Hung Nguyen `(hungn2)`
Anish Saha `(saha9)`
Sharanya Balaji `(sbalaji3)`

1. Points received for initial Phase-1 submission : 94 (out of 100)
2. Our team chooses the following Phase-1 option:
   - [ ] (A) No change to Phase-1 report
   - [X] (B) Improved Phase-1 report

Table of Contents

Here are the following changes made to Phase 1 to address the comments made. These additions have also been highlighted in the Phase 1 Report attached to the end of this document.

Phase 1 Remarks (TA feedback):	**Student Comments / Summary:**
Use cases: [-2] Missing a reason why U0 requires "zero data cleaning".; Data quality list: nice work!;	U0 would not require any data cleaning, as our original dataset D can still be used to find a listing. However, the experience would be a lot less efficient as there would be many undesirable listings.
Initial plan: [-2] Did not specify methods for checking whether the original dataset has been cleaned for your use case after cleaning (e.g. using integrity constraints).	We will be utilizing SQL Integrity Constraints at the end of our procedures to confirm that our cleaned dataset matches the conditions specified in use case U1. We will run the SQL commands prior to data cleaning as well so the change can be visibly understood.
[-2] Did not specify how you will document changes made to the original dataset after cleaning (e.g. using a summary table).	The flowchart and the Jupyter Notebook will document changes made to the original dataset after cleaning by adding a statement to print how many rows are removed with each step. We will add a summary table to our Phase 2 report that clearly delineates all the changes made.

CS 513 Final Project - PHASE II

Description of Dataset and Use Cases

The dataset of interest **D** is the New York City Airbnb dataset (04/07/21 - 04/12/21), which depicts 36,905 Airbnb listings from various locations in New York City. The motivation and target use case **U1** for **D** is described below.

U1: In our case, **U1** describes a use case where a user is seeking to take a college trip, so the user is part of a group of 4 friends seeking to stay at a place in New York City for 6 nights within a specific price range (less than $800 total per night, equivalent to $200 per person per night), neighborhood (Manhattan), and is well-reviewed. In addition, the user would prefer that the place be an entire private setup (without the host), that there be at least 2 separate beds, at least 1.5 bathrooms, and that the host provides at least 1 image (to ensure veracity). Finally, to ensure that both the host and users involved feel comfortable, users would prefer experienced, verified hosts with response times that are less than a day.

Data Cleaning Performed

Our team performed all data cleaning steps using Python. Here are the following data cleaning steps with the corresponding explanations:

The first data cleaning step was filtering by the number of nights. Our use case requires listings that allow their guests to stay for 6 nights. The database is filtered so that minimum_nights > 0, minimum_nights <=6 and maximum_nights >= 6 because our U1 asks for a place for 6 nights. This eliminates all invalid Airbnb entries that should not be shown, while also removing rows that would not allow the guests to stay the desired number of nights. This was a required step to satisfy U1.

The next data cleaning step is preprocessing and cleaning the bathroom_text column. The dataset originally included a bathrooms_text column. This column was structured with the numeric value followed by text like “bath” or “baths”. For example, “1.5 baths,” “2 baths,” and “1 bath.” The text in the column does not add any useful information and would, in fact, require additional checks to compare strings as opposed to comparing simple integers. Therefore, this column was cleaned to remove the text. After cleaning this column, we added a filter to ensure all listings had at least 1.5 bathrooms. Given that we made the column easily comparable through floats, this filter step was significantly easier. This was a required step to satisfy U1. This was not necessary for U1 but improves the user experience to easily determine the number of bathrooms available.

The next data cleaning step is filtering by price, number of guests, and beds. To identify valid records in the data for U1, the dataset needs to be filtered so that listings included accommodations for at least 4 people, 2 beds and less than 800 dollars. Therefore, we check if the *accommodates* column is greater than or equal to 4, the *beds* column is greater than or equal to 2, and the price column is greater than 0 and less than 800. The purpose behind this data cleaning step is to ensure that all listings in the dataset only include the desired accommodation, bed count, and correct pricing. This was a required step to satisfy U1.

The next data cleaning step is filtering by specific neighborhood (Manhattan). Our use case requires listings that are within the Manhattan neighborhood. The dataset includes listings from every neighborhood in New York City. Therefore, we filtered the *neighbourhoods_cleansed* column to the following options "Financial District," "Greenwich Village," "Soho," "Lower East Side," "Chinatown," "Hamilton," "Clinton," "Chelsea," "Midtown," "Stuyvesant Town," "Turtle Bay," "Upper West Side," "Inwood," "Upper East Side," "Central Harlem," "East Harlem," "Morningside Heights," and "Washington Heights." The purpose behind this data cleaning step is to ensure that all listings in the dataset only include the desired neighborhoods of Manhattan. This was a required step to satisfy U1.

The next data cleaning step is filtering by well-reviewed rentals. Our use case requires listings that are considered "well-reviewed." We have defined a "well-reviewed" listing as a listing that has over 10 reviews with a review score rating of over 80.0. The purpose behind this data cleaning step is to ensure that all listings in the dataset only include well-reviewed listings. This ensures that the users will easily be able to select a place that will give them the best possible experience. Additionally, this data cleaning step was required to satisfy the conditions for U1.

The next data cleaning step is filtering by room type and removing entries for shared rooms, apartments, and homes. Our use case requires listings that are considered a "private setup." This means that the listing should not be shared with the host. While the property_type column is more descriptive, there are far more possible values in this column, many of which may not have been shown in this dataset. Moreover, there is ambiguity in which specific types would best satisfy the conditions for UI, so we performed filtering on the room_type column instead. To do this, we filter out all rows where the room_type parameter is not equal to "Entire home/apt." This was a required step to satisfy U1.

The next data cleaning step is filtering out places that have no images. We wanted to ensure that all listings included had actual images that could be looked through in order to make the best decision. Any listings that did not have pictures were considered a "bad listing" as it would not satisfy our use case. This was a required step to satisfy U1.

