# COMP3055
# Machine Learning

## Topic 3 – Data Collection

**Ying Weng**

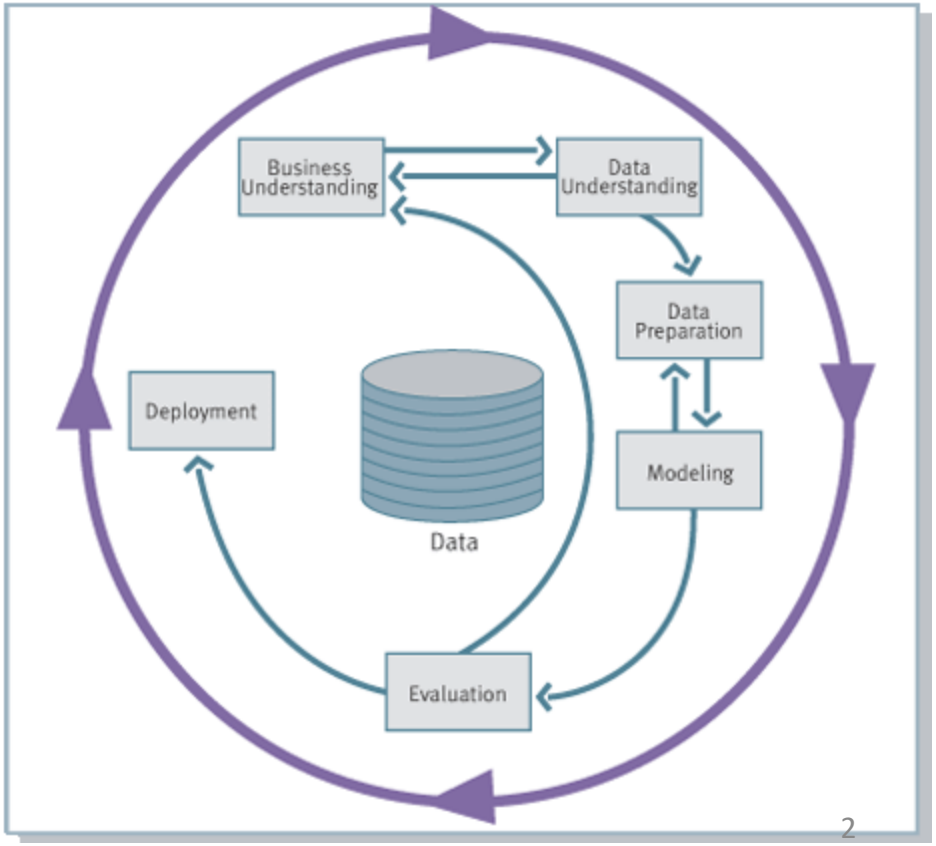
2024 Autumn

# Data Mining Process Model

CRoss Industry Standard Process for Data Mining (**CRISP-DM**)

