REPORT ON INSTAGRAM USER ANALYTICS SQL FUNDAMENTALS

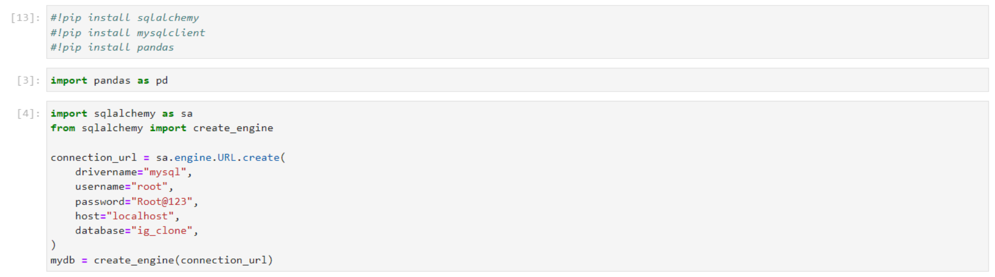
Project Description :
The project focuses on analysing Instagram users in order to provide business insights for marketing, product development, and general growth. Instagram is a well-known digital platform. The software or mobile application of the platform should be used to measure user engagement and interactions. The study seeks to find the contest winners, inactive users, oldest users, most popular hashtags, and the best days for ad campaign start dates. The initiative also evaluates user activity and spots any possible bot or phoney accounts on the network.

Approach:
The approach involves utilising software like MySQL Workbench, Jupyter Notebook, and libraries like SQLAlchemy and Pandas, the technique analyses the Instagram database. To glean pertinent insights, the data will be searched, cleansed, and converted. In order to respond to particular inquiries and offer practical advice, a variety of SQL queries and data manipulation techniques will be used.

Tech Stack Used:
The following tech stack is utilised for this project:

- The Instagram database is accessed and queried using MySQL Workbench.
- Jupyter Notebook is used as the development environment for data analysis and documentation
- Python package called SQLAlchemy is used to connect to databases and run SQL statements.
- Pandas is a potent python data manipulation package that may be used to clean, manipulate, and analyse data.

Insights :
Connection of Database :

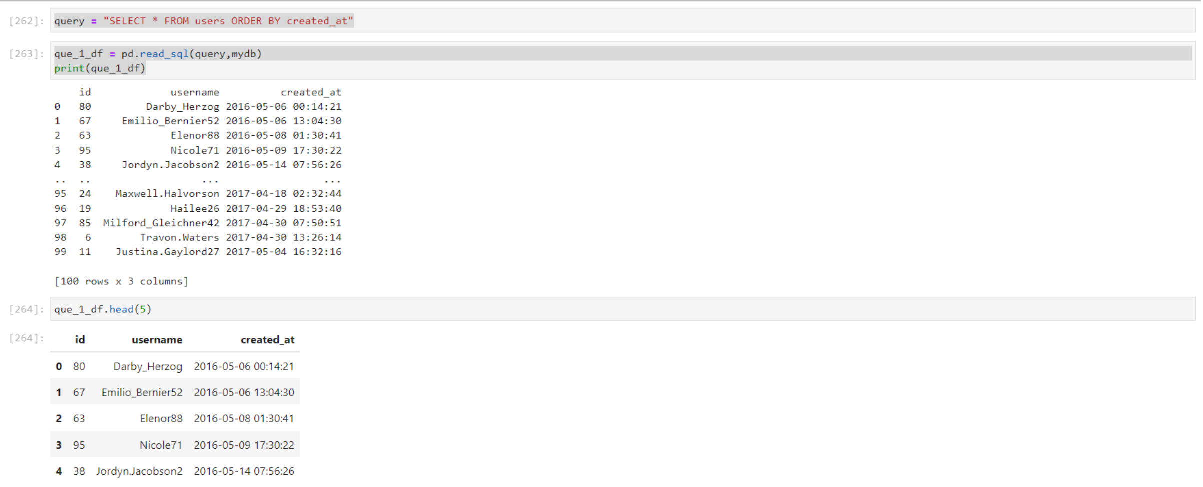


```
[13]: #!pip install sqlalchemy
      #!pip install mysqlclient
      #!pip install pandas

[3]:  import pandas as pd

[4]:  import sqlalchemy as sa
      from sqlalchemy import create_engine

      connection_url = sa.engine.URL.create(
          drivername="mysql",
          username="root",
          password="Root@123",
          host="localhost",
          database="ig_clone",
      )
      mydb = create_engine(connection_url)
```

Performing Analysis :

Oldest Users:

I used a SQL query using the ORDER BY clause to sort the users based on their registration dates in ascending order in order to discover the five oldest Instagram users. I was able to find out information about the five earliest users by choosing the required fields, such usernames or user IDs, and restricting the results to the top five entries.



