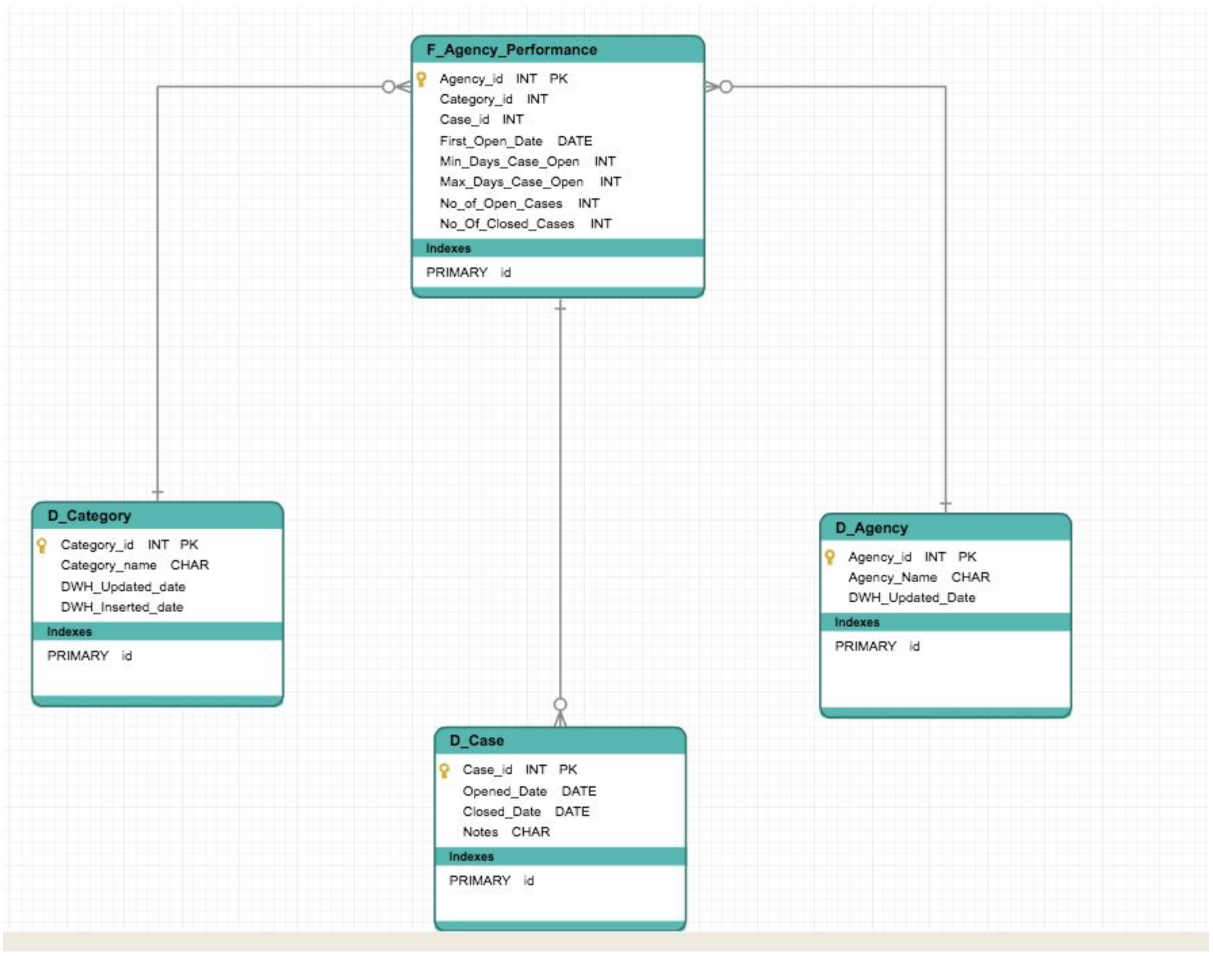
First task -

The data in general details the type of issues being reported in the city of SFO. My idea of developing a ETL pipeline would be through microservices.

