1. Your name and Columbia UNI, and your teammate's name and Columbia UNI.
    - Claire Dawson, cd3417
    - Ashwin Somasundaram, as7116
2. A list of all the files that you are submitting.
    - mining.py
    - create_csv.py (has code for csv modification)
    - INTEGRATED_DATASET.csv
    - output.txt
3. A clear description of how to run your program. Note that your project must **compile/run in a Google Cloud VM** **set up exactly following our instructions**; if you need more memory, please indicate so clearly in the README file and specify what configuration we should use for your Google Cloud VM. Provide all commands necessary to install the required software and dependencies for your program.
   Execute the following command to run the program:
   python mining.py INTEGRATED_DATASET.py 0.1 0.7
   All the required libraries to run the program are listed below:
    - Numpy,pandas,itertools,math,heapq,sys
    - pip install numpy
    - pip install pandas
4. A detailed description explaining: (a) which NYC Open Data data set(s) you used to generate the INTEGRATED-DATASET file; (b) what (high-level) procedure you used to map the original NYC Open Data data set(s) into your INTEGRATED-DATASET file; (c) what makes your choice of INTEGRATED-DATASET file compelling (in other words, justify your choice of NYC Open Data data set(s)). The explanation should be detailed enough to allow us to recreate your INTEGRATED-DATASET file exactly from scratch from the NYC Open Data site

a. The NYC Open Data Set we chose from was Affordable Housing Production by Building (from https://data.cityofnewyork.us/Housing-Development/Affordable-Housing-Production-by-Building/hg8x-zxpr/about_data). It was chosen just out of fascination with the housing market in New York. Particularly, we were curious to learn more about the housing projects specific to New York, as we were unfamiliar with it before moving to New York. Specifically, we were curious about learning if there was a variety of income levels at most or if there were varying and where most were located. The original file had a lot of columns (or items) but many were numbers. So to make the a priori algorithm more efficient and effective, columns deemed not useful were removed and many were merged. Specifically, we created three major buckets of new cell values (based on a combination of other

values) and kept three of the original columns (project name, burrow, and NTA (Neighborhood Tabulation Area)). The first generated column was called Income. This column took from 5 columns that stored a count of Extremely Low Income Units, Very Low Income Units, Low Income Units, Moderate Income Units, and Middle Income units (other income units were ignored because they were not indicative). Using a weighted count vector (1,2,3,4,5) and dotting if with the value of each of the respective units listed before and then divided by the total sum of all 5 different income units, we got a new value. The higher this value is the higher the general income would be for that specific project. Thus, we created 3 buckets based on where this value falls: very_low, low and moderately_low. The thresholds ended up being <1.85 for very_low, >= 1.85 but <= 3 for low and moderately_low for >3. The specific values were guess and check until there was a relatively even distribution with a slightly higher count for “low.” The second created bucket was if most of the affordable units in the building were rented vs owned. Of the units for a given housing project there was a count that was rented and a count that was owned. Using these values, we simply took a ratio of these to the total. If the rental ratio was greater than .5 we labeled that one as mostly_rented, if the owner ratio was greater than .5 we labeled that as mostly_owned, and if neither were greater than .5 (meaning there was other types of units factoring into the total unit count) we simply labeled that one as unclear. To clarify, nothing else was labeled as unclear so if that appeared in item sets we would know what it indicated. The last made bucket was the ratio of affordable units to total units in the building that the project takes place in. For this one, we simply divided the number of affordable over total and if the value was less than .3 it was labeled as mostly_standardd (ie standard priced units in the building), if it was greater than .7 it was labeled as mostly_affordable (because most of the units were apart of the affordable housing project), and if it fell between .3 and .7 it was labeled as mixed_aff_and_stand. The reason we kept the 3 original columns and made the 3 new columns is because we were curious about how the projects varied from different neighborhoods and burrows, and how the correlations of them changed (with things like income, rentals vs owned, and ultimately if the it was just part of the building or the whole thing). Since we did a lot of manipulation to the original file We are including the original file, the new file, and the code used to modify and create these new buckets. Note: we discussed in office hours that it would be okay to include the code because it helps clarify the modifications explained above.

