

IS 537 Final Project Report

Google Play Store Dataset

1. Overview and Initial Assessment of Dataset

Objective

The aim is to work on Google Play Store Dataset perform data cleaning tasks to prepare for exploratory Data Analysis. Secondly, familiarize us with data cleaning tools like OpenRefine and explore alternative options like SQL and Python for the same. We will then work on data visualizations using tools like Tableau or PowerBI.

Significance

Significance of the project is to explore different tools for data cleaning and have a clean, structured, and well formatted file for further analysis. We then use the clean data file to find some insights from the data using visualization. E.g.: Finding the average rating of applications based on the category column from our data set. Our data cleaning project would be useful for organizations which have their applications listed on the Google Play Store. They can look at the reach and performance of their competitors in the respective categories based on metrics like total downloads, average rating, number of reviews and application size.

Similarly, someone planning to launch an application can look at potential competitors in the market. The main purpose would be to help users who want to download an Android application for a certain task like travel booking or video calling. They can look at alternatives based on review, rating and price and make an informed decision.

Technology

Python, OpenRefine for Data Cleaning

SQLite for Data Profiling and Integrity Constraints

Tableau for Data Analysis (Visualizations)

Draw.io, Creately for Workflow

MS Excel for basic dataset assessment of messy dataset

Link to the box folder – <https://uofi.box.com/s/rdxsrs7b7l5ywwquvihen1rpi8sumcv>

- Source messy dataset - googleplaystore.csv
- Cleaned dataset – clean.csv

IS 537 Final Project Deliverables Zip File Contains – *Data Cleaning Project -*

googleplaystore.csv - python code.IPNYB (Python Code), Queries.txt, cleaning.json file (History file retrieved from OpenRefine).

a. Structure and Content of Dataset

Our dataset can be found on Kaggle. Please use the link to access the same:

<https://www.kaggle.com/lava18/google-play-store-apps> The dataset contains details about the Android applications available on Google Play Store. The following are the attributes/ fields of the dataset – App, Category, Rating, Reviews, Size, Installs, Type, Price, Content Rating, Genres, Last Updated, Current Version and Android Version.

There are 13 columns and about 10,841 rows. Our dataset is a comma separated values (.CSV) file type.

b. Data Quality Issues

- Inconsistent data - for the data values in the current version column, the data values are very different and inconsistent.
- Incorrect data - for one record we can confirm that the data was incorrect when we checked the details for an application on google play store
- Redundancy - there are duplicate values of the same application in the dataset, creating confusion
- Inconsistent formats - some of the columns have both numeric and text data in the dataset.

c. Use Case(s):

Our use case is to create an interactive dashboard with customizable filters for the end users, based on various columns, after we are finished with the data cleaning section. Any user (e.g., business organization) can just make use of our dashboard to derive insights instead of being limited to just reading through the original data.

- **Already Good Enough Use Cases**
 - Top applications based on highest ratings received basis the category of the application, price, type, and genre of the application.
 - Top applications based on highest reviews received basis the category of the application, price, type, and genre of the application.
 - Top Play Store applications in particular year.
- **Never Good Enough Use Cases**
 - Which application current versions have had the most users?
 - How to compare the applications based on the current version.
- **Middle of the Road Use Cases**
 - Top n genres based on count of the applications
 - Top n categories based on the average user ratings.
 - Top n applications downloaded based on the count of the reviews received.
 - Top n download categories

d. Data Cleaning Goals

- Formatting the data values across columns for standardization
- Renaming any field names for better audience readability and understanding such as App and Installs
- Formulating process to handle NaN values in the dataset
- Applying advanced sorting functions to get top applications on our dataset
- Rectifying any incorrect data types of the columns such as for Reviews
- Perform data value clustering on column Type
- Handling blank and incorrect values

2. Data Cleaning with OpenRefine

SNO	Task	Cells/ records affected
1.	Trim function on Application column to eliminate white spaces	3
2.	Trim function on Category column to eliminate white spaces	0
3.	Trim function on Type column to eliminate white spaces	0
4.	Trim function on Content Rating column to eliminate white spaces	0
5.	Trim function on Genre column to eliminate white spaces	0
6.	toNumber function on Rating column to convert all rows into number	9639
7.	toNumber function on Review column to convert all rows into number	9639
8.	toDate function on Last Updated column to convert all rows into date format	9639
9.	Replace '+' sign with '' using value.replace function on installs column	9638

10.	Replace '\$' sign with '' using value.replace function on price column	753
11.	Replace '_' sign with '' using value.replace function on Application column	2
12.	Using clustering on Genre columns to reduce total Genres in the dataset	780
13.	Clustering was iteratively repeated on the Genre column for 9 times with new groupings	5135

- **Important Cleaning Steps:**
 - Standardizing column data types
 - Using Columnar transformations
 - Using RegEx for Pattern matching/replacement
 - Clustering options in Facets to group similar cells
- **Pros:**
 - Ideal for preliminary analysis of data
 - Easy to learn and interpret cleaning options
- **Cons:**
 - Not suitable for performing complex cleaning functions
 - Regular Expressions knowledge is needed

Cleaners United: Shubhangi Singhal (ss100), Niken Shah (niken2)

Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated
4.4	5234162	Varies with device	100000000	Free	0	Everyone	Action	2017-11-14T00:00:00Z
4.3	8118609	62M	500000000	Free	0	Everyone	Action	2018-07-05T00:00:00Z
4.2	1497361	33M	100000000	Free	0	Everyone	Action	2018-04-09T00:00:00Z
4.3	10306	50M	1000000	Free	0	Mature 17+	Action	2018-08-01T00:00:00Z
4.3	1000417	Varies with device	500000000	Free	0	Teen	Action	2018-07-05T00:00:00Z
4.7	990491	75M	10000000	Free	0	Everyone	Action	2018-05-24T00:00:00Z
4.6	7671249	Varies with device	100000000	Free	0	Mature 17+	Action	2018-08-02T00:00:00Z
4.5	183846	99M	10000000	Free	0	Everyone	Action	2018-07-30T00:00:00Z
4.5	5465624	53M	100000000	Free	0	Teen	Action	2018-08-03T00:00:00Z
4.7	1534466	Varies with device	50000000	Free	0	Teen	Action	2018-07-23T00:00:00Z

In the above screenshot, you can see how the rating, reviews columns have become green after they have been converted to number format. Similarly, the last updated columns too have become green after it was converted into date. For the installs column, you can see that the '+' has been dropped for all the records.