1. The application offers the facility to connect and access the data through SODA API. I would use this API to connect to the data source (probably using Python, or any other reliable language) and pull the data in a relational database or dump it in S3 bucket (OR) Use queuing system like Kafka to push all the data from various sources and project the Kafka queue to all the teams through microservices.
2. Once the data is pushed into a database (this database would act as the data lake, which contain the source of truth) we can work on developing further data models on top of it.
3. The data from the data lake should be pushed into a data warehouse environment - where all the data marts and data models can be developed.
4. A sample example of a data model -



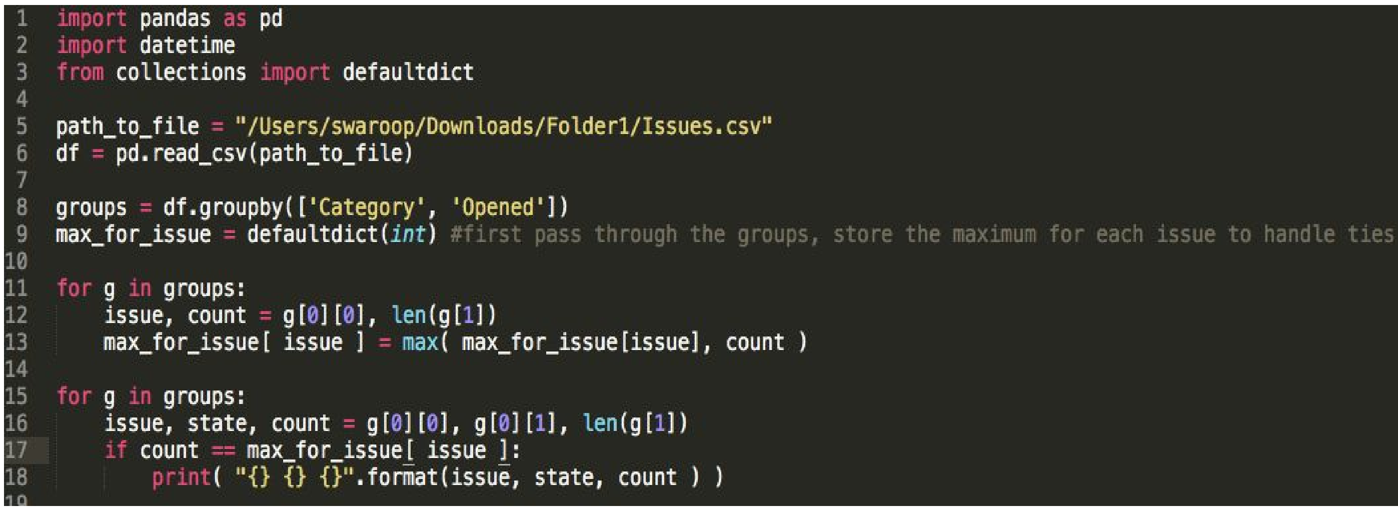
The above data model helps in generating a visualisation of how an agency is doing based on particular category. Its helps answers KPI such as -

- Which agency has the highest response time
- Which agency has the highest open cases
- Which agency has high number of severity cases

Similarly, we can create many such data models which are catered towards answering various KPI's. We can use the longitude and latitude given in the sample data to chart a heat-map that would help us predict which localities are prone to particularly category of issues and focus the marketing on those localities which are prone to more claims.

Once we have our data-marts defined in the above manner - we can extend it further to include other types of data that can later be joined with this data-mart to give more insights and predict future trends.

