Nir Ben Eliahu – DAU Project

An SQL Project in Jupyter Notebook as the Frontend.
In order to produce **visualizations**, a combination of both Python and R was used

Tableau Interactive Dashboard

[DAU Interactive Dashboard (https://public.tableau.com/views/DAUStatus/DAUStatus?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link)](https://public.tableau.com/views/DAUStatus/DAUStatus?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link)

The project includes the following steps:

1. *Establish a connection between jupyter and MSSQL:*
   A. Create a connection through an Open Database Connectivity (ODBC), Define an API for accessing the MSSQL database.
   B. Load the extensions required to run SQL queries in jupyter.
   C. Load the extensions required to run R syntax in jupyter.
2. *Data Validation:*
   A. Import data from the excel source file into jupyter.
   B. Data investigation: Understanding its structure.
   C. Fix incompatiable data entries.
3. *Data Processing:*
   A. Create tables in the database for the data (A table, as the data is consists of 1 table).
   B. Load the validated data from jupyter to MSSQL.
4. *Data Manipulation, Analysis and Visualization:*
   Define questions about the behavior of the product, reflected by the data.

   ```
       - Manipulate the data to produce the best data presentation form.
       - Create SQL View for Tableau application.
       - Visualize the data and answer the questions regarding the product. <br
   >
   ```

5. **Create a Dashboard with Tableau.**

SQL syntax and strategies that are used in the project:

1. Finding sequence of events using windowed Row_Number.
2. dynamic (DECLARE, SET, EXECUTE) PIVOT with FOR XML PATH.
3. multiple CTEs
4. Recursive CTE and a FOR XML PATH Technique
5. Subqueries
6. Multiple inner joins inside a multiple CTEs and a Recursive CTE query
7. Casting and connecting date (dates, str)
8. Creating VIEWS

How Python is used in the project:

1. loading excel data file.
2. investing the data and converting incompatiable data entries with a def function
3. visualizations with seaborn.

How R is used in the project:

1. Visualizations with ggplot2

In [1]:
```
# Import python libraries
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline

# Import sql Extensions librarys
import sqlalchemy
import pyodbc
from sqlalchemy_utils import database_exists, create_database
```

1. Establish a connection between jupyter and MSSQL

1. A. Create a connection througe an Open Database Connectivity (ODBC), Define an API for accessing the MSSQL database

In [2]:
```
SERVER = "DESKTOP-NIR\SQLEXPRESS"
DATABASE = "Nir Ben Eliahu DAU Project"
DRIVER = "SQL Server"
USERNAME = "nir"
PASSWORD = "nir123"
DATABASE_CONNECTION = f'mssql://{USERNAME}:{PASSWORD}@{SERVER}/{DATABASE}?driver={DRIVER
```

In [3]:
```
engine = sqlalchemy.create_engine(DATABASE_CONNECTION)

# if the database is does not exist, sqlalchemy will create the database
if not database_exists(DATABASE_CONNECTION):
    create_database(DATABASE_CONNECTION)
else:
    engine.connect()

# the connection command and variable
connection = engine.connect()
```

1. B. Load the extensions required to run SQL queries in jupyter

In [4]:
```
%load_ext sql
%sql mssql+pyodbc://nir:nir123@mssql
#parameters: username=nir password=nir123
```

1. C. Load the extensions required to run R syntax in jupyter

In [5]:
```
%load_ext rpy2.ipython
```

```
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\rpy2\robjects\packages.
py:366: UserWarning: The symbol 'quartz' is not in this R namespace/package.
  warnings.warn(
```

2. Data Import and Validation

2. A. Import the data that was provided in an excel file into jupyter

In [6]:
```
# Note that the sheet from the excel file is named 'DAU_Data'. There is another sheet ca
DAU_Data_df = pd.read_excel("DAU_Data.xlsx", sheet_name = "DAU_Data")
```

2. B. Data investigation: Understanding its structure and fix incompatiable data

In [7]:
```
DAU_Data_df.head(3)
```

Out[7]:

	Calc_date	DAU	Platform_id	Version	Tier
0	2020-01-02	47613.0	0	3.34	2
1	2020-01-02	19196.0	1	3.35.1	1
2	2020-01-02	9239.0	6	3.35.1	2

In [8]:
```
DAU_Data_df.tail(3)
```

Out[8]:

	Calc_date	DAU	Platform_id	Version	Tier
5733	2020-06-29	13799.0	9	3.61.1	3
5734	2020-06-29	74796.0	1	3.61.1	3
5735	2020-06-29	14960.0	0	3.60.1	5

The data:

1. Six months of data, starts from January 2020 to the end of June 2020, i.e. 02/01/2020-29/06/2020.
2. The Dates are on the level of DAY.
3. Every date has a data entry of DAU, Platform_id, Version, and Tier.
4. There are several combinations of the data, and each has a data entry, causing data recurrence for some items.

> Add a note to remember: There are several values for the same date. This subject needs to be addressed if values are visualized over time.

In [9]:
```
DAU_Data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5736 entries, 0 to 5735
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Calc_date    5736 non-null   datetime64[ns]
 1   DAU          5736 non-null   float64
 2   Platform_id  5736 non-null   int64
 3   Version      5736 non-null   object
 4   Tier         5736 non-null   object
dtypes: datetime64[ns](1), float64(1), int64(1), object(2)
memory usage: 224.2+ KB
```

Summary of the data investigation:

1. Calc_date is classified as datetime.
2. DAU is classified as float type.
3. Platform_id is classified as int, while it might be better assigned as a category.
4. Version is classified as an object type.
5. Tier is classified as an object type while it appears as int type, suggesting a str value among the values.
6. It appears as there are no nulls, 5736 non-nulls out of 5736 entries. but a 'value_count()' will identify more info

In [10]:
```
DAU_Data_df["Calc_date"].value_counts()
```

Out[10]:
```
2020-02-11    47
2020-01-15    47
2020-01-09    44
2020-02-08    42
2020-03-19    42
              ..
2020-05-13    23
2020-06-02    23
2020-06-03    23
2020-06-09    23
2020-06-04    21
Name: Calc_date, Length: 180, dtype: int64
```

In [11]:
```
DAU_Data_df["DAU"].value_counts()
```

Out[11]:
```
8594.0     4
8141.0     3
8915.0     3
9263.0     3
12242.0    3
          ..
10041.0    1
45146.0    1
18371.0    1
27836.0    1
14960.0    1
Name: DAU, Length: 5341, dtype: int64
```

In [12]:
```
DAU_Data_df["Platform_id"].value_counts()
```

Out[12]:
```
2    2160
1    1441
0     876
9     573
3     539
6     147
Name: Platform_id, dtype: int64
```

Summary

1. Platforms are classified 0,1,2,3,6,9.
2. All of the data has Platform_id classification

In [13]:
```
DAU_Data_df["Version"].value_counts()
```

Out[13]:
```
3.39      272
3.38      209
3.5       196
3.48.1    171
3.46      168
         ... 
3.54.2     13
3.34       10
3.34.2      8
3.53        5
(null)      1
Name: Version, Length: 78, dtype: int64
```

Summary

1. Versions varie from 3.39 to 3.53.
2. There is 1 string of (null) entrie.

In [14]:
```
DAU_Data_df["Tier"].value_counts()
```

Out[14]:
```
2         1761
3         1561
4         1136
5          649
1          561
6           64
(null)       4
Name: Tier, dtype: int64
```

Summary

1. Tiers are classified from 1-6.
2. There are 4 strings of (null) entry.

Summary of the data investigation:

1. Version is classified as an object type because it has a str (object) that represents null values, i.e. (null) entry.
2. Tier is classified as an object type because it has a str (object) that represents null values, i.e. (null) entry.

2. C. Fix incompatiable data

Define a function to replace the (null) str to a str that makes more sense i.e. NON-RANKED

In [15]:
```
# Define a functione to  replace the (null) str to a NON-RANKED str
def str_to_str(series):
    if series == "(null)":
        return 'NON-RANKED'
    else:
        return series
```

Use the defined function to clean the data

```
In [16]: DAU_Data_df["Tier"] = DAU_Data_df["Tier"].apply(str_to_str)
         DAU_Data_df["Version"] = DAU_Data_df["Version"].apply(str_to_str)
```

```
In [17]: DAU_Data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5736 entries, 0 to 5735
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Calc_date    5736 non-null   datetime64[ns]
 1   DAU          5736 non-null   float64
 2   Platform_id  5736 non-null   int64
 3   Version      5736 non-null   object
 4   Tier         5736 non-null   object
dtypes: datetime64[ns](1), float64(1), int64(1), object(2)
memory usage: 224.2+ KB
```

Address categorial and continuos data types:

Platform_id should be stored as object type.