Industry Standard

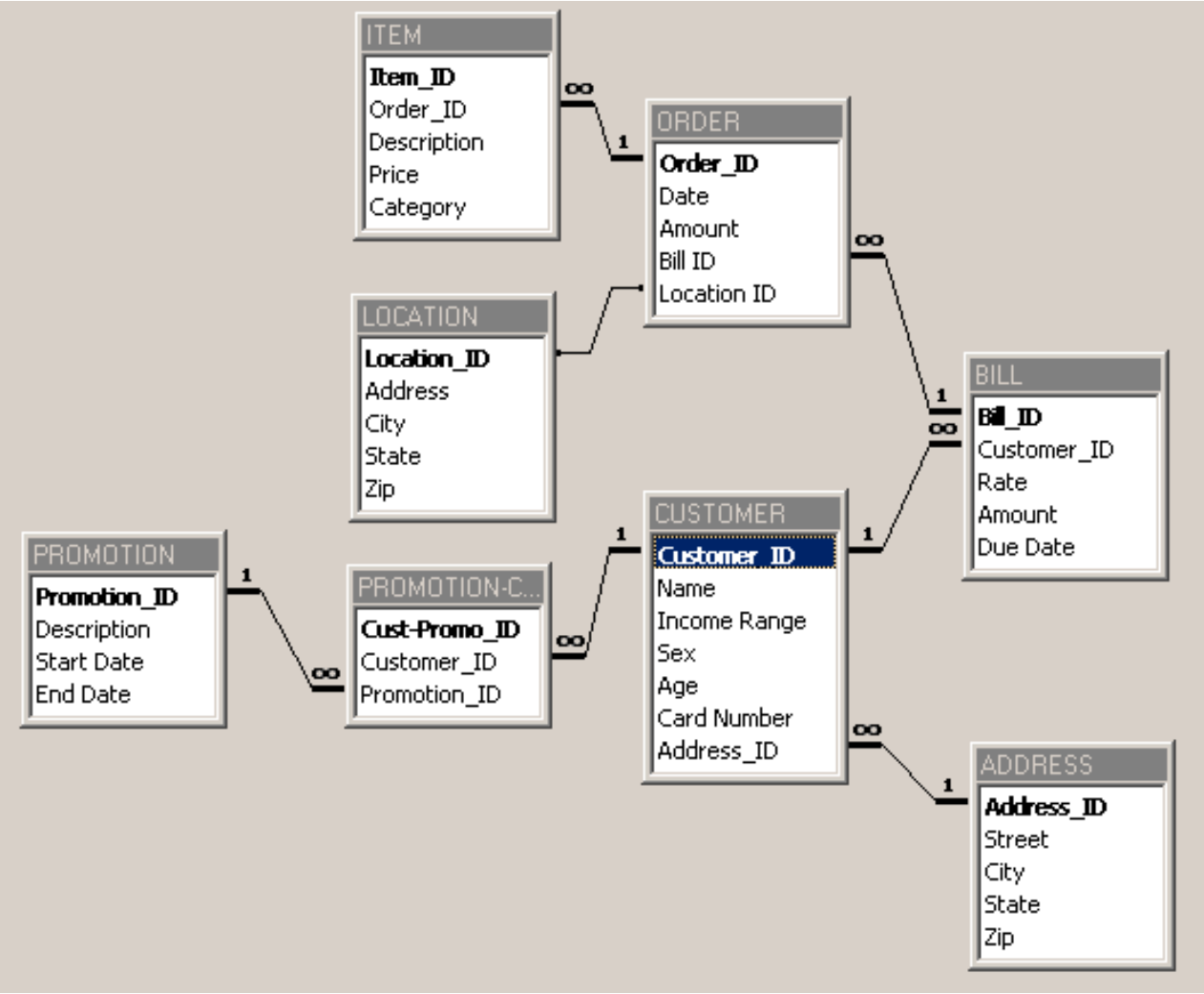
The most popular data mining methodology model according to [KDNuggets.com](KDNuggets.com)



# Step 1: Business Understanding

- Define the problem
- Choose a machine learning model(s)
- Estimate project cost
- Estimate project completion time
- Address legal issues
- Develop a maintenance plan

# Step 2: Data Understanding



# Step 3: Data Preprocessing

- Noisy data
  - Locate duplicate records
  - Locate incorrect attribute values
  - Smooth data
- Missing data
  - Discard records with missing values
  - Replace missing real-valued items with the class mean
  - Replace missing values with values found within *highly similar* instances
- Data transformation
  - Data normalization
  - Data type conversion
  - Attribute and instance selection

# Step 4: Modeling

- Choose training and test data
- Designate a set of input attributes
- If learning is supervised, choose one or more output attributes
- Select learning parameter values
- Train the model

# Step 5: Evaluation

- Statistical analysis
- Heuristic analysis
- Experimental analysis
- Human analysis

