NYU Center for Data Science

DS-GA 1007
Programming for Data Science

Lecture 11
pandas II + SQL I – Operations on Tables

Package for manipulating and accessing data in tabular format

DS-GA 1007
Programming for Data Science

Lecture 11
pandas II + SQL I – Operations on Tables

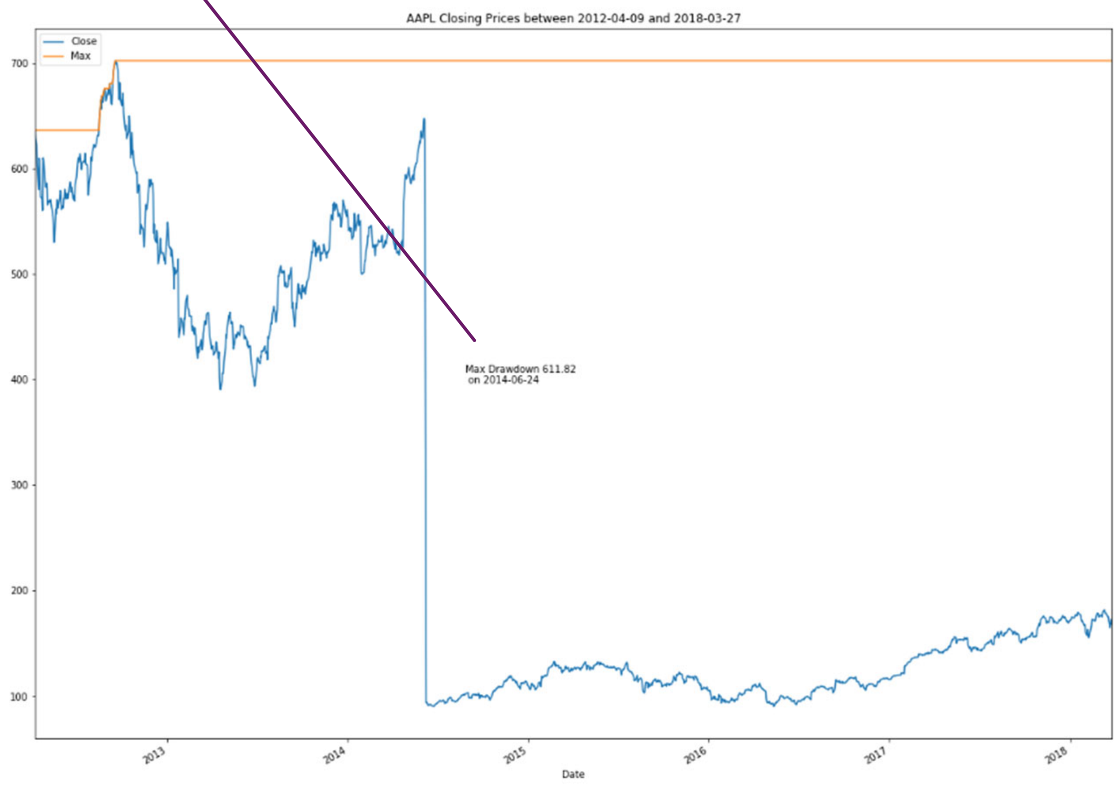
Announcements

- Homework 8 due **Sunday November 17** at 11:59pm
- Project
  - Milestone due **Thursday November 28** at 11:59pm
  - Background and Plans
- Labs
  - Submit on Jupyter Hub under Assignments tab
  - Access scores from Submitted Assignments under Assignments tab

Review

- Tabular data consisting of rows and columns
    - Common in data analysis
    - Rows are observations in sample
    - Columns are features of the data
- Often ***panel data*** with rows consisting of timestamps

June 6 2014



Review

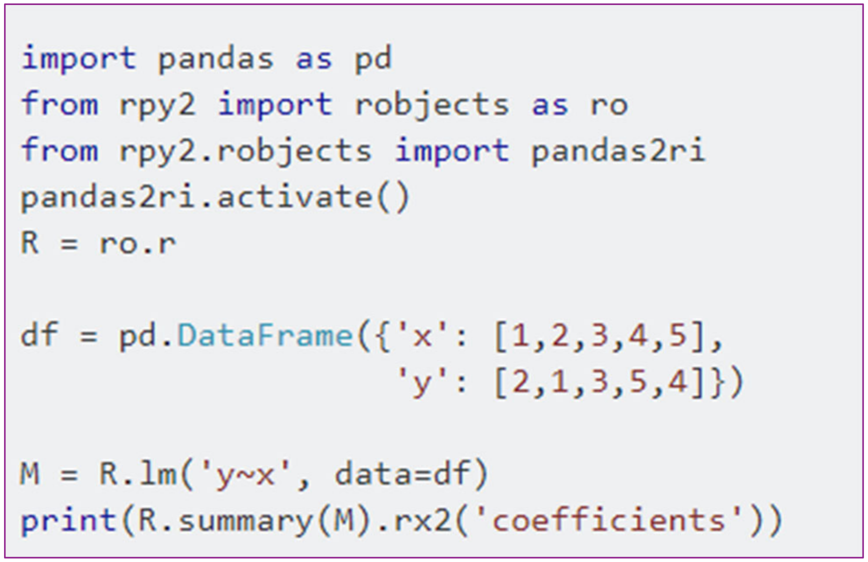
- pandas is a package for manipulating and accessing data in tabular format
- Builds on functionality of
  - numpy
  - scipy
  - components of matplotlib
- Resembles approaches in
  - R programming language
  - SQL database query language

Open source implementation of the S/S plus programming languages for statistics

Review

- pandas is a package for manipulating and accessing data in tabular format
- Builds on functionality of
  - numpy
  - scipy
  - components of matplotlib
- Resembles approaches in
  - R programming language
  - SQL database query language

