Clash Royale Data Intern Test

by Bence Bánsághi

Introduction of the process

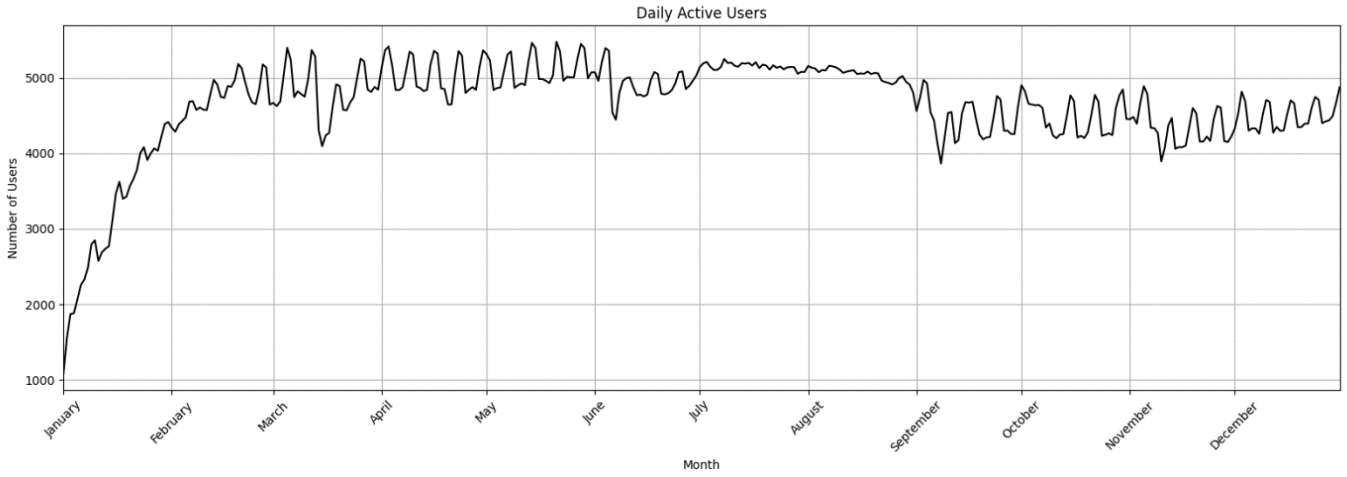
For this task, I chose to work in python using the sqlite3 library. All the graphs and specific numbers in the first part of the document will be referenced to the appropriate part of the code in the second part of the document.

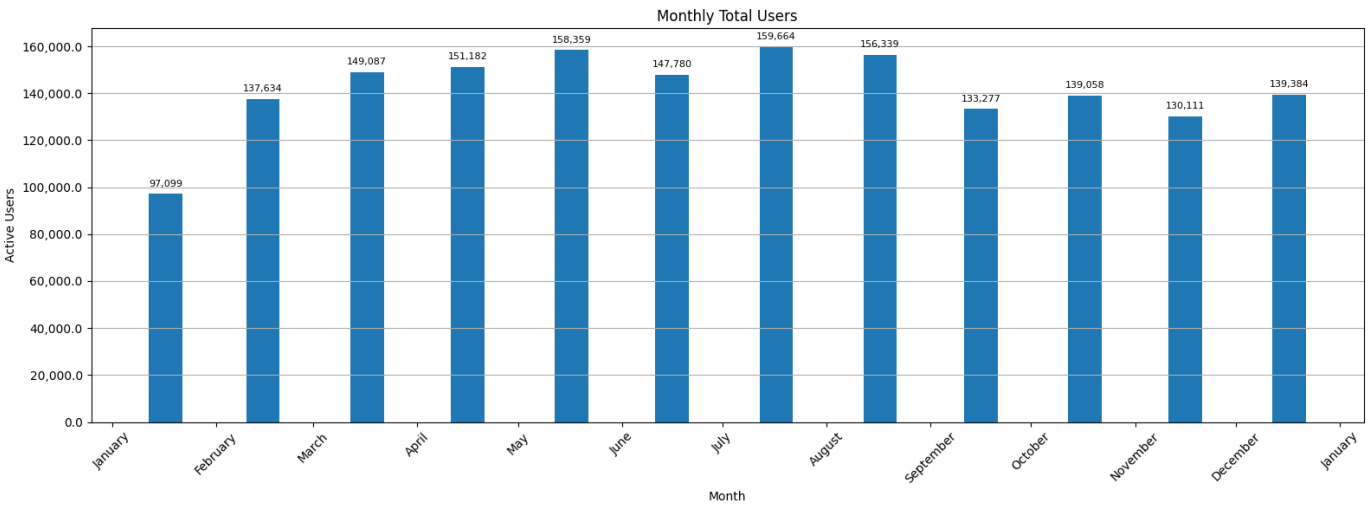
Short description of datasets

There are three tables in the ‘sample.sqlite’ database: 'account', 'account_date_session', and 'iap_purchase'. The 'account' table contains information about the user's account, including when it was created, on which device and platform it was created, in which country it was created, and the app store ID. The 'account_date_session' table details information about game sessions per day, how many times the user opened the application, and how much time they spent on it in total. The 'iap_purchase' table has information on when the purchase was made, what its hash and price are, and the app store ID. The tables are linked together by account ID.