Size	Installs	Type	Price
46M	100000	Paid	6.99
64M	1000	Paid	1.99
29M	10000000	Paid	0.99
34M	10000	Paid	4.99
25M	50000	Paid	0.99
96M	1000000	Paid	0.99
92M	500000	Paid	1.99
65M	1000	Paid	0.99
50M	1000000	Paid	2.99
99M	100000	Paid	2.99

In the above screenshot, you can see that the '\$' sign has been dropped for all the paid applications, so that the column can be used as a numeric column during the visualization process.

After applying clustering, the number of Genre types were limited to 50 as seen below. The number on the right displays the number of records present in each of these Genres after the final clustering process.

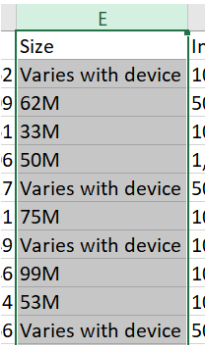
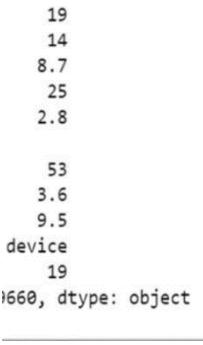
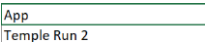
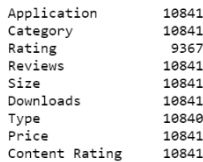
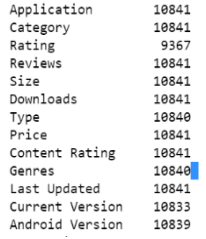

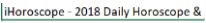
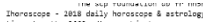
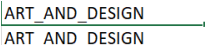

Action	299
Action & Adventure	62
Adventure	80
Arcade	198
Art & Design	65
Auto & Vehicles	85
Beauty	53
Board	56
Books & Reference	225
Brain Games	29
Business	419
Card	47
Casino	39
Casual	165
Comics	56
Communication	316
Creativity	8
Dating	171
Education	688
Entertainment	588
Events	64
Finance	342
Food & Drink	111
Health & Fitness	289
House & Home	74
Libraries & Demo	84
Lifestyle	369

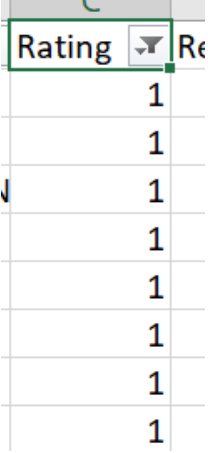
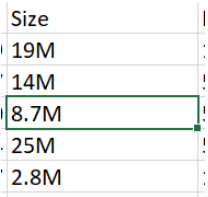
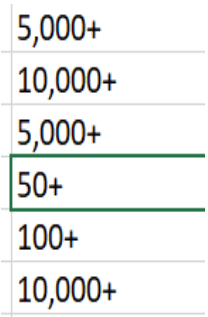
Maps & Navigation	131
Medical	394
Music	24
News & Magazines	254
Parenting	60
Personalization	375
Photography	279
Pretend Play	27
Productivity	374
Puzzle	119
Racing	91
Role Playing	115
Shopping	202
Simulation	199
Social	239
Sports	331
Strategy	96
Tools	823
Travel & Local	219
Trivia	38
Video Players & Editors	165
Weather	79
Word	23

3. Data Cleaning with Python

We intend to perform data cleaning in python along with OpeRefine to meet data cleaning goals additionally which were not covered by OpenRefine. We identified the data cleaning steps using the dataset assessment in MS Excel and tried to resolve quality issues step by step using each column in the dataset. You will see first we have approached the duplicate values of the application column and then gradually move towards the other columns to change their formatting, data type so that they can be used for further analysis in Tableau.

You may find a summary of the data cleaning steps we implemented in python along with the before and after scenario and the number of records affected by them.

SNO	Task	Before	After	Records affected
1.	Formatting the data values across columns for standardization - <i>Size</i> text values to float except for “Varies with the device”.			8413
2.	Renaming any field names for better audience readability and understanding such as <i>App</i> , <i>Installs</i> , <i>Android Ver</i> and <i>Current Ver</i> .			4
3.	Remove duplicate <i>Application</i> records.			1180
4.	Capitalize first letter of data values in column <i>Application</i> .			
5.	Trim leading and trailing spaces off qualitative columns.	“ Paint Splash!”	“Paint Splash!”	
6.	Removing underscores from column <i>Category</i> .			1875

7.	Converted <i>Rating</i> from int to float.		<pre> 0 4.1 1 3.9 2 4.7 3 4.5 4 4.3 ... 10836 4.5 10837 5.0 10838 NaN 10839 4.5 10840 4.5 Name: Rating, Length: 10841, dtype: float64 Wall time: 9.35 ms </pre>	8202
8.	Converted KB to MB in <i>Size</i> and saved as Float.		<pre> 0 19 1 14 2 8.7 3 25 4 2.8 ... 10836 53 10837 3.6 10838 9.5 10839 Varies with device 10840 19 Name: Size, Length: 9660, dtype: float64 </pre>	316
9.	Dropped “+” and “,” from <i>Downloads</i> and converted to Integer.		<pre> 10000 500000 5000000 50000000 100000 ... 5000 100 1000 1000 10000000 Downloads, Length: 9660, dtype: object </pre>	9659

10.	Replaced 0 with “Free” in the “Type” column.	0	Free Free Free Free Free ... Free Free Free Free Free Type, Length : 7 80 1	1
11.	Drop “\$” from the “Price” column.	\$2.99 0 0 0	Price 0 E 0 E 0 E 0 T	800
12.	Formulating process to handle NaN values in the dataset e.g., updated to “Unrated” in <i>Content Rating</i> .	0 4.1 1 3.9 2 4.7 3 4.5 4 4.3 ... 10836 4.5 10837 5.0 10838 NaN 10839 4.5 10840 4.5		1463
13.	Handling blank and incorrect values.			1

Use Cases in Python

- Already good enough use cases
 - Top applications based on highest ratings received basis the category of the application, price, type, and genre of the application.

