

# Exploratory Data Analysis

---

Once I imported the dataset into SQLite Online Studio, I began my EDA process. My goal with this process was to overview the data so I know what I should analyze in the data analysis process.



```
10 **EXPLORATORY DATA ANALYSIS**
11
12 --Check the number of unique app in both tables
13
14 SELECT COUNT(DISTINCT id) AS UniqueAppIDs
15 FROM AppleStore
16
17 SELECT COUNT(DISTINCT id) AS UniqueAppIDs
18 FROM appleStore_description_combined
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

UniqueAppIDs

7197

- I started off by checking the number of unique apps in both tables by using COUNT(DISTINCT).
  - The unique count for each table was 7,197 apps; I don't have to worry about sampling bias since the distribution will be approximately Normal.



```
20 --Check for any missing values in key fields
21
22 SELECT COUNT(*) AS MissingValues
23 FROM AppleStore
24 WHERE track_name IS null OR user_rating IS null OR prime_genre IS NULL
25
26 SELECT COUNT(*) AS MissingValues
27 FROM appleStore_description_combined
28 WHERE app_desc IS null
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

MissingValues

0

- Next, I checked for any NULL values.
  - I made sure to only retrieve NULL values by creating WHERE statements.

SQLite

```

30 --Find out the number of apps per genre
31
32 SELECT prime_genre, COUNT(*) AS NumApps
33 FROM AppleStore
34 GROUP BY prime_genre
35 ORDER BY NumApps DESC
36

```

prime_genre	NumApps
Games	3862
Entertainment	535
Education	453
Photo & Video	349
Utilities	248
Health & Fitness	180
Productivity	178
Social Networking	167
Lifestyle	144
Music	138
Shopping	122
Sports	114
...	...

- Then, I found the number of apps per genre by using COUNT(\*).
  - I created a GROUP BY statement to group all the genres into its distinct values.
- I noticed that the Games and Entertainment genres have the most apps.
  - This means that these genres are extremely competitive and saturated.

SQLite

```

38 --Get an overview of the apps' ratings
39
40 SELECT min(user_rating) AS MinRating,
41        max(user_rating) AS MaxRating,
42        avg(user_rating) AS AvgRating
43 FROM AppleStore
44

```

MinRating	MaxRating	AvgRating
0	5	3.526955675976101

- Finally, I queried for the minimum, maximum, and average app ratings.
  - This tells me that the top 50% of apps had a rating above 3.5.

# Data Analysis

After going through the EDA process, the most important information I obtained was that the top 50% of apps had a rating above 3.5. And since the app developer wants to maximize user ratings, I came up with a few questions:

1. How do paid-app ratings compare with free-app ratings?
2. Which genres have the lowest app ratings?
3. What relationship do additional features like language support and app description have with app ratings?

```
SQLite
45 **DATA ANALYSIS**
46 --Determine whether paid apps have higher ratings than free apps
47
48
49 SELECT CASE
50     WHEN price > 0 THEN 'Paid'
51     ELSE 'Free'
52     END AS App_Type,
53     avg(user_rating) AS Avg_Rating
54 FROM AppleStore
55 GROUP BY App_Type
```

App_Type	Avg_Rating
Free	3.3767258382642997
Paid	3.728948742438714

- I compared the average ratings of free and paid apps by creating a CASE statement where I assigned variables (Paid/Free) under a column called App\_Type. Then, I group the Paid/Free apps into one field with their average ratings.
- I noticed that the average rating for paid apps is higher than the average rating for free apps.
  - This may be because paying for an app leads to more engagement, thus higher user satisfaction.

```
SQLite
57 --Check if apps with more supported languages have higher ratings
58
59 SELECT CASE
60     WHEN lang_num < 10 THEN '<10 languages'
61     WHEN lang_num BETWEEN 10 AND 30 THEN '10-30 languages'
62     ELSE '>30 languages'
63     END AS language_bucket,
64     avg(user_rating) AS Avg_rating
65 FROM AppleStore
66 GROUP BY language_bucket
67 ORDER BY Avg_Rating DESC
```

language_bucket	Avg_rating
10-30 languages	4.1385120918384066
>30 languages	3.7777777777777777
<10 languages	3.368327402135231

- This was a similar query from the “Free App rating vs. Paid App rating” query. The only thing different was that I created language support amount brackets.
- I noticed that apps that support 10-30 languages have the highest ratings.
  - This shows that it’s not about how many languages an app supports. It’s about supporting certain languages to reach a wider audience.

69 --Check genres with low ratings  
70  
71 SELECT prime\_genre,  
72 avg(user\_rating) AS Avg\_Rating  
73 FROM AppleStore  
74 GROUP BY prime\_genre  
75 ORDER BY Avg\_Rating ASC  
76 LIMIT 10

prime_genre	Avg_Rating
Catalogs	2.1
Finance	2.4326923876923875
Book	2.4776785714285716
Navigation	2.6847826886956523
Lifestyle	2.8055555555555554
News	2.98
Sports	2.982456140350877
Social Networking	2.9850299401197606
Food & Drink	3.1825396825396823
Entertainment	3.2467289719626167

- I retrieved genres with the lowest average ratings by grouping genres into distinct fields and sorting the average rating by ascending order.
- This query showed me that the lowest-rated genres like Catalogs and Finance had customers that weren’t satisfied by the services and features within a genre.
  - Thus, creating Catalogs or Finance apps are a great opportunity to maximize user ratings because there aren’t many apps within those genres (less competition), and it’ll be a chance to fulfill missing user satisfaction.

78 --Check if there is correlation between the length of the app description and the user rating  
79  
80 SELECT CASE  
81 WHEN length(b.app\_desc) < 500 THEN 'Short'  
82 WHEN length(b.app\_desc) BETWEEN 500 AND 1000 THEN 'Medium'  
83 ELSE 'Long'  
84 END AS description\_length\_bucket,  
85 avg(a.user\_rating) AS average\_rating  
86 FROM  
87 AppleStore AS a  
88 JOIN  
89 appleStore\_description\_combined AS b  
90 ON  
91 a.id = b.id  
92 GROUP BY description\_length\_bucket  
93 ORDER BY average\_rating DESC

description_length_bucket	average_rating
Long	3.855946944988041
Medium	3.232809430255403
Short	2.533613445378151

- Again, I used a CASE statement to assign data into specific fields (Long/Short/Medium) under a column. Then, I used an INNER JOIN between both tables to join data that have matching IDs.

- Here, I noticed that a long app description has the highest average rating compared to medium and short app descriptions.
  - This could be the fact that a long app description gives a clearer understanding of an app's feature and services, leading to better user satisfaction.

## Insights

---

After exploring and analyzing the App Store dataset, here are the insights and recommendations:

1. Paid apps are more likely to have higher ratings than free apps.
  - a. People who pay for an app are more likely to have engagement with the app = more satisfied.
2. Apps between 10-30 languages have the highest ratings.
  - a. It's not about how many languages that an app can support; it's about supporting languages that can reach a wider audience.
3. Finance and Book apps have the lowest ratings.
  - a. This is a great opportunity to create quality Finance/Book apps to fulfill missing customer satisfaction.
4. Apps with longer descriptions have better ratings.
  - a. People are more satisfied with an app if they clearly understand its services and features.
5. New apps should aim for a rating above 3.5.
  - a. Having a rating above 3.5 will help the app stand out (top 50%).
6. Games and Entertainment genres have the highest competition.
  - a. It's extremely tough to stand out due to saturation.
  - b. However, these genres have a high demand = high risk, high reward.