The next data cleaning step is filtering out any hosts that are not experienced, not verified, and have slow response times. We categorized lack of experience as any host that has been a host only since 2020 or 2021. We wanted to remove any listings that had hosts that fell in this category. We also ensured that the host_identity_verified column was checked to true as we only wanted hosts whose identities have been verified. Finally, we only wanted to include listings that had response times from hosts that were less than a day. Therefore, we only wanted to include listings that had values for host_response_time as either “within an hour” or “within a few hours.” All of these steps were required to satisfy U1.

The next data cleaning step is to remove listings where smoking is not allowed. By searching for keywords (`‘smok’`: to account for various forms of the word smoke) in the name and description column, a new column was added to flag places with no smoking. This column was then filtered so that places with no smoking are removed. This is useful because U1 is for college students looking to have a good time and be allowed to smoke.

The next data cleaning step is to remove listings where partying is not allowed. Our use case requires listings where partying is allowed. By searching for keywords (regex: `'no.{0,5}party'`) in the name and description column, a new column was added to flag places with no partying. This column was then filtered so that places with no partying are removed. This is useful because U1 is for college students looking to have a good time and party.

Removing unnecessary columns is the final data cleaning step in our project. Many columns are not required for the use case or do not require any useful additional information to the user. Having so many unnecessary columns creates confusion to the user and makes the dataset hard to understand. Therefore, we removed 54 columns. The names of the columns removed can be found in the Jupyter Notebook. While this is erasing data, we believe that removing these columns significantly enhances the customer’s experience as it will be very hard for a customer to comprehend 80 categories when weighing the pros and cons of listings.

Data Quality Changes

Our dataset originally included 36905 rows and 74 columns.

Name of Data Cleaning Step	Column Associated to Cleaning	Cells (per column) changed
Filter by number of nights	Minimum_nights & maximum_nights	26312 rows removed.
Preprocess and clean the bathrooms_text column	Bathrooms_text changed to bathrooms	New column created & 8911 rows removed.
Filter by price, number of guests, and number of beds	Price, accommodates & beds	869 rows removed.
Filter by specific neighborhood (Manhattan)	neighbourhood_cleansed	671 rows removed.
Filter by well-reviewed rentals	number_of _reviews & review_scores_ratings	74 rows removed.

Filter by room type, remove entries for shared rooms, apartments, and homes	room_type	6 rows removed.
Filtering out places with no images	picture_url	0 rows removed
Filtering out places with no parties allowed	name & description scraped to add a column (no party?) that flags no parties allowed	0 rows removed.
Filtering out places with no smoking allowed	name & description scraped to add a column (no smoking?) that flags no smoking allowed	2 rows removed
Filtering out hosts that are not verified	host_identity_verified	9 rows removed
Filtering out hosts that do not respond within a few hours	host_response_time	7 rows removed
Filtering out hosts that have only been hosts since 2020 or 2021	host_since	14 rows removed
Removing all unnecessary columns		

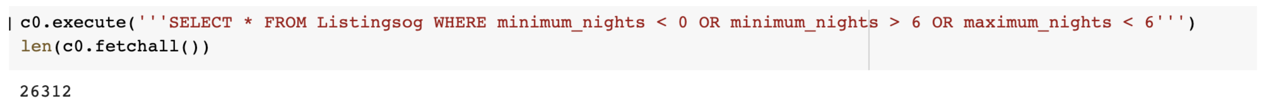
Integrity Constraints:

In order to ensure the data quality has actually improved through the steps outlined above, we ran the following SQL query integrity constraints in our Jupyter Notebook to ensure that these following conditions are not met. If the queries returned, 0 rows then we were confident that our data cleaning goal was met.

All of these queries were run prior to performing data cleaning as well so we could see the difference. The screenshots of the results from Before and After data cleaning are attached below.

SELECT * FROM Listings WHERE minimum_nights < 0 OR minimum_nights > 6 OR maximum_nights < 6

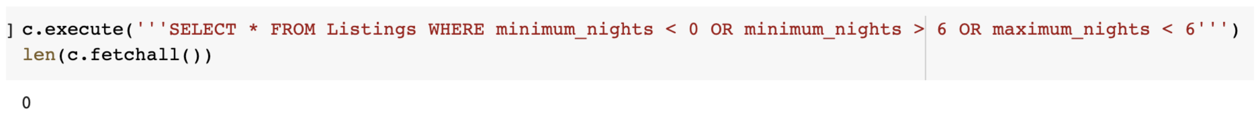
BEFORE



```
c0.execute('''SELECT * FROM Listingsog WHERE minimum_nights < 0 OR minimum_nights > 6 OR maximum_nights < 6''')
len(c0.fetchall())

26312
```

AFTER

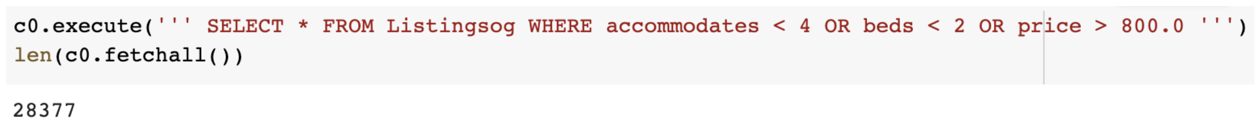


```
] c.execute('''SELECT * FROM Listings WHERE minimum_nights < 0 OR minimum_nights > 6 OR maximum_nights < 6''')
  len(c.fetchall())

  0
```

SELECT * FROM Listings WHERE accommodates < 4 OR beds < 2 OR price > 800.0

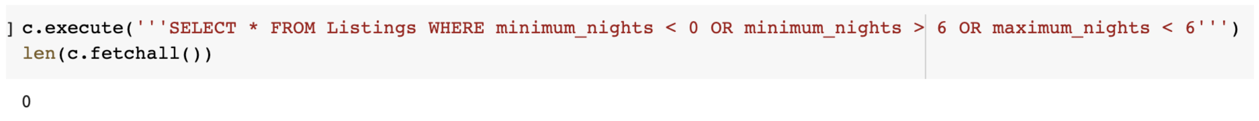
<u>BEFORE</u>



```
c0.execute(''' SELECT * FROM Listingsog WHERE accommodates < 4 OR beds < 2 OR price > 800.0 ''')
len(c0.fetchall())

28377
```