- We did not clean the ratings column as it could have been used as it is with a mix of integer and float values.
 - We do not need to clean the price, type, or genre for this use case because we can obtain good enough results if we avoid the NaN values in our final data visualization or analysis.
- Top applications based on highest reviews received basis the category of the application, price, type, and genre of the application.
- We do not need to clean the reviews column as it could have been used directly.
- Top play Store applications in particular year.
 - We do not need to clean the year column as it is already standardized.
- **Never good enough use cases -**
 - Which application current versions have had the most users?
 - How to compare the applications based on the current version
 - For both these use cases, due to the very high number of different values in the set of Current Version, it is not feasible to clean this column.
- **Middle of the road use cases -**
 - Top n genres based on count of the applications
 - We can clean the genres column based on clustering concepts in OpenRefine to find out the count of applications in each category.
 - Top n categories based on the average user ratings.
 - We could clean the ratings column to convert all values to float for us to get an accurate average user rating value per category instead of an approximate value.
 - Top n applications downloaded based on the count of the reviews received.
 - We cleaned the downloads column to remove the plus and the comma characters to get the core integer values that could be used for data analysis.
 - Top n download categories
 - We cleaned the downloads column to remove the plus and the comma characters to get the core integer values that could be used for data analysis.

Hence, cleaning of some of the above-mentioned columns help with the middle of the road use cases.

- **Utilized**
 - Pandas and Math library
 - Dataframe Utility functions e.g., head, columns, count, drop_duplicates etc.
 - String utility functions e.g., str.strip, str typecasting, and len etc.
















- List and array data structure to implement iterative data cleaning algorithms
- Time taken for code execution - 1.327 seconds
- **Pros**
 - Lesser processing time
 - Bigger user community as compared to OpenRefine
 - Presence of data cleaning libraries e.g., Pandas and NumPy
 - Code Readability
- **Cons**
 - Absence of drag and drop user interface
 - Absence of some data cleaning menus and features present in OpenRefine
 - Specialty is not data cleaning as it is a programming language

4. Developing a relational schema

We used <https://sqliteonline.com/> - an online SQLite3 server for loading our dataset.

a. SQL code for creating relational schema:

```
DROP TABLE IF EXISTS `clean`; CREATE TABLE `clean` (`ID` mediumint(9), `Application` varchar(194) DEFAULT NULL, `Category` varchar(19) DEFAULT NULL, `Rating` varchar(3) DEFAULT NULL, `Reviews` int(11) DEFAULT NULL, `Size` varchar(18) DEFAULT NULL, `Downloads` bigint(20) DEFAULT NULL, `Type` varchar(4) DEFAULT NULL, `Price` varchar(8) DEFAULT NULL, `Content Rating` varchar(15) DEFAULT NULL, `Genres` varchar(37) DEFAULT NULL, `Last Updated` date(8) DEFAULT NULL, `Current Version` varchar(50) DEFAULT NULL, `Android Version` varchar(18) DEFAULT NULL);
```

Table	
 clean	▼
Column	
 ID	mediumint(9)
 Application	varchar...
 Category	varchar(19)
 Rating	varchar(3)
 Reviews	int(11)
 Size	varchar(18)
 Downloads	bigint(20)
 Type	varchar(4)
 Price	varchar(8)
 Content Rating	var...
 Genres	varchar(37)
 Last Updated	date(8)
 Current Version	va...
 Android Version	v...

b. Sample Insert Query:

```
INSERT INTO `clean`
```

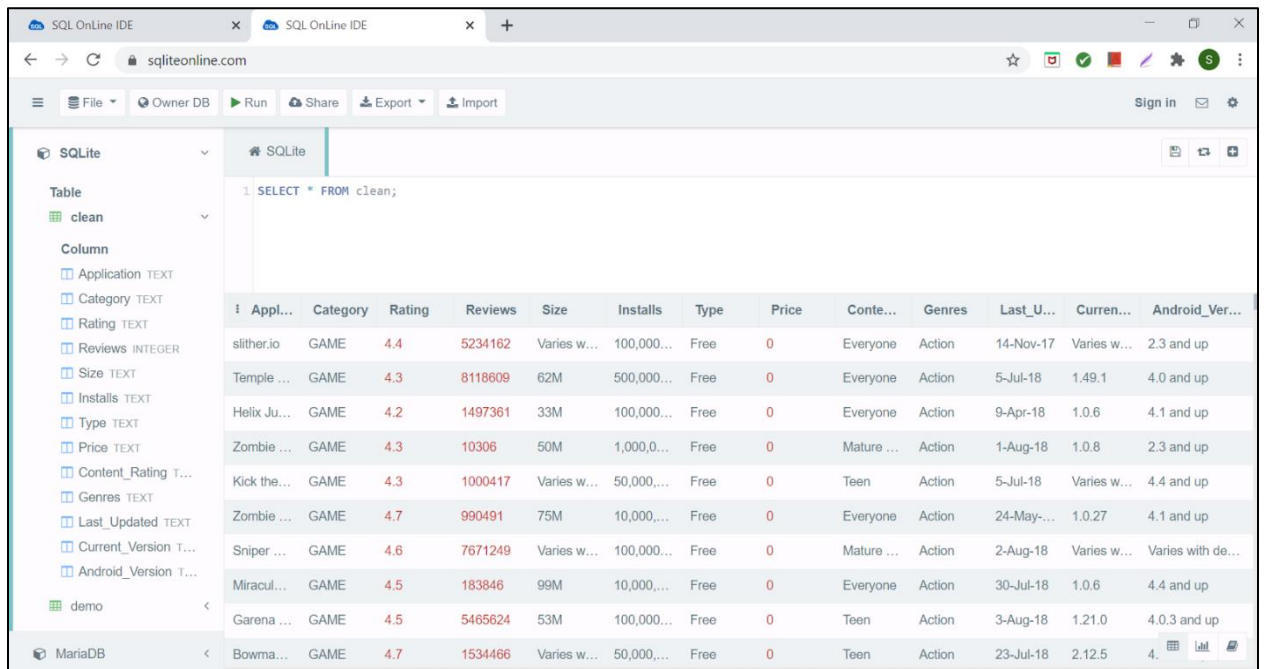
```
(`ID`,`Application`,`Category`,`Rating`,`Reviews`,`Size`,`Downloads`,`Type`,`Price`,`Content Rating`,`Genres`,`Last Updated`,`Current Version`,`Android Version`) VALUES ('0','Photo editor & candy camera & grid & scrapbook','ART_AND_DESIGN','4.1','159','19','10000','Free','0','Everyone','Art & Design','2018-01-07','1.0.0','4.0.3 and up');
```