Interfaces with Python through rpy2

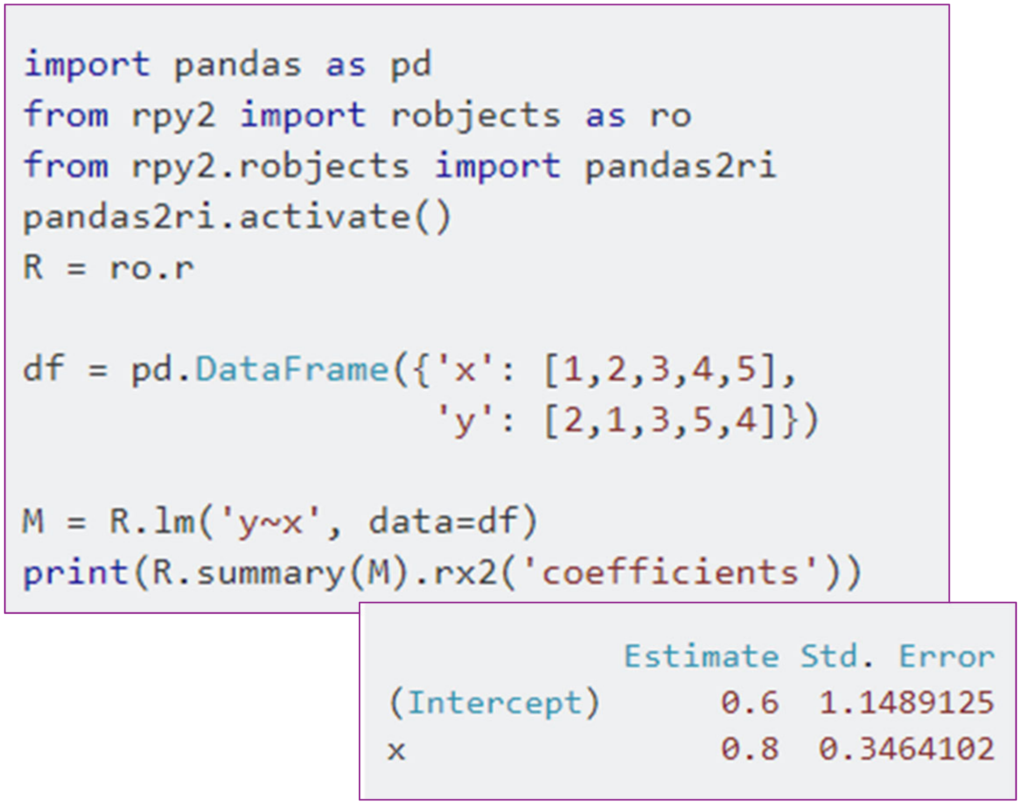


```
import pandas as pd
from rpy2 import robjects as ro
from rpy2.robjects import pandas2ri
pandas2ri.activate()
R = ro.r

df = pd.DataFrame({'x': [1,2,3,4,5],
                   'y': [2,1,3,5,4]})

M = R.lm('y~x', data=df)
print(R.summary(M).rx2('coefficients'))
```

Review

- pandas is a package for manipulating and accessing data in tabular format
- Builds on functionality of
  - numpy
  - scipy
  - components of matplotlib
- Resembles approaches in
  - R programming language
  - SQL database query language

Interfaces with Python through rpy2



```python
import pandas as pd
from rpy2 import robjects as ro
from rpy2.robjects import pandas2ri
pandas2ri.activate()
R = ro.r

df = pd.DataFrame({'x': [1,2,3,4,5],
                   'y': [2,1,3,5,4]})

M = R.lm('y~x', data=df)
print(R.summary(M).rx2('coefficients'))
```

```
            Estimate Std. Error
(Intercept)      0.6  1.1489125
x                0.8  0.3464102
```

Review

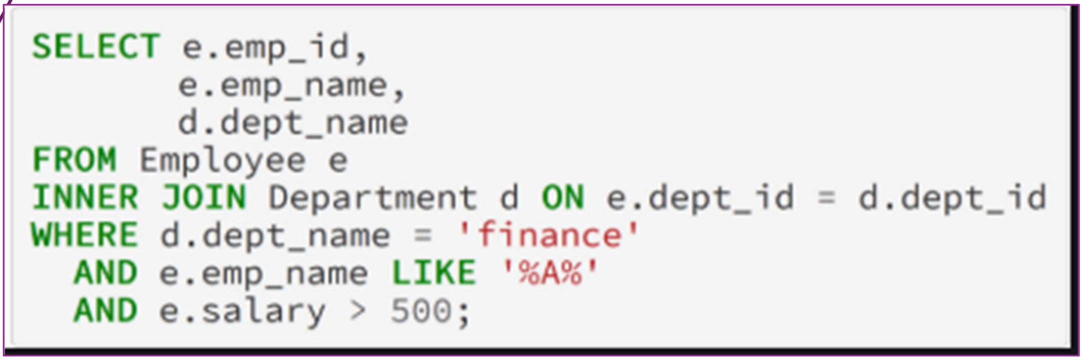
- pandas is a package for manipulating and accessing data in tabular format
- Builds on functionality of
  - numpy
  - scipy
  - components of matplotlib
- Resembles approaches in
  - R programming language
  - SQL database query language

Structured Query Language protocol for storing, accessing and managing information in database

Review

- pandas is a package for manipulating and accessing data in tabular format
- Builds on functionality of
  - numpy
  - scipy
  - components of matplotlib
- Resembles approaches in
  - R programming language
  - SQL database query language

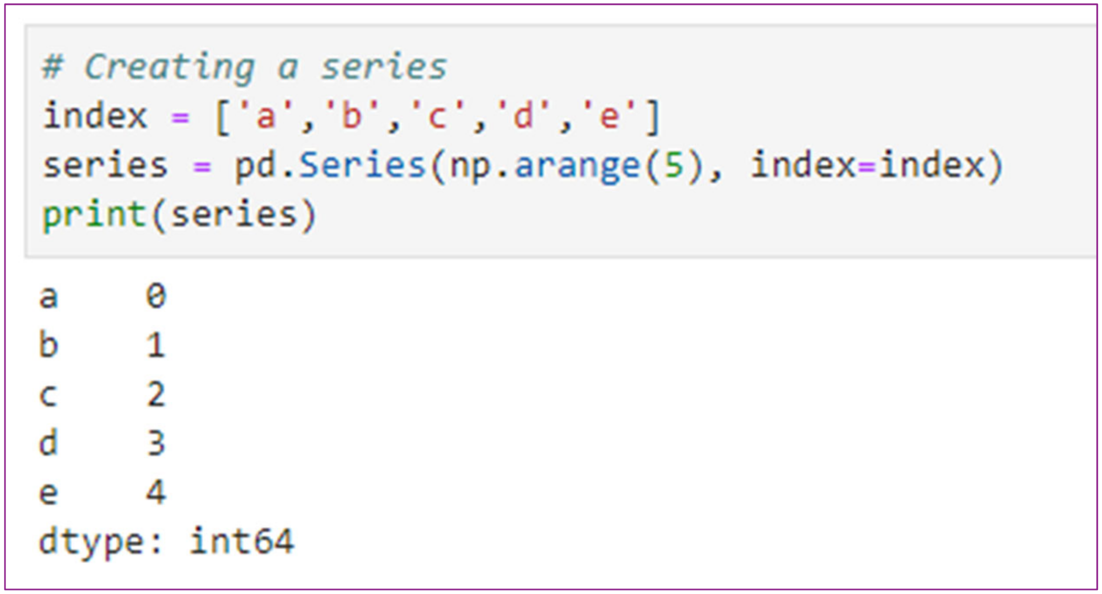
Declare commands to retrieve information from databases



```
SELECT e.emp_id,
       e.emp_name,
       d.dept_name
FROM Employee e
INNER JOIN Department d ON e.dept_id = d.dept_id
WHERE d.dept_name = 'finance'
  AND e.emp_name LIKE '%A%'
  AND e.salary > 500;
```