DAU is INT and not Float

```
In [18]: DAU_Data_df["Platform_id"] = DAU_Data_df["Platform_id"].astype("str")
         DAU_Data_df["DAU"] = DAU_Data_df["DAU"].astype("int")
         DAU_Data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5736 entries, 0 to 5735
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Calc_date    5736 non-null   datetime64[ns]
 1   DAU          5736 non-null   int32
 2   Platform_id  5736 non-null   object
 3   Version      5736 non-null   object
 4   Tier         5736 non-null   object
dtypes: datetime64[ns](1), int32(1), object(3)
memory usage: 201.8+ KB
```

3. Data Processing

3. A. Create tables in the database for the data

In [19]:
```
%%sql

IF OBJECT_ID('Nir Ben Eliahu DAU Project..DAU_Data') IS NOT NULL
DROP TABLE dbo.DAU_Data

CREATE TABLE stock
(
Calc_date datetime,
DAU int,
Platform_id varchar(max),
Version varchar(max),
Tier varchar(max)
);
```

```
 * mssql+pyodbc://nir:***@mssql
(pyodbc.ProgrammingError) ('42S02', "[42S02] [Microsoft][SQL Server Native Client 11.0]
[SQL Server]Cannot drop the table 'dbo.DAU_Data', because it does not exist or you do n
ot have permission. (3701) (SQLExecDirectW)")
[SQL: IF OBJECT_ID('Nir Ben Eliahu DAU Project..DAU_Data') IS NOT NULL
DROP TABLE dbo.DAU_Data

CREATE TABLE stock
(
Calc_date datetime,
DAU int,
Platform_id varchar(max),
Version varchar(max),
Tier varchar(max)
);]
(Background on this error at: https://sqlalche.me/e/14/f405) (https://sqlalche.me/e/14/
f405))
```

3. B. Load the validated-data from jupyter to MSSQL

In [20]:
```
DAU_Data_df.to_sql(name="DAU_Data", schema="dbo", con=engine, if_exists="replace", index
```

In [21]:
```
%%sql
USE "Nir Ben Eliahu DAU Project"
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[21]:
```
[]
```

Create a Variable called "DAU_TABLE" with a predefined query and a connection string (connection to MSSQL), to get all of the data in DAU Table from MSSQL
With the DAU_TABLE variable, we can use our DAU table inside Jupyter notebook.

```
In [22]: DAU_TABLE_query = """

         SELECT * FROM DAU_Data

         """

         DAU_TABLE = pd.read_sql_query(DAU_TABLE_query, connection)
         DAU_TABLE
```

```
Out[22]:
```

	Calc_date	DAU	Platform_id	Version	Tier
0	2020-01-02	47613	0	3.34	2
1	2020-01-02	19196	1	3.35.1	1
2	2020-01-02	9239	6	3.35.1	2
3	2020-01-02	82682	0	3.34	3
4	2020-01-02	65444	1	3.35.1	4
...	...	...	...	...	...
5731	2020-06-29	60383	0	3.60.1	3
5732	2020-06-29	17981	3	3.60.1	3
5733	2020-06-29	13799	9	3.61.1	3
5734	2020-06-29	74796	1	3.61.1	3
5735	2020-06-29	14960	0	3.60.1	5

5736 rows × 5 columns

4. Data Manipulation, Analysis, and Visualization

```
- Manipulate the data to produce the best data presentation form.
- Create an SQL View (to allow Tableau to connect to that view).
- Visualize the data and answer the questions regarding the product.
```

4. A. Define questions about the behavior of the product, reflected by the data

DAU:

1. How does the DAU change over time?
2. How does the DAU change over time, across the different Platforms?

Tiers:

1. What is a Tier?
2. What is a Platform?
3. Which is the largest to smallest (in terms of DAU) Platform?
4. What are the Tier ranks (in terms of DAU), in total time and by each month?
5. Does the Tiers rank varies between Platforms?

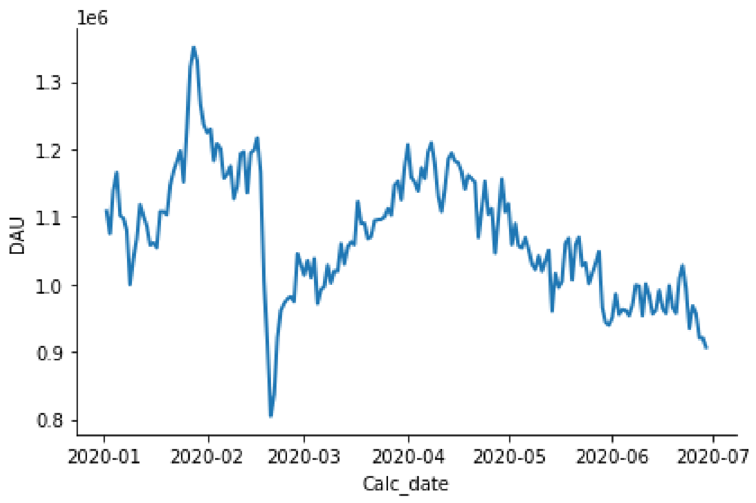
4. A1. DAU:

1. How does the DAU change over time?
2. How does the DAU change over time, across the different Platforms?

Start with a simple plot of DAU change over time:

In [23]:
```
sns.relplot(data = DAU_TABLE, x = "Calc_date", y = "DAU", kind = "line",
            ci = None,
            estimator = sum, #Estimator set to sum - because there are several data entr
            height = 4,
            aspect = 1.5,
            lw=2
           )
```

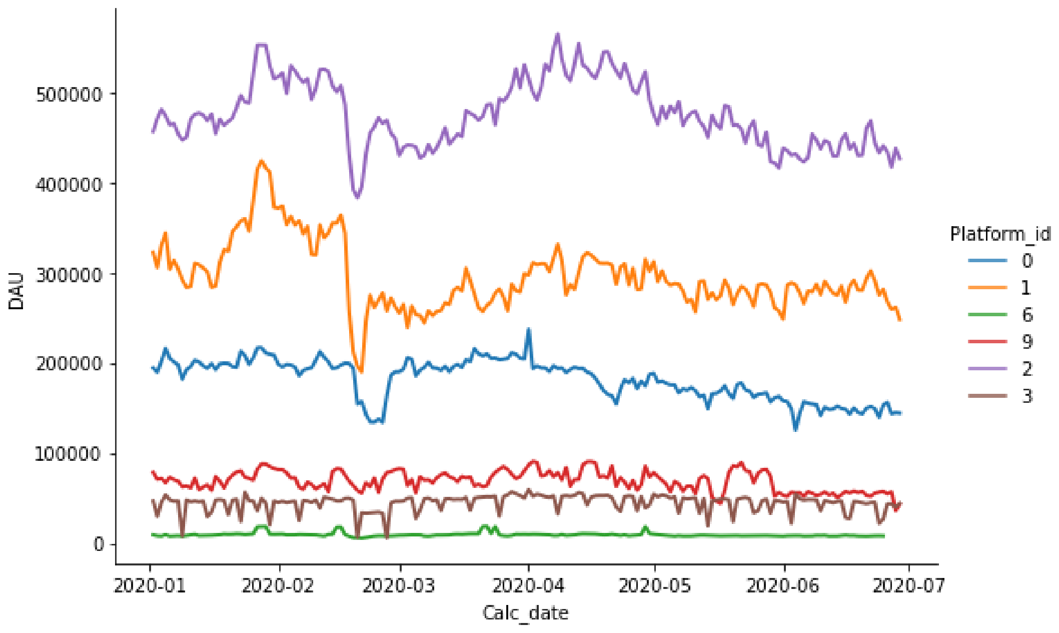
Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x167d21ffd90>
```



DAU have decreased dramatically between February and March
Does this change appear in all of the platforms?