Cleaners United: Shubhangi Singhal (ss100), Niken Shah (niken2)

```
SQLite
1 INSERT INTO `clean` (`ID`,`Application`,`Category`,`Rating`,`Reviews`,`Size`,`Downloads`,`Type`,`Price`,`Content Rating`,`C
2 INSERT INTO `clean` (`ID`,`Application`,`Category`,`Rating`,`Reviews`,`Size`,`Downloads`,`Type`,`Price`,`Content Rating`,`C
3 INSERT INTO `clean` (`ID`,`Application`,`Category`,`Rating`,`Reviews`,`Size`,`Downloads`,`Type`,`Price`,`Content Rating`,`C
4 INSERT INTO `clean` (`ID`,`Application`,`Category`,`Rating`,`Reviews`,`Size`,`Downloads`,`Type`,`Price`,`Content Rating`,`C
5 INSERT INTO `clean` (`ID`,`Application`,`Category`,`Rating`,`Reviews`,`Size`,`Downloads`,`Type`,`Price`,`Content Rating`,`C
6
```

c. Data Profiling (SQL)

Using Import Dataset Function



The screenshot shows the SQL Online IDE interface. On the left, a sidebar lists the database 'SQLite' and a table 'clean'. The table's columns are: Application (TEXT), Category (TEXT), Rating (TEXT), Reviews (INTEGER), Size (TEXT), Installs (TEXT), Type (TEXT), Price (TEXT), Content_Rating (TEXT), Genres (TEXT), Last_Updated (TEXT), Current_Version (TEXT), and Android_Version (TEXT). The main editor shows a SQL query: `SELECT * FROM clean;`. Below the query, a table of application data is displayed.

Appl...	Category	Rating	Reviews	Size	Installs	Type	Price	Conte...	Genres	Last_U...	Curren...	Android_Ver...
slither.io	GAME	4.4	5234162	Varies w...	100,000...	Free	0	Everyone	Action	14-Nov-17	Varies w...	2.3 and up
Temple ...	GAME	4.3	8118609	62M	500,000...	Free	0	Everyone	Action	5-Jul-18	1.49.1	4.0 and up
Helix Ju...	GAME	4.2	1497361	33M	100,000...	Free	0	Everyone	Action	9-Apr-18	1.0.6	4.1 and up
Zombie ...	GAME	4.3	10306	50M	1,000,0...	Free	0	Mature ...	Action	1-Aug-18	1.0.8	2.3 and up
Kick the...	GAME	4.3	1000417	Varies w...	50,000...	Free	0	Teen	Action	5-Jul-18	Varies w...	4.4 and up
Zombie ...	GAME	4.7	990491	75M	10,000...	Free	0	Everyone	Action	24-May-...	1.0.27	4.1 and up
Sniper ...	GAME	4.6	7671249	Varies w...	100,000...	Free	0	Mature ...	Action	2-Aug-18	Varies w...	Varies with de...
Miracul...	GAME	4.5	183846	99M	10,000...	Free	0	Everyone	Action	30-Jul-18	1.0.6	4.4 and up
Garena ...	GAME	4.5	5465624	53M	100,000...	Free	0	Teen	Action	3-Aug-18	1.21.0	4.0.3 and up
Bowma...	GAME	4.7	1534466	Varies w...	50,000...	Free	0	Teen	Action	23-Jul-18	2.12.5	4.0 and up

To find applications which lie in the game category and have ratings greater than 4 for kids and video game enthusiasts

SQLite							
1 SELECT * FROM clean WHERE category = "GAME" AND rating >4.0;							
ID	Application	Category	Rating	Revie...	Size	Do	
1	Out of Bounds BH De...	GAME	NaN	159	19	100	
1	Out of Bounds BH De...	GAME	NaN	159	19	100	
2	Animal Hunting: Snipe...	GAME	NaN	0	48	100	
3	Super ball DZ	GAME	NaN	0	48	100	
4	FK (FlightKid)	GAME	NaN	0	48	100	

To find applications which are free on google play store

1 SELECT * FROM clean WHERE type = "Free";									
ID	Application	Category	Rating	Revie...	Size	Downl...	Type	Price	
0	Photo editor & candy ...	ART_AND_...	4.1	159	19	10000	Free	0	
1	Out of Bounds BH De...	GAME	NaN	159	19	10000	Free	0	
1	Out of Bounds BH De...	GAME	NaN	159	19	10000	Free	0	
2	Animal Hunting: Snipe...	GAME	NaN	0	48	10000	Free	0	

Find applications with size greater than 10 MB

SQLite						
1 SELECT * FROM clean WHERE size>10;						
ID	Application	Category	Rating	Revie...	Size	
0	Photo editor & candy ...	ART_AND_...	4.1	159	19	
1	Out of Bounds BH De...	GAME	NaN	159	19	
1	Out of Bounds BH De...	GAME	NaN	159	19	
2	Animal Hunting: Snipe...	GAME	NaN	0	48	
3	Super ball DZ	GAME	NaN	0	48	

Where application names start with letter S

SQLite											
1	SELECT * FROM clean WHERE application LIKE 's%';										
	ID	Application	Category	Rating	Revie...	Size	Downl...	Type			
3		Super ball DZ	GAME	NaN	0	48	10000	Free			

Count of Ratings with NaN values

1	SELECT COUNT(*) FROM clean WHERE rating = "NaN";										
	COUNT(*)										
	1458										

Finding count of empty cells in the dataset

1	SELECT COUNT(*) FROM clean WHERE rating = "" OR content_rating = "" OR genres = "" OR current_version = "" OR android_version = "";										
	COUNT(*)										
	1										

d. Integrity Check Constraints and Denial Queries (SQL)

Rating should be between 0 and 5

🏠 SQLite												
1 SELECT * FROM clean WHERE Rating NOT BETWEEN 0 AND 5;												
⌵	ID	Appli...	Categ...	Rating	Revie...	Size	Downl...	Type	Price	Conte...	Genres	Last ...
1		Out of ...	GAME	NaN	159	19	10000	Free	0	Everyone	Art & D...	2018-0...
1		Out of ...	GAME	NaN	159	19	10000	Free	0	Everyone	Art & D...	2018-0...
2		Animal ...	GAME	NaN	0	48	10000	Free	0	Everyone	Art & D...	2018-0...
3		Super ...	GAME	NaN	0	48	10000	Free	0	Everyone	Art & D...	2018-0...
4		FK (Fli...	GAME	NaN	0	48	10000	Free	0	Everyone	Art & D...	2018-0...
5		DM Ad...	GAME	NaN	0	48	10000	Free	0	Everyone	Art & D...	2018-0...