Review

- Series is one-dimensional object containing
  - Data
  - Labels (called ***index***)

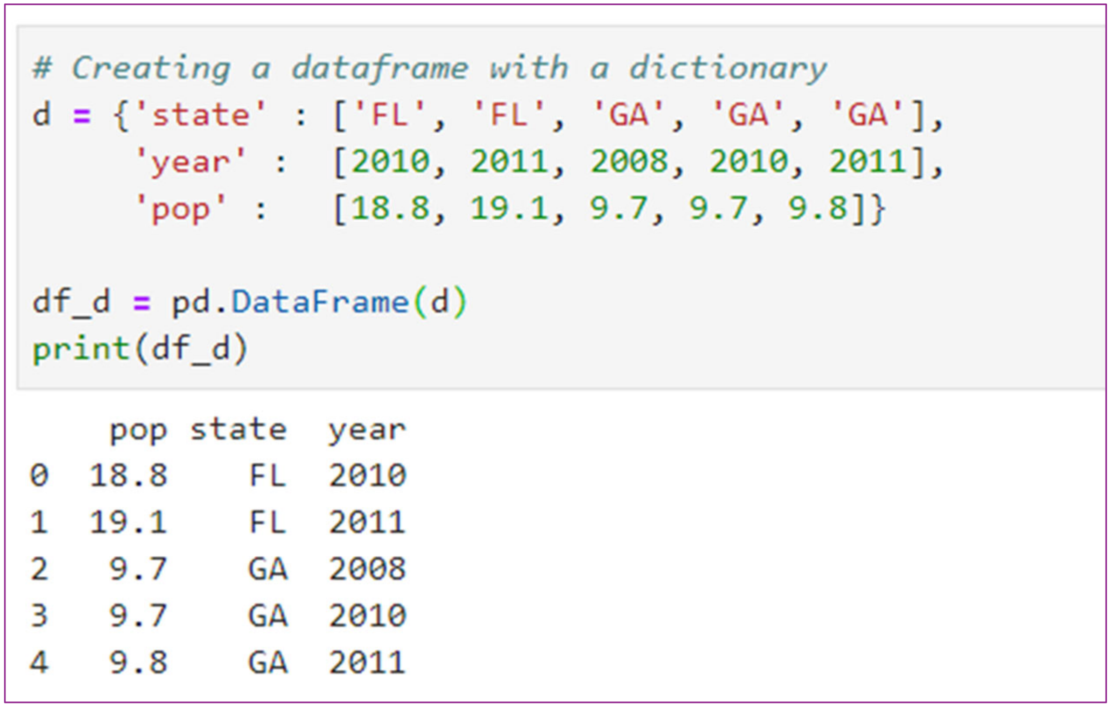


```
# Creating a series
index = ['a','b','c','d','e']
series = pd.Series(np.arange(5), index=index)
print(series)
```

```
a    0
b    1
c    2
d    3
e    4
dtype: int64
```

Review

- Series is one-dimensional object containing
  - Data
  - Labels (called ***index***)
- Dataframe is two-dimensional object containing
  - Data
  - Labels (called ***index***)
  - Columns (ordered)

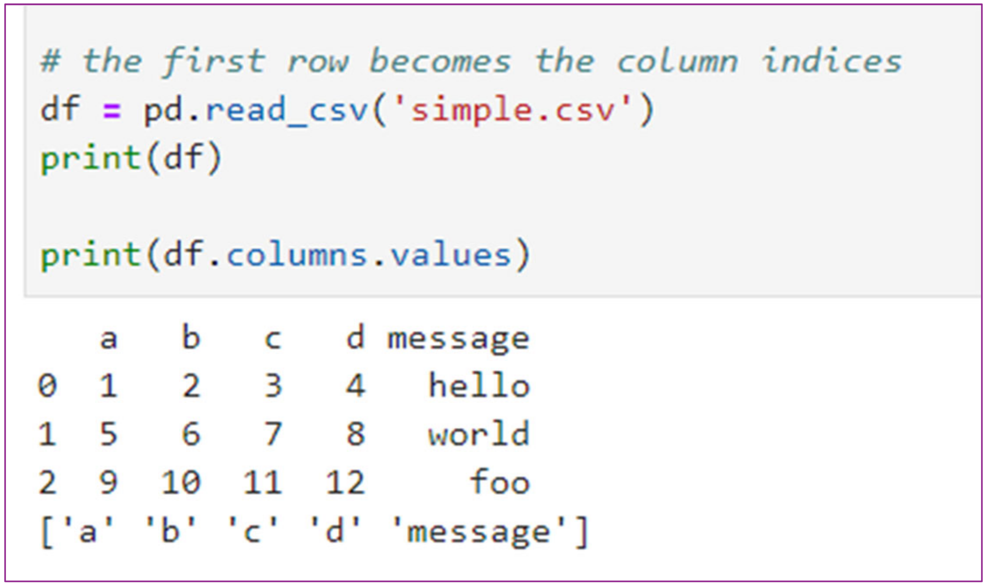


```
# Creating a dataframe with a dictionary
d = {'state' : ['FL', 'FL', 'GA', 'GA', 'GA'],
     'year' :  [2010, 2011, 2008, 2010, 2011],
     'pop' :   [18.8, 19.1, 9.7, 9.7, 9.8]}

df_d = pd.DataFrame(d)
print(df_d)
```

```
    pop state  year
0  18.8    FL  2010
1  19.1    FL  2011
2   9.7    GA  2008
3   9.7    GA  2010
4   9.8    GA  2011
```

Review

- We can store tabular data in many formats
  - Comma Separated Values (CSV)
  - Tab Separated Values (TSV)
- Note that these file formats are not nested
  - Each row and column contains one entry

