MATH1042: Peluang dan Statistika

Laporan Proyek 1B: Telaga Game Play Store

Kelompok 2

- Joy Milliaan/10102190103/IEE
- Hizkia Lesmana/10102190733/IEE
- Elson Rouslie/10101190694/IBDA
- Yosia Farianto/10101190478/IBDA
- Jabez Joeniko/10101200141/IBDA

Secara keseluruhan, kami telah berhasil mendeskripsikan secara statistika dataset aplikasi game play store yang beredar di Amerika Serikat. Dalam proyek ini, kami menggunakan empat library yaitu pandas, numpy, matplotlib dan seaborn. Kami juga telah menyelesaikan rangkuman statistika yang mana kami telah mendapatkan hasil dari ukuran pusat (mean, median dan modus), ukuran variasi (standar deviasi) dan ukuran lokasi (kuartil, jangkauan interkuartil, dan pencilan) dari empat parameter, yaitu #total ratings, #installs, #great, dan #poor. Berikut adalah hasil dari rangkuman statistika kuantitatif dari parameter-parameter di atas.

```
total ratings
Rata - rata: 1101181.5410404624
Median: 457675.0
Modus: [38238, 138238, 311]
Variansi: 10967611812693.422
Standar Deviasi: 3311738.4879687317
Kuartil 1: 187998.75
Kuartil 3: 944334.25
Jarak Interkuartil: 756335.5
Pencilan:
0 data in low outlier
165 in high outlier
```

```
int installs
Rata - rata: 28894624.27745665
Median: 10000000.0
Modus: [10000000, 10100000, 795]
Variansi: 3375661341819422.0
Standar Deviasi: 58100441.83841825
Kuartil 1: 5000000.0
Kuartil 3: 50000000.0
Jarak Interkuartil: 45000000.0
Pencilan:
```

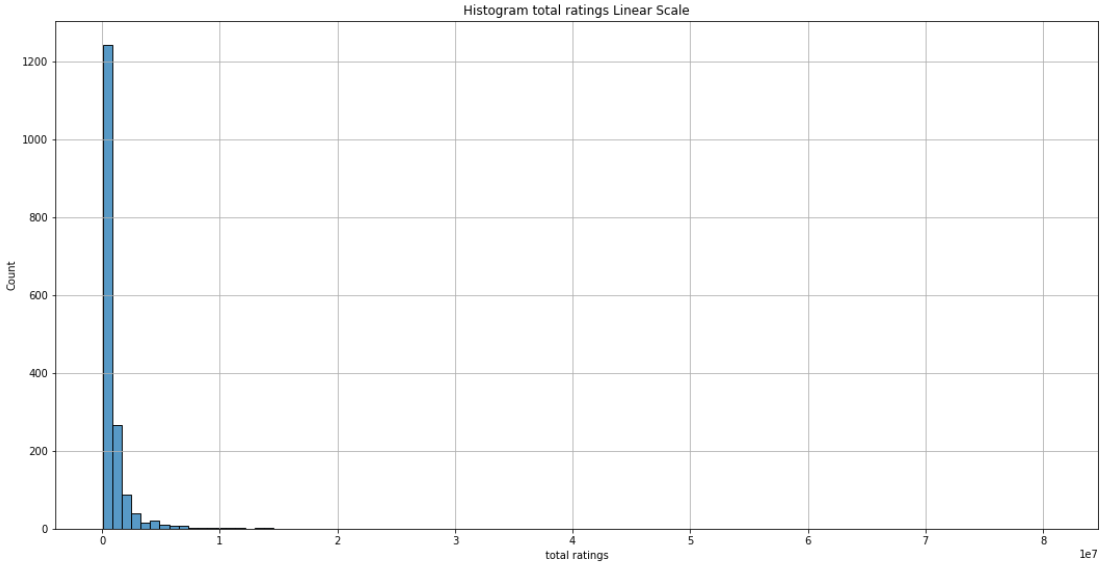
```
0 data in low outlier
12 in high outlier



great
Rata - rata: 910030.3271676301
Median: 367617.0
Modus: [26635, 1026635, 1415]
Variansi: 7625611211070.817
Standar Deviasi: 2761450.9249796234
Kuartil 1: 157423.25
Kuartil 3: 775977.75
Jarak Interkuartil: 618554.5
Pencilan:
0 data in low outlier
167 in high outlier



poor
Rata - rata: 131598.83468208092
Median: 47221.5
Modus: [815, 1000815, 1700]
Variansi: 188561019782.99524
Standar Deviasi: 434236.1336680715
Kuartil 1: 18421.5
Kuartil 3: 111270.0
Jarak Interkuartil: 92848.5
Pencilan:
0 data in low outlier
164 in high outlier
```

Dalam pengerjaan parameter #great, kami menambahkan semua nilai pada 5 star ratings dan 4 star ratings. Begitu juga ketika kami mengerjakan parameter #poor, kami menambahkan nilai pada kolom 2 star ratings dan 1 star ratings . Dan untuk parameter #total ratings dan #installs, kami menggunakan kolom yang sudah tersedia. Dalam pengerjaan proyek ini, kami juga menggunakan metode Object Oriented Programming, yang mana memudahkan kami dalam mengidentifikasi dan membuat kode kami terlihat lebih rapi dan terstruktur. Selanjutnya, kami juga membuat histogram dan boxplot untuk menganalisis dataset yang kami kerjakan. Dalam pengerjaan histogram, kami menemukan beberapa kendala yaitu sulit menganalisa ketika menggunakan count maupun frekuensi relatif. Sehingga, pada akhirnya kami memutuskan untuk menggunakan log scale agar visualisasi lebih mudah untuk dianalisa. Berikut adalah hasil visualisasi menggunakan histogram dan boxplot.