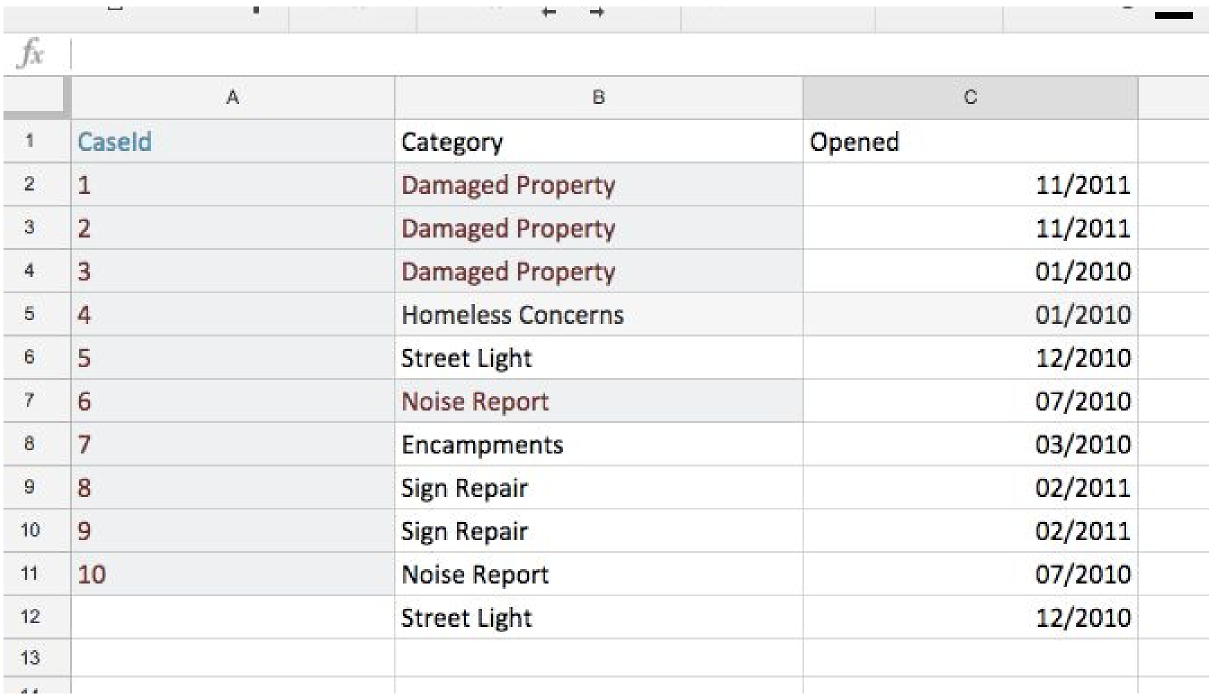
Second Task -



```
import pandas as pd
import datetime
from collections import defaultdict

path_to_file = "/Users/swaroop/Downloads/Folder1/Issues.csv"
df = pd.read_csv(path_to_file)

groups = df.groupby(['Category', 'Opened'])
max_for_issue = defaultdict(int) #first pass through the groups, store the maximum for each issue to handle ties

for g in groups:
    issue, count = g[0][0], len(g[1])
    max_for_issue[ issue ] = max( max_for_issue[issue], count )

for g in groups:
    issue, state, count = g[0][0], g[0][1], len(g[1])
    if count == max_for_issue[ issue ]:
        print( "{} {} {}".format(issue, state, count ) )
```