# Measures of Effectiveness of the Model

- Accuracy

  Percentage of total predictions that were correct
- Return on investment

  Cost-benefit ratios
- Explanation

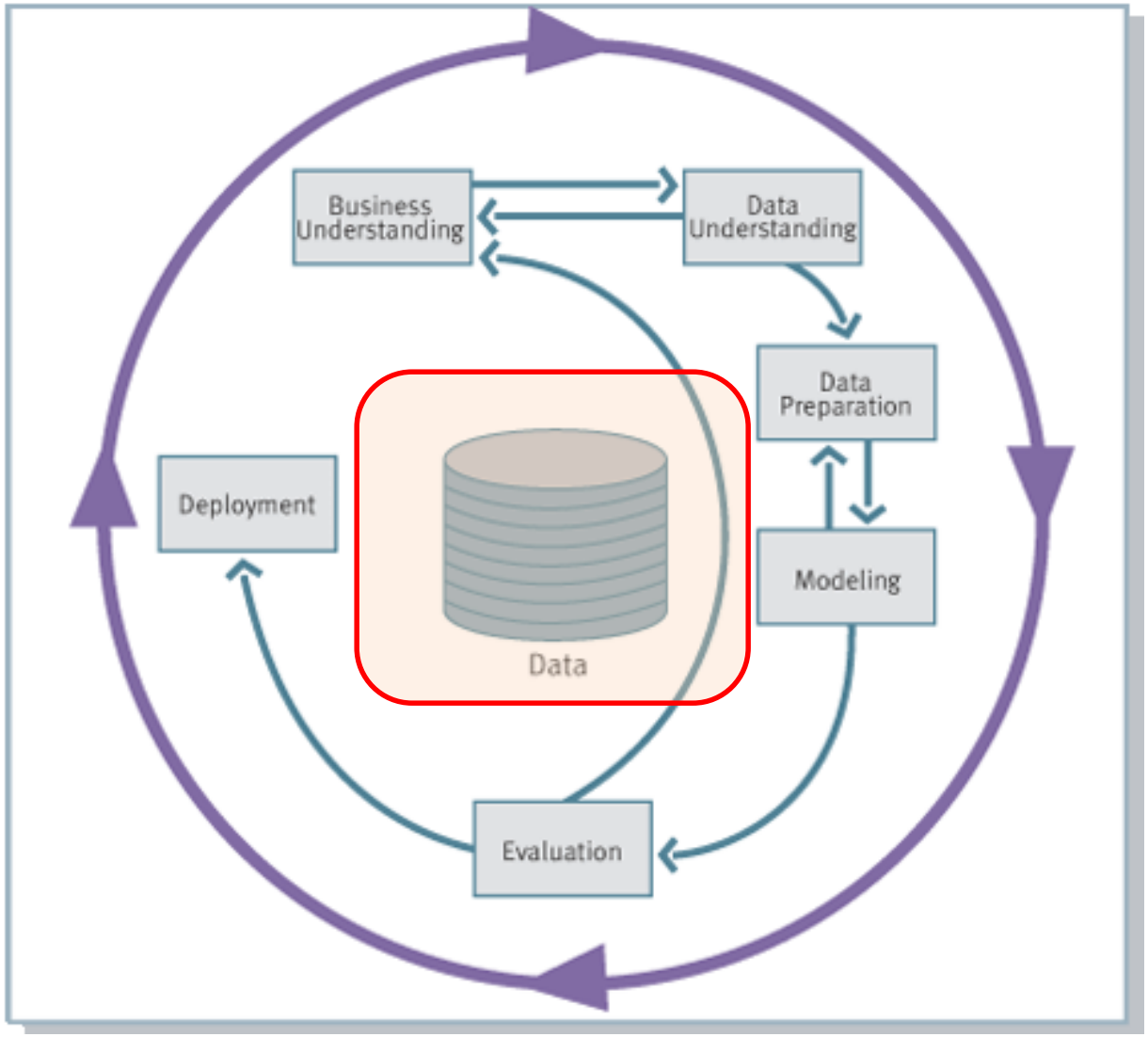
  Able to justify intuition
- Validation

  Automated checking of correctness, indexes

# Step 6: Deployment

- Apply the model to real world usage
- Apps, API, etc.
- Regularly update the model with new data
- …

# You Need Collect Data before Learning Starts!



# What is Data?

**Collection of data objects** and their associated attributes

An **attribute** is a property or characteristic of an object

- Examples: eye color of a person, temperature, etc.
- Attribute is also known as variable, field, characteristic, or feature