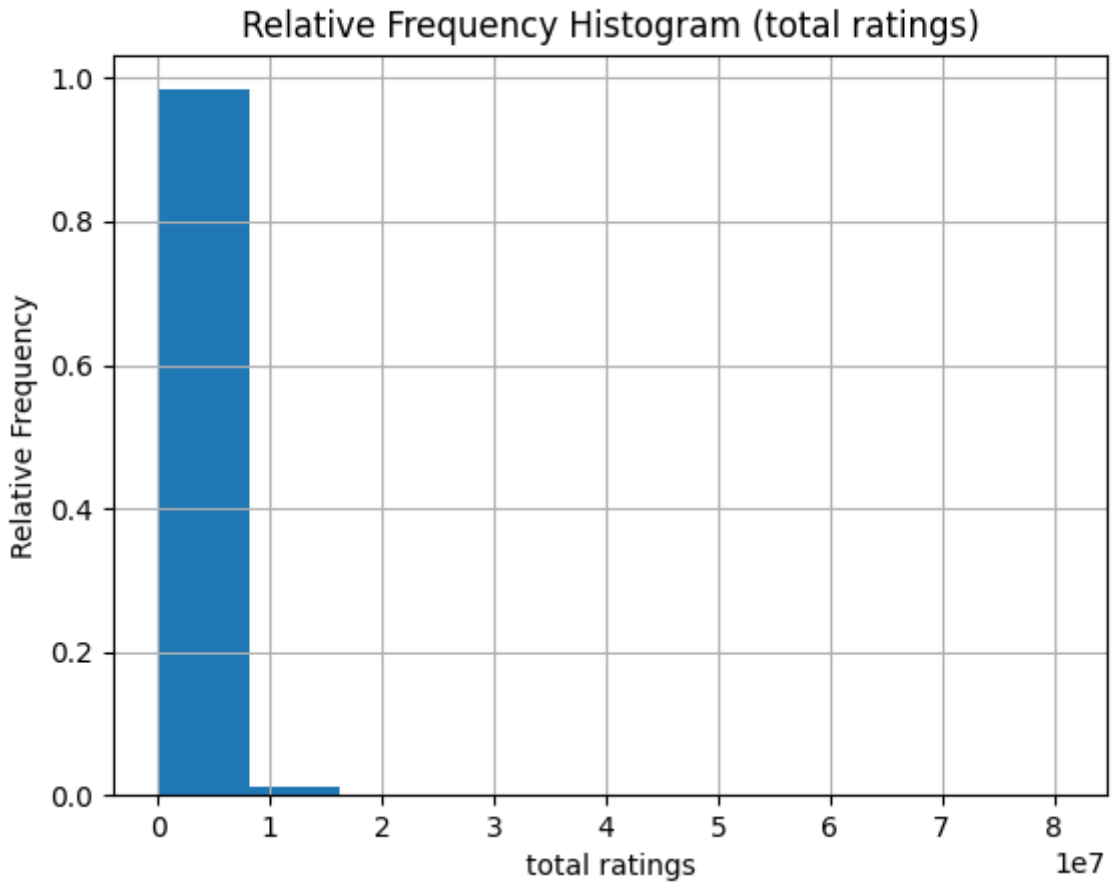
Total Ratings

Histogram

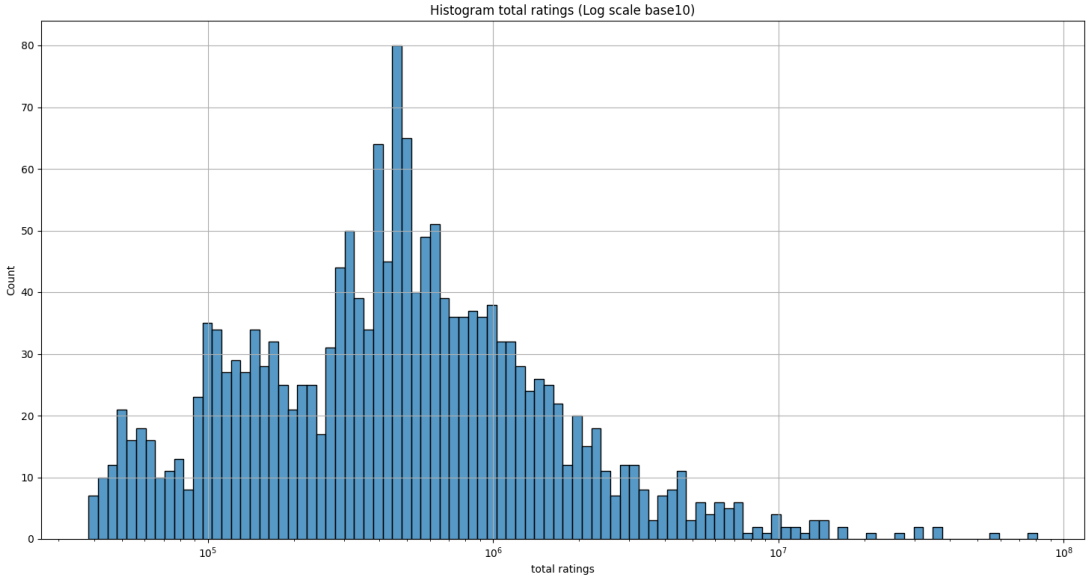
Histogram Skala Linier



Histogram Relative Frequency

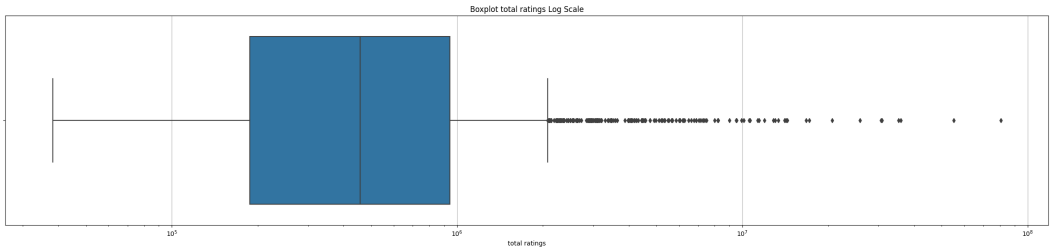


Histogram Skala Log10



Boxplot

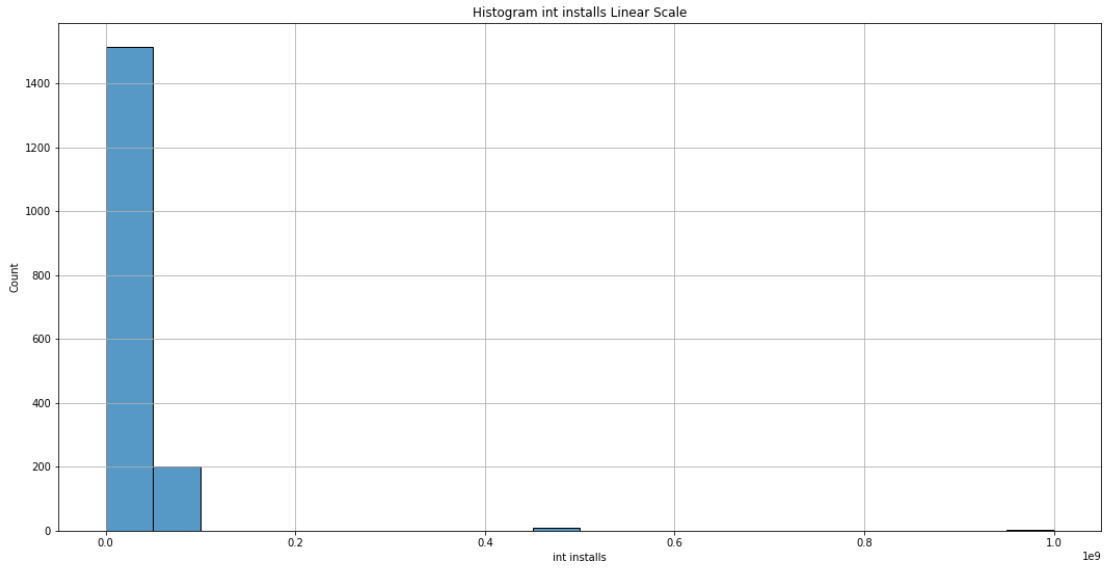
Boxplot Skala Log10



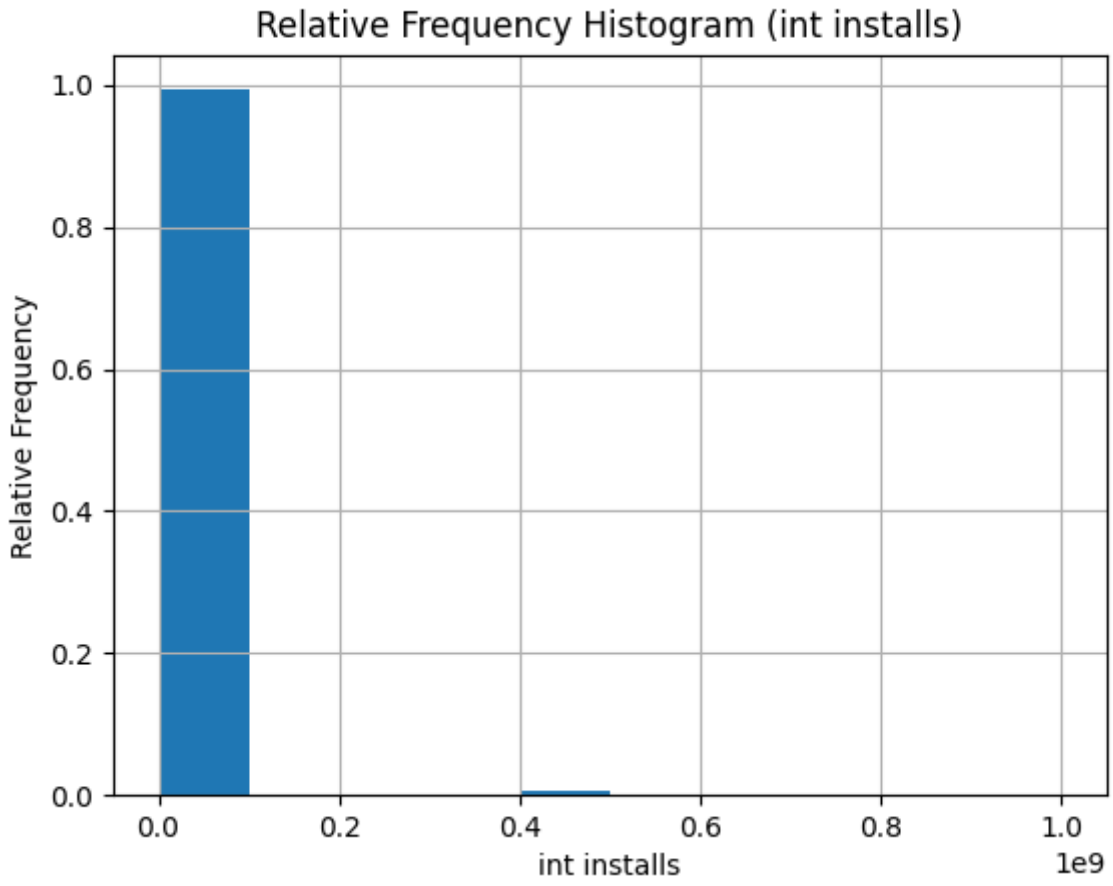
Installs

Histogram

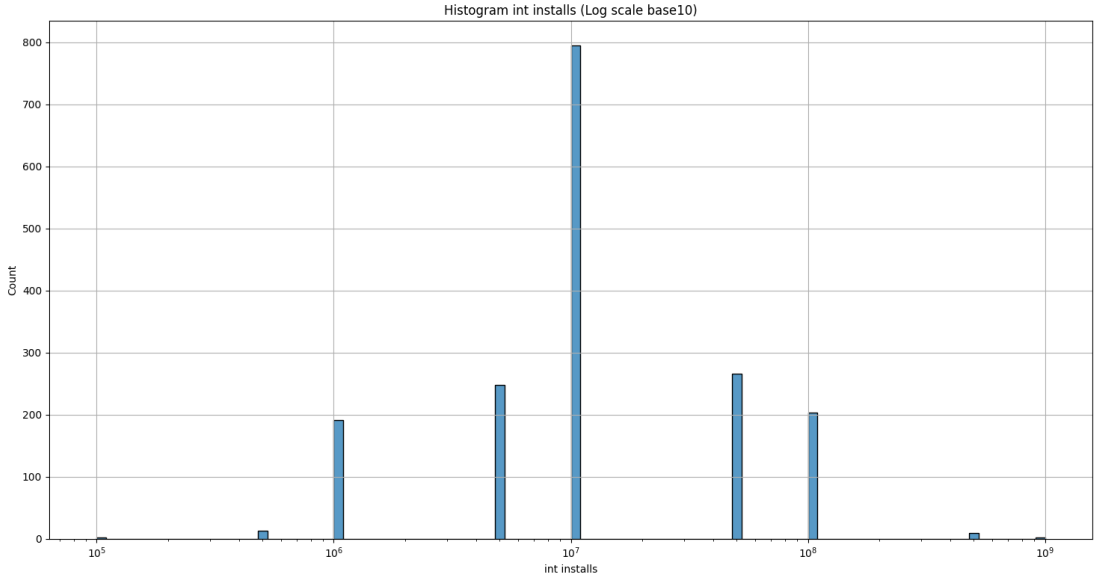
Histogram Skala Linier