<u>AFTER</u>

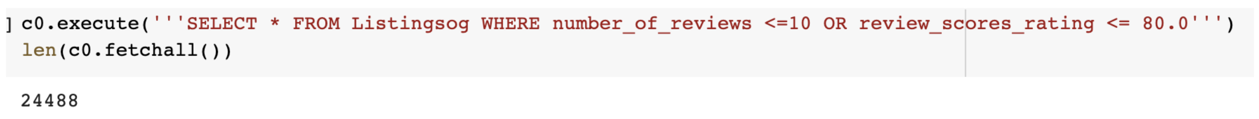


```
] c.execute('''SELECT * FROM Listings WHERE minimum_nights < 0 OR minimum_nights > 6 OR maximum_nights < 6''')
  len(c.fetchall())

  0
```

SELECT * FROM Listings WHERE number_of_reviews <=10 OR review_scores_rating <= 80.0

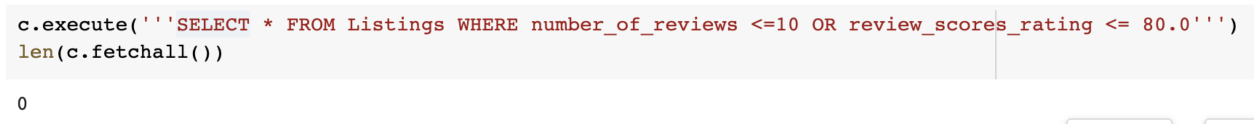
<u>BEFORE</u>



```
] c0.execute('''SELECT * FROM Listingsog WHERE number_of_reviews <=10 OR review_scores_rating <= 80.0''')
  len(c0.fetchall())

  24488
```

<u>AFTER</u>

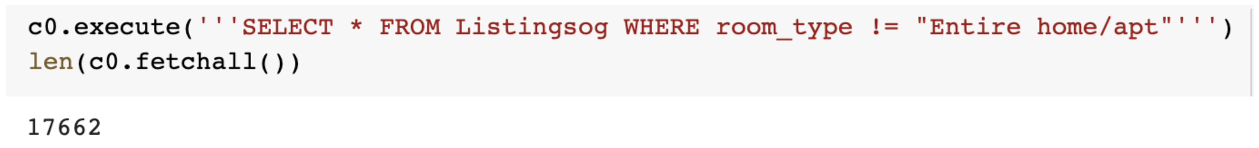


```
c.execute('''SELECT * FROM Listings WHERE number_of_reviews <=10 OR review_scores_rating <= 80.0''')
len(c.fetchall())

0
```

SELECT * FROM Listings WHERE room_type != “Entire home/apt”

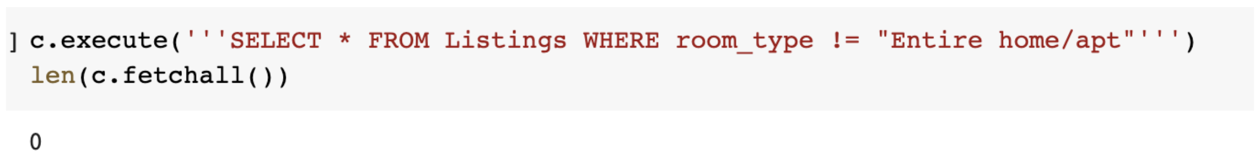
<u>BEFORE</u>



```
c0.execute('''SELECT * FROM Listingsog WHERE room_type != "Entire home/apt"''')
len(c0.fetchall())

17662
```

<u>AFTER</u>

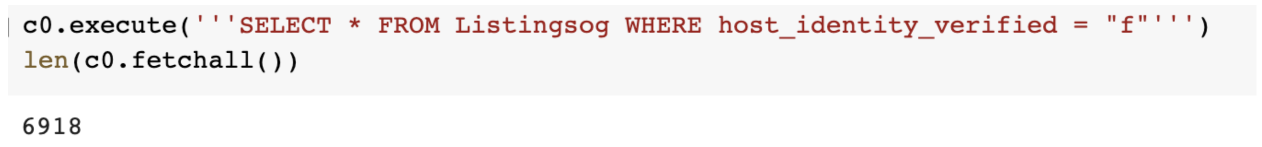


```
] c.execute('''SELECT * FROM Listings WHERE room_type != "Entire home/apt"''')
  len(c.fetchall())

  0
```

SELECT * FROM Listings WHERE host_identity_verified = “f”

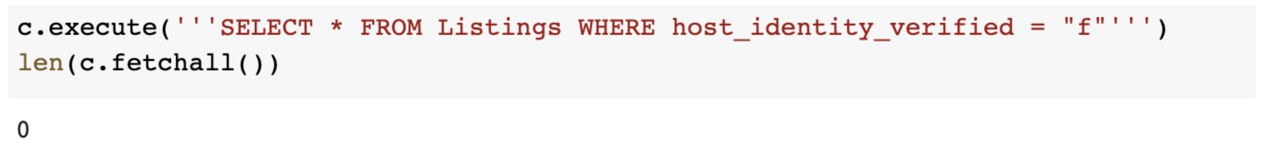
<u>BEFORE</u>



```
c0.execute('''SELECT * FROM Listingsog WHERE host_identity_verified = "f"''')
len(c0.fetchall())
```

```
6918
```

<u>AFTER</u>

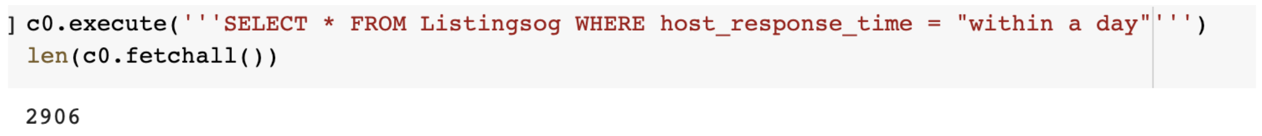


```
c.execute('''SELECT * FROM Listings WHERE host_identity_verified = "f"''')
len(c.fetchall())
```

```
0
```