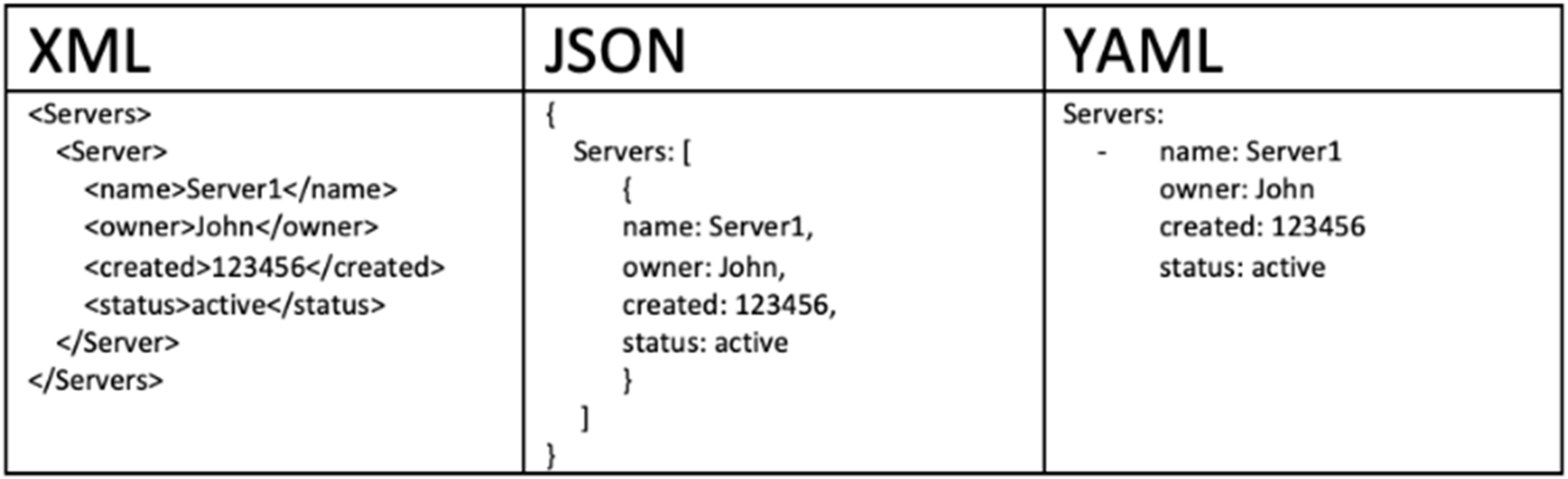


```python
# the first row becomes the column indices
df = pd.read_csv('simple.csv')
print(df)

print(df.columns.values)
```

```
   a   b   c   d message
0  1   2   3   4   hello
1  5   6   7   8   world
2  9  10  11  12     foo
['a' 'b' 'c' 'd' 'message']
```

Review

- Other formats are nested meaning each entry could contains many other entries
  - Tree structure
  - Dictionary structure

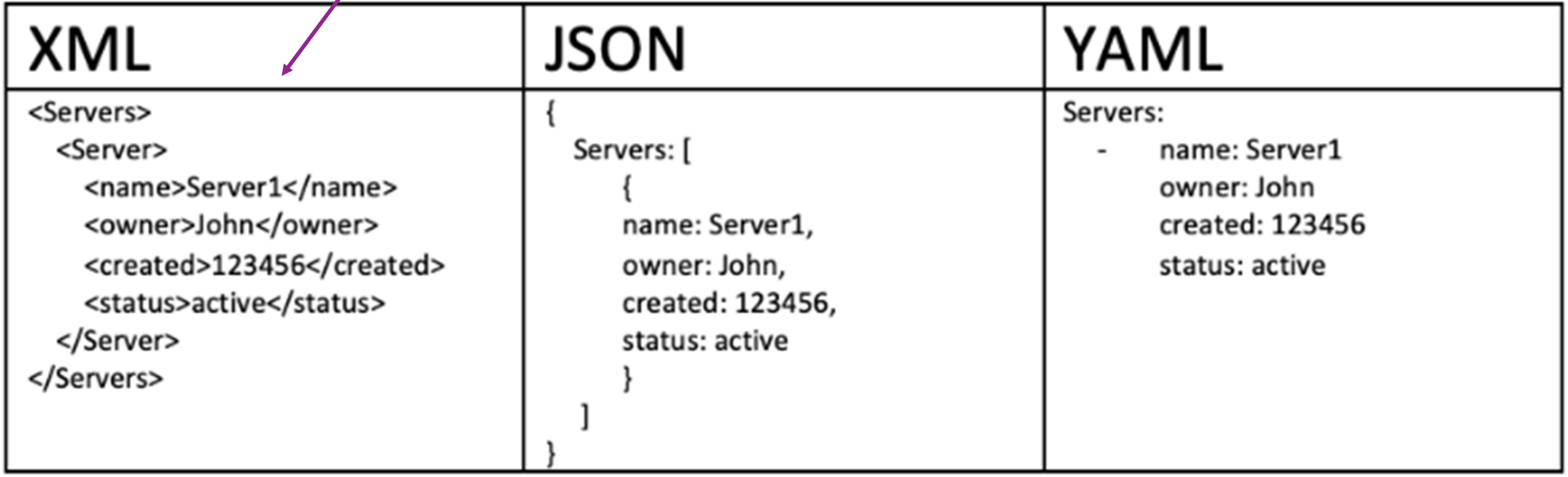


XML	JSON	YAML

<Servers>
<Server>
<name>Server1</name>
<owner>John</owner>
<created>123456</created>
<status>active</status>
</Server>
</Servers>
	{
	Servers: [
	{
	name: Server1,
	owner: John,
	created: 123456,
	status: active
	}
	]
	}
		Servers:
		- name: Server1
		owner: John
		created: 123456
		status: active

Review

- Other formats are nested meaning each entry could contains many other entries
  - Tree structure
  - Dictionary structure