Histogram Relative Frequency

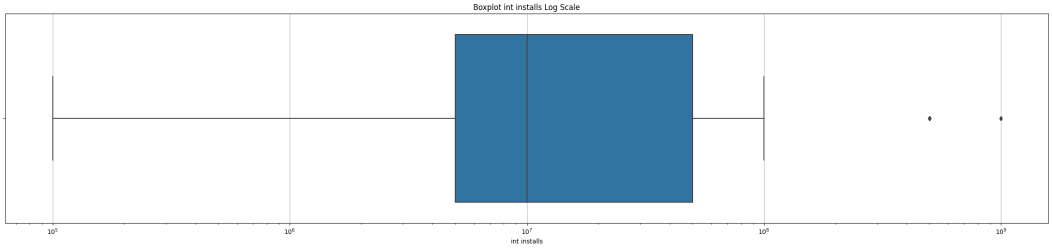


Histogram Skala Log4



Boxplot

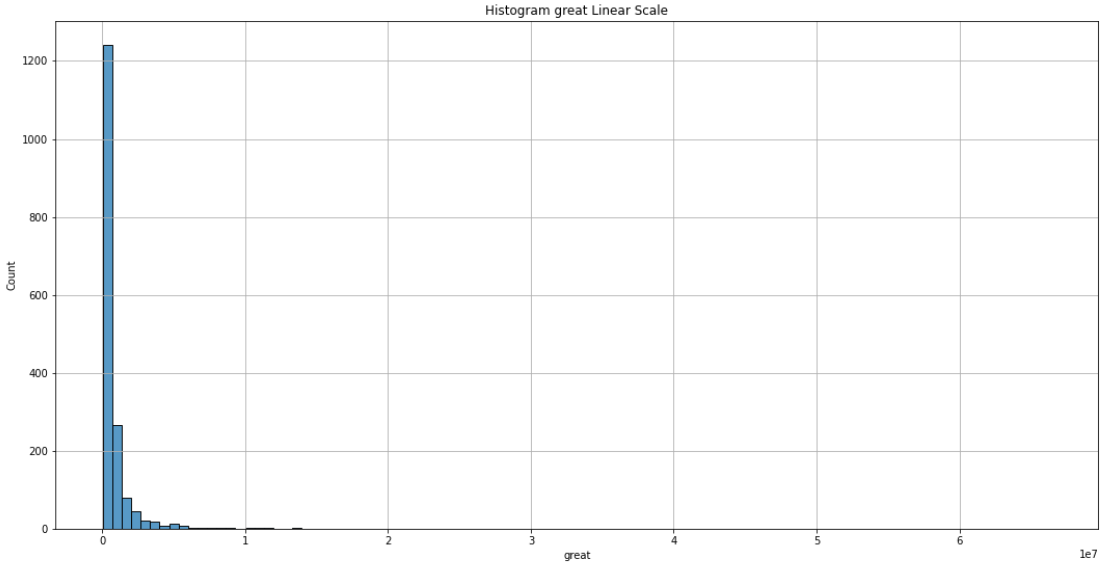
Boxplot Skala Log10



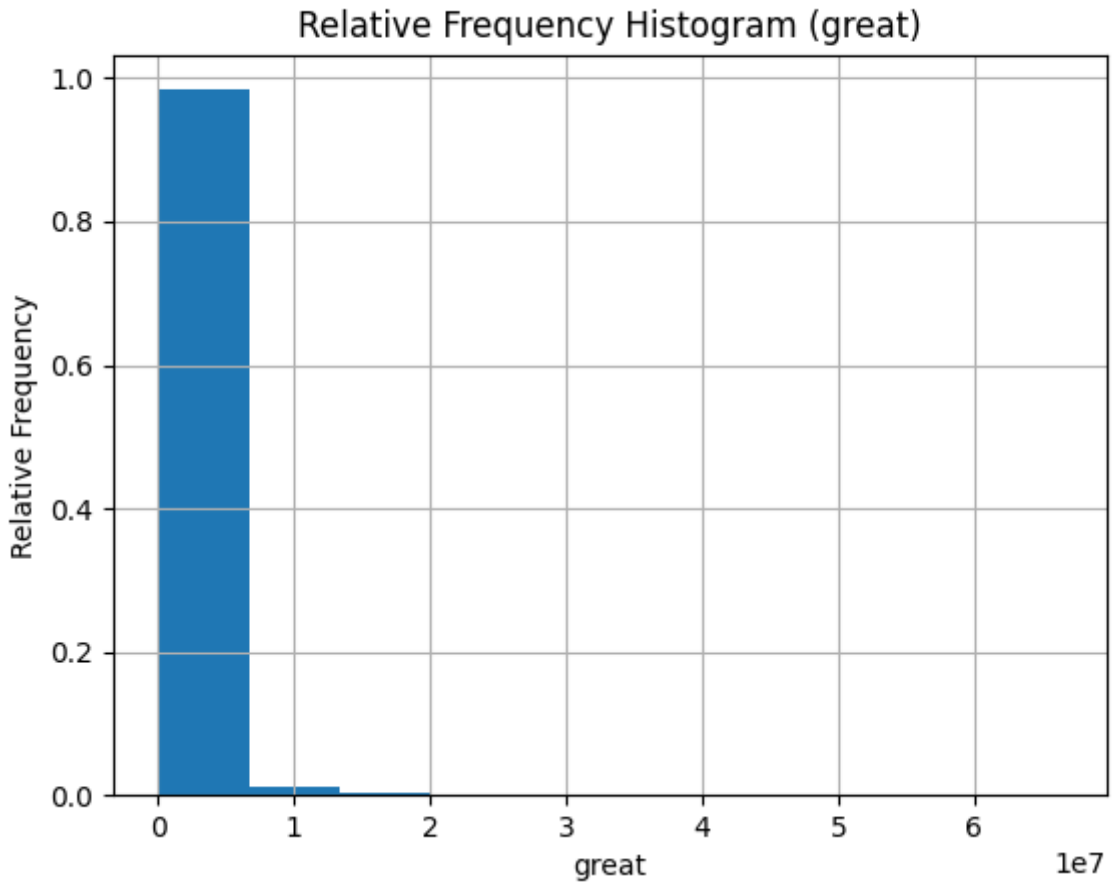
Great

Histogram

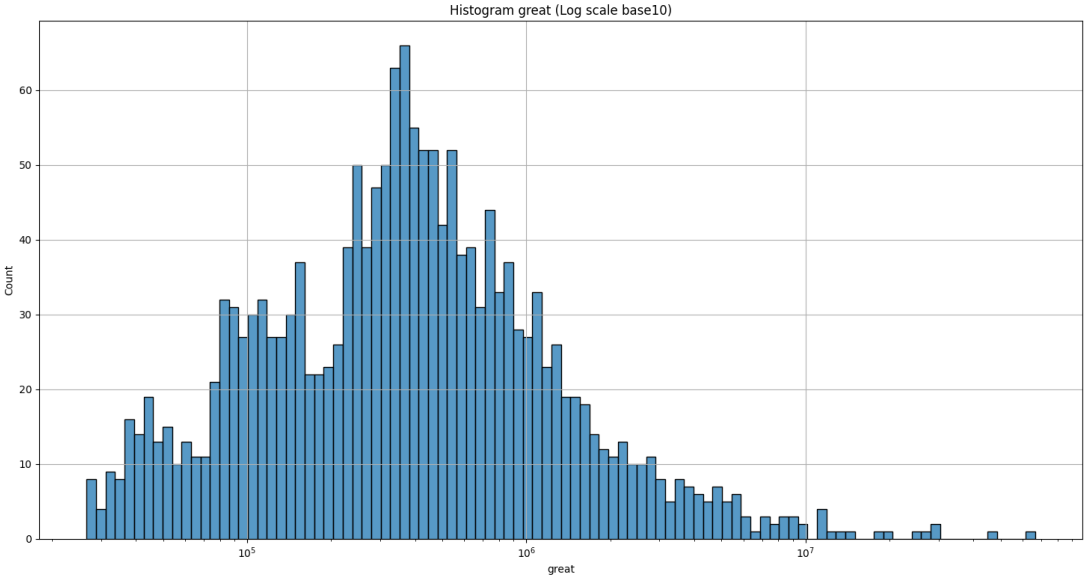
Histogram Skala Linier



Histogram Relative Frequency

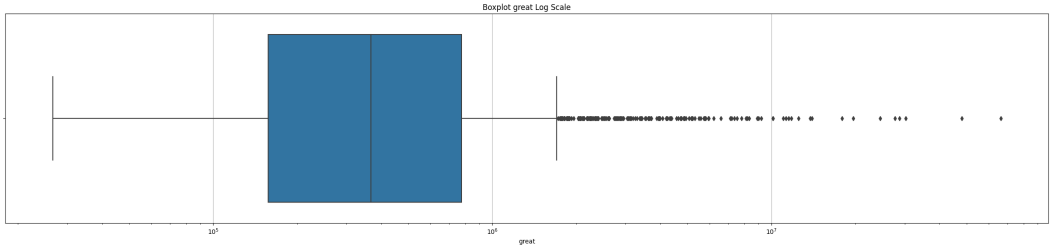


Histogram Skala Log10



Boxplot

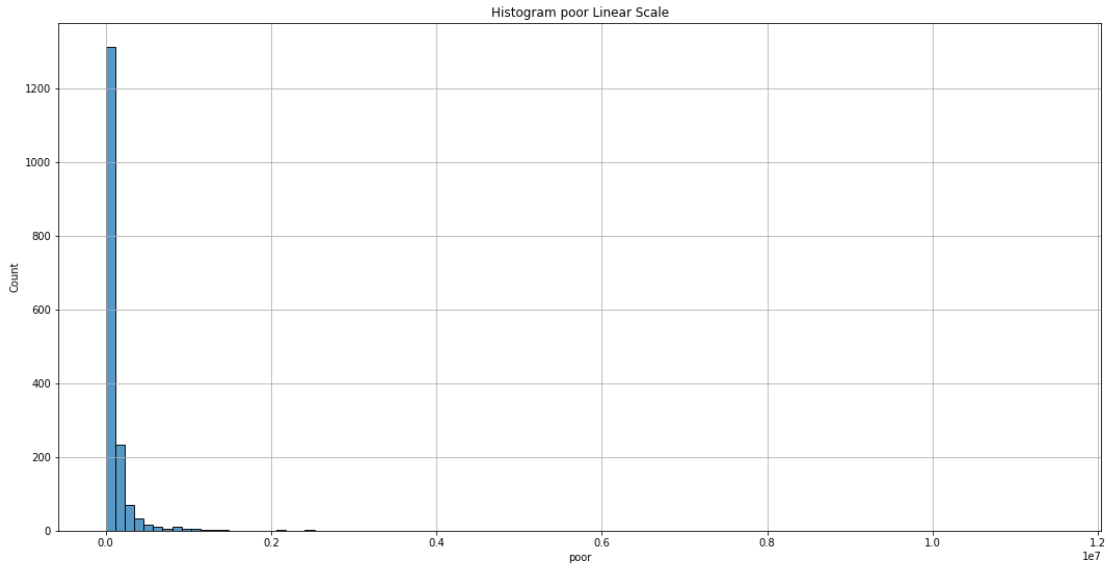