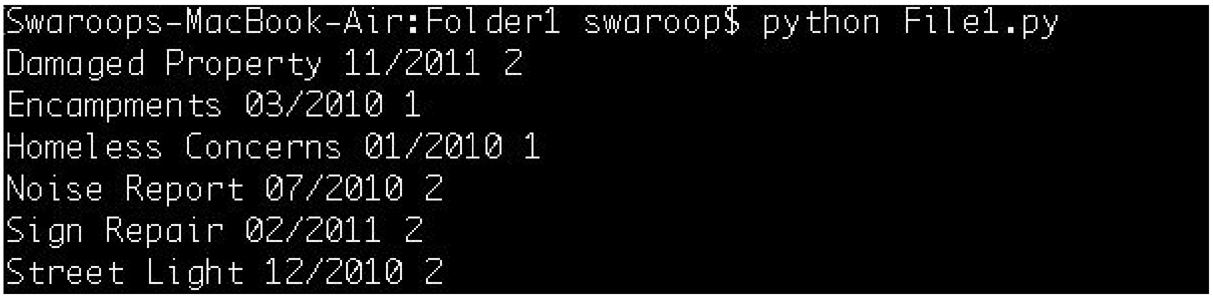
I have written a python procedure that would read the data from the csv file and group them based on the category. This would help us in plotting a trend/graph based on how many issues are being reported in each category per month. This can further help in predicting which category of claims have a likelihood of being used in the future.

For the sake of this example - I have taken the data which constraints to 3 column to help obtain the result.
The sample CSV looks as below :



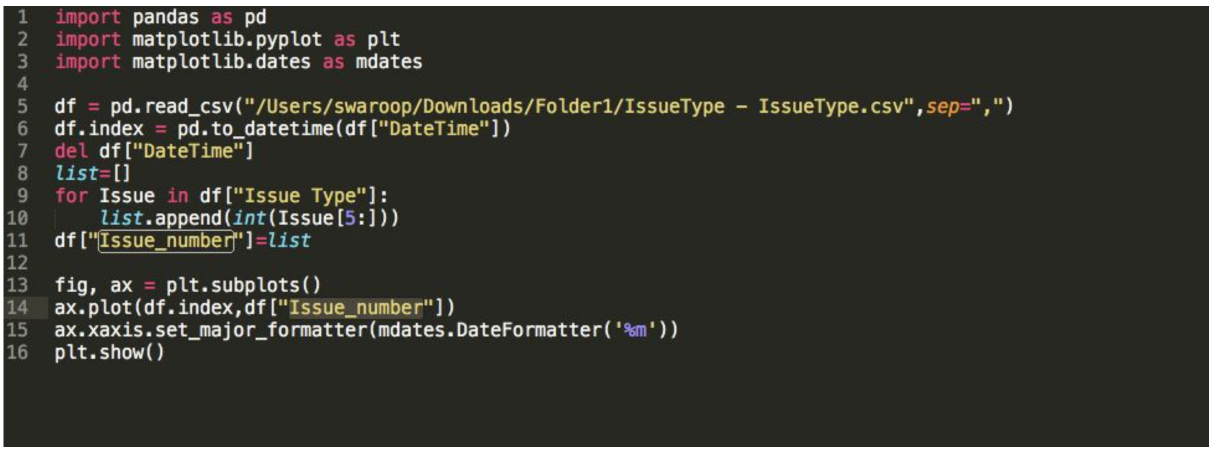
	A	B	C
1	CaseId	Category	Opened
2	1	Damaged Property	11/2011
3	2	Damaged Property	11/2011
4	3	Damaged Property	01/2010
5	4	Homeless Concerns	01/2010
6	5	Street Light	12/2010
7	6	Noise Report	07/2010
8	7	Encampments	03/2010
9	8	Sign Repair	02/2011
10	9	Sign Repair	02/2011
11	10	Noise Report	07/2010
12		Street Light	12/2010
13			

And the output of the code :