Analysing the daily active users

To start analyzing the data, first we will plot the daily active users and monthly total users for the whole year to get a clear overview.



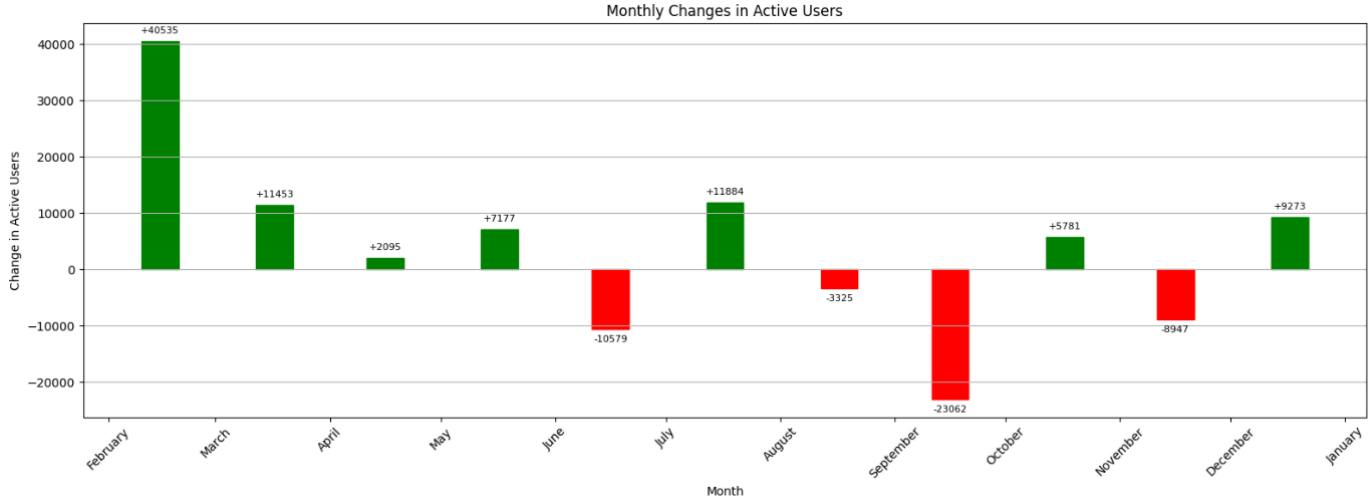


First glances

At a first glance, we can see that the application started gaining popularity at the start of the year, as the number of active daily users starts from around a thousand in January and grows to 4-5 times its size by the middle of February. It is also visible immediately that in the summer the variation between weekdays is less than during the rest of the year. This could be due to the player base being on summer vacation and not needing to stick to the 5/2-week structure. After the summer, as people got back to work and school, the weekly pattern returns to a similar looking one as before. I have also noticed a few bigger drops that seem like outliers. They are middle of March, beginning of June, beginning of September and beginning of November.

Monthly changes

Next, we will look at the monthly changes in active users, to see what changes we can observe.

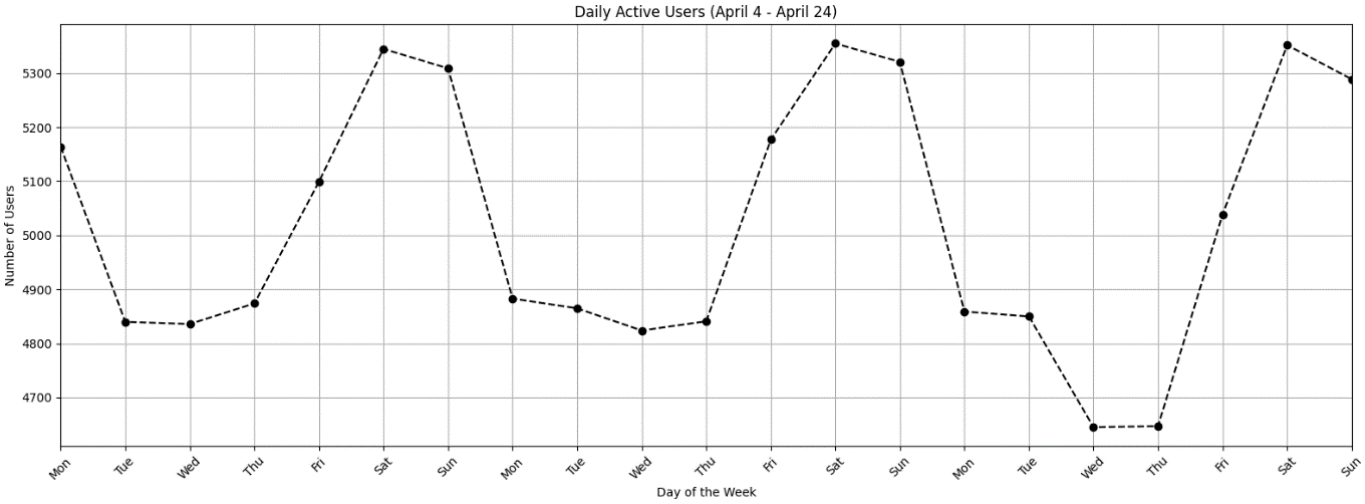


In February and March, the application was still gaining traction, so the high user gains are easily explained. It seems that at the beginning of the summer, the numbers fell, but climbed back up again. I believe this is due to the fact of going on vacation in the early summer days and coming back to games as it becomes a habit as well, or it could be due to exam season being in the beginning of summer. After the summer, there is a big decline, where it mostly stays for the rest of the year. It is easily explained by people going back to work and school and having less time to spend on games.