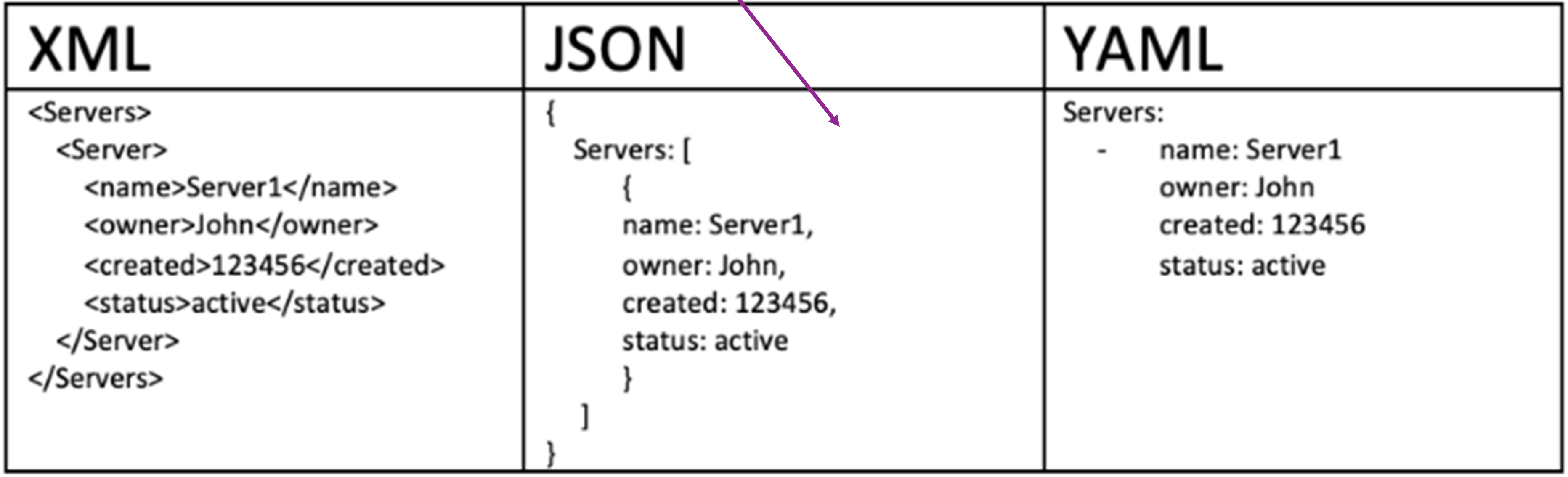


XML	JSON	YAML

<Servers>
  <Server>
    <name>Server1</name>
    <owner>John</owner>
    <created>123456</created>
    <status>active</status>
  </Server>
</Servers>
	{
	  Servers: [
	    {
	    name: Server1,
	    owner: John,
	    created: 123456,
	    status: active
	    }
	  ]
	}
		Servers:
		  - name: Server1
		      owner: John
		      created: 123456
		      status: active

Review

- Other formats are nested meaning each entry could contains many other entries
  - Tree structure
  - Dictionary structure



XML	JSON	YAML

<Servers>
<Server>
<name>Server1</name>
<owner>John</owner>
<created>123456</created>
<status>active</status>
</Server>
</Servers>
	{
	Servers: [
	{
	name: Server1,
	owner: John,
	created: 123456,
	status: active
	}
	]
	}
		Servers:
		- name: Server1
		owner: John
		created: 123456
		status: active

Agenda

- Lesson
  - pandas
  - SQL
- Demos
  - Operations on tables
  - Working with databases
- Readings
  - Python for Data Analysis by Wes McKinney
  - http://pandas.pydata.org/pandas-docs/stable/index.html

Look at input / output from different file formats

ectives

- How can we store data in pandas? How can we access and manipulate data in pandas?
- What is databases? Why would we need to work with databases through query languages?
- What are the similarities and differences between pandas and SQL?

Agenda

- Lesson
  - pandas
  - SQL
- Demos
  - Operations on tables
  - Working with databases
- Readings
  - Python for Data Analysis by Wes McKinney
  - http://pandas.pydata.org/pandas-docs/stable/index.html

Objectives

- **How can we store data in pandas? How can we access and manipulate data in pandas?**
  - Importing / Exporting Data
  - Filling or Dropping Missing Data
  - Multiple Indexes
  - Categorical Data
  - Plotting

Agenda

- Lesson
  - pandas
  - SQL
- Demos
  - Operations on tables
  - Working with databases
- Readings
  - Python for Data Analysis by Wes McKinney
  - http://pandas.pydata.org/pandas-docs/stable/index.html

Objectives

- **What is databases? Why would we need to work with databases through query languages?**
  - Define the terms Atomicity, Consistency, Isolation, Durability.
  - What is a schema? What is a primary / foreign key?
  - How can we combine tables?

SQL

- Text Files
  - Issue with Many Users
  - Inconsistency
  - Lack of Scale
  - Examples
    - Comma Separated Value
    - Tab Separated Value

nicknamed
flat files

SQL

- Text Files
  - Issue with Many Users
  - Inconsistency
  - Lack of Scale
  - Examples
    - Comma Separated Value
    - Tab Separated Value

- Database
  - Any collection of structured data. Organized into tables.
  - Database Management Systems (DBMS) **store**, **manage** and facilitate **access** to one or more databases.

SQL

Examples include sqlite and MySQL

- Text Files
  - Issue with Many Users
  - Inconsistency
  - Lack of Scale
  - Examples
    - Comma Separated Value
    - Tab Separated Value

- Database
  - Any collection of structured data. Organized into tables.
  - Database Management Systems (DBMS) **store**, **manage** and facilitate **access** to one or more databases.

SQL

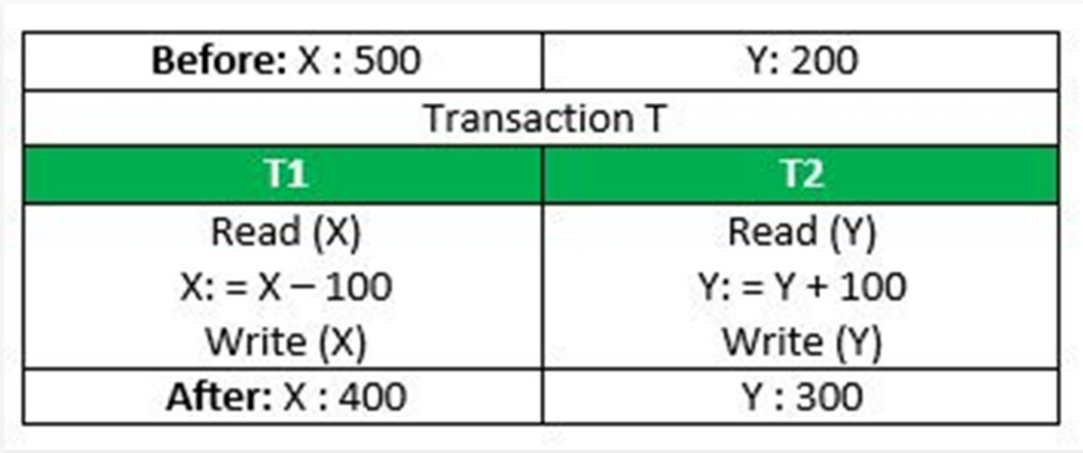
What is NoSQL?

- Text Files
  - Issue with Many Users
  - Inconsistency
  - Lack of Scale
  - Examples
    - Comma Separated Value
    - Tab Separated Value

- Database
  - Any collection of structured data. Organized into tables.
  - Database Management Systems (DBMS) **store**, **manage** and facilitate **access** to one or more databases.