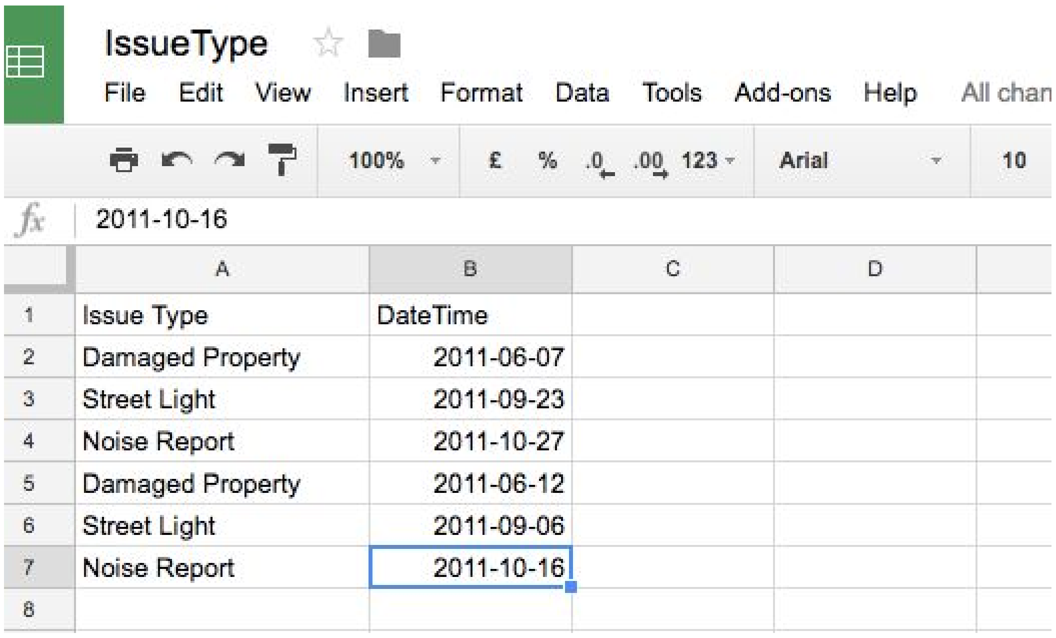
```
Swaroops-MacBook-Air:Folder1 swaroop$ python File1.py
Damaged Property 11/2011 2
Encampments 03/2010 1
Homeless Concerns 01/2010 1
Noise Report 07/2010 2
Sign Repair 02/2011 2
Street Light 12/2010 2
```

A second approach to evaluate this task would be through graphical programming in python or any other language.