Variation between regular and summer weeks

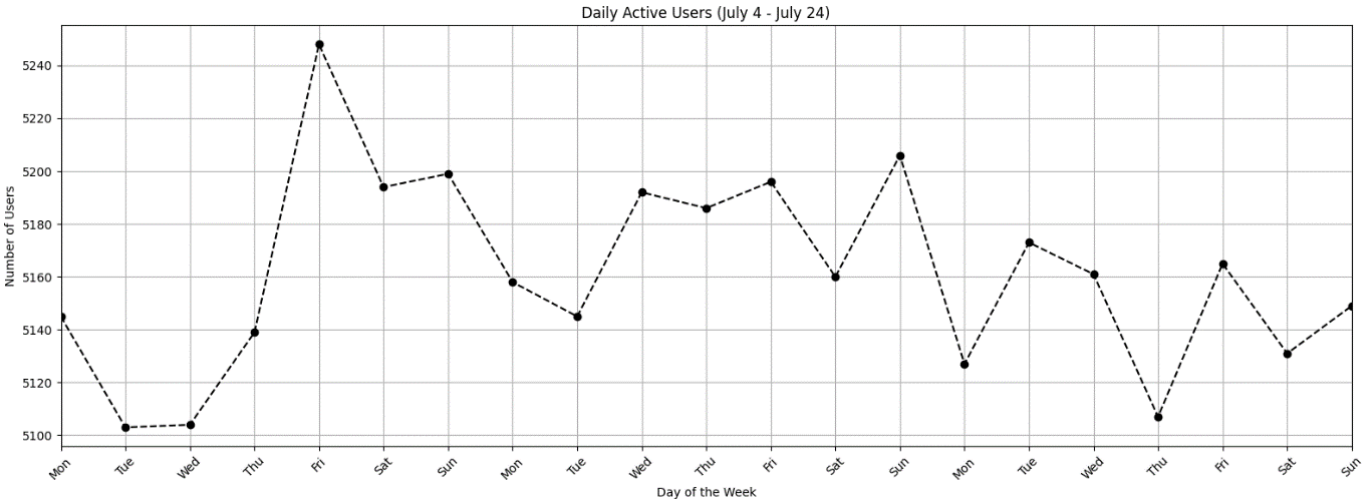
In this section, we will look at how the weekly ups and downs change between regular and summer weeks by selecting an arbitrary three-week period from both.

Regular operation:



From this 3-week segment of normal operations, it is clearly visible that the player base is more active during weekends than weekdays. I counted Friday in the weekdays category, but it is visible that the daily activity is higher than on other weekdays, however not as much as weekends.
During this three-week period, the average difference between weekdays and weekends was 432.3.

Summer operation:



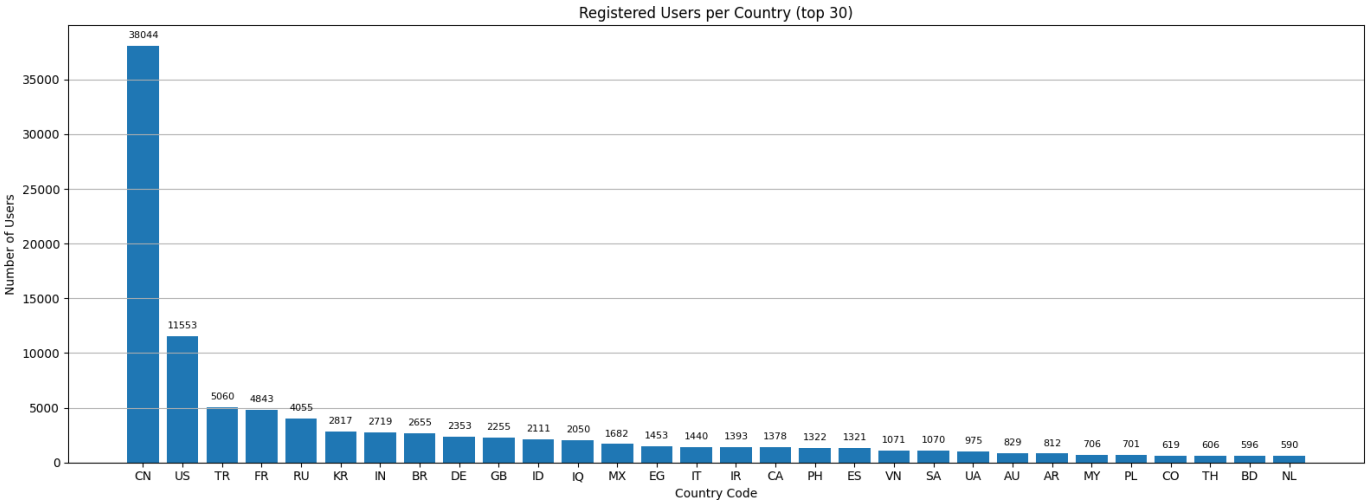
From this 3-week segment of summer operations, there does not seem to be any major differences in daily active users between weekdays and weekends.
During this three-week period, the average difference between weekdays and weekends was 16.6.