SQL Storage

- Store
  - Provide **durable** storage to survive system crashes and disk failures
  - Changes are **atomic** meaning all-or-nothing
    - If no error, then changes committed to database
    - If error, then changes are aborted



Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

SQL Management

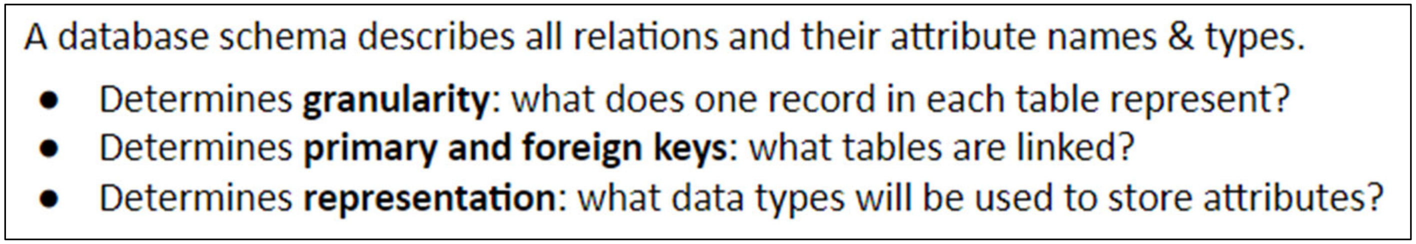
- Manage
  - Configure schema of tables to **organize** data and to ensure **consistent** properties
  - Example
    - GPA is floating point number
    - GPA is between 0 and 4.0
    - GPA in not empty

column names,
data types,
and constraints

SQL Mangement

- Manage
  - Configure schema of tables to **organize** data and to ensure **consistent** properties
  - Example
    - GPA is floating point number
    - GPA is between 0 and 4.0
    - GPA in not empty

column names,
data types,
and constraints

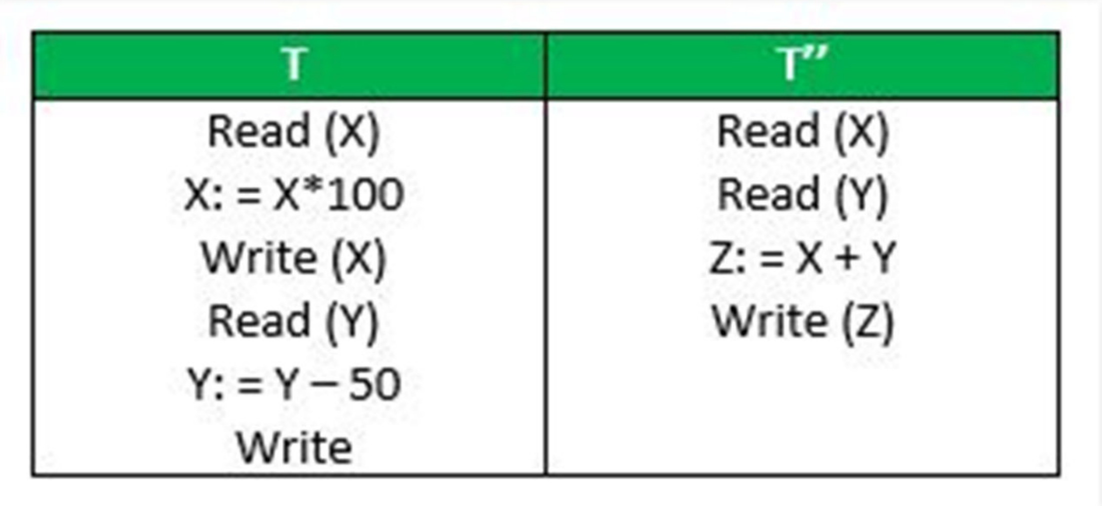


A database schema describes all relations and their attribute names & types.

- Determines **granularity**: what does one record in each table represent?
- Determines **primary and foreign keys**: what tables are linked?
- Determines **representation**: what data types will be used to store attributes?

SQL Access

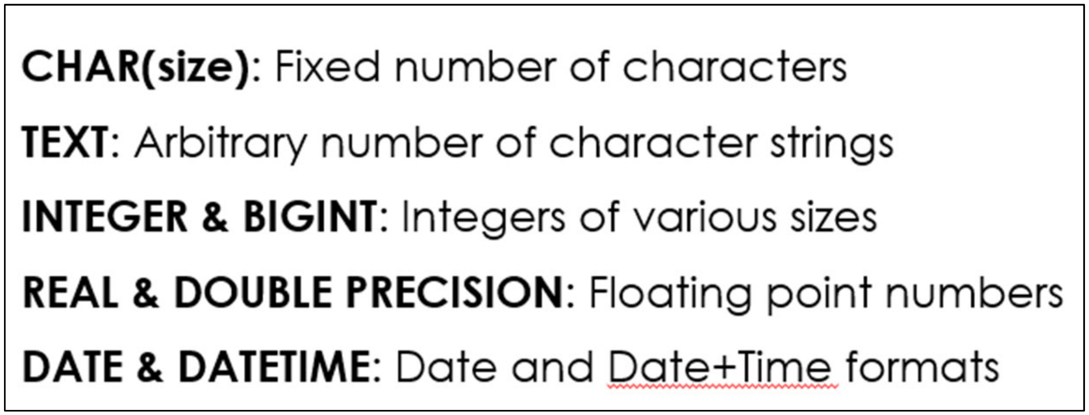
- Access
  - Efficient access to large datasets
  - Enables queries (group, sort, select, join, etc.)
  - Changes are **isolated** meaning concurrent changes are organized in series.



T	T”
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write	

SQL for Relational Database Management Systems

- Tables
  - Called **relations**
- Row
  - Called a **record** or **tuple**
- Column
  - Called an **attribute** or **field**
  - Has a name and data type (strings, integers, etc.). Remember that computational data types correspond to statistical data types.
- Columns ordered in schema. Rows ordered by inclusion with records added to bottom.



CHAR(size): Fixed number of characters

TEXT: Arbitrary number of character strings

INTEGER & BIGINT: Integers of various sizes

REAL & DOUBLE PRECISION: Floating point numbers

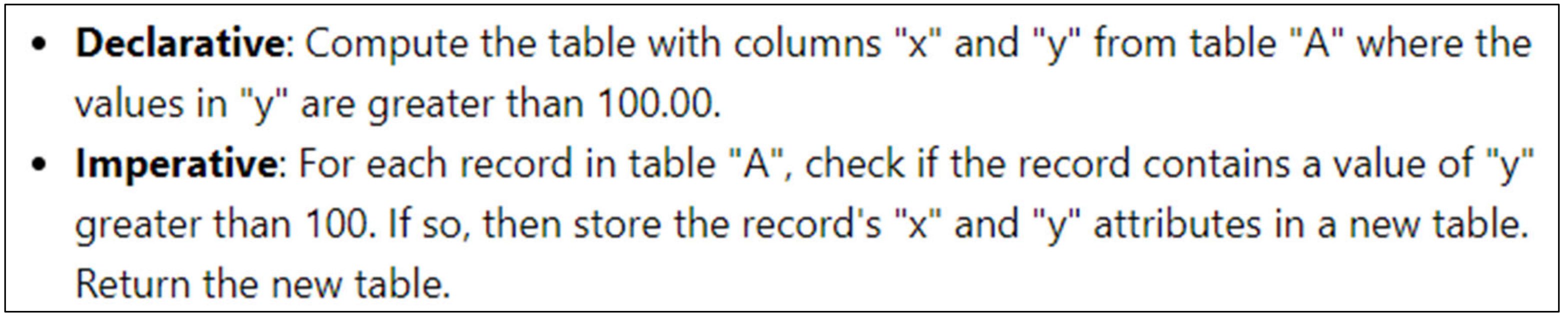
DATE & DATETIME: Date and Date+Time formats