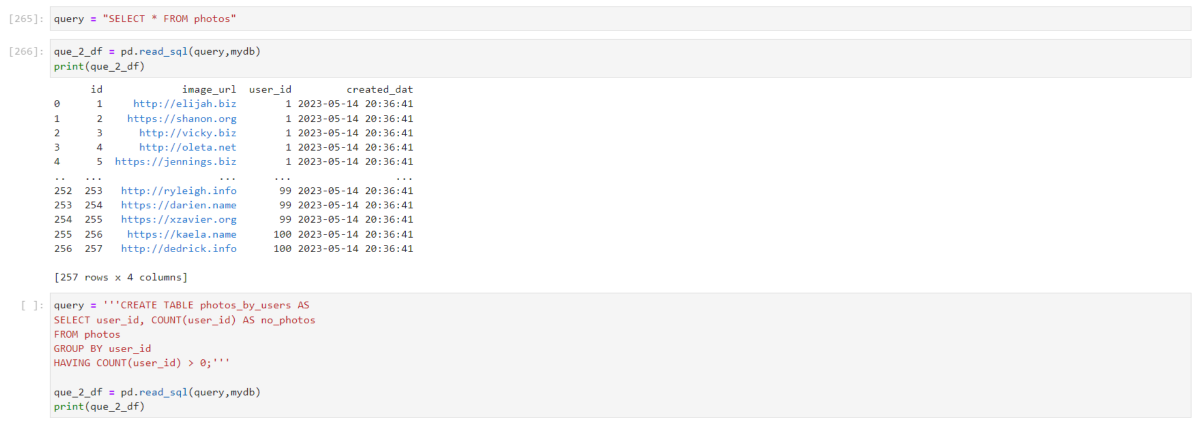
```
[262]: query = "SELECT * FROM users ORDER BY created_at"

[263]: que_1_df = pd.read_sql(query,mydb)
       print(que_1_df)

    id           username          created_at
0   80       Darby_Herzog 2016-05-06 00:14:21
1   67     Emilio_Bernier52 2016-05-06 13:04:30
2   63           Elenor88 2016-05-08 01:30:41
3   95           Nicole71 2016-05-09 17:30:22
4   38     Jordyn.Jacobson2 2016-05-14 07:56:26
..  ..                ...                 ...
95  24    Maxwell.Halvorson 2017-04-18 02:32:44
96  19           Hailee26 2017-04-29 18:53:40
97  85  Milford_Gleichner42 2017-04-30 07:50:51
98   6        Travon.Waters 2017-04-30 13:26:14
99  11     Justina.Gaylord27 2017-05-04 16:32:16

[100 rows x 3 columns]

[264]: que_1_df.head(5)
```

[264]:

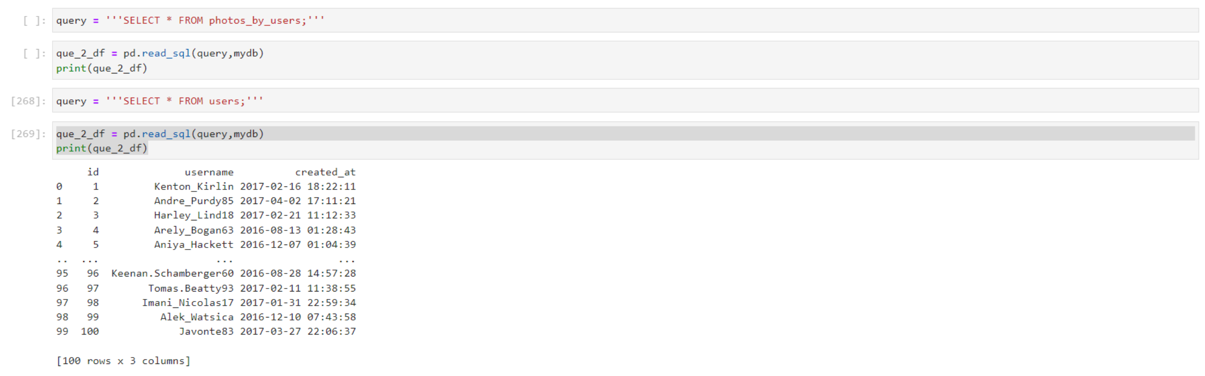
	id	username	created_at
0	80	Darby_Herzog	2016-05-06 00:14:21
1	67	Emilio_Bernier52	2016-05-06 13:04:30
2	63	Elenor88	2016-05-08 01:30:41
3	95	Nicole71	2016-05-09 17:30:22
4	38	Jordyn.Jacobson2	2016-05-14 07:56:26

Inactive Users:

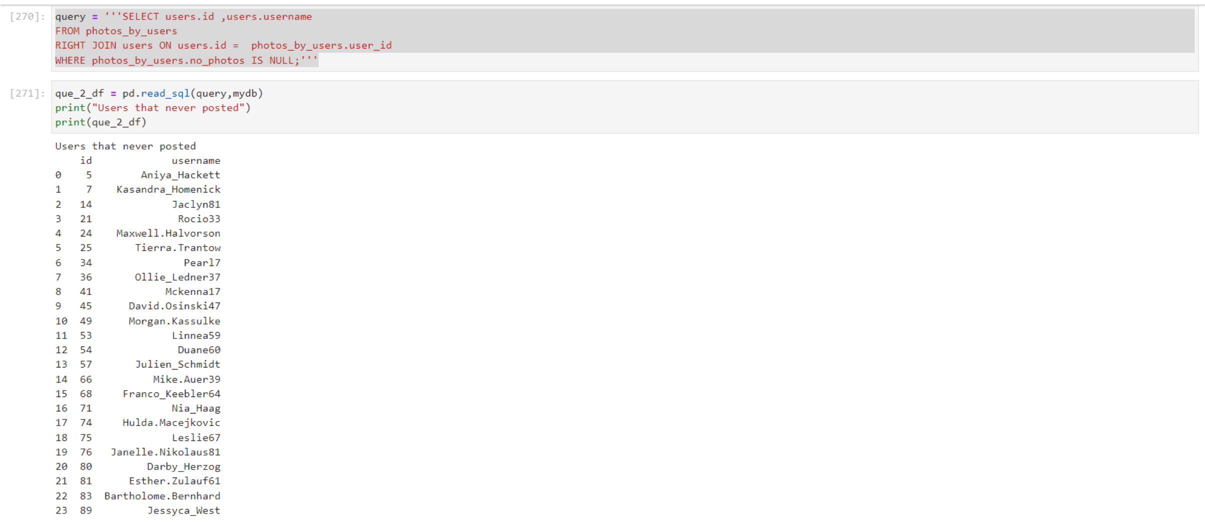
I combined SQL queries using the RIGHT JOIN and COUNT() procedures to find users who have never shared a photo on Instagram. I filtered out users with a count of zero by connecting the user table and the post table and calculating the amount of posts for each user. To persuade these people to begin posting, promotional emails might be sent to them.



```
[265]: query = "SELECT * FROM photos"

[266]: que_2_df = pd.read_sql(query,mydb)
       print(que_2_df)

      id           image_url  user_id          created_dat
0      1    http://elijah.biz        1 2023-05-14 20:36:41
1      2   https://shanon.org        1 2023-05-14 20:36:41
2      3     http://vicky.biz        1 2023-05-14 20:36:41
3      4     http://oleta.net        1 2023-05-14 20:36:41
4      5  https://jennings.biz        1 2023-05-14 20:36:41
..   ...                 ...      ...                 ...
252  253   http://ryleigh.info       99 2023-05-14 20:36:41
253  254   https://darien.name       99 2023-05-14 20:36:41
254  255   https://xzavier.org       99 2023-05-14 20:36:41
255  256    https://kaela.name      100 2023-05-14 20:36:41
256  257   http://dedrick.info      100 2023-05-14 20:36:41

[257 rows x 4 columns]

[ ]: query = '''CREATE TABLE photos_by_users AS
     SELECT user_id, COUNT(user_id) AS no_photos
     FROM photos
     GROUP BY user_id
     HAVING COUNT(user_id) > 0;'''

     que_2_df = pd.read_sql(query,mydb)
     print(que_2_df)
```



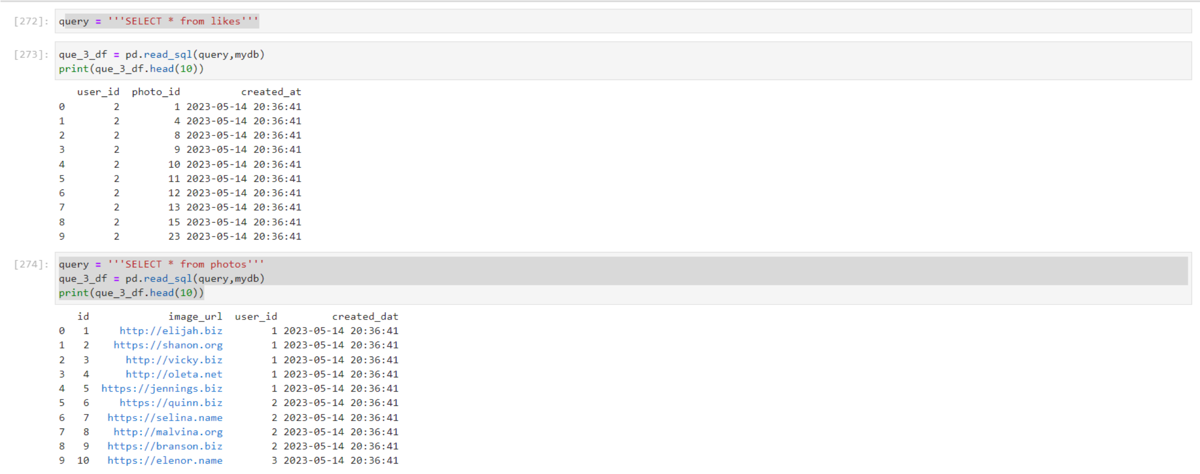
```
[ ]: query = '''SELECT * FROM photos_by_users;'''

[ ]: que_2_df = pd.read_sql(query,mydb)
     print(que_2_df)

[268]: query = '''SELECT * FROM users;'''

[269]: que_2_df = pd.read_sql(query,mydb)
       print(que_2_df)

        id             username          created_at
   0     1         Kenton_Kirlin 2017-02-16 18:22:11
   1     2         Andre_Purdy85 2017-04-02 17:11:21
   2     3         Harley_Lind18 2017-02-21 11:12:33
   3     4         Arely_Bogan63 2016-08-13 01:28:43
   4     5         Aniya_Hackett 2016-12-07 01:04:39
   ..  ...                  ...                 ...
   95   96  Keenan.Schamberger60 2016-08-28 14:57:28
   96   97        Tomas.Beatty93 2017-02-11 11:38:55
   97   98      Imani_Nicolas17 2017-01-31 22:59:34
   98   99          Alek_Watsica 2016-12-10 07:43:58
   99  100           Javonte83 2017-03-27 22:06:37

   [100 rows x 3 columns]
```