SELECT * FROM Listings WHERE host_response_time = “within a day”

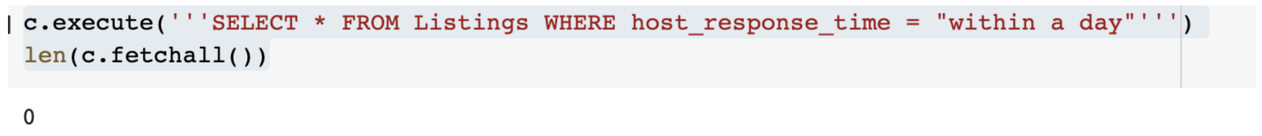
<u>BEFORE</u>



```
c0.execute('''SELECT * FROM Listingsog WHERE host_response_time = "within a day"''')
len(c0.fetchall())
```

```
2906
```

<u>AFTER</u>



```
c.execute('''SELECT * FROM Listings WHERE host_response_time = "within a day"''')
len(c.fetchall())
```

```
0
```

Workflow Model

In this section, the outer/overall workflow model and the inner workflow model are presented.

The outer/overall workflow model is shown in Figure 1 and showcases a high level overview of our project and end-to-end steps. We first profile our dataset using Excel because it provides a very easy interface to manually inspect the dataset columns and entries to devise a plan on cleaning the dataset for U1. The data was then loaded, cleaned, and checked for IC violations all in Python. Python was the tool of choice because of our group's experience in Python and its ability to easily filter and clean data step-by-step. Python also allowed us to easily create a workflow model from our code using the library `pyflowchart`.

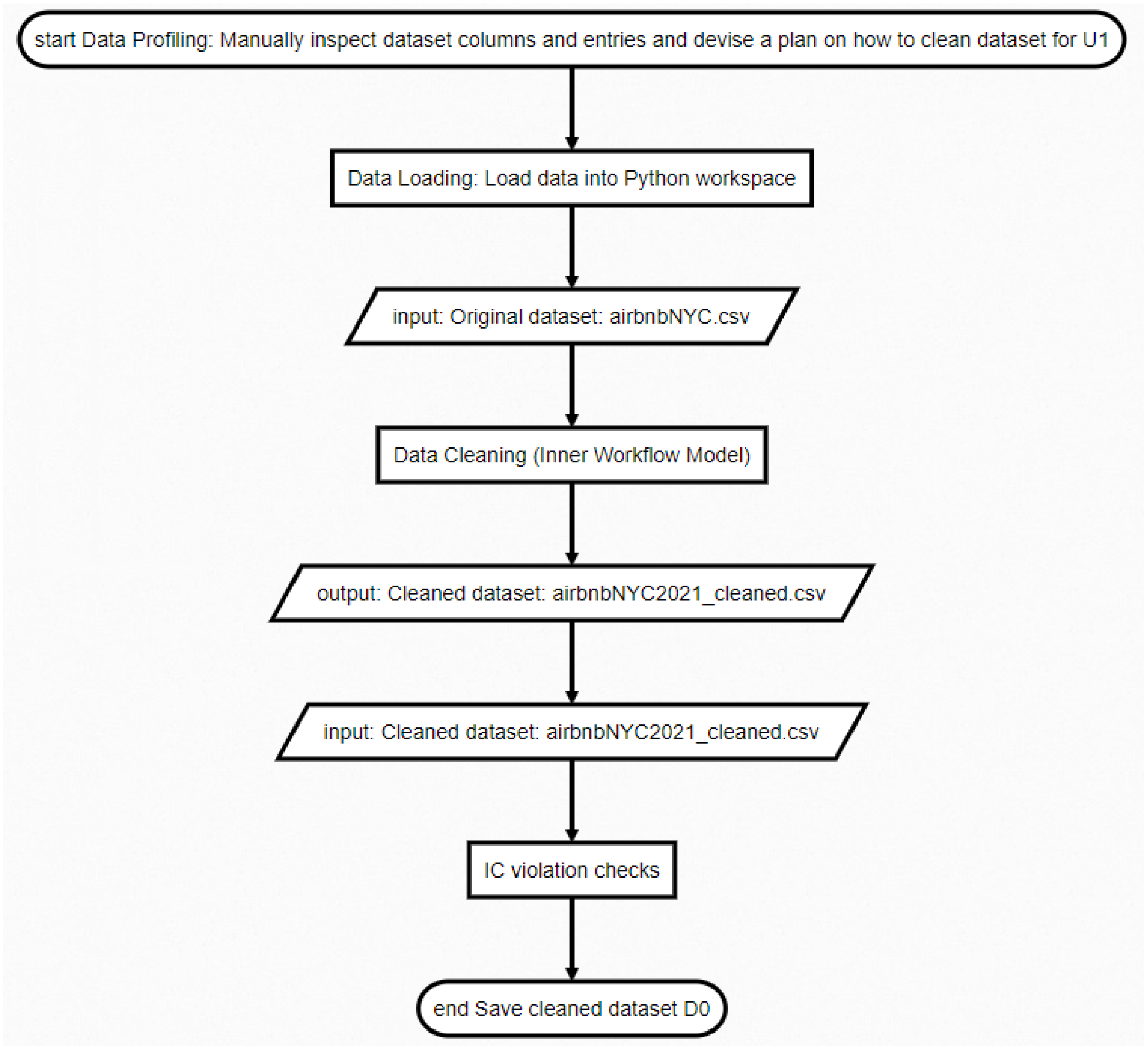

start Data Profiling: Manually inspect dataset columns and entries and devise a plan on how to clean dataset for U1
Data Loading: Load data into Python workspace
input: Original dataset: airbnbNYC.csv
Data Cleaning (Inner Workflow Model)
output: Cleaned dataset: airbnbNYC2021_cleaned.csv
input: Cleaned dataset: airbnbNYC2021_cleaned.csv
IC violation checks
end Save cleaned dataset D0

Figure 1: Outer Workflow Model

The inner workflow model is shown in Figure 2 and provides each step in our data cleaning process. Refer back to the Data Cleaning Section for details of each step.