Differences SQL and pandas

- Storage
  - Databases provide long term storage in specific format
  - pandas reads data from different formats into short term shortage
- Operations
  - pandas has additional operations (e.g. transpose)
  - pandas has index supporting access by location
- Language
  - pandas use Python
  - databases use SQL

Differences SQL and pandas

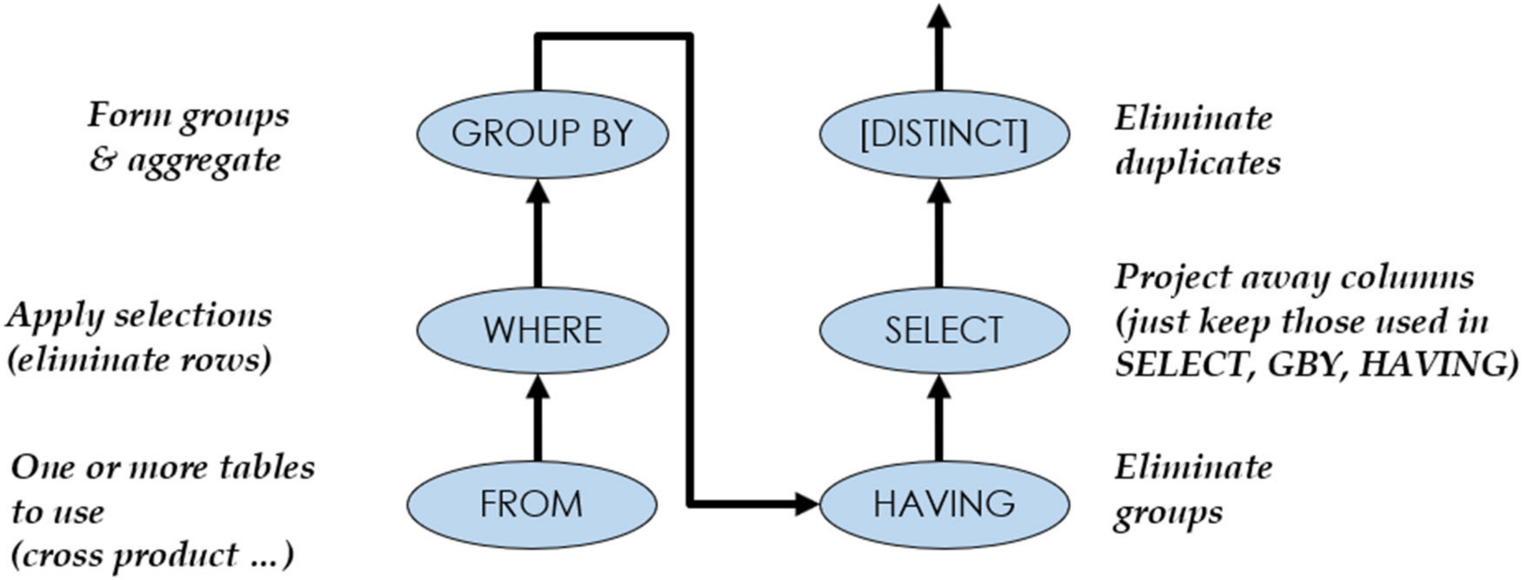
- Storage
  - Databases provide long term storage in specific format
  - pandas reads data from different formats into short term shortage
- Operations
  - pandas has additional operations (e.g. transpose)
  - pandas has index supporting access by location



- **Declarative**: Compute the table with columns "x" and "y" from table "A" where the values in "y" are greater than 100.00.
- **Imperative**: For each record in table "A", check if the record contains a value of "y" greater than 100. If so, then store the record's "x" and "y" attributes in a new table. Return the new table.

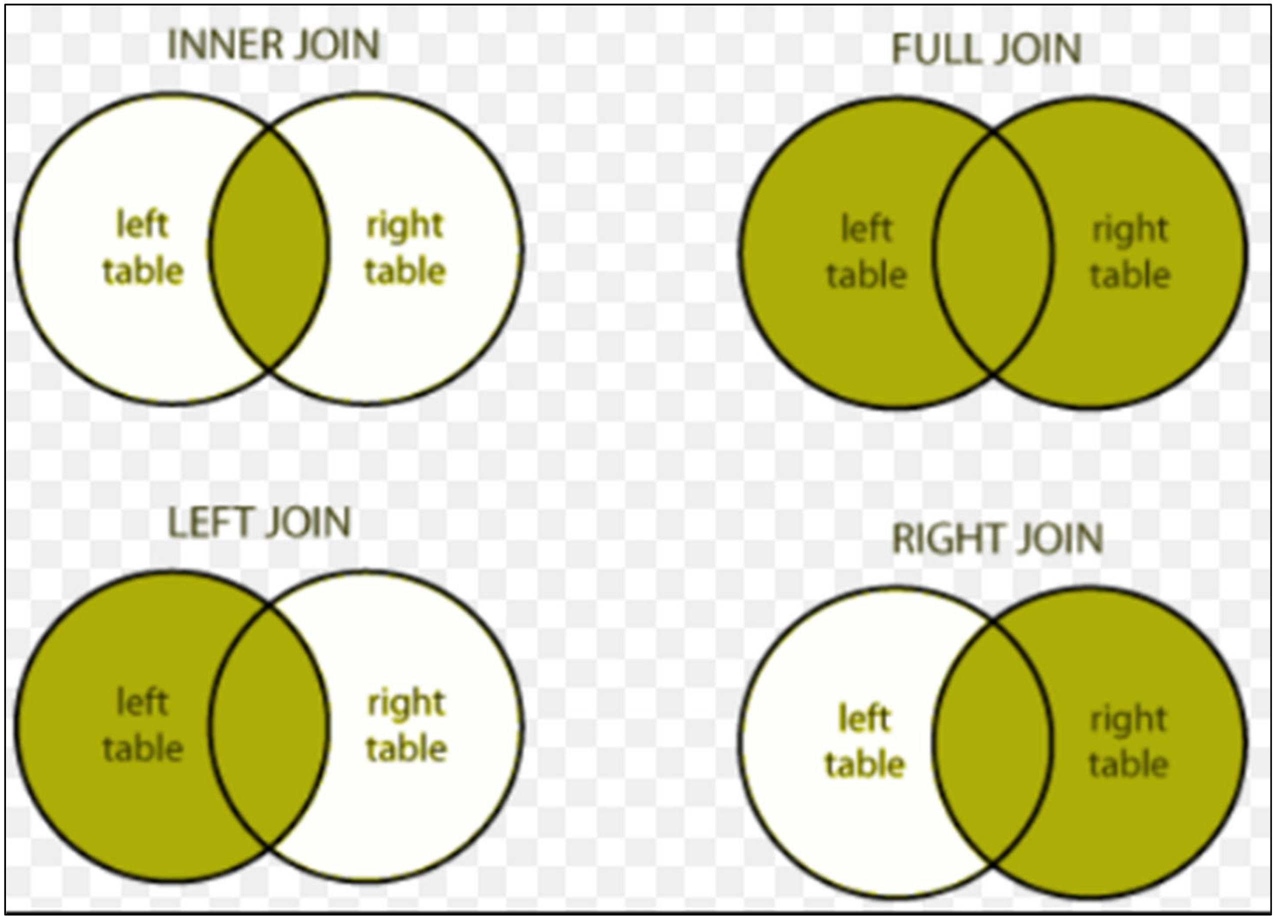
SQL Commands

```
SELECT      [DISTINCT] target-list
FROM        relation-list
WHERE       qualification
GROUP BY    grouping-list
HAVING      group-qualification
```