We can see that the assumption made in the first glances section was proven, there really is a difference between weekdays and weekends in normal operation, while this does not hold true in the summer.

Sales analysis

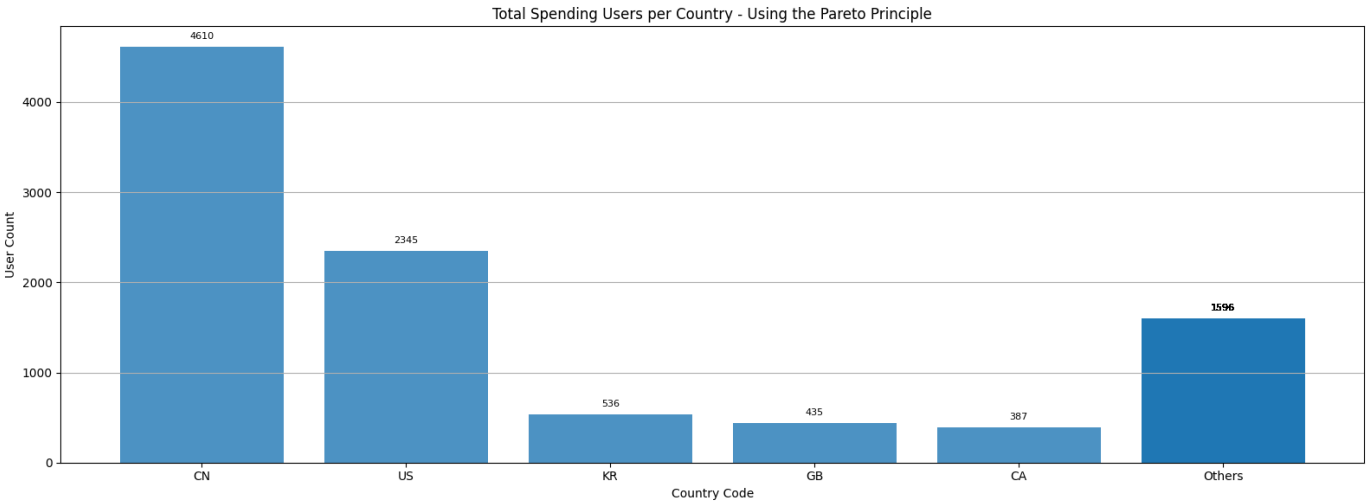
User analysis

First, we will look at the geographic split of users between the countries. Only the top 30 countries were selected for better visibility.



We can see, that the vast amount (33.76%) of the player base is from China and 10.25% is from the USA, so the company probably needs to be focused on these markets the most as they make up a large percentage of the total player base. However, the other 56.03% of the players are distributed between 188 countries, so aside from focus on the Chinese and American markets, there has to be an encompassing focus on the others. We will see this more clearly in the next two graphs, which detail total paying users per country and total spending per country.

In the next graph, we will see the total users who pay money per country. From now on, I will be using the Pareto principle to group together countries in plots which are related to earnings.

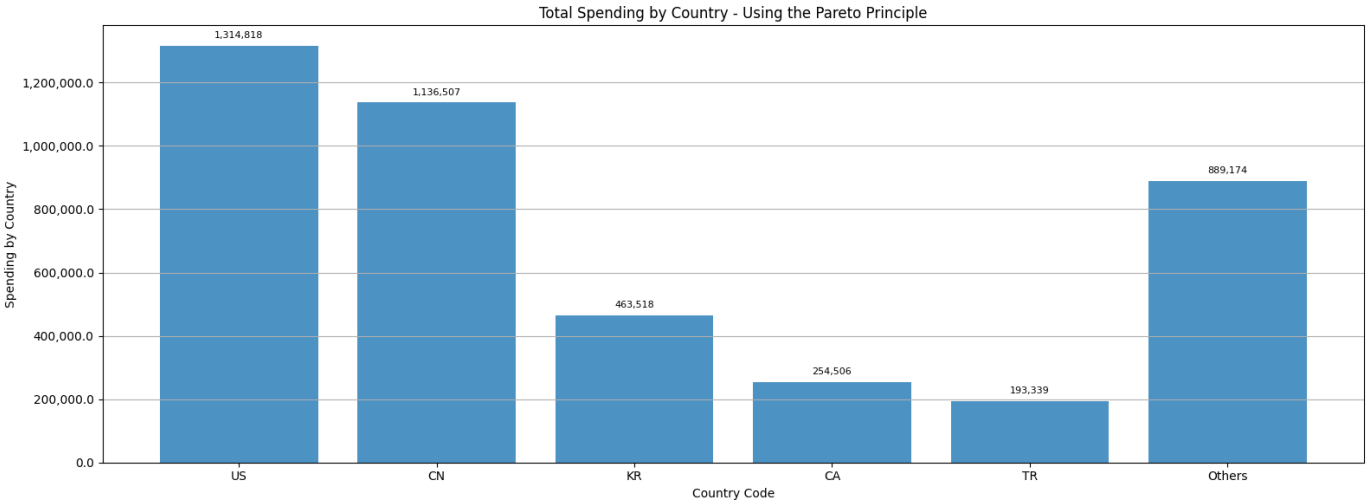


The total paying customers per country is mostly consistent with the total user base per country.