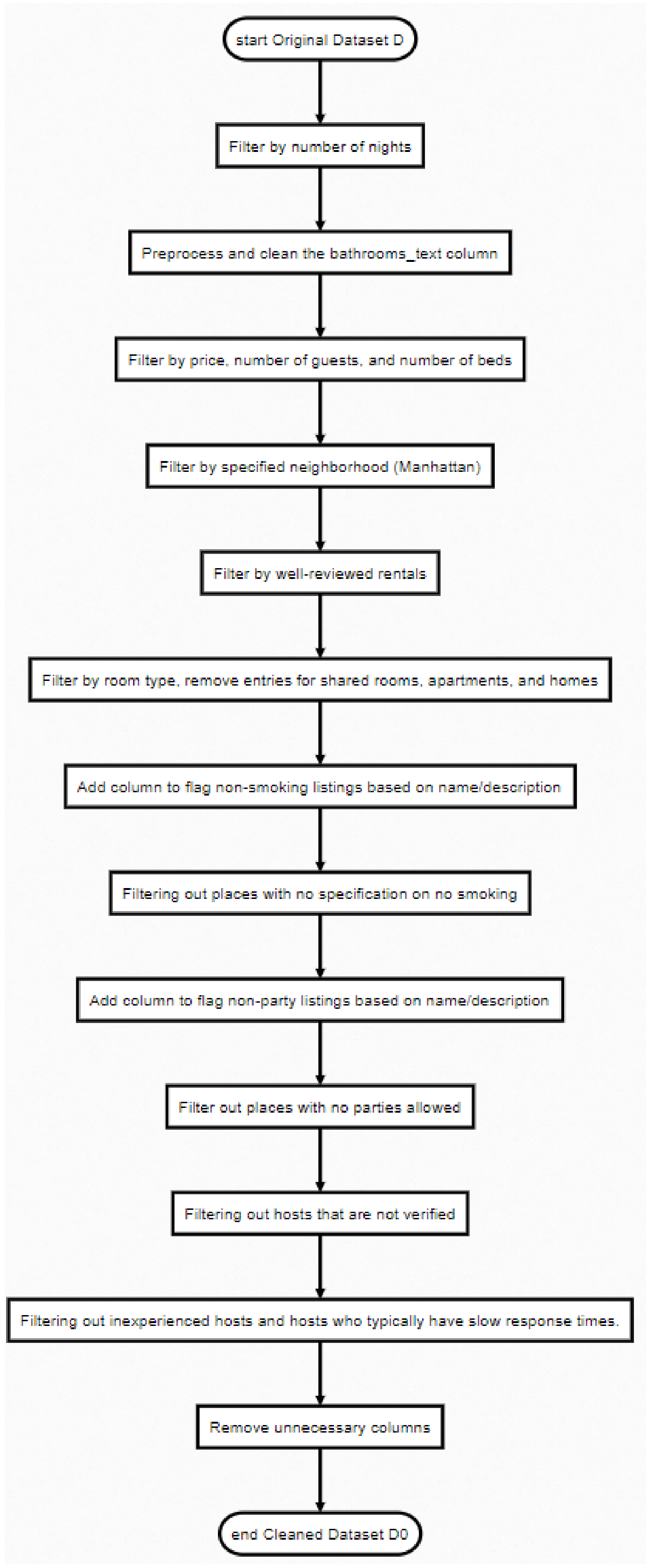

start Original Dataset D
Filter by number of nights
Preprocess and clean the bathrooms_text column
Filter by price, number of guests, and number of beds
Filter by specified neighborhood (Manhattan)
Filter by well-reviewed rentals
Filter by room type, remove entries for shared rooms, apartments, and homes
Add column to flag non-smoking listings based on name/description
Filtering out places with no specification on no smoking
Add column to flag non-party listings based on name/description
Filter out places with no parties allowed
Filtering out hosts that are not verified
Filtering out inexperienced hosts and hosts who typically have slow response times.
Remove unnecessary columns
end Cleaned Dataset D0

Figure 2: Inner Workflow Model

Conclusions & Summary

After completing this project, we were left with 30 wonderful listings and we chose the Big 🍎 NYC STAY & TRAVEL like ROYALTY w/ Elevator listing with id of 305777. We are so excited!

This project was a great opportunity to understand all the nuances regarding data cleaning. Through this project, we were able to select a dataset, create a target use case, and outline all the required steps to create the most accurate, clean and concise dataset that would allow us to answer our case use with the best customer experience. Additionally, we learned about what is required to double-check data cleaning steps with the use of integrity constraints. This is super valuable to ensure that you have cleaned the data the way you intended it to be cleaned. All the skills required to complete this project will prove valuable in future projects.

The following tasks were worked on by the corresponding team members:

- Sharanya Balaji - Some data cleaning steps, Integrity Constraint Queries, and Phase 1 & 2 Report
- Anish Saha - Setting up Jupyter notebook, some data cleaning steps, and Phase 1 & 2 Report
- Hung Nguyen - Some data cleaning steps, Workflow models, and Phase 1 & 2 Report

Supplementary Materials

1. Conceptual Model / Database Schema - attached in zip
2. Operation History - attached in zip
3. Queries - attached in zip
4. Outer and Inner Workflow Models - attached in zip
5. Raw Dataset - uploaded to Box
6. Cleaned Dataset - uploaded to Box

CS 513 Final Project - PHASE I

*1. Identify a dataset **D** of interest. This can be a provided dataset (NYPL menus; PPP loan applications; US Farmers Markets), or a new dataset that your team would like to work with.*

We will be using the New York City Airbnb dataset (04/07/21 - 04/12/21), which depicts 36,905 Airbnb listings from various locations in New York City.

