supercell data intern test

subham das

November 2023

1 description of datasets

The database contains 3 datasets : account, iap_purchase, account_date_session.

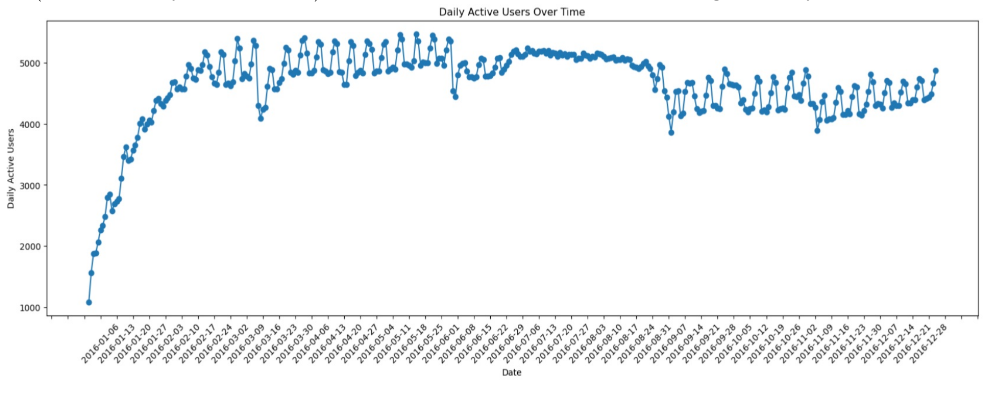
account dataset contains basic user information attributes related to the account, such as account_id, created_time, created_device, created_platform , country_code and created_app_store_id.

iap_purchase dataset contains attributes regarding purchase information of a given account id, which allows to link an account id to the app store id and other related attributes . The attributes here are: account_id, created_time,package_id_hash, iap_price_usd_cents,app_store_id

account_date_session dataset contains information regarding the online activities of a user on their account such as the number of sessions and time duration of the sessions. the attributes here are: account_id, date, session_count, session_duration_sec.

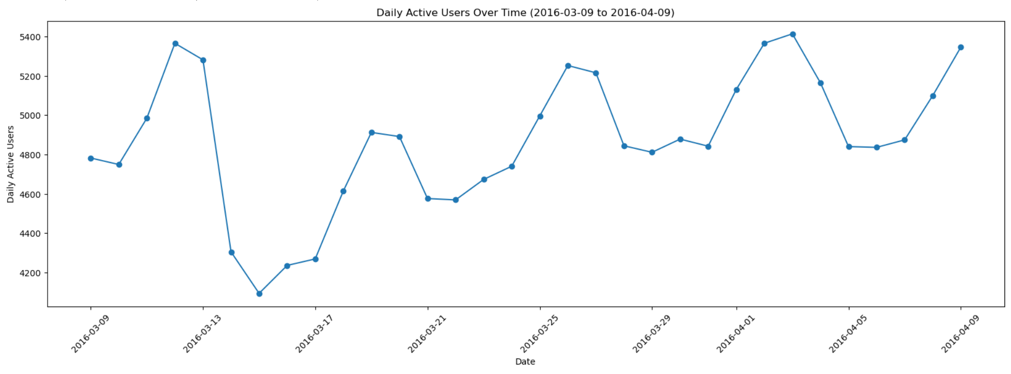
2 Analysis of daily active users

- In our case, I assume that DAU is the sum of the number of unique users(identified by account_id) with at least 1 session count in a given day.