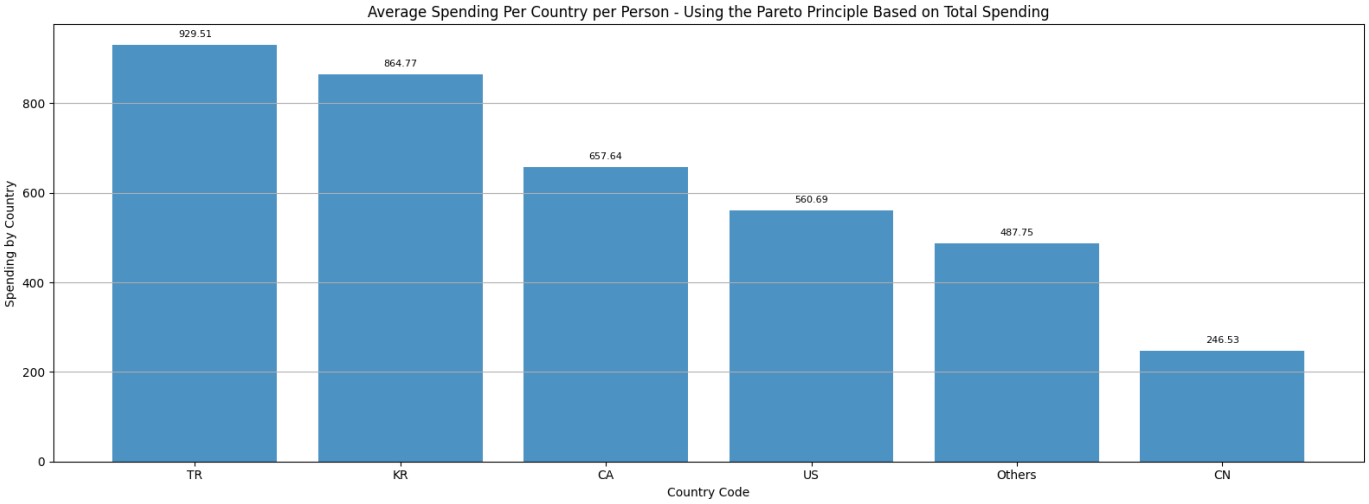
In total, out of 112792 total users there are 9909 spending customers, which is about 8.79% of users.

Spending analysis

In this section, we will first have a look at the total spending and then the average spending per person in the different markets. The average spending across all markets is 429.09, and I will refer to this number when I say global average.



