1 Opening

- pandas
- as spreadsheet
- relational database
- application

2 pandas.ipynb part

2.1 Numpy

- Python. duck typing. if something looks like sequence and acts like a sequence. iteration over, length
- numpy array was not designed. numpy functions implemented in C.
- base Python is not well designed for data science
- floats by default unless otherwise told
- numpy array 8-byte cells adjacent
- numpy is vectorized.
- adding a list to a list will concatenate in base Python. numpy is vectorized elementwise numerical summation.
- Data Science from Scratch
- Joel Grus - Livecoding Madness - Let's Build a Deep Learning Library
- "*" is elementwise, np.dot is matrix multiplication
- Python list is heterogeneous. just list of pointers.
- base Python iteration on a numpy array element. overhead of unboxing and boxing.
- garbage collected language. if reference count is zero, memory is freed
- do not work elementwise. if iterating may help to convert to a base Python list up front

2.2 Pandas

- DataFrame: index rows 0,1,.., columns have meaning
- rename column names.
- Hadoop Spark. parallelizing over distributed system. change should happen by copying not by mutating. one failed machine means one part must be repeated (existing structure should not have changed.). if mutating, the entire thing must be run again.
- variables are postit in Python.
- for in-place mutation use inplace=True
- pandas DF is a numpy array with an overall meta-data (not cell level)
- single column (series) extraction with .colname
- len(df) number of rows
- iteration over a df is columnwise operation like R
- Missing is NaN in pandas.
- itertuples to iterate over rows. but usually a bad idea.
- .iloc[0:3] is index location. first three rows
- .loc[0:7] row names 0,. . . ,7 (labels not location index)

- .loc[5:7, []]
- avoid string if possible.
- Python Data Science Handbook
- Chris Albon's web site

3 EDA.ipynb

3.1 EDA

The basic workflow is as follows:

1. **Build** a DataFrame from the data (ideally, put all data in this object)
2. **Clean** the DataFrame. It should have the following properties:
   - Each row describes a single object
   - Each column describes a property of that object
   - Columns are numeric whenever appropriate
   - Columns contain atomic properties that cannot be further decomposed
3. Explore **global properties**. Use histograms, scatter plots, and aggregation functions to summarize the data.
4. Explore **group properties**. Use groupby and small multiples to compare subsets of the data.

This process transforms your data into a format which is easier to work with, gives you a basic overview of the data's properties, and likely generates several questions for you to followup in subsequent analysis.

4 grammarofdata.ipynb

- categorical transformaton. astype("category")
- describe() describes categorical nicely.
- regression requires dummy variables
- specify SQL type data base schema
  - name of column
  - primary key modifier must be unique
  - NOT EMPTY cannot be empty
  - NOT NULL (must be one of them)

4.1 verbs

VERB	dplyr	pandas	SQL
QUERY/SELECTION	filter() (and slice())	query() (and loc[], iloc[])	SELECT WHERE
SORT	arrange()	sort()	ORDER BY
SELECT-COLUMNS/PROJECTION	select() (and rename())	[](getitem) (and rename())	SELECT COLUMN
SELECT-DISTINCT	distinct()	unique(),drop_duplicates()	SELECT DISTINCT COLUMN
ASSIGN	mutate() (and transmute())	assign	ALTER/UPDATE
AGGREGATE	summarise()	describe(), mean(), max()	None, AVG(),MAX()
SAMPLE	sample_n() and sample_frac()	sample()	implementation dep, use RAND()
GROUP-AGG	group_by/summarize	groupby/agg, count, mean	GROUP BY
DELETE	?	drop/masking	DELETE/WHERE

pandas is faster if on-memory storage is possible.

4.1.1 QUERY

- .isin() for categoricals

4.1.2 SORT

4.1.3 SELECT-COLUMNS/PROJECTION

4.1.4 SELECT-DISTINCT

4.1.5 ASSIGN

- inplace

4.1.6 AGGREGATE

- .describe() on the entire df R's summary()

4.1.7 SAMPLE

4.1.8 GROUP-AGG

- .groupby() will create a nested df
- index is created. can iterate over the index.

4.1.9 DELETE

- inplace manipulation for large datasets.

4.1.10 LIMIT

- used with ORDER BY

4.2 Relationships: JOINs are Cartesian products

- `http://pandas.pydata.org/pandas-docs/stable/merging.html`
- `https://blog.codinghorror.com/a-visual-explanation-of-sql-joins/`