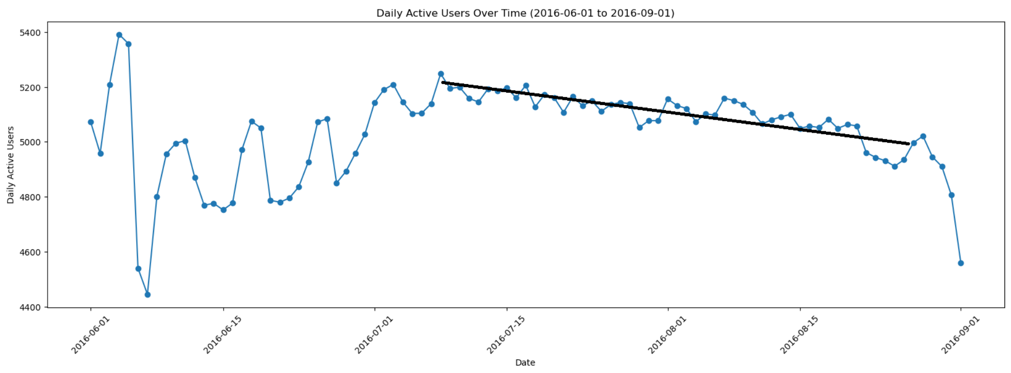


- Observing the data, we can see that the initial rise of players up until 2016-02-17 is most likely due to the launch of a new game and the number of players saturates and remains about the same after that date.

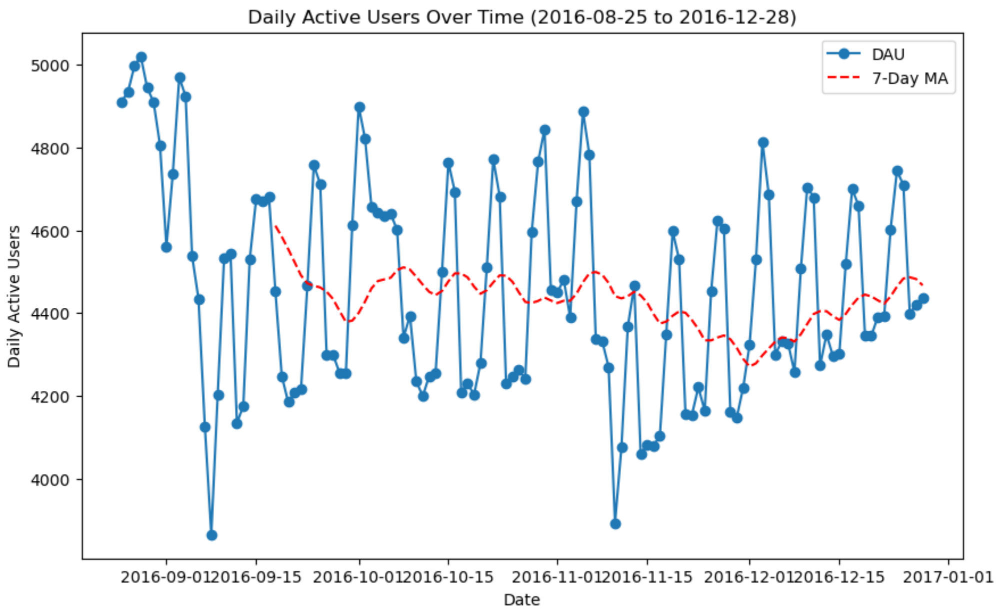
- For most of the data we can see a constant zig-zag or up-down pattern except for the initial first few weeks after the launch of the game and in between the dates from 2016-06-22 to 2016-08-24.
  This can be explained as follows: Most of the people who play this game are children under the age of 18 who are school going people and a lower percentage of other age groups.If we zoom in one such pattern, we can observe that consistently 3 peak days, Friday is when the DAU starts increasing and DAU reaches its peak on Saturdays and Sundays over the weekend when people of this age group are usually free to play this game. The DAU then consistently drops for the remaining 4 days of the week(Schooldays/workdays) which makes sense.



- The relatively constant looking line(2016-06-22 to 2016-08-24) where the prevalent weekend spikes and weekday dips are absent can be explained as follows :
  2016-06-22 to 2016-08-24, the months in these dates June, August-September are the months of Summers where children usually receive month-long breaks for summer from school and thus they have more time to play the game, therefore the data does not spike on the weekends because weekends and weekdays are usually the same on vacation.
  The graph shows a slight negative slope (black line) during this period, which may indicate that users slowly lose interest in the game after playing for a long time in summer.