A **collection of attributes** describe an object

- Object is also known as record, point, case, sample, entity, or instance

**Attributes**

**Objects**

| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

# Types of Attributes

There are different types of attributes

- Nominal
  - Examples: ID numbers, eye color, zip codes
- Ordinal
  - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
- Interval
  - Examples: calendar dates, temperatures in Celsius or Fahrenheit.
- Ratio
  - Examples: temperature in Kelvin, length, time, counts

| Attribute Type | Description | Examples | Operations |
|---|---|---|---|
| Nominal | The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another. | zip codes, employee ID numbers, eye color, sex: {*male*, *female*} | mode, entropy, contingency correlation, $\chi^2$ test |
| Ordinal | The values of an ordinal attribute provide enough information to order objects. | hardness of minerals, {*good*, *better*, *best*}, grades, street numbers | median, percentiles, rank correlation, run tests, sign tests |
| Interval | For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. | calendar dates, temperature in Celsius or Fahrenheit | mean, standard deviation, Pearson's correlation, *t* and *F* tests |
| Ratio | For ratio variables, both differences and ratios are meaningful. | temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current | geometric mean, harmonic mean, percent variation |

# Discrete and Continuous Attributes

## Discrete attribute

- Has only a **finite** or countable set of values.
- Examples: zip codes, number of employees, or sale counts.
- Often represented as **integer** variables.
- Note: binary attributes are a special case of discrete attributes.

## Continuous attribute

- Has **real** numbers as attribute values.
- Examples: temperature, stock prices, or net income.
- Continuous attributes are typically represented as **floating-point** variables.

# Types of Data

**Record**

- Data matrix
- Transaction data

**Graph**

- Social networks
- Molecular structures

**Ordered**

- Spatial data
- Temporal data
- Sequential data
- Genetic sequence data

**We often deal with a mixture of different types data**

# Record Data

- Data that consists of a collection of records, each of which consists of a fixed set of attributes.

| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

# Data Matrix

- If data objects have the same fixed set of **numeric** attributes, the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute.

- Such data set can be represented as an `m` by `n` matrix, where there are `m` rows (one for each object) and `n` columns (one for each attribute).

| Projection of x Load | Projection of y load | Distance | Load | Thickness |
|---|---|---|---|---|
| 10.23 | 5.27 | 15.22 | 2.7 | 1.2 |
| 12.65 | 6.25 | 16.22 | 2.2 | 1.1 |

# Transaction Data

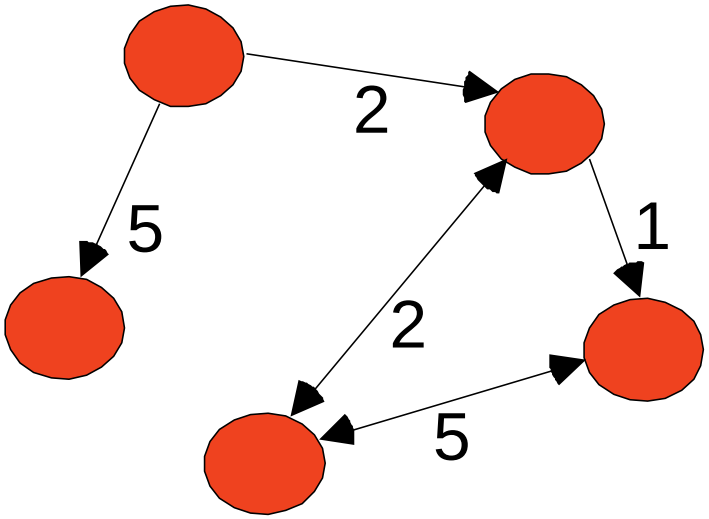
- A special type of record data, where
  - Each record (transaction) involves a set of items.
  - For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

| *TID* | *Items* |
|---|---|
| 1 | Bread, Coke, Milk |
| 2 | Beer, Bread |
| 3 | Beer, Coke, Diaper, Milk |
| 4 | Beer, Bread, Diaper, Milk |
| 5 | Coke, Diaper, Milk |

# Graph Data

Examples:

- Representation of HTML Links
- Social Networks



# Ordered Data

Genomic sequence data

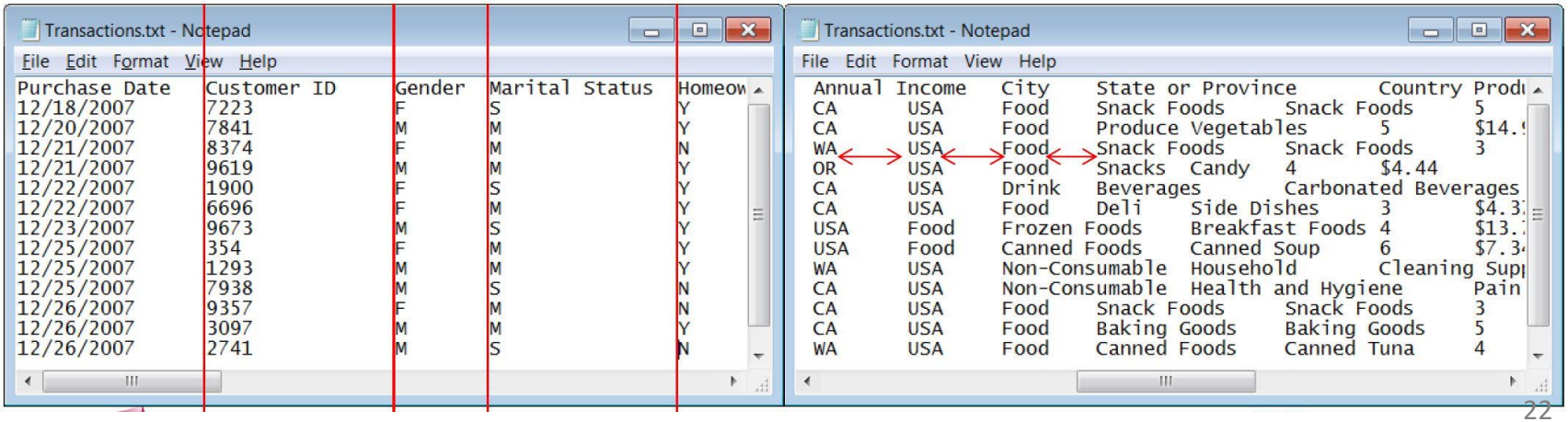
```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

# Obtaining Data

- **Data sources**
  - Obtained directly from owner: text file or relational database.
  - Collect public available data: web.
- **Unstructured VS. structured**
  - Text file: can be unstructured or structured.
  - Relational database: structured.
  - Web: unstructured.
  - Need to structure the data if necessary.
- **Data need to be cleaned**
- **Tools for data collection and cleaning**
  - Python, R, Excel, SQL, etc.

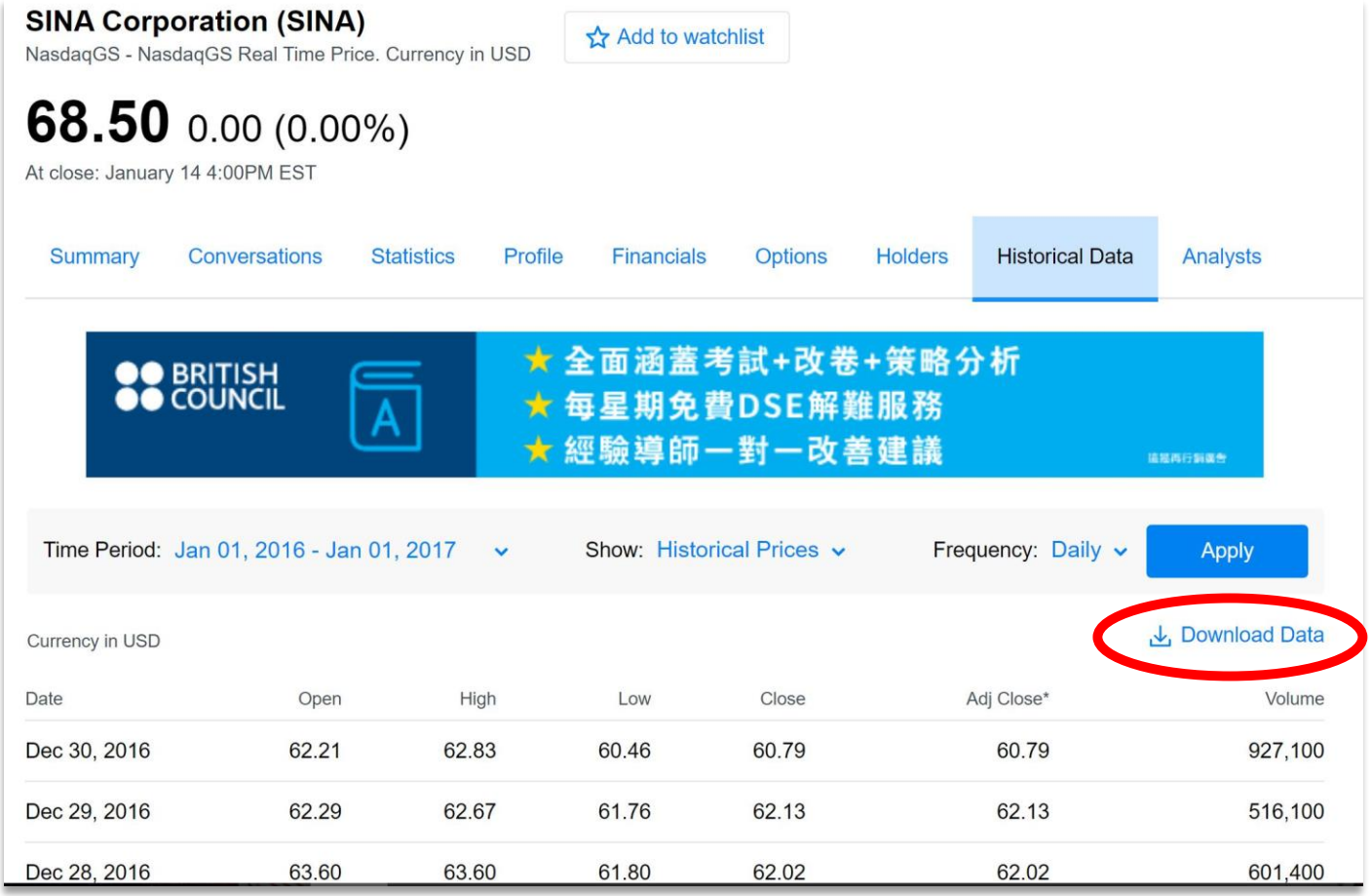
# Text Files

- Most companies use proprietary software to store data that can be exported into text files.
  - Usually with .txt extension.
  - Sometimes with .csv extension
- Commonly used text file format
  - Fixed-width: each attribute value starts and stops a fixed positions (columns) in the lines.
  - Delimited: there is a delimiter character, usually a tab, space, or **comma**, that separates different values in the lines.
  - **CSV**: comma separated values



# CSV Example

- Yahoo stock price historical data
  - Example: Sina stock price from Jan 1st, 2016 to Jan 1st 2017
  - http://finance.yahoo.com/quote/SINA/history?period1=1451577600&period2=1483200000&interval=1d&filter=history&frequency=1d




| Date | Open | High | Low | Close | Adj Close* | Volume |
|---|---|---|---|---|---|---|
| Dec 30, 2016 | 62.21 | 62.83 | 60.46 | 60.79 | 60.79 | 927,100 |
| Dec 29, 2016 | 62.29 | 62.67 | 61.76 | 62.13 | 62.13 | 516,100 |
| Dec 28, 2016 | 63.60 | 63.60 | 61.80 | 62.02 | 62.02 | 601,400 |

# CSV Example

- Manual download from Yahoo
  - http://chart.finance.yahoo.com/table.csv?s=SINA&a=0&b=1&c=2016&d=0&e=1&f=2017&g=d&ignore=.csv
  - Save to local disk as csv file, e.g. “table.csv”
  - Use Python to read csv file
- Automatically download from Yahoo
  - Use Python for both downloading and reading