Boxplot Skala Log10



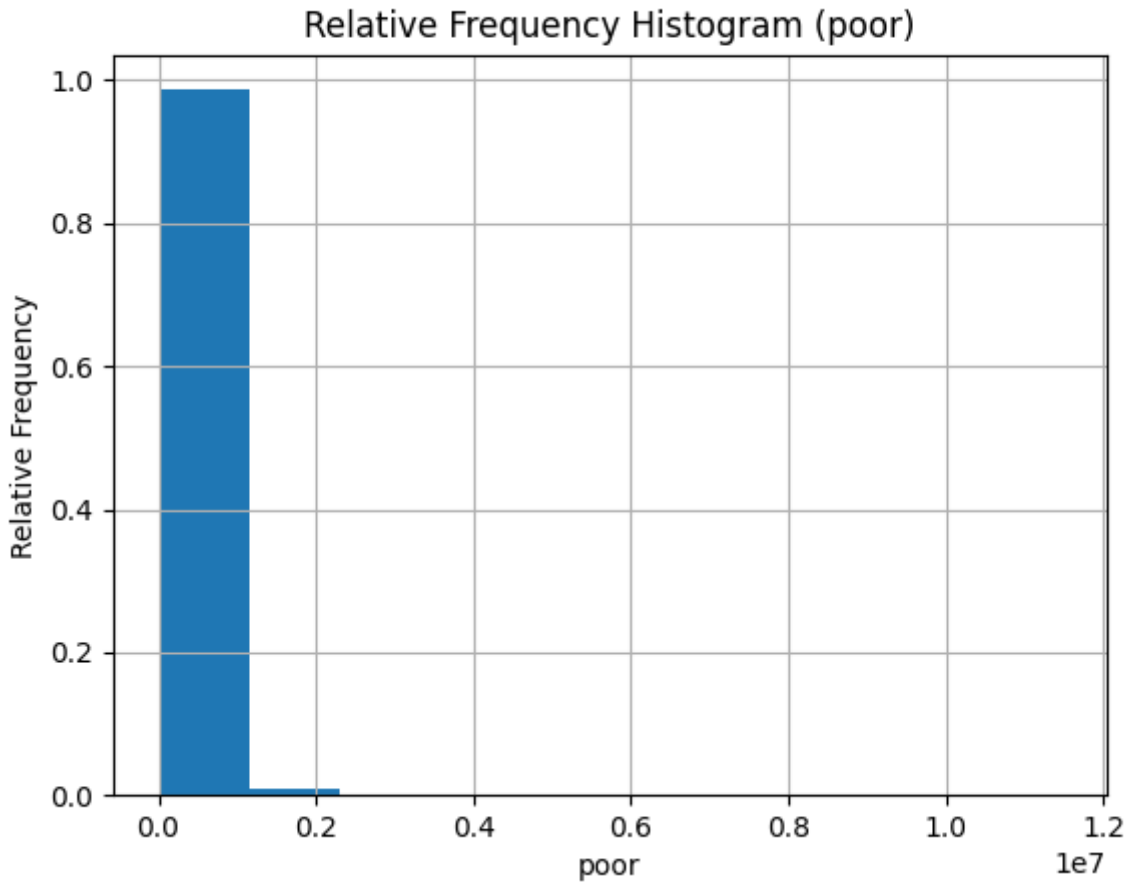
Poor

Histogram

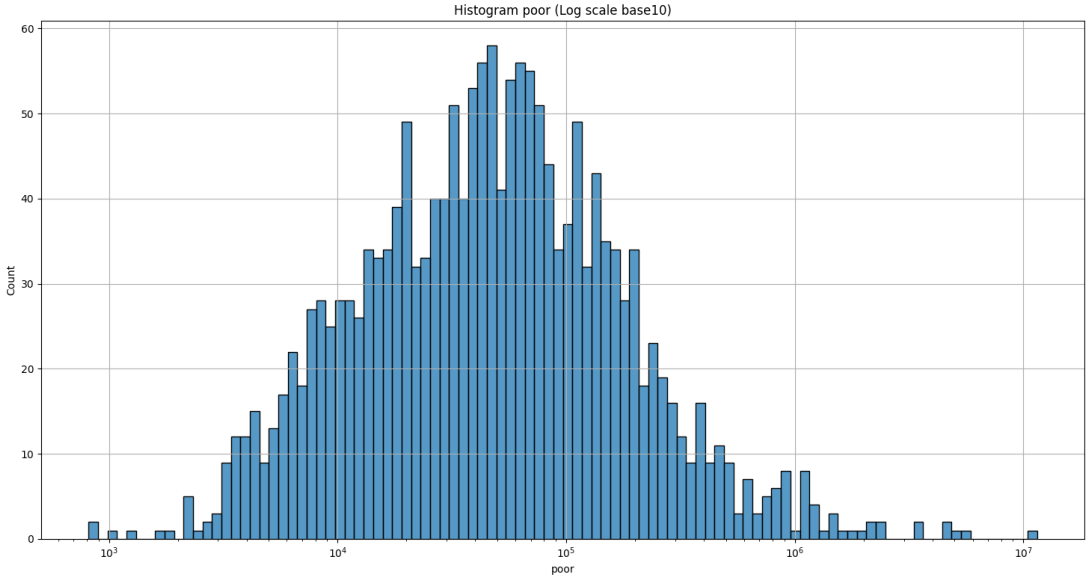
Histogram Skala Linier



Histogram Relative Frequency

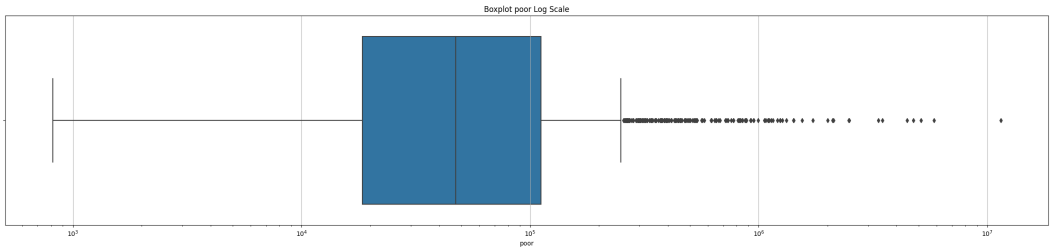


Histogram Skala Log10



Boxplot

Boxplot Skala Log10



Kesimpulan

Dari pengerjaan proyek ini kami pun menarik beberapa kesimpulan sebagai berikut:

1. Kebanyakan data berada pada interval nilai yang kecil.