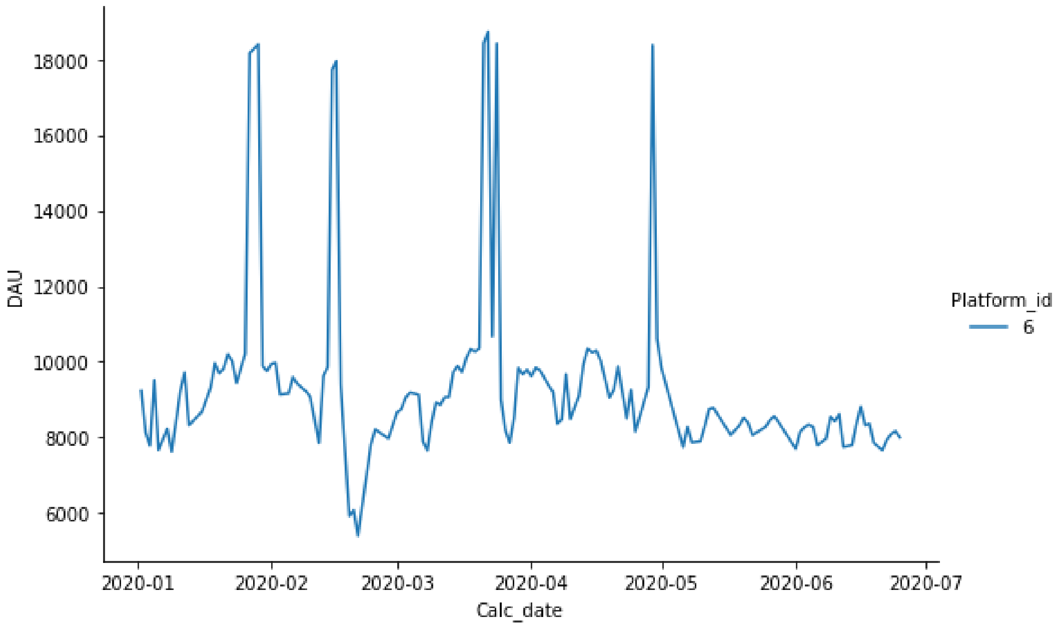
```
In [24]: sns.relplot(data = DAU_TABLE, x = "Calc_date", y = "DAU", kind = "line",
                     hue = "Platform_id",
                     palette = "tab10",
                     ci = None,
                     estimator = sum, #Estimator set to sum - because there are several values pe
                     height = 5,
                     aspect = 1.5,
                     lw=2
                     )
```

```
Out[24]: <seaborn.axisgrid.FacetGrid at 0x167d2d8ec40>
```

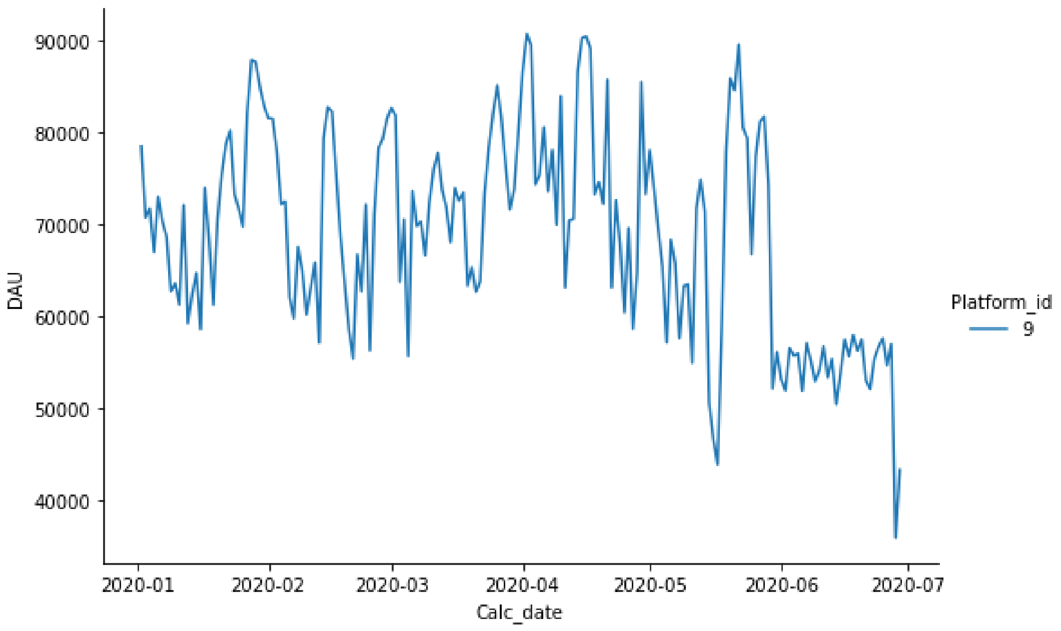


1. The DAU change over time is similar between Platforms 0,1,2 where the DAU is larger.
2. There is also, a similarity to platform 9, which has a smaller DAU than Platforms 0,1,2.
3. Platforms 3 and 6, each has a unique change of DAU over time.
   and also:
4. The decrease in DAU is observed in most of the platforms, but does it also occurs in platforms 6 and 9?
5. the DAU large data spread and the difference between the Platforms does not allow to determine this, so these platforms should be visualized individually.

```
In [25]: # DAU in platform 6
         sns.relplot(data = DAU_TABLE.loc[DAU_TABLE["Platform_id"]=="6"], x = "Calc_date", y = "D
                     hue = "Platform_id",
                     palette = "tab10",
                     ci = None,
                     estimator = sum, #Estimator set to sum - because there are several values pe
                     height = 5,
                     aspect = 1.5)

Out[25]: <seaborn.axisgrid.FacetGrid at 0x167d4908bb0>
```



```
In [26]: # DAU in platform 9
         sns.relplot(data = DAU_TABLE.loc[DAU_TABLE["Platform_id"]=="9"], x = "Calc_date", y = "D
                     hue = "Platform_id",
                     palette = "tab10",
                     ci = None,
                     estimator = sum, #Estimator set to sum - because there are several values pe
                     height = 5,
                     aspect = 1.5)

Out[26]: <seaborn.axisgrid.FacetGrid at 0x167d4967b80>
```



1. The DAU decrease is <u>also</u> observed in platform 6
2. The DAU decrease is <u>not</u> observed in platform 9

Filtering the dates where the DAU has decreased

It is important to also include the dates before and after the decrease, so the DAU decrease is observed.

In [27]:
```
# Define a functione to Filter the dates with the decrease in DAU and the bounding dates
def between_two_dates(df, date_column, start_date, end_date):

    return df[df[date_column].isin(pd.date_range(start_date, end_date))]
```

Testing the between_two_dates function

In [28]:
```
between_two_dates(DAU_TABLE, "Calc_date", "2020-02-01", "2020-03-01")
```

Out[28]:

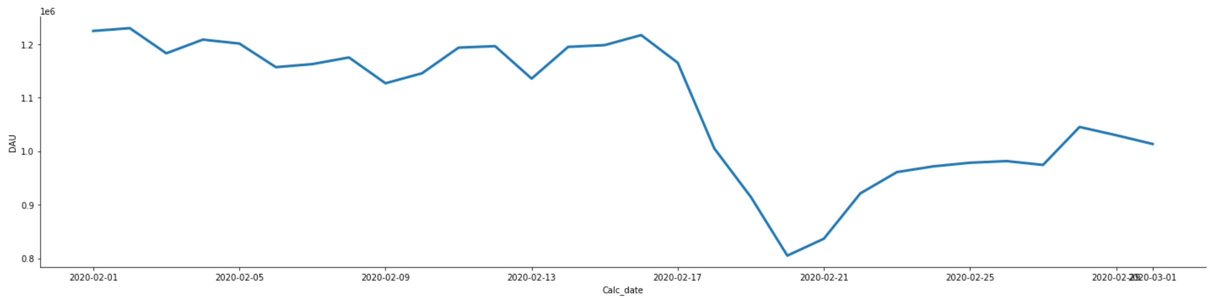
	Calc_date	DAU	Platform_id	Version	Tier
969	2020-02-01	11885	2	3.38	4
970	2020-02-01	105967	1	3.39	2
971	2020-02-01	8376	1	3.38	3
972	2020-02-01	44174	9	3.39	2
973	2020-02-01	9945	6	3.39	2
...	...	...	...	...	...
1971	2020-03-01	8817	9	3.43.2	4
1972	2020-03-01	23231	2	3.43.3	2
1973	2020-03-01	15155	3	3.43.1	2
1974	2020-03-01	17619	2	3.44	5
1975	2020-03-01	34574	2	3.44	1