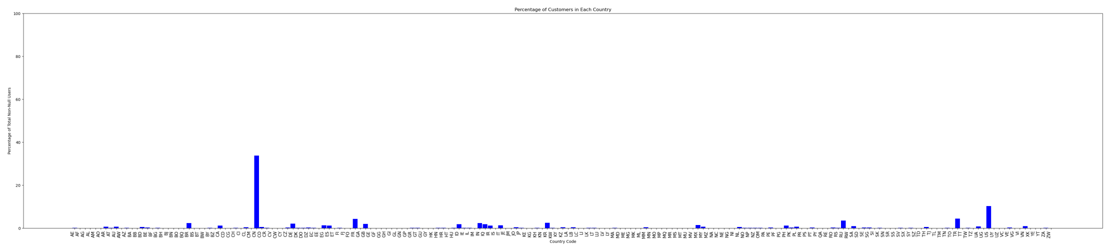


- there is a slight drop in the avg DAU of 2016-11-02 to 2016-12-28 from 2016-08-31 to 2016-11-02. This could be due to various factors such as game updates, seasonal events or technical issues.

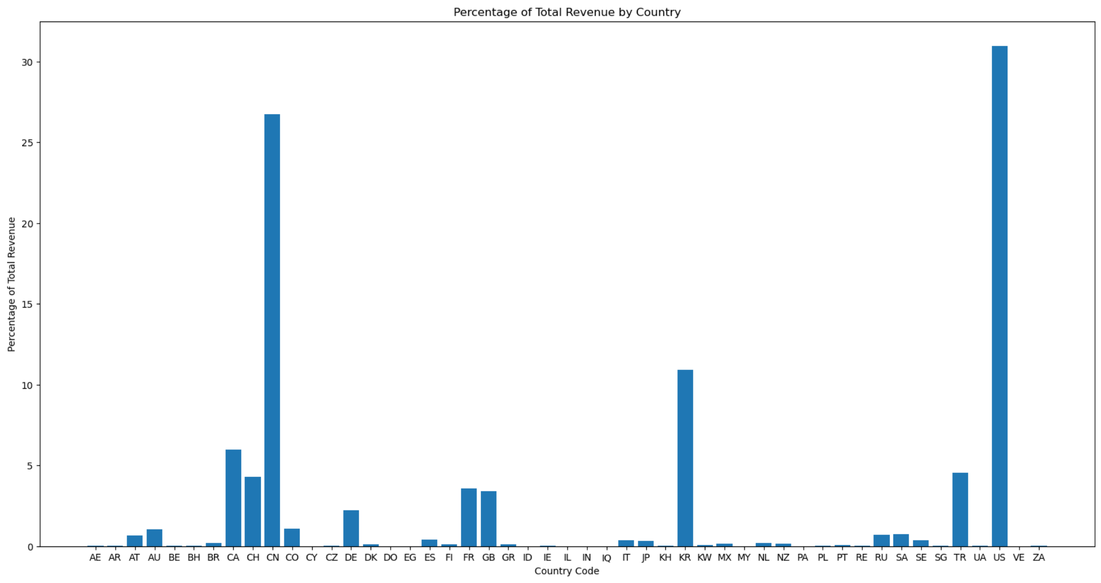


3 Sales analysis:

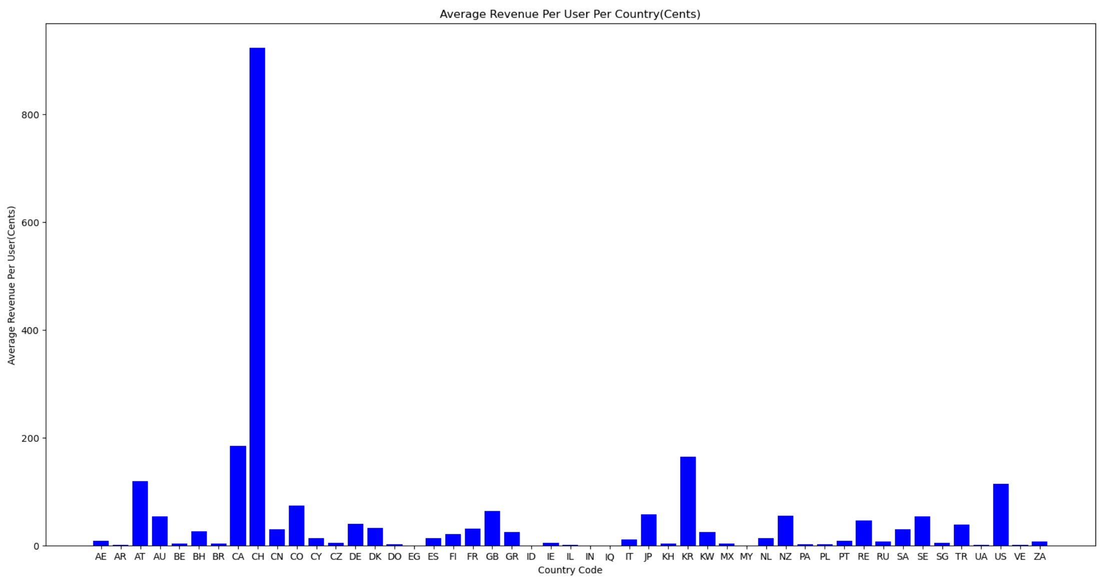
- USER SPLIT:
  We can observe that about 50% of the user traffic comes from China(38%) and USA(10%) combined with the remaining countries bringing in about 2-4 % of the total traffic.



- REVENUE SPLIT:
  Here, we can observe that only about 49 countries out of the users from 189 countries who play this game pay for any in game content.
  Around 30% of total revenue comes from USA even though its users constitute only about 10% of total users, China comes in next with around 27% of total revenue compared to its 38% user split. Combined, USA and China account for more than 50% of the total revenue split. Surprisingly South Korea accounts for about 10% of net revenue even though it accounts for only 2.5% of total users.



- AVG REVENUE PER USER PER REGION:
  Switzerland has an unusually high avg revenue per user of 923 Cents which is about 5 times more than the avg in Canada which is 184 Cents and is next highest.
  In general, it can be observed that usually, users form more developed western countries are usually the most likely to spend above 100 cents on the game.



- CONCLUSION:
  Generally speaking, we can conclude that a vast majority of players are from China, USA and India, who are likely minors and are students (evident from the peaks and lows), the peaks and lows are negligible during the summer months when users are on vacation. Almost half of the traffic and revenue comes from China and USA, with china having more traffic and less revenue and vice-versa for USA. The other countries individually constitute small amounts of traffic and revenue(except Korea 10%).
  The avg revenue per user per region is surprising since switzerland with only 198 users and 182774 Cents total revenue is by far the highest with next coming in at Canada with 184 Cents

4 Appendix

A DAU code

```
import sqlite3
import matplotlib.pyplot as plt
from datetime import datetime
import pandas as pd

from matplotlib.dates import DateFormatter, WeekdayLocator
connection = sqlite3.connect('sample.sqlite')
cursor = connection.cursor()
```

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")

# Fetch all the table names
tables = cursor.fetchall()

# Print the table names
for table in tables:
    print(table[0])
# Query to retrieve Daily Active Users (DAU) with respect to dates
cursor.execute("SELECT date, COUNT(DISTINCT account_id) AS dau FROM account_date

# Fetch all the data
data = cursor.fetchall()

# Separate the dates and DAU values for plotting
dates, dau_values = zip(*data)

# Plotting the data
fig, ax = plt.subplots(figsize=(20, 6))
ax.plot(dates, dau_values, marker='o', linestyle='-')

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('Daily Active Users')
plt.title('Daily Active Users Over Time')