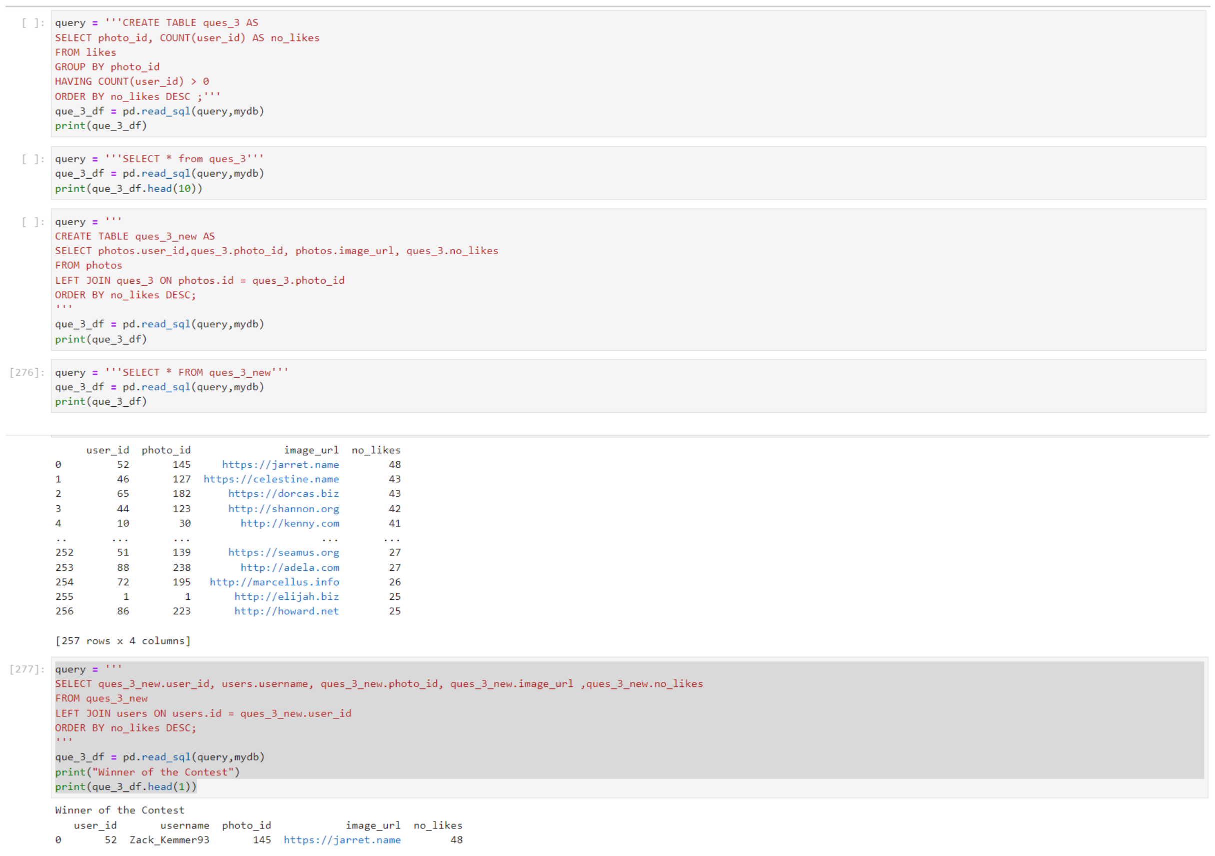
```
[270]: query = '''SELECT users.id ,users.username
       FROM photos_by_users
       RIGHT JOIN users ON users.id =  photos_by_users.user_id
       WHERE photos_by_users.no_photos IS NULL;'''

[271]: que_2_df = pd.read_sql(query,mydb)
       print("Users that never posted")
       print(que_2_df)

       Users that never posted
           id             username
       0    5        Aniya_Hackett
       1    7    Kasandra_Homenick
       2   14            Jaclyn81
       3   21             Rocio33
       4   24     Maxwell.Halvorson
       5   25       Tierra.Trantow
       6   34              Pearl7
       7   36       Ollie_Ledner37
       8   41           Mckenna17
       9   45       David.Osinski47
       10  49       Morgan.Kassulke
       11  53            Linnea59
       12  54             Duane60
       13  57       Julien_Schmidt
       14  66          Mike.Auer39
       15  68     Franco_Keebler64
       16  71            Nia_Haag
       17  74     Hulda.Macejkovic
       18  75            Leslie67
       19  76   Janelle.Nikolaus81
       20  80          Darby_Herzog
       21  81      Esther.Zulauf61
       22  83  Bartholome.Bernhard
       23  89          Jessyca_West
```

Contest Winner:

I used a SQL query with the COUNT() function and an LEFT link to link the user table with the likes table in order to decide the contest winner based on the person who received the most likes on a specific photo. I tallied the number of likes for each photo by categorising the data by the person and the image. I was able to find out the contest winner's information by using ORDER BY to sort the results in descending order and restricting it to the top record.



```
[272]: query = '''SELECT * from likes'''

[273]: que_3_df = pd.read_sql(query,mydb)
       print(que_3_df.head(10))

          user_id  photo_id          created_at
       0        2         1 2023-05-14 20:36:41
       1        2         4 2023-05-14 20:36:41
       2        2         8 2023-05-14 20:36:41
       3        2         9 2023-05-14 20:36:41
       4        2        10 2023-05-14 20:36:41
       5        2        11 2023-05-14 20:36:41
       6        2        12 2023-05-14 20:36:41
       7        2        13 2023-05-14 20:36:41
       8        2        15 2023-05-14 20:36:41
       9        2        23 2023-05-14 20:36:41

[274]: query = '''SELECT * from photos'''
       que_3_df = pd.read_sql(query,mydb)
       print(que_3_df.head(10))

          id            image_url  user_id         created_dat
       0   1      http://elijah.biz        1 2023-05-14 20:36:41
       1   2     https://shanon.org        1 2023-05-14 20:36:41
       2   3       http://vicky.biz        1 2023-05-14 20:36:41
       3   4       http://oleta.net        1 2023-05-14 20:36:41
       4   5   https://jennings.biz        1 2023-05-14 20:36:41
       5   6      https://quinn.biz        2 2023-05-14 20:36:41
       6   7    https://selina.name        2 2023-05-14 20:36:41
       7   8     http://malvina.org        2 2023-05-14 20:36:41
       8   9    https://branson.biz        2 2023-05-14 20:36:41
       9  10    https://elenor.name        3 2023-05-14 20:36:41
```



```
[ ]: query = '''CREATE TABLE ques_3 AS
     SELECT photo_id, COUNT(user_id) AS no_likes
     FROM likes
     GROUP BY photo_id
     HAVING COUNT(user_id) > 0
     ORDER BY no_likes DESC ;'''
     que_3_df = pd.read_sql(query,mydb)
     print(que_3_df)

[ ]: query = '''SELECT * from ques_3'''
     que_3_df = pd.read_sql(query,mydb)
     print(que_3_df.head(10))

[ ]: query = '''
     CREATE TABLE ques_3_new AS
     SELECT photos.user_id,ques_3.photo_id, photos.image_url, ques_3.no_likes
     FROM photos
     LEFT JOIN ques_3 ON photos.id = ques_3.photo_id
     ORDER BY no_likes DESC;
     '''
     que_3_df = pd.read_sql(query,mydb)
     print(que_3_df)

[276]: query = '''SELECT * FROM ques_3_new'''
       que_3_df = pd.read_sql(query,mydb)
       print(que_3_df)
```

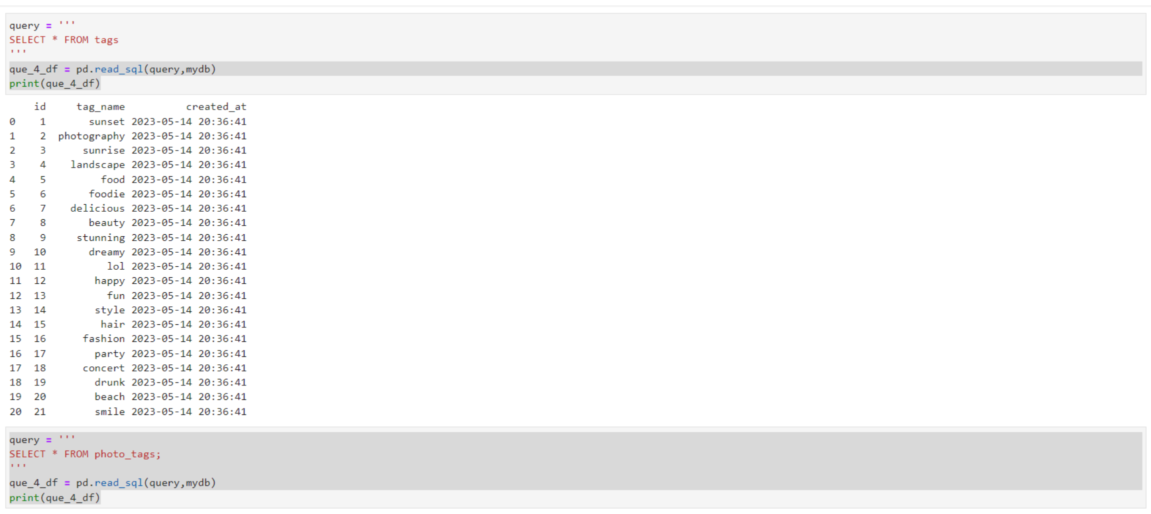
```
     user_id  photo_id               image_url  no_likes
0         52       145      https://jarret.name        48
1         46       127    https://celestine.name      43
2         65       182        https://dorcas.biz       43
3         44       123        http://shannon.org       42
4         10        30          http://kenny.com       41
..       ...       ...                      ...       ...
252       51       139        https://seamus.org       27
253       88       238          http://adela.com       27
254       72       195    http://marcellus.info        26
255        1         1         http://elijah.biz       25
256       86       223         http://howard.net       25

[257 rows x 4 columns]
```

```
[277]: query = '''
       SELECT ques_3_new.user_id, users.username, ques_3_new.photo_id, ques_3_new.image_url ,ques_3_new.no_likes
       FROM ques_3_new
       LEFT JOIN users ON users.id = ques_3_new.user_id
       ORDER BY no_likes DESC;
       '''
       que_3_df = pd.read_sql(query,mydb)
       print("Winner of the Contest")
       print(que_3_df.head(1))
```

```
Winner of the Contest
   user_id       username  photo_id           image_url  no_likes
0       52  Zack_Kemmer93       145  https://jarret.name        48
```