2. Tidak ada low outlier karena semua data berada pada interval nilai yang kecil.
3. Menggunakan log scale pada histogram mempermudah kita melihat visualisasi data dibanding menggunakan frekuensi relatif.

Pertanyaan Diskusi A

Untuk mengkalkulasi nilai terpencilnya game dengan jumlah install lebih dari 100 juta, kami menggunakan cara berikut:

1. Hitung rata-rata jumlah installs bagi game dengan >100 juta installs. Nilai rata-rata ini digunakan untuk mendapatkan lokasi pusat jumlah install >100 juta dalam garis bilangan.
2. Bagi nilai rata-rata installs >100 juta dengan rata-rata seluruh installs. Semakin besar hasil pembagian ini, semakin terpencil game dengan > 100 juta install.
3. (Tidak diperlukan dalam kasus ini) Jika kita mau melihat nilai ke-terpencilan sekelompok data lain, kita bisa menambahkan tanda (+) jika rata-rata target lebih besar dari rata-rata seluruh data dan tanda (-)

jika rata-rata target lebih kecil dari rata-rata seluruh data.

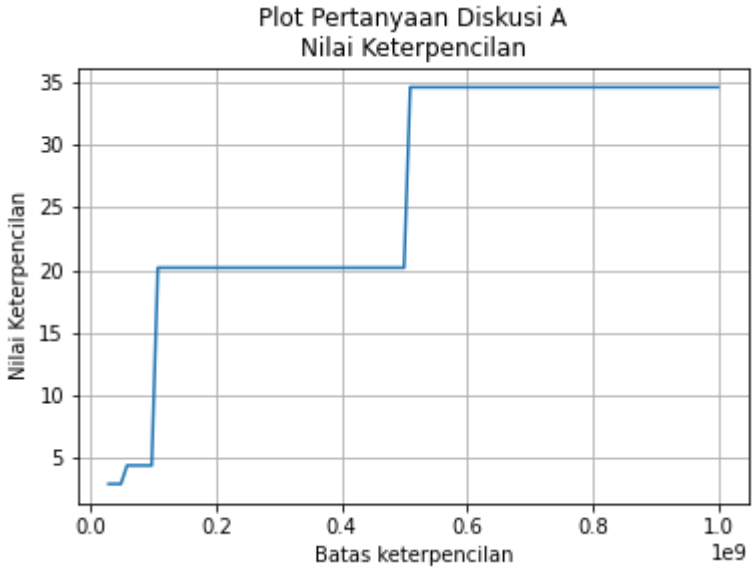
O : Nilai keterpencilan
n : Batas jumlah installs
$\overline{X}_n$: Rata-rata installs di atas batas
$\overline{X}$: Rata-rata installs seluruh data
$O = \frac{\overline{X}_n}{\overline{X}}$

```
installs.pertanyaan_a()
```

```
20.19
```

Nilai keterpencilan dari game dengan jumlah install di atas 100 juta adalah 20.19.