1007 rows × 5 columns

Filtering the dates with the decrease in DAU and the bounding dates with still high DAU

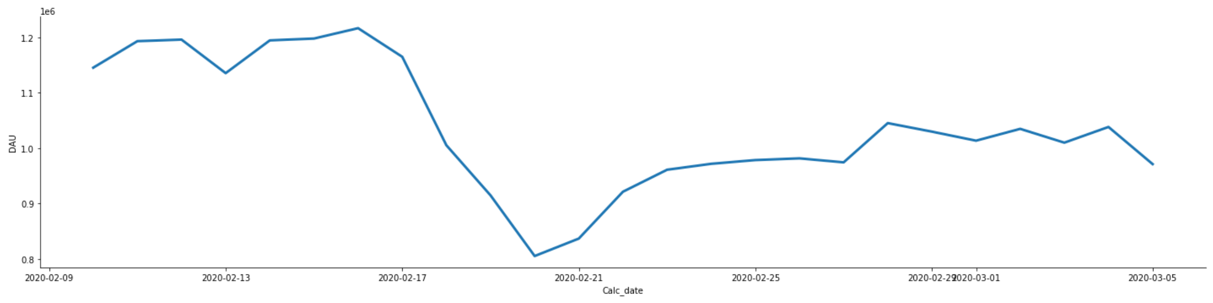
In [29]:
```
# Filter the dates with the decrease in DAU and the bounding dates with still high DAU
sns.relplot(data = between_two_dates(DAU_TABLE, "Calc_date", "2020-02-01", "2020-03-01")
            x = "Calc_date", y = "DAU", kind = "line",
            ci = None,
            estimator = sum, #Estimator set to sum - because there are several values pe
            height = 5,
            aspect = 4,
            lw=3
            )
```

Out[29]:
```
<seaborn.axisgrid.FacetGrid at 0x167d4908460>
```



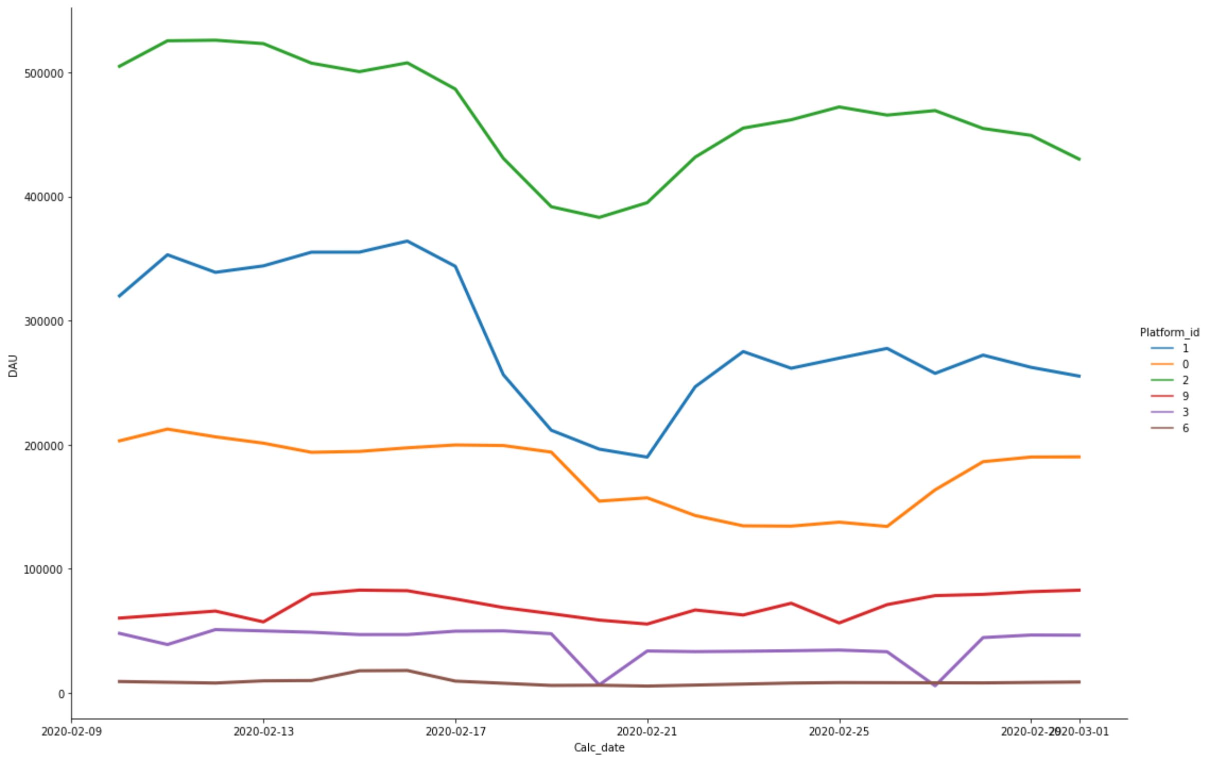
```
In [30]: # Filter the dates with the decrease in DAU and the bounding dates with still high DAU
         sns.relplot(data = between_two_dates(DAU_TABLE, "Calc_date", "2020-02-10", "2020-03-05")
                     x = "Calc_date", y = "DAU", kind = "line",
                     ci = None,
                     estimator = sum, #Estimator set to sum - because there are several values pe
                     height = 5,
                     aspect = 4,
                     lw=3
                     )
```

```
Out[30]: <seaborn.axisgrid.FacetGrid at 0x167d498fe50>
```



```
In [31]: sns.relplot(data = between_two_dates(DAU_TABLE, "Calc_date", "2020-02-10", "2020-03-01")
                     x = "Calc_date", y = "DAU", kind = "line",
                     hue = "Platform_id",
                     palette = "tab10",
                     ci = None,
                     estimator = sum, #Estimator set to sum - because there are several values pe
                     height = 10,
                     aspect = 1.5,
                     lw=3
                     )
```

```
Out[31]: <seaborn.axisgrid.FacetGrid at 0x167d49679a0>
```



In [32]:

```
%%sql
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'DAU_Decrease') DR
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[32]:

```
[]
```

In [33]:

```
%%sql
CREATE VIEW DAU_Decrease AS
SELECT *
FROM DAU_Data
WHERE CAST(Calc_date AS DATE) BETWEEN '2020-02-10' AND '2020-03-01'
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[33]:

```
[]
```

In [34]:

```
DAU_Decrease_query = """

SELECT *
FROM DAU_Decrease

"""

DAU_Decrease = pd.read_sql_query(DAU_Decrease_query, connection)
DAU_Decrease
```

Out[34]:

	Calc_date	DAU	Platform_id	Version	Tier
0	2020-02-10	90946	1	3.41.1	3
1	2020-02-10	48290	0	3.40.1	4
2	2020-02-10	31100	2	3.4	2
3	2020-02-10	35417	2	3.4	3
4	2020-02-10	58779	1	3.41.1	4
...	...	...	...	...	...
719	2020-03-01	8817	9	3.43.2	4
720	2020-03-01	23231	2	3.43.3	2
721	2020-03-01	15155	3	3.43.1	2
722	2020-03-01	17619	2	3.44	5
723	2020-03-01	34574	2	3.44	1

724 rows × 5 columns

In [35]:

```
DAU_Decrease.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 724 entries, 0 to 723
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Calc_date    724 non-null    datetime64[ns]
 1   DAU          724 non-null    int64
 2   Platform_id  724 non-null    object
 3   Version      724 non-null    object
 4   Tier         724 non-null    object
dtypes: datetime64[ns](1), int64(1), object(3)
memory usage: 28.4+ KB
```

<u>The preferred visualization of the data, to understand what happened during the DAU decrease is:</u>