Top Hashtags:

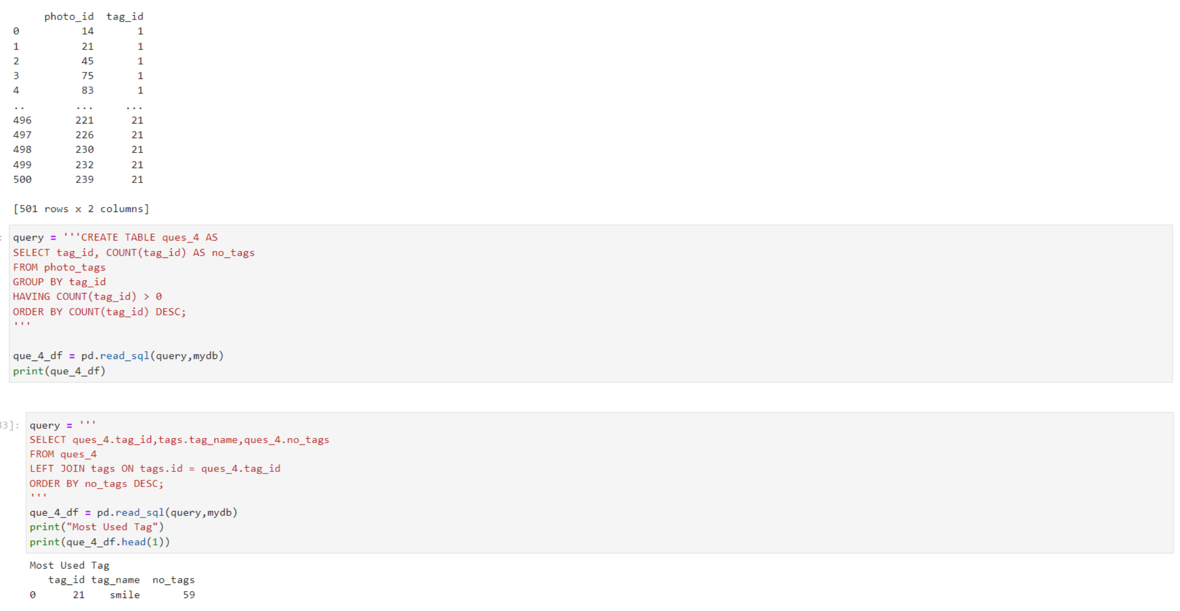
I joined the post table and the hashtag table using an LEFT JOIN and a SQL query with the COUNT() function to get the top five most popular hashtags on the site. I used ORDER BY to group the data by hashtag and count the number of occurrences before sorting the outcome. On the basis of this outcome, the top five hashtags were chosen.



```
[278]: query = '''
       SELECT * FROM tags
       '''
       que_4_df = pd.read_sql(query,mydb)
       print(que_4_df)
```

```
    id     tag_name          created_at
0    1       sunset 2023-05-14 20:36:41
1    2  photography 2023-05-14 20:36:41
2    3      sunrise 2023-05-14 20:36:41
3    4    landscape 2023-05-14 20:36:41
4    5         food 2023-05-14 20:36:41
5    6       foodie 2023-05-14 20:36:41
6    7    delicious 2023-05-14 20:36:41
7    8       beauty 2023-05-14 20:36:41
8    9     stunning 2023-05-14 20:36:41
9   10       dreamy 2023-05-14 20:36:41
10  11          lol 2023-05-14 20:36:41
11  12        happy 2023-05-14 20:36:41
12  13          fun 2023-05-14 20:36:41
13  14        style 2023-05-14 20:36:41
14  15         hair 2023-05-14 20:36:41
15  16      fashion 2023-05-14 20:36:41
16  17        party 2023-05-14 20:36:41
17  18      concert 2023-05-14 20:36:41
18  19        drunk 2023-05-14 20:36:41
19  20        beach 2023-05-14 20:36:41
20  21        smile 2023-05-14 20:36:41
```

```
[279]: query = '''
       SELECT * FROM photo_tags;
       '''
       que_4_df = pd.read_sql(query,mydb)
       print(que_4_df)
```