Untuk memperjelas bagian ini, kami membuat plot yang menunjukkan nilai keterpencilan dengan jangkauan batas dari rata-rata installs sampai maksimum installs.



Bisa dilihat bahwa nilai terkecil pada plot ini adalah 1 yaitu ketika rata-rata dibagi dengan dirinya sendiri. Kemudian nilai tertinggi sedikit kurang dari 35. Nilai tertinggi adalah ketika rata-rata jumlah installs bernilai $1.000.000.000$ dibagi dengan rata-rata seluruh data yaitu $28.894.624.28$ dengan hasil 34.61 (sedikit kurang dari 35).

Pertanyaan Diskusi B

Untuk menentukan batas tertinggi 0.1%, kami menggunakan cara berikut:

1. Kalikan 0.1% dengan jumlah data. Tahap ini digunakan untuk menetukan banyaknya data yang ada di 0.1% paling tinggi. Perhitungan ini kita sebut m.
2. Kemungkinan besar sekali hasil perhitungan no. 1 bernilai desimal, maka hasil perhitungan tersebut dibulatkan ke bawah.

3. Urutkan dataset dari nilai terbesar sampai terkecil berdasarkan kolom total ratings
4. Ambil n data paling besar. Dengan n adalah jumlah data 0.1% paling tinggi.
5. Untuk menentukan batas nilai 0.1%, sebenarnya kita bisa langsung mengambil nilai paling kecil dari data 0.1% tersebut (karena jumlah data harus bersifat diskrit). Tetapi jika kita menganggap data ini kontinu, kita bisa menggunakan cara berikut:
   A. Ambil komponen desimal dari perhitungan nomor 1. Misal ada 1354 data, maka $m = 1.354$ dan $n = 1$. Komponen desimal adalah $d = m - n$, maka $d = 0.354$.
   B. Mengikuti contoh 5.A, misalkan nilai terbesar data tersebut adalah sebagai berikut: $[\ldots, 732, 856, 987]$. Jelas data 0.1% adalah 987 saja. Tetapi batasan untuk data persentil ke 0.1% ada pada jarak ke-0.354 dari 987 ke data sebelumnya yaitu 856.
   C. Untuk menghitung batas ini, kami menggunakan formula berikut:
      X_{n-1} : data persentil 0.1% terkecil (ditambah -1 karena menggunakan penomoran dari 0)
      X_n : data terbesar yang tidak masuk persentil 0.1%. Bisa dikatakan juga data persis sebelum X_{n-1}.
      X_d : batasan persentil 0.1%. Jika data lebih besar dari X_d, maka data termasuk 0.1% paling besar.
      Formula:
      $X_d = X_{n-1} - d(X_{n-1} - X_n)$
      Contoh Perhitungan:
      $X_d = 987 - 0.354 \cdot (987 - 856)$
      $X_d = 940.63$

```
total_ratings.pertanyaan_b()
```

```
80678661 - 0.73 * 44706700
1 0    80678661
Name: total ratings, dtype: int64
48042770.0
```

Batas minimal #totalrating untuk mendapatkan 0.1% game favorit adalah 48.042.770 rating.

Pertanyaan Diskusi C

Parameter total ratings, installs, great, dan poor penting untuk dipertimbangkan semua karena:

- jika total ratings ditinggalkan, kita bisa mendapatkan game dengan respon positif dan banyak installs tetapi yang menilai sedikit sehingga penilaian kualitas game kurang akurat.
- jika installs ditinggalkan, kita bisa mendapatkan game dengan rating positif (kelihatannya bagus) tetapi sedikit yang memainkan
- great dan poor harus diikuti ke perhitungan karena sudah jelas kedua parameter ini menunjukkan kualitas game tersebut.

Dari ide ini, kami mengatakan bahwa:

- semakin banyak total ratings, semakin bagus (berbanding lurus dengan nilai keseluruhan)
- semakin banyak installs, semakin bagus (berbanding lurus dengan nilai keseluruhan)

- semakin banyak great, semakin bagus (berbanding lurus dengan nilai keseluruhan)
- semakin banyak poor, semakin buruk (berbanding terbalik dengan nilai keseluruhan)

Maka itu, kami membuat formula berikut untuk menentukan nilai keseluruhan dari satu game:
N : Nilai keseluruhan satu game
t_r : total ratings
i : jumlah install
g : jumlah rating 4 dan 5 bintang
p : jumlah rating 2 dan 1 bintang

$$N = t_r \times i \times \frac{g}{p}$$

```
data_final = data.sort_values(by='final game score', ascending=False)
data_final.head()
```

title	total ratings	installs	5 star ratings	4 star ratings	2 star ratings	1 star ratings	great	poor	int installs	rati ind
Candy Crush Saga	30859251	1000.0 M	23521533	4128497	457810	1266557	27650030	1724367	1000000000	16.034.8
Subway Surfers	35305263	1000.0 M	26840330	3377645	801027	2667798	30217975	3468825	1000000000	8.711.3
Clash of Clans	55170976	500.0 M	42736445	5397273	978099	3773793	48133718	4751892	500000000	10.129.3
Garena Free Fire - The Cobra	80678661	500.0 M	61935712	4478738	1814999	9654037	66414450	11469036	500000000	5.790.7
My Talking Tom	16715691	500.0 M	12245585	1517427	445084	1661606	13763012	2106690	500000000	6.533.0

Ditemukan bahwa game **Candy Crush Saga** memiliki nilai kualitas paling tinggi dari semua game yaitu sebesar 4.95×10^{23}.