# Show only one week intervals on the x-axis
ax.xaxis.set_major_locator(WeekdayLocator(interval=1))
plt.xticks(rotation=45)

# Show the plot
plt.show()
connection.close
```

B Zoomed in DAU plot for peaks and lows

```
connection = sqlite3.connect('sample.sqlite')
cursor = connection.cursor()
# Query to retrieve DAU data for the specified date range
cursor.execute("SELECT date, COUNT(DISTINCT account_id) AS dau FROM account_date

# Fetch all the data
data = cursor.fetchall()
```

```
# Unpack the data
dates, dau_values = zip(*data)

# Convert dates to datetime objects
dates = [datetime.strptime(date, "%Y-%m-%d") for date in dates]


# Plotting the data with a larger figure size
fig, ax = plt.subplots(figsize=(20, 6))
ax.plot(dates, dau_values, marker='o', linestyle='-')

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('Daily Active Users')
plt.title('Daily Active Users Over Time (2016-03-09 to 2016-04-09)')
plt.xticks(rotation=45)

# Show the plot
plt.show()
connection.close()
```

C DAU changes for 2016-06-01 to 2016-09-01 Summer

```
connection = sqlite3.connect('sample.sqlite')
cursor = connection.cursor()
# Query to retrieve DAU data for the specified date range
cursor.execute("SELECT date, COUNT(DISTINCT account_id) AS dau FROM account_date

# Fetch all the data
data = cursor.fetchall()

# Unpack the data
dates, dau_values = zip(*data)

# Convert dates to datetime objects
dates = [datetime.strptime(date, "%Y-%m-%d") for date in dates]


# Plotting the data with a larger figure size
fig, ax = plt.subplots(figsize=(20, 6))
ax.plot(dates, dau_values, marker='o', linestyle='-')

# Adding labels and title
```

```
plt.xlabel('Date')
plt.ylabel('Daily Active Users')
plt.title('Daily Active Users Over Time (2016-06-01 to 2016-09-01)')
plt.xticks(rotation=45)

# Show the plot
plt.show()

connection.close()
```

D DAU changes 2016-08-25 to 2016-12-28 end of year

```
connection = sqlite3.connect('sample.sqlite')
cursor = connection.cursor()
# Query to retrieve DAU data for the specified date range
cursor.execute("SELECT date, COUNT(DISTINCT account_id) AS dau FROM account_date

# Fetch all the data
data = cursor.fetchall()

# Unpack the data
dates, dau_values = zip(*data)

# Convert dates to datetime objects
dates = [datetime.strptime(date, "%Y-%m-%d") for date in dates]
# Create a DataFrame for easier manipulation
df = pd.DataFrame({'Date': dates, 'DAU': dau_values})

# Calculate the 7-day moving average
df['MA'] = df['DAU'].rolling(window=25).mean()

# Plotting the data with a larger figure size
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(dates, dau_values, marker='o', linestyle='-', label='DAU')
ax.plot(dates, df['MA'], linestyle='--', label='7-Day MA', color='red')

# Adding labels, title, and legend
plt.xlabel('Date')
plt.ylabel('Daily Active Users')
plt.title('Daily Active Users Over Time (2016-08-25 to 2016-12-28)')
plt.legend()

# Show the plot
```

```
plt.show()

connection.close()
```

E USER SPLIT

```
connection = sqlite3.connect('sample.sqlite')
cursor = connection.cursor()

# SQL query to get the number of customers in each country
sql_query = '''
    SELECT
        country_code,
        COUNT(DISTINCT account_id) AS number_of_customers
    FROM
        account
    WHERE
        country_code IS NOT NULL
    GROUP BY
        country_code;
'''

# Execute the query
cursor.execute(sql_query)

# Fetch all the results

results = cursor.fetchall()
check=0
# Print the results
for row in results:
    check=check+1
    country_code, number_of_customers = row
    print(f"Country Code: {country_code}, Number of Customers: {number_of_custom
print(check)