Primary Key ID should uniquely determine all other attributes:

SQLite										
<pre> 1 SELECT a.* FROM clean a 2 JOIN (SELECT *, COUNT(*)) 3 FROM clean b 4 GROUP BY ID 5 HAVING COUNT(*) > 1) b 6 ON a.ID = b.ID ORDER BY a.ID; 7 </pre>										
ID	Appli...	Categ...	Rating	Revie...	Size	Downl...	Type	Price	Conte...	Genres
1	Out of ...	GAME	NaN	159	19	10000	Free	0	Everyone	Art & D...
1	Out of ...	GAME	NaN	159	19	10000	Free	0	Everyone	Art & D...

Price should be non-negative:

SQLite										
<pre> 1 SELECT * FROM clean WHERE Price < 0; </pre>										

Count of reviews should be greater than or equal to the count of downloads:

SQLite										
<pre> 1 SELECT * FROM clean WHERE Reviews > Downloads; </pre>										
ID	Applic...	Categ...	Rating	Reviews	Size	Downl...	Type	Price		
6	FK (Flig...	GAME	NaN	10	48	1	Free	0		
7	FK (Flig...	GAME	NaN	100	48	10	Free	0		
8	FK (Flig...	GAME	NaN	10000	48	10	Free	0		
9	FK (Flig...	GAME	NaN	100	48	1	Free	0		

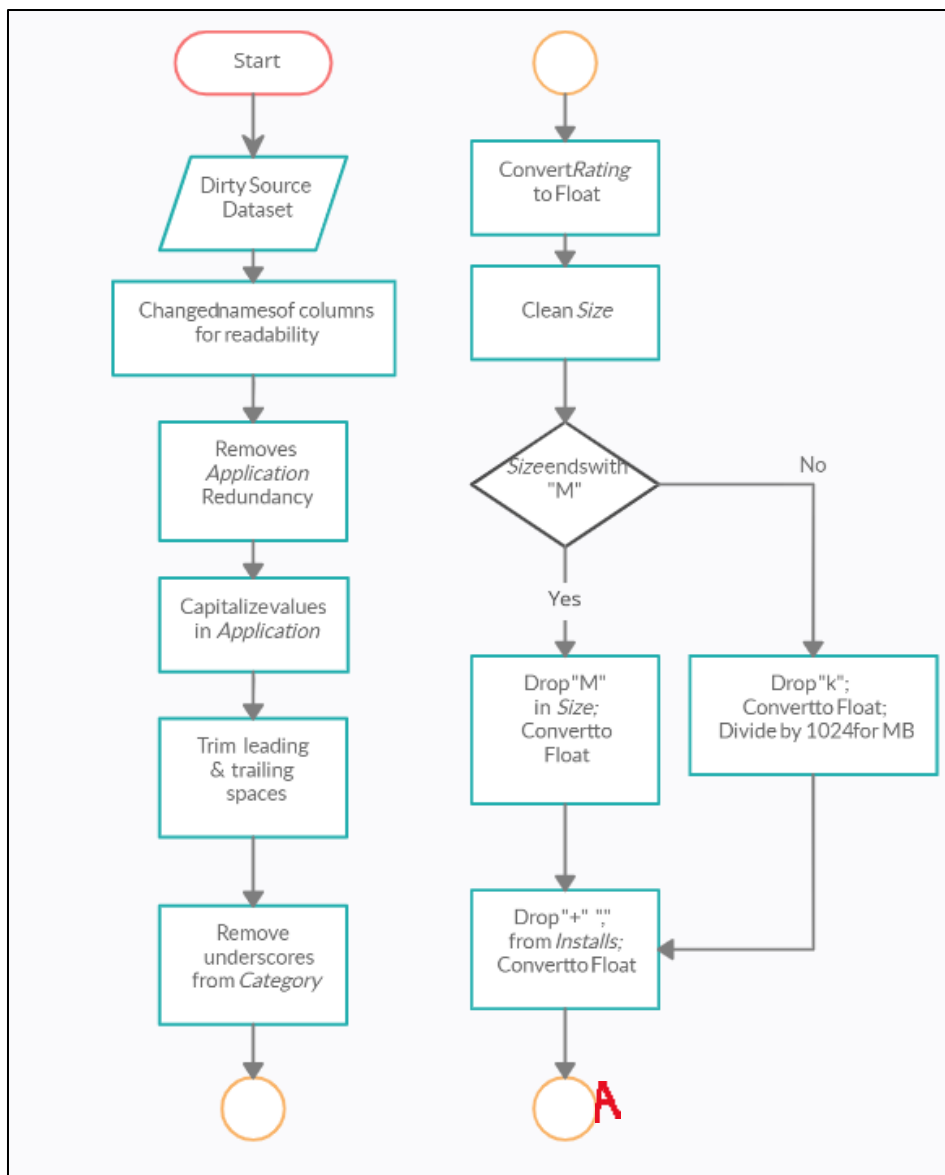
Applications which are free must have the corresponding price as 0:

```
1 SELECT * FROM clean WHERE Type = 'Free' AND Price > 0;
```