5. A clear description of the internal design of your project; in particular, if you decided to implement variations of the original a-priori algorithm (see above), you must explain precisely what variations you have implemented and why.

We implemented the a-priori algorithm specified in the Agrawal and Srikant paper in VLDB 1994. Specifically, we followed the general approach of incrementally creating candidate itemsets of k, by utilizing previously computed k-1 size frequent itemsets and then pruning to retrieve the plausible itemsets of size k. We further filter out some elements from this set to retrieve the frequent itemsets which satisfy the minimum threshold for the support as specified by the user. We keep repeating this process until we are unable to generate any more new itemsets.

Once we have received all of the frequent itemsets, we then proceed to calculate the association rules based on the minimum confidence specified by the user. We consider each frequent item set, and remove each item one by one to consider it as the RHS (or) the result of the association rule. We then proceed to see if any combinations of the remaining items in the itemset (or) RHS, form a good predicate and have the specified minimum confidence. We calculate this by taking a ratio of the support of the LHSxRHS union vs the support of the LHS. We add the resulting association rules to a max heap, to be able to retrieve the association rules in order of decreasing confidence.

6. The command line specification of a compelling sample run (i.e., a *min_sup*, *min_conf* combination that produces association rules that are revealing, surprising, useful, or helpful; see above). Briefly explain why the results are indeed compelling. 7. Include any information you consider significant.

   After a lot of trial and error, we finalized the most meaningful results with a minimum support of 10% and a minimum confidence of 70%. Having a high support of 10% ensures that a significant number of records/rows in the dataset are representative of the association rules which are being mined. We also want a high confidence to ensure that the rule itself is indicative of some assocation. We obtain some very interesting results when querying the dataset with these metrics. Below, you can see the most meaningful ones:

   ('Brooklyn', 'low')=>('mostly_rented',) (Conf: 96.888%, Supp: 20.296%)
   ('Bronx', 'low')=>('mostly_affordable',) (Conf: 94.608%, Supp: 13.601%)
   ('Brooklyn',)=>('mostly_rented',) (Conf: 91.941%, Supp: 40.803%)
   ('Bronx',)=>('mostly_rented',) (Conf: 90.135%, Supp: 23.502%)
   ('moderately_low',)=>('mostly_rented',) (Conf: 88.004%, Supp: 33.862%)
   ('low',)=>('mostly_rented',) (Conf: 86.735%, Supp: 42.970%)
   ('low',)=>('mostly_affordable',) (Conf: 86.700%, Supp: 42.953%)
   ('Manhattan',)=>('mostly_rented',) (Conf: 86.124%, Supp: 16.949%)
   ('Brooklyn', 'moderately_low')=>('mostly_rented',) (Conf: 85.675%, Supp: 16.332%)

('Bronx', 'low')=>('mostly_rented',) (Conf: 85.539%, Supp: 12.297%)
('Brooklyn', 'low')=>('mostly_affordable',) (Conf: 83.347%, Supp: 17.459%)
('Manhattan',)=>('mostly_affordable',) (Conf: 82.005%, Supp: 16.138%)
('Bronx',)=>('mostly_affordable',) (Conf: 78.243%, Supp: 20.402%)

We see that in Brooklyn, people in low income households tend to live in mostly rented apartments/buildings, while these apartments (when inhabited by low income people) are mostly affordable with a lower confidence. We see that in the Bronx, people with a low income tend to live in apartments that are usually mostly affordable. The next set of significant rules state that both Bronx and Brooklyn are mostly rented, and only a few people really own the places they live in. An interesting observation is that low income households both live in mostly rented and mostly affordable households, with an overwhelming support of more than 40%! A separate rule states that Bronx is mostly affordable in general based on a solid support of 20%. This intuitively makes sense as Bronx is often viewed to be one of the more affordable boroughs in New York City. There is an interesting observation that Manhattan is mostly affordable with a support of 16%, but we suspect that this is due to the nature of the dataset itself, that lower income households and affordable housing options comprise the majority of the records in the dataset itself.
That being said, modifying the support and confidence has seemed to yield more meaningful and smaller association rules which seem to be created with more items in the dataset.