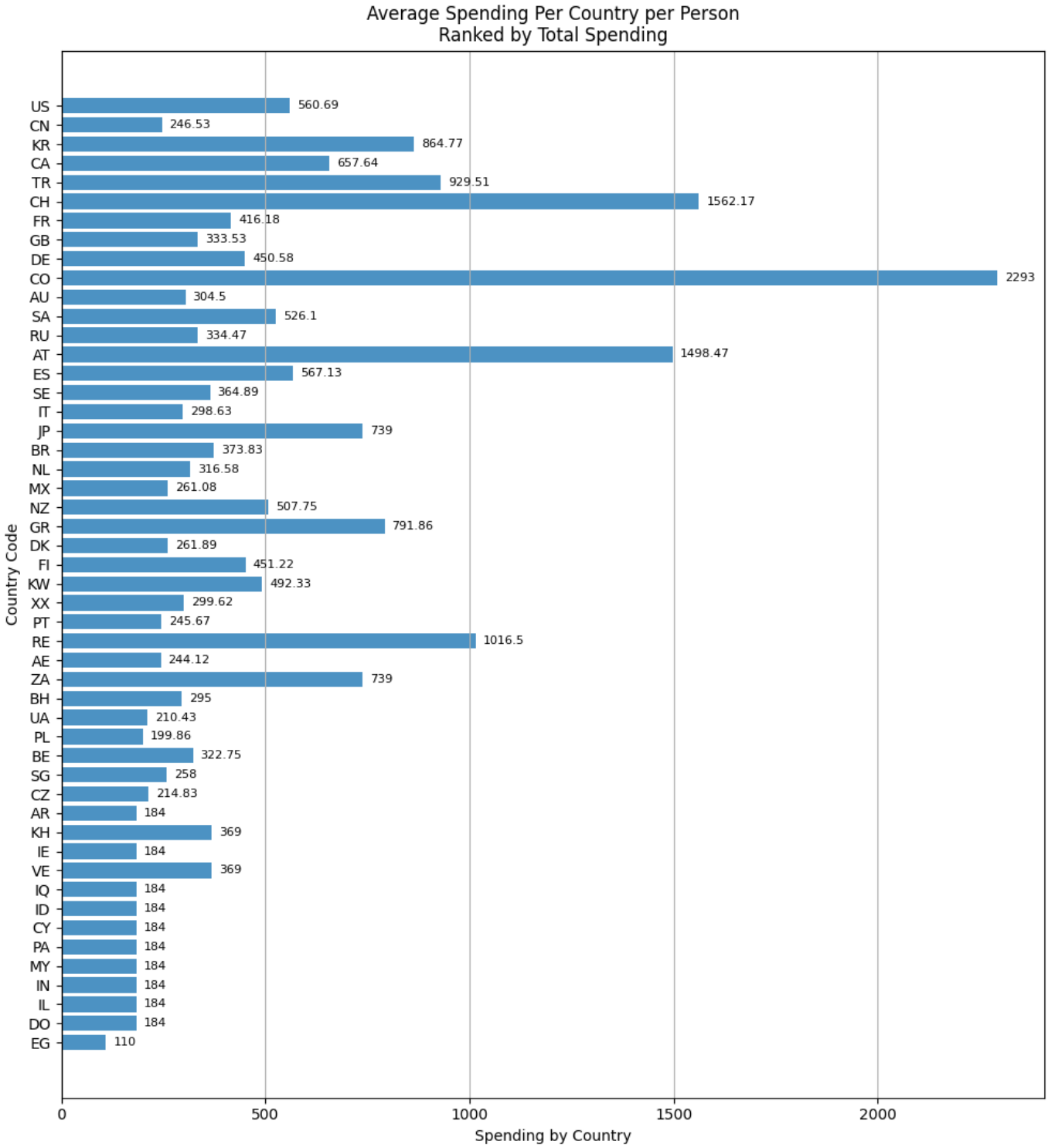
This graph shows something quite interesting; while the US only has about half of the paying players, it spends more in total. This is most likely because the standards of living are higher than in China, so users have more money to spend. However, because of the sheer size of the player base, China is still barely behind the USA.



We can see that the average spending does not necessarily depend on total spending. One thing that is important to notice is how much the average spending in China falls behind the rest. This can signal an opportunity to entice the Chinese player base to spend more, as even a little more spending per person would lead to a huge profit margin.

Average revenue per user per market

Here, I will showcase the full graph of the average revenue per user per market of all the countries with paying players. The data without countries shown as XX. I do not know if this is an error or intentional, as there wasn't any information whether the country was a requirement, so I will leave it as is.