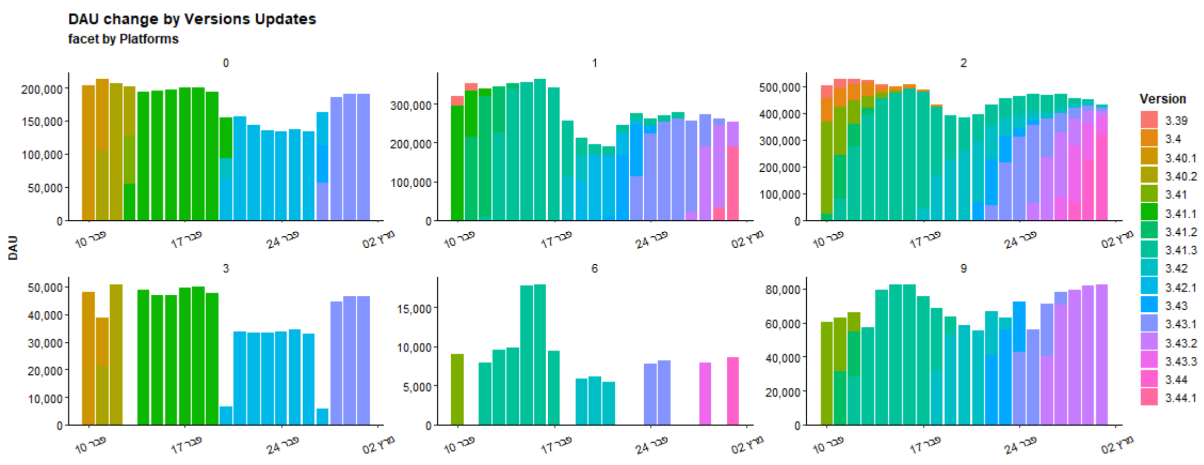
- A stacked bar-plot of DAU over time;
- Grouped by the Versions;
- and faceted by the Platforms.

This type of plot will show the DAU change over time, by Versions update sequence, and will show that sequence over each Platform.

Seaborn package of python has a limitation of faceting a stacked bar-plot. it is possible but requires specific rather than a general syntax.
Hence, The following plot is produced in R - ggplot2, which has this function built-in.

```
In [36]: %%R -i DAU_Decrease -w 40 -h 15 --units cm

         library(ggplot2)
         p <- ggplot(
           data = DAU_Decrease,
           aes(
             x=Calc_date,
             y=DAU,
             fill= as.factor(Version)
             )
           )

         bar <- geom_bar(
           stat="identity", # for count use stat="bin", i.e. for histograms
           position= "stack", # disable position dodge for stacked bars.
           #color = "black", size = 0.1
           )

         axis_gap_y <-scale_y_continuous(expand = c(0, 0, 0.05, 0), labels = scales::label_comma(
         axis_gap_x <-scale_x_continuous(expand = c(0, 0, 0.05, 0))

         facet_w <- facet_wrap(
           as.factor(DAU_Decrease$Platform_id),
           scales = "free") #("fixed", "free", "free_x", "free_y")

         titles <-   labs(
             y = "DAU",
             fill = "Version",
             title = "DAU change by Versions Updates",
             subtitle = "facet by Platforms"
             )


         white_theme <- theme(
           title = element_text(size = 12, color = "black", face = "bold"),
           axis.text.x = element_text(vjust= 0.5, angle = 22.5, color = "black", size = 11),
           axis.text.y = element_text(size = 11, color = "black"),
           axis.title.x = element_blank(),
           axis.title.y = element_text(size = 11, color = "black"),

           legend.position = "right", #(right, left, bottom, top, none)
           legend.title.align = NULL,
           legend.justification = "center",
           #legend.title=element_blank(),
           legend.text = element_text(size=11, color = "black"),
           legend.background = element_rect(fill = NA),
           legend.box.background = element_rect(fill = NA, color = NA),
           legend.key = element_rect(fill = NA),

           panel.border = element_rect(fill = NA, color = NA),
           panel.grid.major = element_line(color = NA),
           panel.grid.minor = element_line(color = NA),
           panel.background = element_rect(fill = NA, color = NA),

           strip.text.x = element_text(size = 11, color = "black", face = "plain"),
           strip.text.y = element_text(size = 11, color = "black", face = "plain"),
           strip.background =element_rect(fill= NA),

           axis.line = element_line(colour = "black")
         )

         p+bar+titles+facet_w+white_theme+axis_gap_y
```

```
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\arrays\d
atetimes.py:2192: PytzUsageWarning: The zone attribute is specific to pytz's interfa
ce; please migrate to a new time zone provider. For more details on how to do so, se
e https://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pyt
z-deprecation-shim.readthedocs.io/en/latest/migration.html)
  values, tz_parsed = conversion.datetime_to_datetime64(data.ravel("K"))
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\indexes
\base.py:6411: PytzUsageWarning: The zone attribute is specific to pytz's interface;
please migrate to a new time zone provider. For more details on how to do so, see ht
tps://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-de
precation-shim.readthedocs.io/en/latest/migration.html)
  result = lib.maybe_convert_objects(
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\arrays\d
atetimes.py:2307: PytzUsageWarning: The zone attribute is specific to pytz's interfa
ce; please migrate to a new time zone provider. For more details on how to do so, se
e https://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pyt
z-deprecation-shim.readthedocs.io/en/latest/migration.html)
  elif not timezones.tz_compare(tz, inferred_tz):
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\arrays\d
atetimes.py:2388: PytzUsageWarning: The zone attribute is specific to pytz's interfa
ce; please migrate to a new time zone provider. For more details on how to do so, se
e https://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pyt
z-deprecation-shim.readthedocs.io/en/latest/migration.html)
  if tz is not None and not timezones.tz_compare(tz, dtz):
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\indexes
\base.py:5363: PytzUsageWarning: The zone attribute is specific to pytz's interface;
please migrate to a new time zone provider. For more details on how to do so, see ht
tps://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-de
precation-shim.readthedocs.io/en/latest/migration.html)
  and not tz_compare(self.tz, other.tz)
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\dtypes\d
types.py:796: PytzUsageWarning: The zone attribute is specific to pytz's interface;
 please migrate to a new time zone provider. For more details on how to do so, see h
ttps://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-d
eprecation-shim.readthedocs.io/en/latest/migration.html)
  and tz_compare(self.tz, other.tz)
```



Explanation of the findings

1. Platform 0: the DAU decrease is with Version 3.42.1
2. Platform 1: the DAU decrease is with Versions 3.42, 3.42.1
3. Platform 2: the DAU decrease is with Versions 3.42
4. Platform 3: the DAU decrease is with Versions 3.42.1
5. Platform 6: the DAU decrease is with Versions 3.42
6. Platform 9: there is a DAU decrease with Versions 3.42 But, in the previous section, this decrease does not appear unique.

Conclusion:

1. The decrease in DAU is caused by the update of versions 3.42 and 3.42.1
2. And is resolved with the update of versions: 3.43, 3.43.1

Answer the DAU questions:

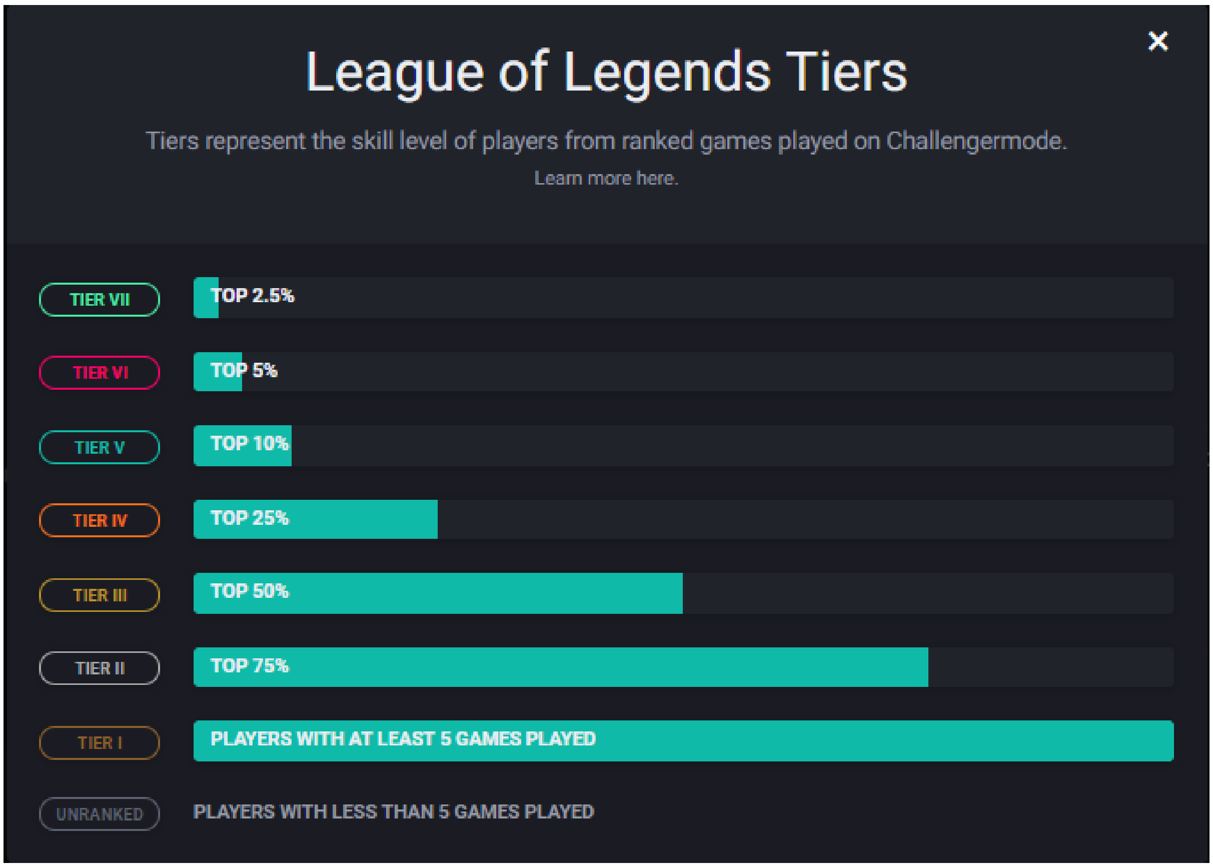
1. **How does the DAU change over time?**
   Answer:
   A. DAU has monthly trends of increase and decrease, but the overall size doesn't change significantly.
   B. The data consists of only six months, so a change across the year is not possible to analyze, in this matter, the influence of holidays, seasons, etc.
   C. Between the 10th of February until 3rd of March, there was a significant decrease in DAU. This event occurs over all of the platforms, but platform 9, where the event is in the scope of regular changes.
   D. The decrease in DAU is caused by the update of versions 3.42 and 3.42.1
   E. And is resolved with the update of versions: 3.43, 3.43.1
2. **How does the DAU change by over time, accross the different Platforms?**
   Answer:
   A. DAU size is different between the platforms, the largest to smallest Platforms in terms of DAU are 2,1,0,9,3,6.
   B. DAU trend of change over time is relatively similar among the different platforms except in platform 9.

4. A2. Tiers and Platforms:

1. What is a Tier?
2. What is a Platform?
3. Which is the largest to smallest (in terms of DAU) Platform?
4. What are the Tier ranks (in terms of DAU), in total time and by each month?
5. Does the Tiers rank varies between Platforms?

What is a Tier? An Example:
Based on the example below, Tiers represent the segmentation of players by their in-game level.

- most of the players are low level, thus the Tier number that represents those players is supposed to have the highest DAU;
- While only a few of the players are high level, their Tier should have the lowest DAU.
- Hence, the Tiers are supposed to be ranked in order, i.e. 1,2,3,4,5, etc., or in a descending order i.e. 5,4,3,2,1.


League of Legends Tiers

Tiers represent the skill level of players from ranked games played on Challengermode.

Learn more here.

TIER VII	TOP 2.5%
TIER VI	TOP 5%
TIER V	TOP 10%
TIER IV	TOP 25%
TIER III	TOP 50%
TIER II	TOP 75%
TIER I	PLAYERS WITH AT LEAST 5 GAMES PLAYED
UNRANKED	PLAYERS WITH LESS THAN 5 GAMES PLAYED

In [37]:
```
%%sql
select distinct Tier, SUM(DAU) AS Total_DAU, AVG(DAU) AS Avg_day_DAU
FROM DAU_Data
GROUP BY Tier
ORDER BY SUM(DAU) DESC
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[37]:

Tier	Total_DAU	Avg_day_DAU
3	69293163	44390
2	63921853	36298
4	35507401	31256
1	12185179	21720
5	11751225	18106
6	533623	8337
NON-RANKED	65260	16315

In the data:

1. There are six Tiers, and also a few un-ranked users.
2. The largest to smallest Tiers in terms of Total_DAU are: 3,2,4,1,5,6, Unranked.

The Tiers are not ranked in the anticipated order, i.e. Tier 2 is larger than Tier 4 but smaller than Tier 3.

How does the Tiers rank (in terms of DAu) Varie each month?

In [38]:
```
%%sql
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'Tier_Rnk') DROP V
```
```
 * mssql+pyodbc://nir:***@mssql
Done.
```
Out[38]:
```
[]
```

In [39]:
```
%%sql
CREATE VIEW Tier_Rnk AS
WITH "Largest_Tier_1" AS
(
SELECT Tier,
       CAST(YEAR(Calc_Date) AS NVARCHAR(4)) + '-' + CAST(MONTH(Calc_Date) AS NVARCHAR(2)
       SUM(DAU) AS DAU
FROM DAU_Data
GROUP BY Tier, CAST(YEAR(Calc_Date) AS NVARCHAR(4)) + '-' + CAST(MONTH(Calc_Date) AS NVA
),
"Largest_Tier_2"
AS
(
SELECT *,
        'Rank_' + CAST(Row_Number() OVER (PARTITION BY Date ORDER BY DAU DESC) AS NVARCH
FROM Largest_Tier_1
)
SELECT Date,
       Tier,
       DAU,
       Rnk
FROM Largest_Tier_2
```
```
 * mssql+pyodbc://nir:***@mssql
Done.
```
Out[39]:
```
[]
```

In [40]:
```
%%sql

DECLARE @columns AS VARCHAR(2000),
        @query AS VARCHAR(2000)

SET @columns = STUFF((SELECT ',' + Rnk FROM
                (SELECT DISTINCT QUOTENAME(Rnk) AS Rnk FROM Tier_Rnk) TBL_Tier_Rnk ORDER
                 FOR XML PATH ('')),1 ,1 ,'')

SET @query = 'SELECT * FROM (
        SELECT Date,
               Tier,
               Rnk
        FROM Tier_Rnk
     ) AS TBL_Tier_Rnk
PIVOT
(
MAX(Tier)
FOR [Rnk] IN ( '+ @columns +'  )
) AS PVT_Tier_Rnk
ORDER BY Date'

EXECUTE(@query)
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[40]:

Date	Rank_1	Rank_2	Rank_3	Rank_4	Rank_5	Rank_6	Rank_7
2020-1	3	2	4	5	1	6	None
2020-2	3	2	4	5	1	6	None
2020-3	3	2	4	1	5	6	None
2020-4	3	2	4	1	5	6	None
2020-5	3	2	4	1	5	6	None
2020-6	3	2	4	5	1	6	NON-RANKED

Summary of The Tiers Ranks:

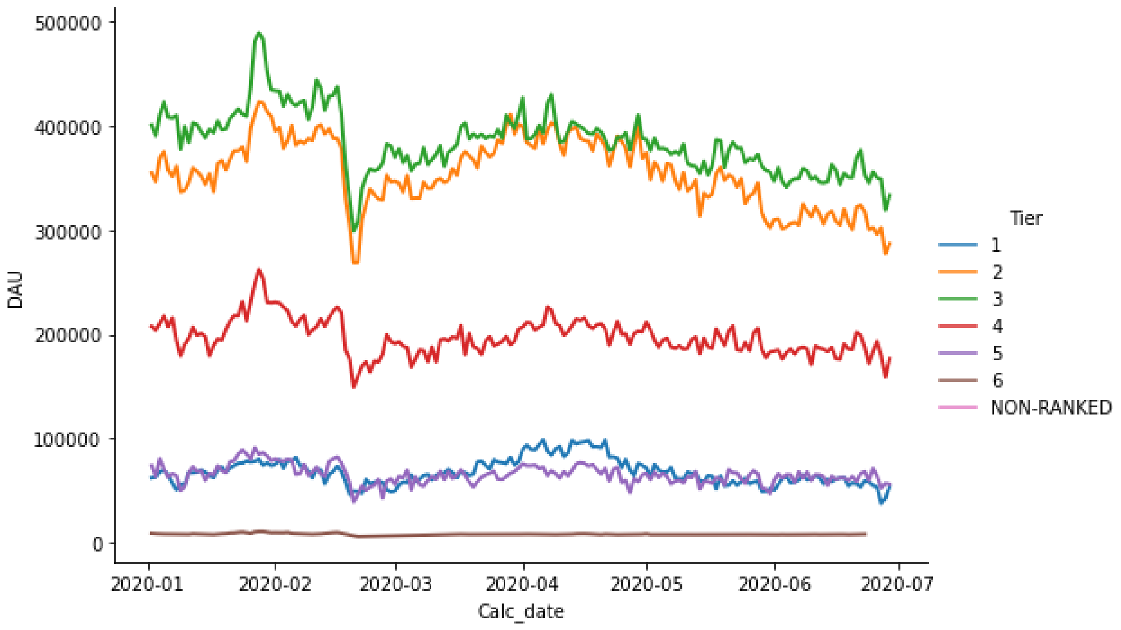
1. Tier 3 is always the largest.
2. Tier 2 is always the 2nd largest Tier.
3. Tier 4 is always the 3rd largest Tier.
4. Tiers 1 and 5 are changing rank position between themself in Ranks 4 and 5.
5. Tier 6 is always the smallest.