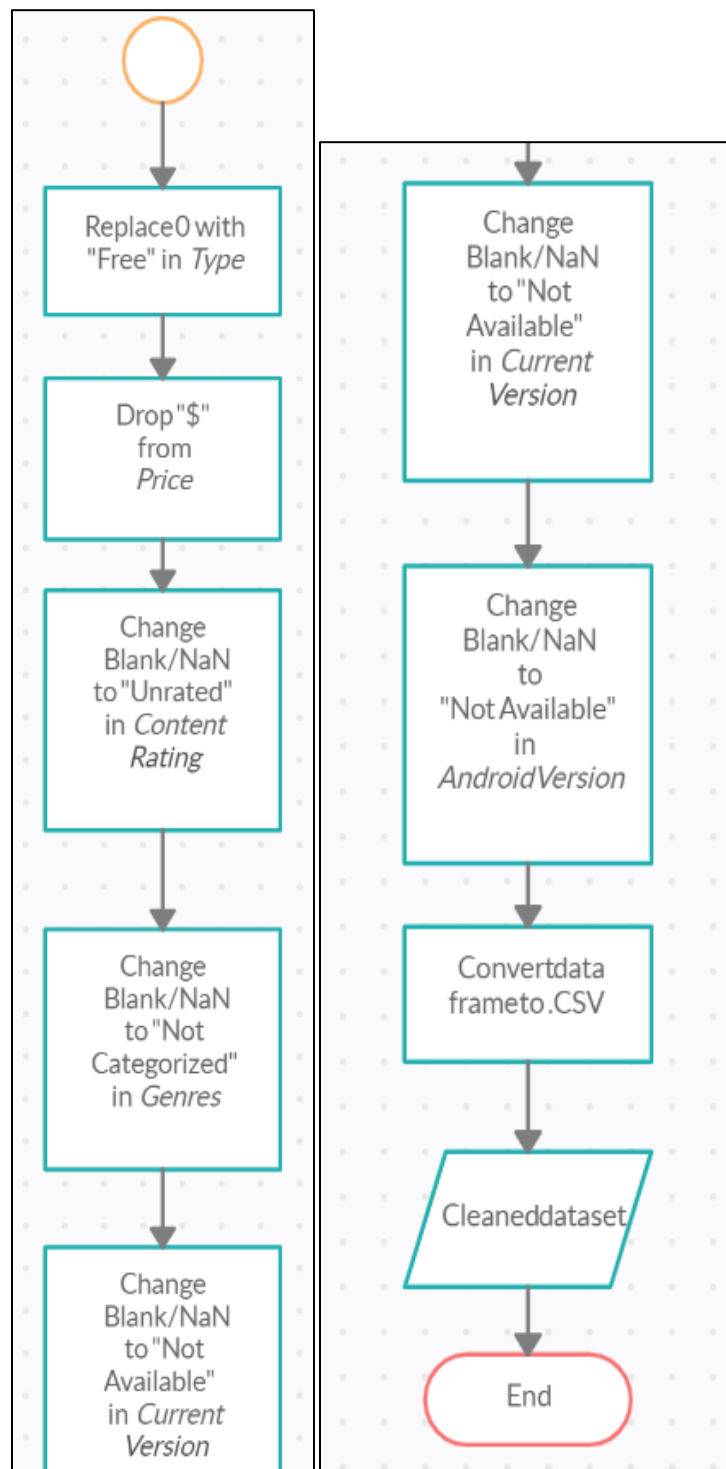
- Repairing the dataset based on IC check-
 - For the first integrity constraint, we will be required to either remove records with NaN values or replace bad records with an average rating of the same category of applications or only allow the values between 0 and 5 on entering the records in the table.
 - For the 2nd integrity constraint, we can set the primary key constraint on creating the table.
 - For the 3rd integrity constraint, we can again either replace nonnegative values with blanks, or drop the negatives and keep the magnitude value or we can keep an IC check on inserting the records.
 - For 4th integrity constraint we must only allow records which have reviews > count of downloads on insertion.
 - For the 5th integrity constraint, we can repair the records by replacing the price as 0 for all the places with type as “free” after insertion of records using cleaning steps like we mentioned in python.

5. Creating Workflow Model

- **Data cleaning workflow Python – This workflow has been created using www.creately.com**



In continuation from the point A

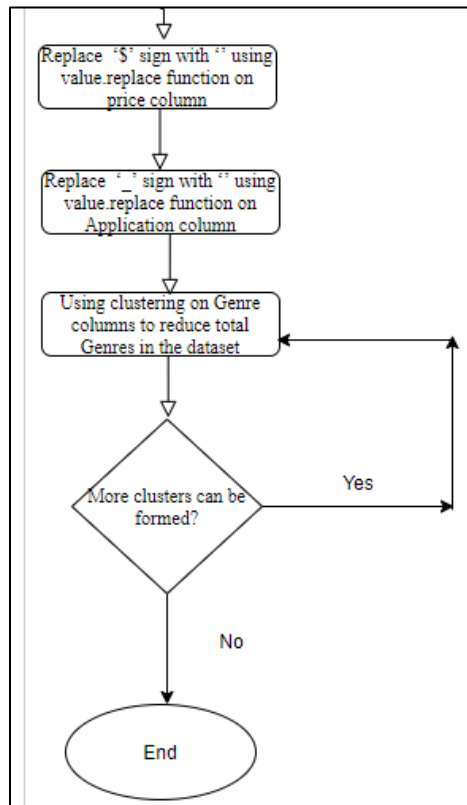
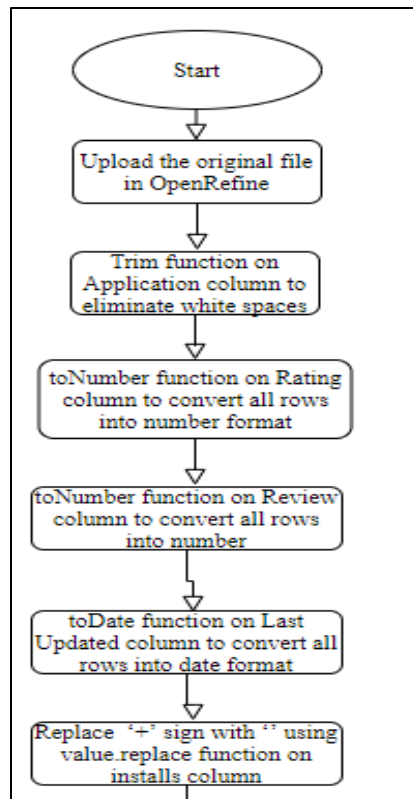


We start with a messy dataset. We first changed the names of some of the columns in the dataset for better user readability. Then, we removed duplicated records by removing duplicate application names from the dataset. We also capitalized the first letter of the application's names for formatting the data. We trimmed the leading and trailing spaces off the qualitative columns, so that in case we must apply any text analytics algorithms in the future, this would be helpful. We then removed the underscores from the Category column so that the values look like the other text values in other columns. We then converted the Rating data values, wherever they were text or int to float. We then cleaned the size column where we dropped the MB and kB from the values, converted the remaining text to float values. in case of kB, we converted the float values to MB by dividing them by 1024 We also removed "+", ",", symbols from the installs/ downloads column for us to convert them to a long integer data type so that those values can be used in our data visualizations and analysis.