http://insideairbnb.com/get-the-data.html

*2. Develop a target use case **U1** for **D** such that data cleaning is necessary and sufficient to support the data analysis use case. Thus, after performing data cleaning, your cleaned data **D0** is fit-for-purpose (i.e., for **U1**). In addition to your main use case **U1** you should briefly describe two minor use cases: **U0** should be a use case that requires "zero data cleaning", i.e., **D** is "good enough as it is". In contrast, **U2** is a use case for which the given dataset **D** is "never (good) enough", i.e., no amount of data cleaning or wrangling will make **D** suitable for **U2** (even though at first sight one might think so). The purpose of the corner cases **U0** (data cleaning is not necessary) and **U2** (data cleaning is not sufficient) is to reinforce the concept that data cleaning should be done with a purpose in mind, i.e., a use case such as your target use case **U1**, where data cleaning really makes a difference.*

U0: In our case, **U0** describes a use case where a user is seeking to stay at any place in New York City, where the user has not specified many preferences in terms of price, location, etc. This would require no data cleaning, as it would still be useful for the user to book a trip, but the experience would be a lot less efficient as there would be many undesirable listings. U0 would not require any data cleaning, as our original dataset D can still be used to find a listing. However, the experience would be a lot less efficient as there would be many undesirable listings.

U1: In our case, **U1** describes a use case where a user is seeking to take a college trip, so the user is part of a group of 5 friends seeking to stay at a place in New York City for 6 nights within a specific price range (<$200/night), neighborhood (Lower Manhattan), and is well-reviewed. In addition, the user would prefer that the place be a private setup, that there be at least 4 beds, and that the host provides at least 2 images (to ensure veracity). Finally, to ensure that both the host and users feel comfortable, the user would likely prefer an experienced, non-foreign host.

U2: In our case, **U2** describes a use case where a user is seeking to visit New York City under the same specifications as **U1**, but would also like to stay at a location within walking distance of at least 5 major tourist attractions. This requires outside knowledge of locations and information regarding all tourist attractions that is not available within the dataset itself, so data cleaning will not be sufficient to resolve the user's needs in this particular use case.

*3. Describe the dataset **D**. For example, you can provide a conceptual model (ER diagram) that depicts the entity types and relationship types, or an ontology that illustrates the main classes and their relationships. Or you can provide a database schema that illustrates and explains the structure and contents of the dataset. You should also add a short narrative, i.e., one or more paragraphs in English to describe the origin of the data and any relevant metadata (e.g., a temporal or spatial extent). A dataset about farmers markets, e.g., can be described with a relational schema (e.g., CREATE TABLE statements); the narrative would then explain what the different columns (attributes) mean. Other metadata may describe, e.g., the spatial extent of the data (only Illinois markets? All of the Midwest? Or the US?), and the temporal extent (for which period is the data correct?), etc.*

Dataset **D** represents all Airbnb listings in the New York Metropolitan Area between April 7, 2021 and April 12, 2021. The dataset includes a lot of information, many of which are redundant or unnecessary for **U1**. A large part of our planned data cleaning process will be to remove the redundant and unnecessary columns. The remaining columns can then be further cleaned per Section 4.

All details are listed below with

- *id* - The unique identifier of that particular AirBnb listing.
- *listing_url* - link to listing on Airbnb.com
- *scrape_id* - ID used when scraping data from Airbnb.com. Not useful
- *last_scraped* - Date when data was last scraped. Not useful
- *name* - The name of the listing
- *description* - This category provides additional information about the listing and describes the listing in more detail.
- *neighborhood_overview* - This category describes the neighborhood in terms of where it is located, the type of ambience, and other descriptive information.
- *picture_url* - check to make sure url is active*
- *host_id* - contains the id for the host of the listing. For context
- *host_url* - link to host's profile on Airbnb.com
- *host_name* - name of the host
- *host_since* - how long the host has been active
- *host_location* - where the host is primarily located. This can be and is most often different from the location of the listing.
- *host_about* - An about paragraph regarding the host. This contains additional information about the host that would not be
- *host_response_time* - how long the host usually takes to reply to a request for booking.
- *host_response_rate* - how often the host responds to booking requests.
- *host_acceptance_rate* - how often the host accepts booking requests.
- *host_is_superhost* - Boolean for whether host is a superhost (T/F).
- *host_thumbnail_url* - Small photo of host used for listing thumbnail
- *host_picture_url* - Medium photo of host used in listing
- *host_neighbourhood* - same as *neighbourhood* column; redundant.
- *host_listings_count* - how many listings the host currently has on AirBnb.

- *host_total_listings_count* - total number of listings host has listed.
- *host_verifications* - Information from host that has been verified (e.g. email, phone, social media, etc.)
- *host_has_profile_pic* - Boolean for whether host has a profile picture (T/F).
- *host_identity_verified* - boolean value of whether the host's identity has been verified.
- *neighbourhood* - Original neighbourhood data as extracted from airbnb listing. Redundant to Neighbourhood_cleansed and Neighbourhood_group_cleansed.
- *neighbourhood_cleansed* - Neighbourhood data that has already been cleansed in the original dataset, D. We assume the original author used latitude/longitude data to generate this.
- *neighbourhood_group_cleansed* - Neighbourhood group data that has already been cleansed in the original dataset, D. We assume the original author used latitude/longitude data to generate this.
- *latitude* - the latitude coordinates of the listing's location.
- *longitude* - the longitude coordinates of the listing's location.
- *property_type* - The type of property (e.g. boat, entire apartment, private room, etc). More descriptive than Room_type.
- *room_type* - This defines whether the listing is a private room or an entire home/apartment.
- *accommodates* - The number of people the listing accommodates.
- *bathrooms* - The number of bathrooms at the listing.
- *bathrooms_text* - This category defines the number of bathrooms as well as if the bathroom is a shared or private bathroom.
- *bedrooms* - The number of bedrooms at the listing.
- *beds* - The number of beds at the listing.
- *amenities* - This category lists all the amenities the listing has in a list. Examples of amenities are kitchen, cable tv, heating, bathtub, etc,
- *price* - The price of the listing.
- *minimum_nights* - minimum nights to stay
- *maximum_nights* - maximum nights to stay
- *minimum_minimum_nights* - minimum minimum nights the customer must stay
- *maximum_minimum_nights* - maximum minimum nights the customer must stay
- *minimum_maximum_nights* - minimum maximum nights the customer must stay
- *maximum_maximum_nights* - maximum maximum nights the customer must stay
- *minimum_nights_avg_ntm* - average minimum nights the host has requested that customer stay in the next twelve months
- *maximum_nights_avg_ntm* - average maximum nights the host has requested that customer stay in the next twelve months
- *calendar_updated* - Blank column
- *has_availability* - A boolean value that indicates if the listing is available or not.
- *availability_30* - Availability in the month
- *availability_60* - Availability in two months
- *availability_90* - Availability in a quarter
- *availability_365* - Availability in a year