These findings are also shown in the next plot when the DAU changes over time:

In [41]:
```
DAU_Tiers_query = """

SELECT Calc_date, Tier,
       SUM(DAU) AS DAU
FROM DAU_Data
GROUP BY Calc_date, Tier

"""

DAU_Tiers = pd.read_sql_query(DAU_Tiers_query, connection)
```

```
In [42]: sns.relplot(data = DAU_Tiers,
                     x = "Calc_date", y = "DAU", kind = "line",
                     hue = "Tier",
                     palette = "tab10",
                     ci = None,
                     estimator = sum, #Estimator set to sum - because there are several values pe
                     height = 5,
                     aspect = 1.5,
                     lw=2
                     )
```

```
Out[42]: <seaborn.axisgrid.FacetGrid at 0x167d50c58e0>
```



The Tiers should probably be partitioned between the different platforms

```
In [43]: %%sql
         IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'Tier_Num_Platform
```

```
          * mssql+pyodbc://nir:***@mssql
         Done.
```

```
Out[43]: []
```

In [44]:
```
%%sql
CREATE VIEW Tier_Num_Platforms AS
WITH "Tier_Count_CTE_1" AS
(
SELECT DISTINCT Platform_id, Tier
FROM DAU_Data
),
"Tier_Count_CTE_2"
AS
(
SELECT Tier, Platform_id,
       Dense_Rank() OVER (PARTITION BY Tier, Platform_id ORDER BY Tier) AS Rnk
FROM Tier_Count_CTE_1
)
SELECT Tier, Sum(Rnk) As Num_of_Platforms
FROM Tier_Count_CTE_2
GROUP BY Tier
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[44]:
```
[]
```

In [45]:
```
%%sql
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'Tier_DAU') DROP V
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[45]:
```
[]
```

In [46]:
```
%%sql
CREATE VIEW Tier_DAU AS
SELECT DISTINCT Tier, SUM(DAU) AS Total_DAU, AVG(DAU) AS Avg_day_DAU
FROM DAU_Data
GROUP BY Tier
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[46]:
```
[]
```

In [47]:
```
%%sql

WITH "CTE_1" AS
(
SELECT DISTINCT Platform_id, Tier
FROM DAU_Data
),
"CTE_2"
AS
(
SELECT Dense_Rank() OVER (ORDER BY Tier) AS Tier_id, Tier, Platform_id
FROM CTE_1
),
"CTE_3" (Tier_id, Platforms)
AS
(
SELECT
    1,
    STUFF( (SELECT ',' + Platform_id
    FROM CTE_2
    WHERE CTE_2.Tier_id = 1
    FOR XML PATH ('') ) , 1, 1, '' ) AS Platforms
UNION ALL
SELECT
     CTE_3.Tier_id+1,
     STUFF( (SELECT ',' + Platform_id
     FROM CTE_2
     WHERE CTE_2.Tier_id = CTE_3.Tier_id+1
     FOR XML PATH ('') ) , 1, 1, '' ) AS Platforms
FROM CTE_3
WHERE CTE_3.Tier_id < 7
)
SELECT DISTINCT
    CTE_2.Tier,
    CTE_3.Platforms,
    Tier_Num_Platforms.Num_of_Platforms,
    Tier_DAU.Total_DAU
FROM CTE_3 JOIN CTE_2
ON  CTE_3.Tier_id =  CTE_2.Tier_id
           JOIN Tier_Num_Platforms
ON CTE_2.Tier = Tier_Num_Platforms.Tier
           JOIN Tier_DAU
ON CTE_2.Tier = Tier_DAU.Tier
ORDER BY Total_DAU DESC
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[47]:

Tier	Platforms	Num_of_Platforms	Total_DAU
3	0,1,2,3,6,9	6	69293163
2	0,1,2,3,6,9	6	63921853
4	0,1,2,3,9	5	35507401
1	1,2,9	3	12185179
5	0,1,2,3	4	11751225
6	0,1	2	533623
NON-RANKED	0,1,2	3	65260

```
In [48]: %%sql
         SELECT Platform_id, Sum(DAU) AS Total_DAU
         FROM DAU_Data
         GROUP BY Platform_id
         ORDER BY Sum(DAU) DESC
```

```
 * mssql+pyodbc://nir:***@mssql
Done.
```

Out[48]:

Platform_id	Total_DAU
2	85701726
1	53196395
0	32672603
9	12402668
3	7977028
6	1307284

Summary

1. The largest Tiers, 2 and 3, are also defined in all of the platforms.
2. The smallest Tier, 6, is only defined in two platforms (0,1).
3. Tiers 1 and 4 are medium-sized (in terms of DAU) and are defined in 3-5 Platforms.
4. There are only a few Unranked players, that are only found in platforms 0,1,2.

A Possible explanation of the findings

1. The smaller the platform (in terms of DAU), the fewer Tiers are defined in it.
2. Tiers 2 and 3 are general Tiers, They fit in every platform and can represent players with low in-game levels.
3. Tier 4 is also general, which might represent a higher in-game level of players. but, Tier 4 is not defined in platform 6, could be because platform 6 is relatively smaller (in terms of DAU) than all other platforms.
4. This explanation also fits Tiers 5 and 6: These Tiers represent high player's in-game level, but they are only included in the large Platforms (Not necessary the largest)
5. Tier 1 is the only Tier not in order because according to the logic described above, it should have the highest DAU and be used in all of the platforms, while it does not. An explanation is that maybe Tier 1 is used to target players only in specific platforms, that share specific attributes.
6. NON-RANKED players: Only a few players are unranked, and only in the largest used platforms. This could mean that over the smallest platforms, a player can't be un-ranked and that there is a rank for every player. while in the largest platforms, there are attributes that were not categorized.

Platforms Rank (in terms of DAU):

1. Platform 2.
2. Platform 1.
3. Platform 0.
4. Platform 9.
5. Platform 3.
6. Platform 6.

Visualize the data and answer the questions regarding the product

<u>The preferred visualization of the data, to show the Tiers change over time, across all of the Platforms is:</u>

- A line plot of DAU over time;
- Grouped by the Tiers;
- and faceted by the Platforms.

