf'
        CSV DELIMITER E'\t';
```

Column	Example Values	Data Type	Description
orderid	1,2,3	Int	Unique ID for the order
userid	1,2,3	int	Unique ID for the user
trans	Double Bar	varchar	Bar type in order
type	Unit, Sub	varchar	Is this part of a subscription or one off transaction?
local	Mexico	varchar	Location of the customer
trans_dt	date	12/22/2016	Date of the transaction
units	1,2,3	int	Number of that trans in the order
coupon	.25	float	the percent coupon applied
months	1,2,3	int	If a subscription, the timing of the subscription
amt	47.96	float	The total price of the transaction

To load the data into Pandas, use the following command:

```
dfTrans = pd.read_csv('<FILEPATH>/soapData.tdf'
                    , sep='\t', engine='python', names = ['orderid', 'userid'
                       , 'trans', 'type', 'local', 'trans_dt', 'units', 'coupon', 'months', 'amt' ]
                     ,parse_dates=['trans_dt'])
```