- *calendar_last_scraped* - same as last_scraped
- *number_of_reviews* - The number of reviews corresponding to that listing.
- *number_of_reviews_ltm* - Number of reviews in the last 12 months.
- *number_of_reviews_l30d* - Number of reviews in the last 30 days.
- *first_review* - The date of the first review
- *last_review* - The data of the last review
- *review_scores_rating* - Overall score of the listing with scale 20 thru 100 (divide by 20 to get 1 - 5 stars).
- *review_scores_accuracy* - Score of the listing's accuracy with scale 2 thru 10 (divide by 2 to get 1 - 5 stars).
- *review_scores_cleanliness* - Score of the listing's cleanliness with scale 2 thru 10 (divide by 2 to get 1 - 5 stars).
- *review_scores_checkin* - Score of the listing's check-in with scale 2 thru 10 (divide by 2 to get 1 - 5 stars).
- *review_scores_communication* - Score of the listing's communication with scale 2 thru 10 (divide by 2 to get 1 - 5 stars).
- *review_scores_location* - Score of the listing's location with scale 2 thru 10 (divide by 2 to get 1 - 5 stars).
- *review_scores_value* - Score of the listing's value with scale 2 thru 10 (divide by 2 to get 1 - 5 stars).
- *license* - blank column; unnecessary
- *instant_bookable* - Boolean on whether the listing is instantly bookable (T/F).
- *calculated_host_listings_count* - Total number of listings from host (based on host_id)
- *calculated_host_listings_count_entire_homes* - Total number of listings that are entire homes from host (based on host_id)
- *calculated_host_listings_count_private_rooms* - Total number of listings that are private rooms from host (based on host_id)
- *calculated_host_listings_count_shared_rooms* - Total number of listings that are shared rooms from host (based on host_id)
- *reviews_per_month* - number of reviews per month

*4. List obvious data quality problems (i.e., which are easy to spot). In order for your dataset **D** and target use case **U1** to match, data cleaning must be necessary and sufficient to implement **U1**. You need to support this claim by documenting data quality problems that your inspection of **D** has revealed that need to be addressed before **U1** can be tackled.*

One data quality problem that is easy to spot is that many of the host names will need to be consolidated -- there are many instances where the multiple listings reference the same host, but they will be treated as separate since they differ by just a few characters.

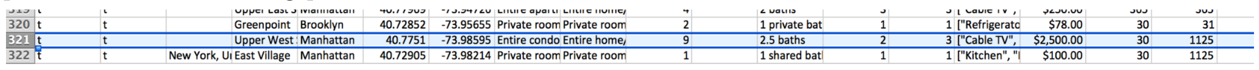
Another data quality issue is that there seem to be a number of nonsensical data points, where some listings are priced unreasonably (>$1000 per night or <$20 per night), so these useless data points will need to be removed. Moreover, there are many preferences specific to our use case (**U1**) that need to be addressed. As such, data filtering will be utilized to query only certain values (or ranges of values) for various columns in order to remove the rows of data that do not fit the use case for our project.

In addition, there is a data quality issue that some hosts do not allow certain behaviors, which is not accounted for by any features/columns in the dataset. For example, some hosts do not permit smoking, parties, social gatherings, etc. but some hosts even encourage this (based on the titles in the listing). We may try to use NLP to extract synthetic features so that our target users do not find listings that might end up incurring fines for not strictly following restrictions.

Finally, there seem to be many listings with titles containing other languages or emojis, which could obfuscate the user's ability to find meaning in the listings. We may need to perform sorts in our data cleaning process pipeline to prioritize the listings that best suit the user's needs.

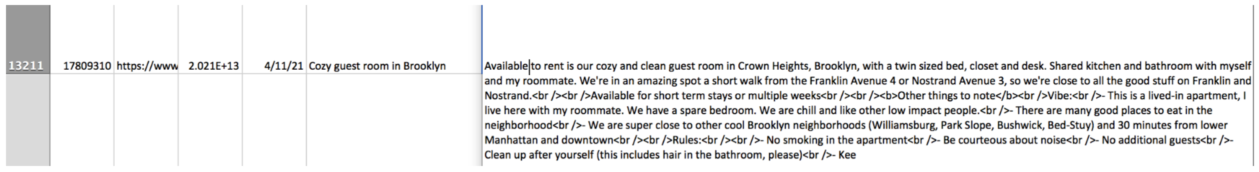
Dirty Data Examples:

[Unreasonable Pricing]



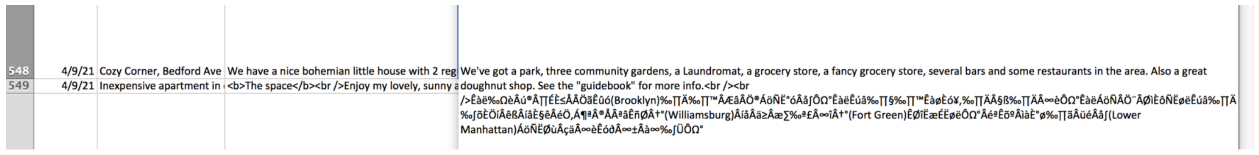
320	t	t		Greenpoint	Brooklyn	40.72852	-73.95655	Private room	Private room	2		1 private bat	1	1	["Refrigerato	$78.00	30	31
321	t	t		Upper West	Manhattan	40.7751	-73.98595	Entire condo	Entire home/	9		2.5 baths	2	3	["Cable TV",	$2,500.00	30	1125
322	t	t	New York, U	East Village	Manhattan	40.72905	-73.98214	Private room	Private room	1		1 shared bat	1	1	["Kitchen", "I	$100.00	30	1125