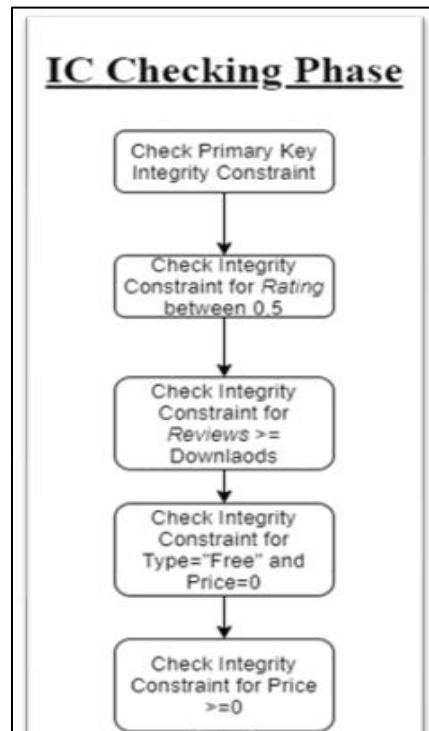
Post this, we replaced any data value "0" with "Free" to remove redundancy in the column. There was only one record that we could change through this. We also dropped the "\$" symbol from the Price column to convert it to float instead of a currency data type in MS Excel. We finally cleaned the Blank/ NaN values from the content rating, genres, android, and current version columns where we changed those values to "Unrated", "Not Categorized" and "Not Available" correspondingly to obtain a cleaner play store applications dataset.

- **Data cleaning workflow OpenRefine** – This workflow has been created using draw.io

We first upload the original file in OpenRefine, then used trim functions in the application column to eliminate white spaces. Then we planned to use toNumber function on Rating column to convert all rows into number format. We use the same function on reviews column and then use toDate function on last updated column to change values to date data type. We then use value.replace function to replace "+" with "" in installs/ downloads column. We then replaced the dollar sign with "" in the price column using the same function and then replaced "_" as well. In the end we used clustering function in the genres column as you can see below.



- **Integrity Constraint Check workflow** - created in www.draw.io

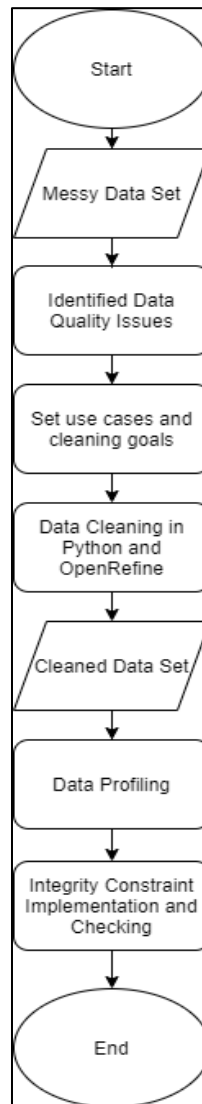


For the IC check workflow, we first check the Primary key integrity constraint from the dataset which will tell us which records are having the same unique identifier. This way we may be able to remove any duplicate records. Then we check for the rating column, whether its values are between 0 and 5. In the original dataset all the rating values were between the 0 and 5 so when we ran the IC check, zero columns were fetched. We then inserted some sample records with rating values which were NaN to check if the constraint was working or not. We were able to fetch multiple such records.

We also implemented an IC for the review count to be greater than or equal to the downloads count. Again, in the original dataset when we did the profiling, we found there were no records where the reviews were lesser than the downloads, so just to check the effectiveness of the constraint we inserted sample records with reviews less than or equal to downloads and were able to fetch those records on running the SQL query successfully. We then tried to implement and check the IC for type = "free" then price = 0. In this we only used the original dataset for the

checking purpose and found that no such records existed. This integrity check is effective only after insertion of records and not at the time of inserting the records. We also checked for the IC price to be non-negative. Again, in the original dataset no such records existed.

Overall Data Cleaning Workflow - created in www.draw.io

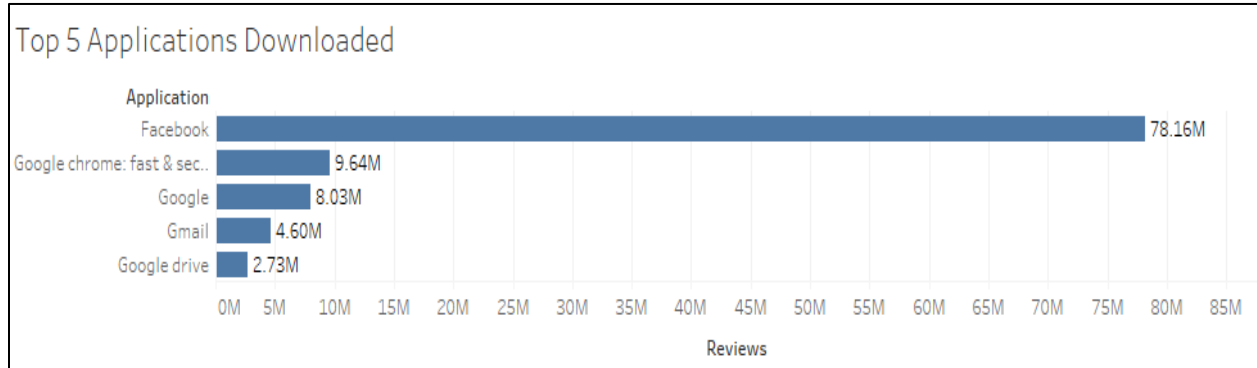


We followed the above approach for the overall data cleaning process. We first took the messy data and did dataset assessment manually in MS Excel, then we figured out the data quality issues and set some data cleaning goals as per those. Going forward, we performed data cleaning in Python and OpenRefine as mentioned in detail earlier. We then on the cleaner dataset created a relational schema in SQLite and performed data profiling to understand the structure and content