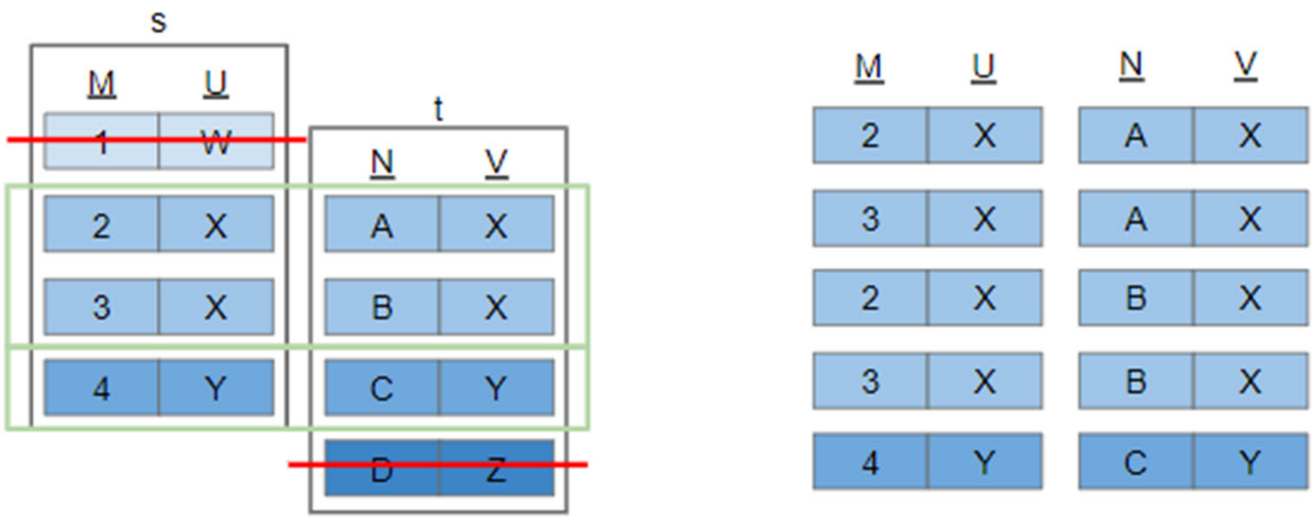


Form groups & aggregate
GROUP BY
Apply selections (eliminate rows)
WHERE
One or more tables to use (cross product ...)
FROM
[DISTINCT]
Eliminate duplicates
SELECT
Project away columns (just keep those used in SELECT, GBY, HAVING)
HAVING
Eliminate groups

SQL Joins


INNER JOIN
left table
right table
FULL JOIN
left table
right table
LEFT JOIN
left table
right table
RIGHT JOIN
left table
right table

SQL Inner Join



s

M	U
~~1~~	~~W~~
2	X
3	X
4	Y

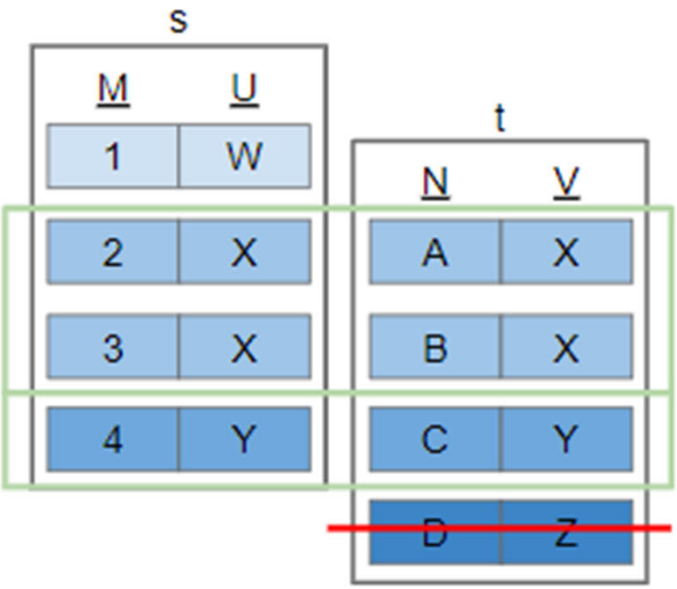
t

N	V
A	X
B	X
C	Y
~~D~~	~~Z~~

M	U	N	V
2	X	A	X
3	X	A	X
2	X	B	X
3	X	B	X
4	Y	C	Y

```
SELECT * FROM s JOIN t ON s.u = t.v;
SELECT * FROM s INNER JOIN t ON s.u = t.v;
SELECT * FROM s, t WHERE s.u = t.v;
```

SQL Left Outer Join



s

M	U
1	W
2	X
3	X
4	Y

t

N	V
A	X
B	X
C	Y
~~D~~	~~Z~~

1	W	null	null
2	X	A	X
3	X	A	X
2	X	B	X
3	X	B	X
4	Y	C	Y

```
SELECT * FROM s LEFT JOIN t ON s.u = t.v;
```

SQL Cross Join

s

M	U
1	W
2	X
3	X
4	Y

t

N	V
A	X
B	X
C	Y
D	Z

```
SELECT * FROM s, t;
```

M	U	N	V
1	W	A	X
2	X	A	X
3	X	A	X
4	Y	A	X
1	W	B	X
2	X	B	X
3	X	B	X
4	Y	B	X

(to be continued ...)

(... continued)

M	U	N	V
1	W	C	Y
2	X	C	Y
3	X	C	Y
4	Y	C	Y
1	W	D	Z
2	X	D	Z
3	X	D	Z
4	Y	D	Z

Summary

- pandas
  - Importing / Exporting Data
  - Filling or Dropping Missing Data
  - Multiple Indexes
  - Categorical Data
  - Plotting
- DBMS and RDBMS
  - Atomic, Consistent, Isolated, Durable
  - Scalable
- SQL
  - Similarities and Differences with Python
  - Commands
    - SELECT, FROM
    - WHERE, GROUP BY, HAVING
    - ORDER BY, LIMIT
    - JOIN