[Restricted Behaviors]



13211	17809310	https://www	2.021E+13	4/11/21	Cozy guest room in Brooklyn	Available to rent is our cozy and clean guest room in Crown Heights, Brooklyn, with a twin sized bed, closet and desk. Shared kitchen and bathroom with myself and my roommate. We're in an amazing spot a short walk from the Franklin Avenue 4 or Nostrand Avenue 3, so we're close to all the good stuff on Franklin and Nostrand.<br /><br />Available for short term stays or multiple weeks<br /><br /><b>Other things to note</b><br />Vibe:<br />- This is a lived-in apartment, I live here with my roommate. We have a spare bedroom. We are chill and like other low impact people.<br />- There are many good places to eat in the neighborhood<br />- We are super close to other cool Brooklyn neighborhoods (Williamsburg, Park Slope, Bushwick, Bed-Stuy) and 30 minutes from lower Manhattan and downtown<br /><br />Rules:<br /><br />- No smoking in the apartment<br />- Be courteous about noise<br />- No additional guests<br />- Clean up after yourself (this includes hair in the bathroom, please)<br />- Kee

[Foreign / Illegible Language or Text]



548	4/9/21	Cozy Corner, Bedford Ave	We have a nice bohemian little house with 2 reg	We've got a park, three community gardens, a Laundromat, a grocery store, a fancy grocery store, several bars and some restaurants in the area. Also a great
549	4/9/21	Inexpensive apartment in	<b>The space</b><br />Enjoy my lovely, sunny a	doughnut shop. See the "guidebook" for more info.<br /><br />Êàë‰ΩèÂú®Â∏ÉÈ≤ÅÂÖãÊûó(Brooklyn)‰∏Ä‰∏™ÂÆâÂÖ®ÁöÑË°óÂå∫ÔΩ°ÊàëÊúâ‰∏§‰∏™ÊàøÈó¥,‰∏ÄÂ§ß‰∏ÄÂ∞èÔΩ°ÊàëÁöÑÂÖ¨ÂØìÈôÑËøëÊúâ‰∏Ä‰ƒõÈÖíÂêßÂíåÈ§êÂéÖ,Á¶ªÂ®ÅÂªâÊñØÂ†°(Williamsburg)ÂíåÂä≥Âæ∑‰ª£Â∞îÂ†°(Fort Green)ÊØîËæÉËøëÔΩ°ÂéªÊõºÂìàÈ°ø‰∏ãÂüéÂåƒ(Lower Manhattan)ÁöÑËØùÂçäÂ∞èÊóðÂ∞±Âà∞‰ƒÜÔΩ°

*5. Devise an initial plan that outlines how you intend to clean the dataset in Phase-II. A typical plan for the overall project will include the following steps: S1: description of dataset **D** and matching use case **U1**; S2: profiling of **D** to identify the quality problems **P** that need to be addressed to support **U1**; S3: performing the data cleaning process using one or more tools to address the problems **P** (here you should describe which tools you are planning to use, e.g., OpenRefine; Python; etc.) S4: checking that your new dataset **D0** is an improved version of **D**, e.g., by documenting that certain problems P are now absent and that **U1** is now supported; S5: documenting the types and amount of changes that have been executed on **D** to obtain **D0**. You should also include a tentative assignments of tasks to team members (who does what).*

Initial Plan:
S1 - [See Questions 1 and 2]

S2 - [See Question 4] - Will be addressed further in Phase II.

S3 - We are planning to primarily use Python to clean the dataset — in particular, we will be manipulating Pandas DataFrames using various `filter`, `loc`, `iloc`, and `apply` operations that perform all the necessary steps efficiently. To filter out unnecessary rows of data as per the requirements of our **U1** use case, `loc` and `iloc` operations will allow us to select all rows with entries corresponding to specific ranges of values in certain columns. Meanwhile, the `apply` operations (used in conjunction with `lambda` functions) will allow us to manipulate the data, allowing us to perform functions such as consolidating host names as well as performing the NLP-related operations on columns containing text data.

S4 - We can check if the cleaned dataset will be better based on if it will be empirically easier to find meaningful data that corresponds to the users' needs. Not only should unreasonable data points be eliminated, but because of simplistic data filtering on various features, in tandem with more complex NLP pipelines, it will mean that data points that are not of interest to the end user should not be preset. As such, by visually inspecting data in the resultant dataset **D0** we could ascertain that **D0** is an improved version of the original dataset **D**. We will be utilizing SQL Integrity Constraints at the end of our procedures to confirm that our cleaned dataset matches the conditions specified in use case U1. We will run the SQL commands prior to data cleaning as well so the change can be visibly understood.

S5 - We will be using a Jupyter Notebook (hosted using Google Colab) to create an executable playground containing all the Python code used to clean the original dataset. This will include a history of all operations necessary to manipulate and clean the data. We will also make sure to include descriptive output statements and comments in the code to inform the user of all of the important libraries, operations, and functionalities involved throughout the step-by-step process of creating the new, improved dataset **D0** from the original dataset **D**. Furthermore, we plan to use the library `pyflowchart` to create the workflow model of our data pipeline. The flowchart and the Jupyter Notebook will document changes made to the original dataset after cleaning. The flowchart and the Jupyter Notebook will document changes made to the original dataset after cleaning by adding a statement to print how many rows are removed with each step. We will add a summary table to our Phase 2 report that clearly delineates all the changes made.

Tentative Assignments:

1. Filter by best price, number of nights, number of people, etc. - *Anish*
2. Develop an NLP pipeline to filter out restrictive hosts (smoking, events, etc.) - *Hung*
3. Consolidate by host name, filter out foreign and poorly reviewed hosts - *Sharanya*