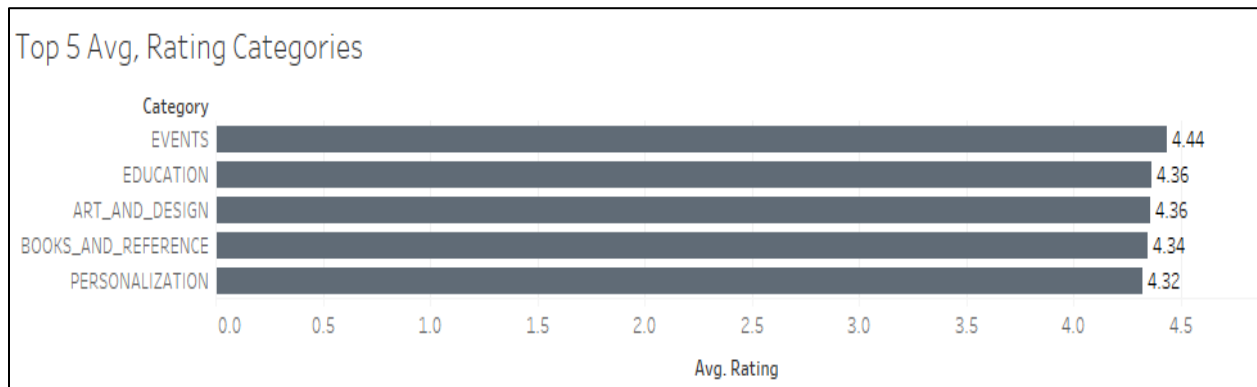
of the cleaner dataset. After that we implemented integrity constraints on that dataset and checked those.

6. Data Analysis

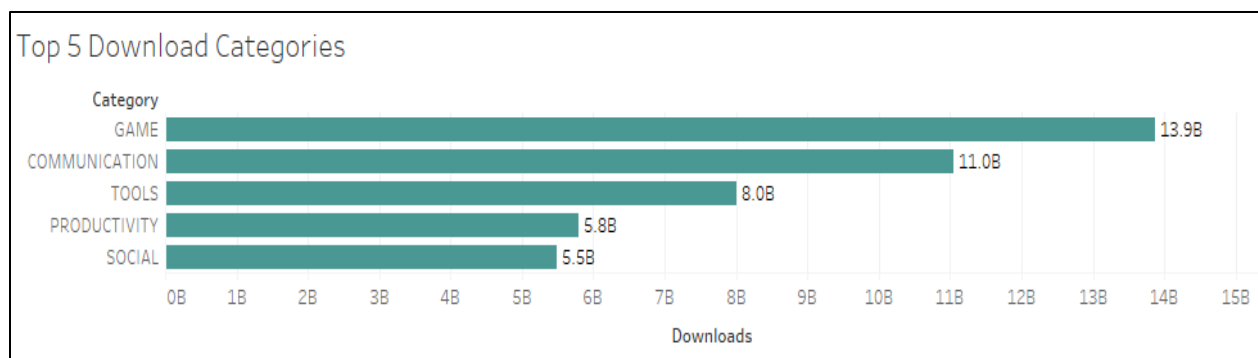
The first visualization displays top 5 applications with the most downloads on the Google Play Store. 4 out of the top 5 applications are owned by Google. However, the application with the highest number of downloads is not from Google but is Facebook.



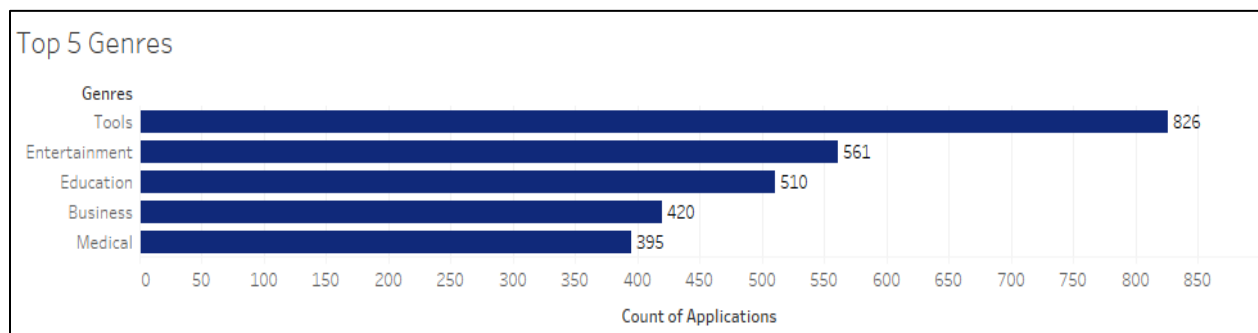
When it comes to Categories; Events, Education and Art & Design have the highest average rating. Hence, if a new company is planning to launch an application, it can easily identify which categories are saturated with good quality applications.



For Downloading Categories, companies would be able to identify which type of applications are very popular in the market. Below you can see that gaming is the most popular category on google play store, followed by communication and tools.



The Top 5 Genres with the greatest number of applications will help companies identify those Genres which are already saturated with many applications and allow them to explore new genres. In the below visualization, we can see that tools, entertainment and education have over 500 applications on the Google Play Store.



7. Contribution

- Shubhangi has worked on the Python cleaning and its corresponding workflow along with the quality issues, use cases, integrity check workflow, data profiling and the conclusion.
- Niken has worked on the Open Refine cleaning and workflow part along with the initial overview and dataset assessment. Niken then worked on the data analysis/visualization part.
- We both jointly worked on creating the cleaning goals, SQL schema, implementing integrity checks in that schema, formulating Queries.txt and the overall workflow.

8. Conclusion

- **Key Takeaways**
 - It is crucial to perform data profiling before starting the data cleaning process to avoid identifying new loopholes in the dataset

- The data might be collected from multiple sources and there may be discrepancies in the data.
- Data cleaning is important because the clean data eases visualizing and exploratory data analysis.
- There is always scope for data cleaning no matter how good the data looks.
- **Challenges Faced**
 - Manually updated one anomaly record in MS Excel to improve execution and space complexity.
 - Poor execution time running Genre Clustering in Python.
 - Scoping and identifying data cleaning problems in the dataset basis the use cases required research of Google Play Store Applications.

9. References

- <https://elitedatascience.com/data-cleaning>
- <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7889976>
- <https://www.dezyre.com/article/data-cleaning-in-python/406>
- <https://medium.com/@jaspreet.ida/benefits-and-advantages-of-data-cleansing-techniques-390d87e1c566>
- <https://www.ringlead.com/blog/7-common-data-quality-issues/>
- <https://www.precisely.com/blog/data-quality/data-quality-problems-errors>
- <https://www.kaggle.com/lava18/google-play-store-apps>