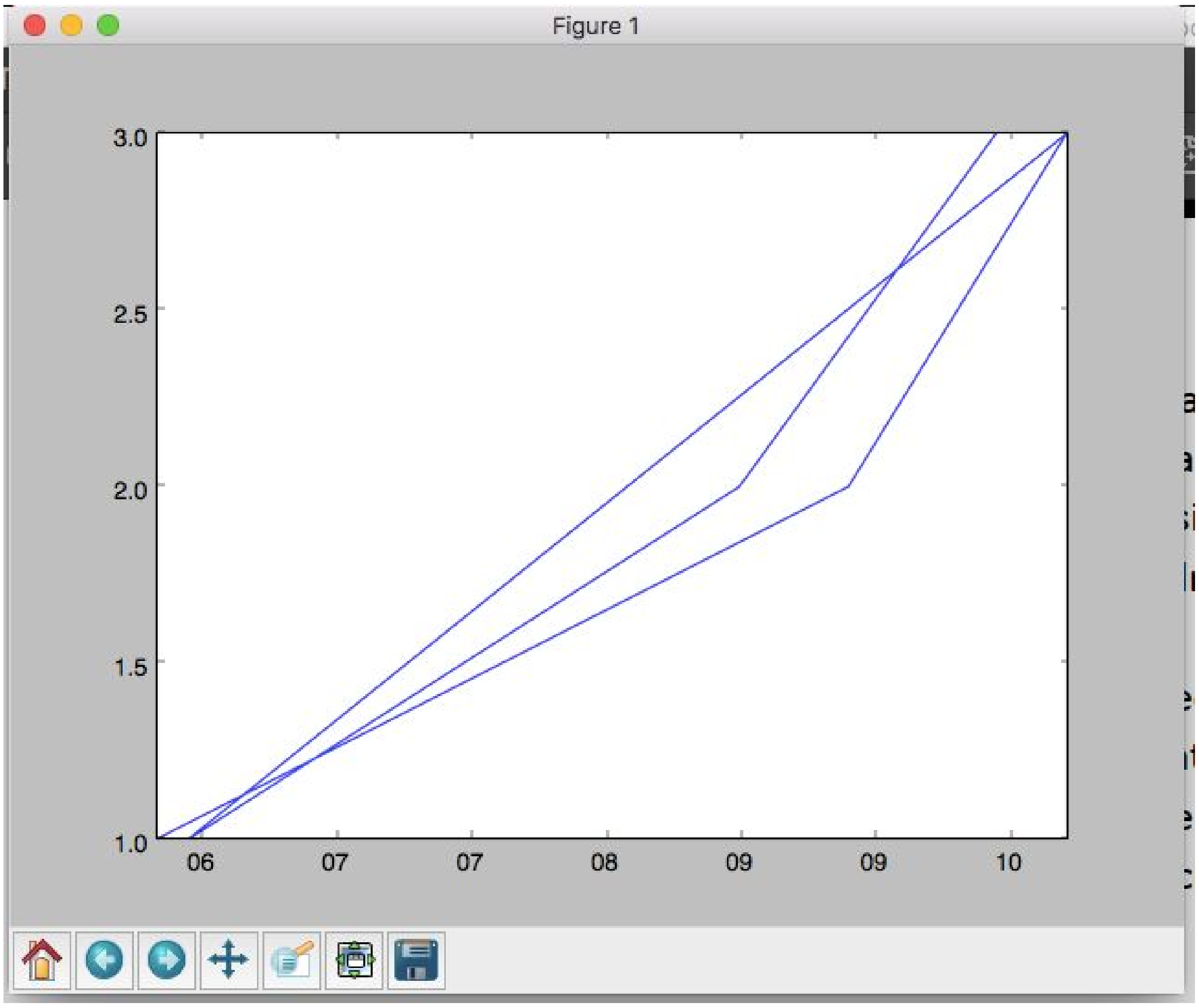
```python
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

df = pd.read_csv("/Users/swaroop/Downloads/Folder1/IssueType - IssueType.csv",sep=",")
df.index = pd.to_datetime(df["DateTime"])
del df["DateTime"]
list=[]
for Issue in df["Issue Type"]:
    list.append(int(Issue[5:]))
df["Issue_number"]=list

fig, ax = plt.subplots()
ax.plot(df.index,df["Issue_number"])
ax.xaxis.set_major_formatter(mdates.DateFormatter('%m'))
plt.show()
```

For the sake of simplicity: The above code uses the same csv as below -


IssueType
File Edit View Insert Format Data Tools Add-ons Help All chan
100% £ % .0 .00 123 Arial 10
fx 2011-10-16

	A	B	C	D
1	Issue Type	DateTime		
2	Damaged Property	2011-06-07		
3	Street Light	2011-09-23		
4	Noise Report	2011-10-27		
5	Damaged Property	2011-06-12		
6	Street Light	2011-09-06		
7	Noise Report	2011-10-16		
8				

And the result would look as below (it's not aesthetically pleasing, but we can work on making it better):



Although I have written the code in python - I feel that any kind of visualisation is better done through tool such as Tableau or QlikView, which hits the database where the data mart and data models are stored. If we're constrained towards using programming language then we can go for R programming language as its has better feature to develop and analyse trends using data mining.

Data Quality - As part of DQ, we need to ensure that all the data that is present in the source is brought into the data lake. Since the data lake acts as the source of truth for all the data - we would have to run automated DQ to check the consistency of the data. The data should not duplicated, not be skipped and partial data shouldn't be brought in.

As a next step in DQ - Once the dataMart and models have been generated - We would have to ensure that the data pipeline between the datalake and data marts are flawless and there is no data loss or duplication - as data mart is the place where stakeholders would generate reports to analyse their KPI - it's quite important to have DQ checks there as well.