Python code

```
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
import numpy as np

# Open sqlite 3
database_connection = sqlite3.connect("sample.sqlite")

#Daily active user data plotting for the whole year
daily_active_users=pd.read_sql_query("SELECT date, COUNT(DISTINCT account_id) AS
active_users FROM account_date_session GROUP BY date", database_connection)
daily_active_users["date"] = pd.to_datetime(daily_active_users["date"],format="%Y-
%m-%d")
plt.figure(figsize=(16, 6))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%B"))
plt.plot(daily_active_users["date"],
daily_active_users["active_users"],color="black", marker="", linestyle="-")
plt.title("Daily Active Users")
plt.ylabel("Number of Users")
plt.xlabel("Month")
plt.xlim(daily_active_users["date"].min(), daily_active_users["date"].max())
plt.grid(True)
plt.xticks(rotation=45)
plt.show()

#Changes from one month to the other
#Resampling to get monthly total active users and monthly changes
monthly_total_users =
daily_active_users.set_index("date").resample("M").agg({"active_users": "sum"})
monthly_total_users["monthly_changes"] = monthly_total_users["active_users"].diff()

#Plotting total monthly users
plt.figure(figsize=(16, 6))
bars=plt.bar(monthly_total_users.index, monthly_total_users["active_users"],
width=10)
plt.title("Monthly Total Users")
plt.xlabel("Month")
plt.ylabel("Active Users")
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%B"))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(bymonthday=15))
plt.xticks(rotation=45)
formatter = ticker.StrMethodFormatter("{x:,}")  # Format long values for better
visibility
plt.gca().yaxis.set_major_formatter(formatter)
labels = ['{:,}'.format(val) for val in monthly_total_users["active_users"]]
plt.bar_label(bars, labels=labels, label_type="edge", fontsize=8,padding=5)
plt.grid(axis="y")
```

```
plt.tight_layout()
plt.show()

#Plotting monthly changes in active users
plt.figure(figsize=(16, 6))
bars=plt.bar(monthly_total_users.index, monthly_total_users["monthly_changes"],
width=10)
plt.title("Monthly Changes in Active Users")
plt.xlabel("Month")
plt.ylabel("Change in Active Users")
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%B"))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(bymonthday=15))
plt.xticks(rotation=45)
for i, bar in enumerate(bars):
    if monthly_total_users["monthly_changes"].iloc[i] >= 0:
        bar.set_color("green")
    else:
        bar.set_color("red")
labels = [("{:+.0f}".format(val) if val > 0 else "{:.0f}".format(val)) for val in
monthly_total_users["monthly_changes"]]
plt.bar_label(bars, labels=labels, label_type="edge", fontsize=8,padding=5)
plt.grid(axis="y")
plt.tight_layout()
plt.show()

#Let's choose and arbitrary three week period to see the differences between the
days of the week.
normal_filtered_DAU = daily_active_users[(daily_active_users["date"] >= "2016-04-
04") & (daily_active_users["date"] <= "2016-04-24")]
plt.figure(figsize=(16, 6))
plt.plot(normal_filtered_DAU["date"],normal_filtered_DAU["active_users"],color="bla
ck", marker="o", linestyle="--")
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%a"))
plt.gca().xaxis.set_major_locator(mdates.DayLocator())
plt.title("Daily Active Users (April 4 - April 24)")
plt.ylabel("Number of Users")
plt.xlabel("Day of the Week")
plt.grid(True)
plt.xlim(normal_filtered_DAU["date"].min(), normal_filtered_DAU["date"].max())
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

#used for getting exact values in the document
normal_weekdays = normal_filtered_DAU[normal_filtered_DAU["date"].dt.weekday < 5]
normal_weekends = normal_filtered_DAU[normal_filtered_DAU["date"].dt.weekday >= 5]
normal_avg_difference = normal_weekends["active_users"].mean() -
normal_weekdays["active_users"].mean()

#Doing the same for summer
```

```
summer_filtered_DAU = daily_active_users[(daily_active_users["date"] >= "2016-07-
04") & (daily_active_users["date"] <= "2016-07-24")]
plt.figure(figsize=(16, 6))
plt.plot(summer_filtered_DAU["date"],summer_filtered_DAU["active_users"],color="bla
ck", marker="o", linestyle="--")
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%a"))
plt.gca().xaxis.set_major_locator(mdates.DayLocator())
plt.title("Daily Active Users (July 4 - July 24)")
plt.ylabel("Number of Users")
plt.xlabel("Day of the Week")
plt.grid(True)
plt.xlim(summer_filtered_DAU["date"].min(), summer_filtered_DAU["date"].max())
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

#used for getting exact values in the document
summer_weekdays = summer_filtered_DAU[summer_filtered_DAU["date"].dt.weekday < 5]
summer_weekends = summer_filtered_DAU[summer_filtered_DAU["date"].dt.weekday >= 5]
summer_avg_difference = summer_weekends["active_users"].mean() -
summer_weekdays["active_users"].mean()


#Sales analysis

#Plotting registered users per country
geographic_spread = pd.read_sql_query("SELECT country_code, COUNT(account_id) AS
total_users FROM account GROUP BY country_code ORDER BY total_users DESC",
database_connection)
geo_head=30
geographic_spread_select=geographic_spread.head(geo_head) #Selecting top 30 for
better visibility
plt.figure(figsize=(16, 6))
bars=plt.bar(geographic_spread_select["country_code"],
geographic_spread_select["total_users"], width=0.8)
labels = geographic_spread_select["total_users"]
plt.title(f"Registered Users per Country (top {geo_head})")
plt.ylabel("Number of Users")
plt.xlabel("Country Code")
plt.grid(axis="y")
plt.bar_label(bars, labels=labels, label_type="edge", fontsize=8,padding=5)
plt.tight_layout()
plt.show()

#used for getting exact values in the document
#Calculating major countries' share of players
total_accounts = geographic_spread['total_users'].sum()
china_accounts = geographic_spread[geographic_spread['country_code'] ==
'CN']['total_users'].values[0]
usa_accounts = geographic_spread[geographic_spread['country_code'] ==
'US']['total_users'].values[0]
```

```
percent_from_china = (china_accounts / total_accounts) * 100
percent_from_usa = (usa_accounts / total_accounts) * 100
percent_from_other = 100 - percent_from_china - percent_from_usa