This type of plot will show the DAU change over time, by the different Tiers in all of the Platform.

```
In [49]: DAU_Tiers_Platforms_query = """

         SELECT Calc_date, Tier, Platform_id,
                SUM(DAU) AS DAU
         FROM DAU_Data
         GROUP BY Calc_date, Tier, Platform_id

         """

         DAU_Tiers_Platforms = pd.read_sql_query(DAU_Tiers_Platforms_query, connection)
         DAU_Tiers_Platforms
```

```
Out[49]:
```

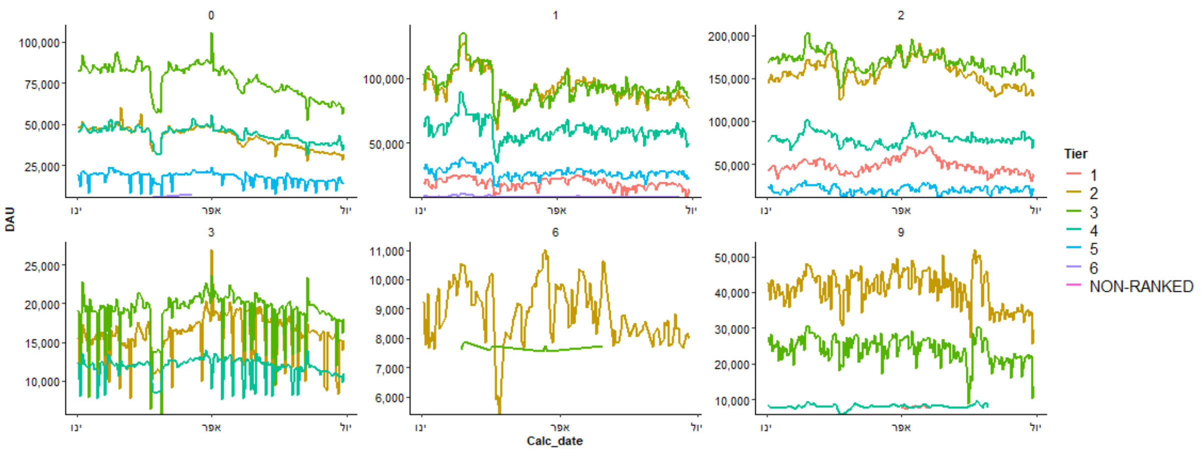
	Calc_date	Tier	Platform_id	DAU
0	2020-01-02	1	1	19196
1	2020-01-02	1	2	42953
2	2020-01-02	2	0	47613
3	2020-01-02	2	1	95435
4	2020-01-02	2	2	144699
...	...	...	...	...
3685	2020-06-29	4	2	78209
3686	2020-06-29	4	3	10921
3687	2020-06-29	5	0	14960
3688	2020-06-29	5	1	23642
3689	2020-06-29	5	2	17289

3690 rows × 4 columns

In [50]:
```
%%R -i DAU_Tiers_Platforms -w 40 -h 15 --units cm


library(ggplot2)

DAU_Tiers_Platforms$Platform_id <- as.factor(DAU_Tiers_Platforms$Platform_id)
DAU_Tiers_Platforms$Tier <- as.factor(DAU_Tiers_Platforms$Tier)

#Create a Blank plot called p, choose your x,y variables
p <- ggplot(
  data = DAU_Tiers_Platforms,
  aes(
    x=Calc_date,
    y=DAU,
    color = Tier
    )
  )

facet_w <- facet_wrap(DAU_Tiers_Platforms$Platform_id, scales = "free") #("fixed", "free

line <- geom_line(
  alpha = 1,
  size = 1
  )

axis_y <- scale_y_continuous(
    expand = c(0, 0, 0.05, 0), labels = scales::label_comma()
)

#Create your plot theme. this example sets the theme to a clean white plot
#to remove an object use: element_blank()
white_theme <- theme(
  title = element_text(size = 12, color = "black", face = "bold"),
  axis.text.x = element_text(vjust= 0.5, angle = 0, color = "black", size = 11),
  axis.text.y = element_text(size = 11, color = "black"),
  axis.title.x = element_text(size = 11, color = "black"),
  axis.title.y = element_text(size = 11, color = "black"),

  legend.position = "right", #(right, left, bottom, top, none)
  legend.title.align = NULL,
  legend.justification = "center",
  #legend.title=element_blank(),
  legend.text = element_text(size=15, color = "black"),
  legend.background = element_rect(fill = NA),
  legend.box.background = element_rect(fill = NA, color = NA),
  legend.key = element_rect(fill = NA),

  panel.border = element_rect(fill = NA, color = NA),
  panel.grid.major = element_line(color = NA),
  panel.grid.minor = element_line(color = NA),
  panel.background = element_rect(fill = NA, color = NA),

  strip.text.x = element_text(size = 11, color = "black", face = "plain"),
  strip.text.y = element_text(size = 11, color = "black", face = "plain"),
  strip.background =element_rect(fill= NA),

  axis.line = element_line(colour = "black")
)

p+line+facet_w+white_theme+axis_y
```

```
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\arrays\date
times.py:2192: PytzUsageWarning: The zone attribute is specific to pytz's interface; pl
ease migrate to a new time zone provider. For more details on how to do so, see http
s://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-depreca
tion-shim.readthedocs.io/en/latest/migration.html)
  values, tz_parsed = conversion.datetime_to_datetime64(data.ravel("K"))
```

```
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\indexes\bas
e.py:6411: PytzUsageWarning: The zone attribute is specific to pytz's interface; please
migrate to a new time zone provider. For more details on how to do so, see https://pytz
-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-deprecation-shi
m.readthedocs.io/en/latest/migration.html)
  result = lib.maybe_convert_objects(
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\arrays\date
times.py:2307: PytzUsageWarning: The zone attribute is specific to pytz's interface; pl
ease migrate to a new time zone provider. For more details on how to do so, see http
s://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-depreca
tion-shim.readthedocs.io/en/latest/migration.html)
  elif not timezones.tz_compare(tz, inferred_tz):
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\arrays\date
times.py:2388: PytzUsageWarning: The zone attribute is specific to pytz's interface; pl
ease migrate to a new time zone provider. For more details on how to do so, see http
s://pytz-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-depreca
tion-shim.readthedocs.io/en/latest/migration.html)
  if tz is not None and not timezones.tz_compare(tz, dtz):
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\indexes\bas
e.py:5363: PytzUsageWarning: The zone attribute is specific to pytz's interface; please
migrate to a new time zone provider. For more details on how to do so, see https://pytz
-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-deprecation-shi
m.readthedocs.io/en/latest/migration.html)
  and not tz_compare(self.tz, other.tz)
C:\Users\nir\Anaconda3\envs\pandas_playground\lib\site-packages\pandas\core\dtypes\dtyp
es.py:796: PytzUsageWarning: The zone attribute is specific to pytz's interface; please
migrate to a new time zone provider. For more details on how to do so, see https://pytz
-deprecation-shim.readthedocs.io/en/latest/migration.html (https://pytz-deprecation-shi
m.readthedocs.io/en/latest/migration.html)
  and tz_compare(self.tz, other.tz)
```



Answer the Tiers and Platforms questions:

1. **What is a Tier?**
   Answer:
   A. A-Tier in this data most likely represent in-game progress such as the number of games played or in-game level of the player.
2. **What is a Platform?**
   Answer:
   A. A platform is the environment where a player is playing the game, could be an Android, an iPhone, social network, different types and versions of them, etc.
3. **Which is the largest to smallest (in terms of DAU) Platform?**
   Answer:
   A. the platforms are ranked as follows:
      - Platform 2
      - Platform 1
      - Platform 0
      - Platform 9

- Platform 3
- Platform 6

4. **What are the Tier ranks (in terms of DAU), in total time and by each month?**
   Answer:
   A. The Tiers are ranked as follows:
      - Tier 3 is always the largest.
      - Tier 2 is always the 2nd largest Tier.
      - Tier 4 is always the 3rd largest Tier.
      - Tiers 1 and 5 are changing rank position between themself in Ranks 4 and 5.
      - Tier 6 is always the smallest.
5. **How does the Tiers varie between Platforms?**
   Answer:

- The largest Tiers, 2 and 3, are also defined in all of the platforms.
- The smallest Tier, 6, is only defined in two platforms (0,1).
- Tiers 1 and 4 are medium-sized (in terms of DAU) and are defined in 3-5 Platforms.
- There are only few Unranked players, that are only found in platforms 0,1,2.

  A Possible explanetion of the finidings

  A. The smaller the platform (in terms of DAU), the fewer Tiers are defined in it.
  B. Tiers 2 and 3 are general Tiers, They fit in every platform and can represent players with low in-game levels.
  C. Tier 4 is also general and might represent a higher in-game level of players. but, Tier 4 is not defined in platform 6, could be because platform 6 is relatively smaller (in terms of DAU) than all other platforms.
  D. This explanation also fits Tiers 5 and 6: These Tiers represent high player's in-game level, but they are only included in the large Platforms (Not necessary the largest)
  E. Tier 1 is the only Tier not in order because according to the logic described above, it should have the highest DAU and be used in all of the platforms, while it does not. An explanation is that maybe Tier 1 is used to target players only in specific platforms, that share specific attributes.
  F. NON-RANKED players: Only a few players are unranked, and only in the largest used platforms. This could mean that over the smallest platforms, a player can't be un-ranked and that there is a rank for every player. while in the largest platforms, there are attributes that were not categorized.

```
In [51]: connection.close()
```