```
# Load Yahoo stock price for Sina
import numpy as np
# URL for SINA from Jan 1st 2016 to Jan 1st 2017
# url = "table.csv"
url = "http://chart.finance.yahoo.com/table.csv?s=SINA&a=0&b=1&c=2016&d=0&e=1&f=2017&g=d&ignore=.csv"
# load the CSV file as a numpy matrix
dataset = np.genfromtxt(url, dtype=None, skip_header=1, delimiter=",")
print(dataset[0])
```

# Relational Database

- A **relational database** is a set of related tables, where each table is a rectangular arrangement of fields and records.
  - Rows corresponding to records.
  - Columns corresponding to fields.
  - Primary key contains unique values to index data.
  - Foreign key links tables and can contain duplicate values.
- Python allows you to import data from many database packages
  - You may need to install drive for different database, e.g. MySQLdb for MySQL database, before use.
- Basic steps
  - Make connection with host name, user name, password, database name, etc.
  - Create a cursor to the connected database
  - Execute SQL queries and fetch the data through the cursor
  - Close the database

# Relational Database

```
import MySQLdb

db = MySQLdb.connect(host="localhost", # your host, usually localhost
user="john", # your username
passwd="megajonhy", # your password
db="jonhydb") # name of the data base

# you must create a Cursor object. It will let
# you execute all the queries you need
cur = db.cursor()

# Use all the SQL you like
cur.execute("SELECT * FROM YOUR_TABLE_NAME")
# print all the first cell of all the rows

for row in cur.fetchall():
    print row[0]

db.close()
```

# Data from the Web

- Web sites containing data are structured in all sorts of ways, and the steps required to collect the data for analysis vary greatly.


- No matter the server side is a program or file, what the client side received (and the browser shows) are all files.
  - Usually in HTML (hypertext markup language) format.
  - HTML uses tags for displaying various items on the web page.

# HTML File and Representation

- Most **html tags** control font, color, images, actions, etc., which are not related to the content of data.
- The layout of html is often controlled using table (old fashion) or CSS (Cascading Style Sheets, new fashion).
- Tablet data are normally put inside table tag, but many data we are interested are not in the table.

```
<html>
<body>
<p>
Each table starts with a table tag.
Each table row starts with a tr tag.
Each table data starts with a td tag.
</p>

<h4>One row and three columns:</h4>
<table border="1">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
</table>

<h4>Two rows and three columns:</h4>
<table border="1">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
```

Each table starts with a table tag.
Each table row starts with a tr tag.
Each table data starts with a td tag.

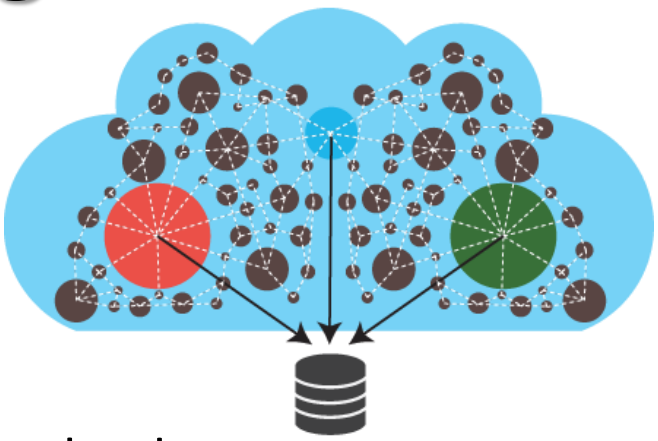
**One row and three columns:**

| | | |
|---|---|---|
| 100 | 200 | 300 |

**Two rows and three columns:**

| | | |
|---|---|---|
| 100 | 200 | 300 |
| 400 | 500 | 600 |

# Web Crawling



- Find out the patterns of URL, e.g.
  - http://finance.yahoo.com/q?s=SINA
  - actual address: http://finance.yahoo.com/q
  - Parameter: **s=SINA**
  - **name=value** pairs separated by &
- Write a program to generate the URLs and download.
  - By changing parameters and name=value pairs
- Sometimes need to find and download the URLs inside a web page (spidering).
  - URLs normally start with “http” in raw html files.
- Parsing the web pages to collect useful data
  - **Identify** where the useful data is in a web page.
  - Finding **patterns** to help the program auto locate the data by looking at raw html files.
  - Use Python Regular Expression to locate the data according to the pattern.

# Unstructured Data Parsing

## Regular Expression

- **Python**, Java, etc.
- Find patterns from the "unstructured" data, including text file, HTML file, or other files.
- Define such patterns using regular expression grammar.
- Read files into a string. The processing engine will extract data from the string that meets this pattern.
- More on this using Python later!

```
import urllib.request
import re

url="http://google.com"

# regular expression for locating title
these_regex=b"<title>(.+?)</title>"
pattern=re.compile(these_regex)

# load the url
with urllib.request.urlopen(url) as response:
    html = response.read()

# find the pattern in the downloaded file
titles=re.findall(pattern, html)
print(titles)
```

# Any Questions?