total_countries=geographic_spread.shape[0]

#used for getting exact values in the document
average_spending_globally = pd.read_sql_query("SELECT AVG(iap_price_usd_cents) as
avg_price FROM iap_purchase",database_connection)

#Plotting the total paying users per country using the pareto principle
spending_users_per_countries = pd.read_sql_query("SELECT account.country_code,
COUNT(account.account_id) AS total_users FROM account INNER JOIN iap_purchase USING
(account_id) GROUP BY country_code ORDER BY total_users DESC", database_connection)

spending_users_per_countries["cumulative_percent"] =
spending_users_per_countries["total_users"].cumsum() /
spending_users_per_countries["total_users"].sum() * 100
pareto_index =
spending_users_per_countries[spending_users_per_countries["cumulative_percent"] >=
80].index[0]
spending_users_per_countries_pareto =
spending_users_per_countries.iloc[:pareto_index + 1].copy()
spending_users_per_countries_other = spending_users_per_countries.iloc[pareto_index
+ 1:].copy()
spending_users_per_countries_other["country_code"] = "Others"
spending_users_per_countries_other["total_users"] =
spending_users_per_countries_other["total_users"].sum()
spending_users_per_countries_all =
pd.concat([spending_users_per_countries_pareto,spending_users_per_countries_other],
ignore_index=True)

plt.figure(figsize=(16, 6))
bars=plt.bar(spending_users_per_countries_all["country_code"],
spending_users_per_countries_all["total_users"], alpha=0.8)
plt.xlabel("Country Code")
plt.ylabel("User Count")
labels = spending_users_per_countries_all["total_users"]
plt.title("Total Spending Users per Country - Using the Pareto Principle")
plt.grid(axis="y")
plt.bar_label(bars, labels=labels, label_type="edge", fontsize=8,padding=5)
plt.tight_layout()
plt.show()

#used for getting exact values in the document
user_spending_percent=spending_users_per_countries['total_users'].sum()/total_accou
nts*100
total_spending_users=spending_users_per_countries['total_users'].sum()

#Plotting the total spending per country using the pareto principle
```

```
total_spending_per_countries = pd.read_sql_query("SELECT account.country_code,
SUM(iap_purchase.iap_price_usd_cents) AS total_spending FROM account INNER JOIN
iap_purchase ON account.account_id=iap_purchase.account_id GROUP BY country_code
ORDER BY total_spending DESC", database_connection)

total_spending_per_countries["cumulative_percent"] =
total_spending_per_countries["total_spending"].cumsum() /
total_spending_per_countries["total_spending"].sum() * 100
total_spending_per_countries_pareto =
total_spending_per_countries[total_spending_per_countries["cumulative_percent"] <=
80]
total_spending_per_countries_other = pd.DataFrame({"country_code": ["Others"],
                        "total_spending":
[total_spending_per_countries[total_spending_per_countries["cumulative_percent"] >
80]["total_spending"].sum()],
                        "cumulative_percent": [100]})
total_spending_per_countries_all = pd.concat([total_spending_per_countries_pareto,
total_spending_per_countries_other], ignore_index=True)

plt.figure(figsize=(16, 6))
bars=plt.bar(total_spending_per_countries_all["country_code"],
total_spending_per_countries_all["total_spending"], alpha=0.8)
plt.xlabel("Country Code")
plt.ylabel("Spending by Country")
plt.title("Total Spending by Country - Using the Pareto Principle")
formatter = ticker.StrMethodFormatter("{x:,}")  # Format long values for better
visibility
plt.gca().yaxis.set_major_formatter(formatter)
plt.grid(axis="y")
labels = ['{:,}'.format(val) for val in
total_spending_per_countries_all["total_spending"]]

plt.bar_label(bars, labels=labels, label_type="edge",
fontsize=8,padding=5,fmt=formatter)
plt.tight_layout()
plt.show()


#Plotting the average spending per person per country using the pareto principle
based on total spending per country
merged_spending= pd.merge(total_spending_per_countries_all,
spending_users_per_countries[['country_code', 'total_users']], on='country_code',
how='left')
merged_spending['total_users'] = merged_spending['total_users'].fillna(0) #filling
empty with 0 to be able to add numbers
total_users_others =
spending_users_per_countries[~spending_users_per_countries['country_code'].isin(tot
al_spending_per_countries_all['country_code'])]['total_users'].sum()
merged_spending.loc[merged_spending['country_code'] == 'Others', 'total_users'] +=
total_users_others
```

```
merged_spending["average_spending"]=merged_spending["total_spending"]/merged_spendi
ng["total_users"]
sorted_data = merged_spending.sort_values('average_spending', ascending=False)

plt.figure(figsize=(16, 6))
bars=plt.bar(sorted_data["country_code"], sorted_data["average_spending"],
alpha=0.8)
plt.xlabel("Country Code")
plt.ylabel("Spending by Country")
plt.title("Average Spending Per Country per Person - Using the Pareto Principle
Based on Total Spending")
plt.grid(axis="y")
labels = ['{:.2f}'.format(val) for val in sorted_data["average_spending"]]
plt.bar_label(bars, labels=labels, label_type="edge", fontsize=8,padding=5)
plt.tight_layout()
plt.show()

#Showing all countries
merged_spending_all=pd.merge(total_spending_per_countries,
spending_users_per_countries[['country_code', 'total_users']], on='country_code',
how='left')
merged_spending_all["average_spending"]=merged_spending_all["total_spending"]/merge
d_spending_all["total_users"]
#I noticed there is some data with no country, so I will replace the empty
countries with "XX"
merged_spending_all=merged_spending_all.fillna("XX")

plt.figure(figsize=(10, 20))
bars=plt.barh(merged_spending_all["country_code"],
merged_spending_all["average_spending"], alpha=0.8)
plt.ylabel("Country Code")
plt.xlabel("Spending by Country")
plt.title("Average Spending Per Country per Person\nRanked by Total Spending")
plt.grid(axis="x")
labels = ['{:.2f}'.format(val).rstrip('0').rstrip('.') if val % 1 else
'{:.0f}'.format(val) for val in merged_spending_all["average_spending"]]
plt.bar_label(bars, labels=labels, label_type="edge", fontsize=8,padding=5)
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()

database_connection.close()
```