# Extract data for plotting
country_codes, customer_counts = zip(*results)



# Enlarge the figure size
plt.figure(figsize=(50, 10))
```

```
# Calculate the percentage of total non-null users
total_users = 112685
percentages = [count / total_users * 100 for count in customer_counts]

# Plotting
plt.bar(country_codes, percentages, color='blue', width=0.9)
plt.xlabel('Country Code')
plt.ylabel('Percentage of Total Non-Null Users')
plt.title('Percentage of Customers in Each Country')
plt.ylim(0, 100)  # Set y-axis limits to 0-100 for percentages



# Rotate x-axis labels for better readability
plt.xticks(rotation=90, ha='right')
plt.tick_params(axis='x', which='major', labelsize=11)

plt.show()

connection.close()
```

F REVENUE SPLIT

```
connection = sqlite3.connect('sample.sqlite')
cursor = connection.cursor()
# Sample SQL query to join the account and iap_purchase tables
query = '''
    SELECT account.country_code, SUM(iap_purchase.iap_price_usd_cents) as total_r
    FROM account
    JOIN iap_purchase ON account.account_id = iap_purchase.account_id
    WHERE account.country_code IS NOT NULL
    GROUP BY account.country_code
'''

# Execute the query and fetch the results into a DataFrame
result_df = pd.read_sql_query(query, connection)

# Calculate the percentage of total revenue for each country
# Total number of countries
total_countries = result_df['country_code'].nunique()
print(f'Total number of countries: {total_countries}')
total_revenue = result_df['total_revenue'].sum()
result_df['percentage_revenue'] = (result_df['total_revenue'] / total_revenue) *
```

```
# Plotting the bar graph
# Enlarge the figure size
plt.figure(figsize=(20, 10))
plt.bar(result_df['country_code'], result_df['percentage_revenue'])
plt.xlabel('Country Code')
plt.ylabel('Percentage of Total Revenue')
plt.title('Percentage of Total Revenue by Country')

plt.show()

# Print total revenue per country# Print total revenue per country
print(result_df[['country_code', 'total_revenue']])

connection.close()
```

G AVG REVENUE PER USER PER REGION

```
# First code to get the number of customers in each country
connection_1 = sqlite3.connect('sample.sqlite')
cursor_1 = connection_1.cursor()

sql_query_1 = '''
    SELECT
         country_code,
        COUNT(DISTINCT account_id) AS number_of_customers
    FROM
         account
    WHERE
         country_code IS NOT NULL
    GROUP BY
         country_code;
'''

cursor_1.execute(sql_query_1)
results_1 = cursor_1.fetchall()

# Create a DataFrame from the results
customers_df = pd.DataFrame(results_1, columns=['country_code', 'number_of_custo

# Second code to get the total revenue per country
connection_2 = sqlite3.connect('sample.sqlite')
cursor_2 = connection_2.cursor()
```

```
query_2 = '''
    SELECT account.country_code, SUM(iap_purchase.iap_price_usd_cents) as total_r
    FROM account
    JOIN iap_purchase ON account.account_id = iap_purchase.account_id
    WHERE account.country_code IS NOT NULL
    GROUP BY account.country_code
'''

result_df = pd.read_sql_query(query_2, connection_2)

# Merge the two DataFrames on 'country_code'
merged_df = pd.merge(result_df, customers_df, on='country_code', how='inner')

# Calculate average revenue per user per country
merged_df['avg_revenue_per_user'] = merged_df['total_revenue'] / merged_df['num

# Print the final result
print(merged_df[['country_code', 'avg_revenue_per_user']])


# Plot the results
plt.figure(figsize=(20, 10))
plt.bar(merged_df['country_code'], merged_df['avg_revenue_per_user'], color='blu
plt.xlabel('Country Code')
plt.ylabel('Average Revenue Per User(Cents)')
plt.title('Average Revenue Per User Per Country(Cents)')
plt.show()
```