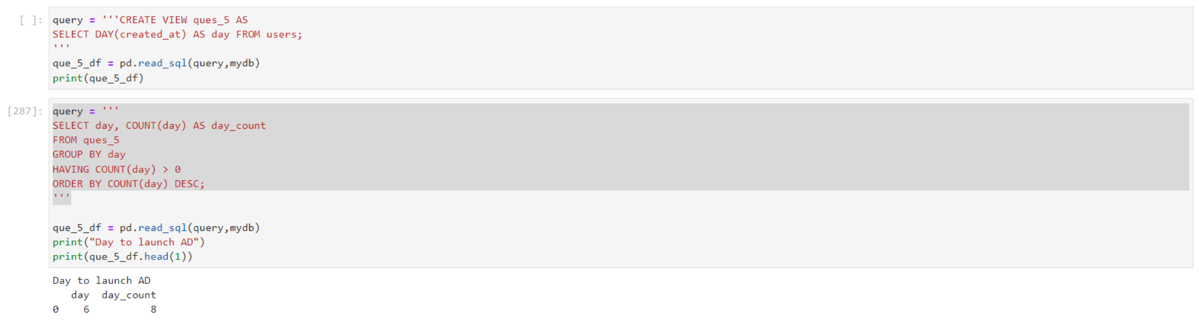
```
     photo_id  tag_id
0          14       1
1          21       1
2          45       1
3          75       1
4          83       1
..        ...     ...
496       221      21
497       226      21
498       230      21
499       232      21
500       239      21

[501 rows x 2 columns]
```

```
[ ]: query = '''CREATE TABLE ques_4 AS
     SELECT tag_id, COUNT(tag_id) AS no_tags
     FROM photo_tags
     GROUP BY tag_id
     HAVING COUNT(tag_id) > 0
     ORDER BY COUNT(tag_id) DESC;
     '''

     que_4_df = pd.read_sql(query,mydb)
     print(que_4_df)
```

```
[283]: query = '''
       SELECT ques_4.tag_id,tags.tag_name,ques_4.no_tags
       FROM ques_4
       LEFT JOIN tags ON tags.id = ques_4.tag_id
       ORDER BY no_tags DESC;
       '''
       que_4_df = pd.read_sql(query,mydb)
       print("Most Used Tag")
       print(que_4_df.head(1))

       Most Used Tag
          tag_id tag_name  no_tags
       0      21    smile       59
```

Optimal AD Campaign Launch:

I examined user registration data with a SQL query to ascertain the most effective day to start advertising campaigns. I tallied the number of registrations for each day by using the DAY() method to extract the day of the week and grouping the data by the registration day. I found the day with the most user registrations by using ORDER BY to arrange the results in descending order. Ad campaigns may be timed for optimal effect and reach using this information.

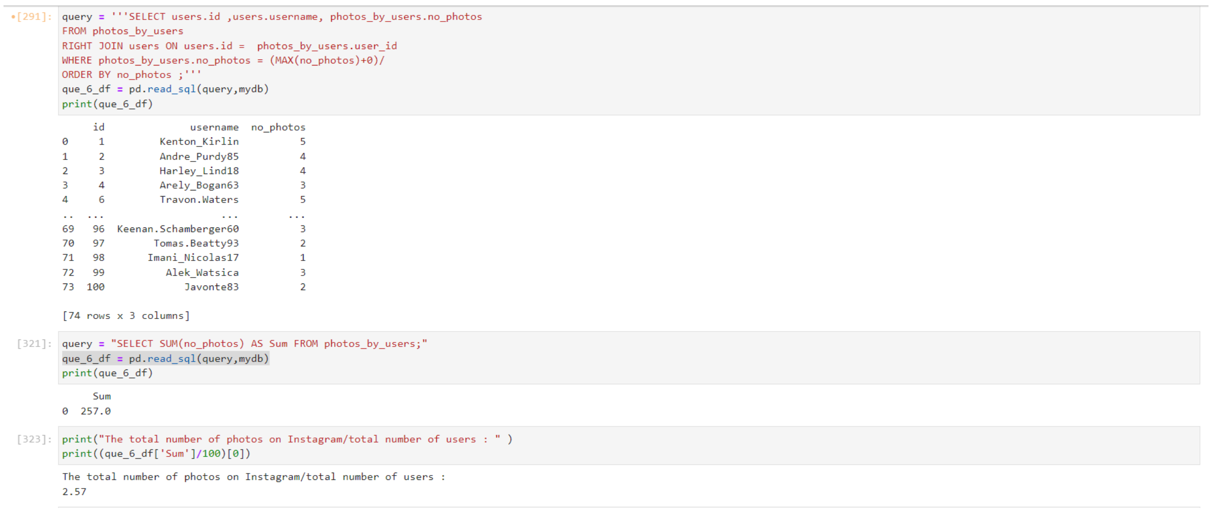


```
[ ]: query = '''CREATE VIEW ques_5 AS
     SELECT DAY(created_at) AS day FROM users;
     '''
     que_5_df = pd.read_sql(query,mydb)
     print(que_5_df)
```

```
[287]: query = '''
       SELECT day, COUNT(day) AS day_count
       FROM ques_5
       GROUP BY day
       HAVING COUNT(day) > 0
       ORDER BY COUNT(day) DESC;
       '''

       que_5_df = pd.read_sql(query,mydb)
       print("Day to launch AD")
       print(que_5_df.head(1))

       Day to launch AD
          day  day_count
       0    6          8
```

User Engagement:

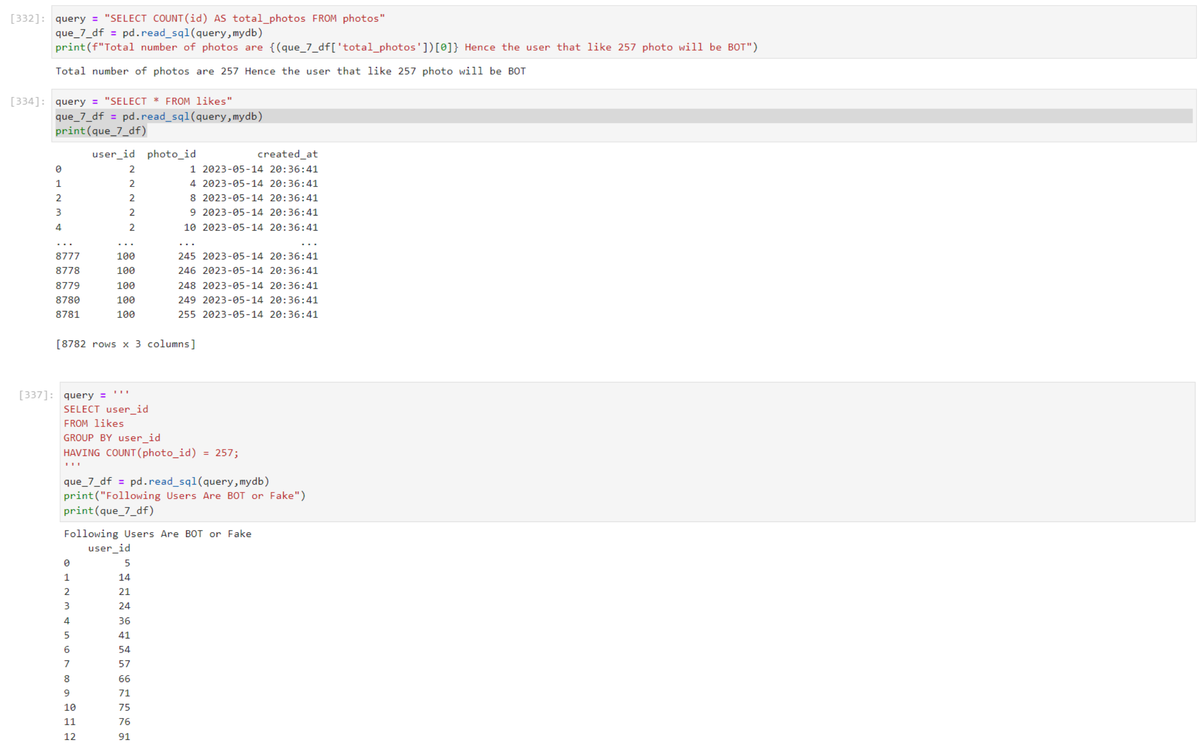
The average number of posts per user on Instagram may be calculated to gauge user engagement. This may be accomplished by counting the total number of users and posts using the COUNT() method. The average posts per user may be calculated by dividing the total number of posts by the total number of users. Additionally, you may count the postings in the database to ascertain the overall quantity of photographs on Instagram.



```
•[291]: query = '''SELECT users.id ,users.username, photos_by_users.no_photos
        FROM photos_by_users
        RIGHT JOIN users ON users.id =  photos_by_users.user_id
        WHERE photos_by_users.no_photos = (MAX(no_photos)+0)/
        ORDER BY no_photos ;'''
        que_6_df = pd.read_sql(query,mydb)
        print(que_6_df)

             id              username  no_photos
        0     1          Kenton_Kirlin          5
        1     2          Andre_Purdy85          4
        2     3          Harley_Lind18          4
        3     4          Arely_Bogan63          3
        4     6          Travon.Waters          5
        ..  ...                    ...        ...
        69   96   Keenan.Schamberger60          3
        70   97          Tomas.Beatty93          2
        71   98        Imani_Nicolas17          1
        72   99           Alek_Watsica          3
        73  100              Javonte83          2

        [74 rows x 3 columns]

 [321]: query = "SELECT SUM(no_photos) AS Sum FROM photos_by_users;"
        que_6_df = pd.read_sql(query,mydb)
        print(que_6_df)

             Sum
        0  257.0

 [323]: print("The total number of photos on Instagram/total number of users : " )
        print((que_6_df['Sum']/100)[0])

        The total number of photos on Instagram/total number of users :
        2.57
```

Bots & Fake Accounts:

You may examine the activities of individuals who have liked every single post on Instagram to spot probable bot or phoney accounts. Finding people who have liked every picture might be a sign of suspicious or automated behaviour because regular users would not be able to carry out this action. You may find people that meet this requirement by running a query on the database and looking at the likes information.



```
 [332]: query = "SELECT COUNT(id) AS total_photos FROM photos"
        que_7_df = pd.read_sql(query,mydb)
        print(f"Total number of photos are {(que_7_df['total_photos'])[0]} Hence the user that like 257 photo will be BOT")

        Total number of photos are 257 Hence the user that like 257 photo will be BOT

 [334]: query = "SELECT * FROM likes"
        que_7_df = pd.read_sql(query,mydb)
        print(que_7_df)

              user_id  photo_id          created_at
        0           2         1 2023-05-14 20:36:41
        1           2         4 2023-05-14 20:36:41
        2           2         8 2023-05-14 20:36:41
        3           2         9 2023-05-14 20:36:41
        4           2        10 2023-05-14 20:36:41
        ...       ...       ...                 ...
        8777      100       245 2023-05-14 20:36:41
        8778      100       246 2023-05-14 20:36:41
        8779      100       248 2023-05-14 20:36:41
        8780      100       249 2023-05-14 20:36:41
        8781      100       255 2023-05-14 20:36:41

        [8782 rows x 3 columns]

 [337]: query = '''
        SELECT user_id
        FROM likes
        GROUP BY user_id
        HAVING COUNT(photo_id) = 257;
        '''
        que_7_df = pd.read_sql(query,mydb)
        print("Following Users Are BOT or Fake")
        print(que_7_df)

        Following Users Are BOT or Fake
            user_id
        0         5
        1        14
        2        21
        3        24
        4        36
        5        41
        6        54
        7        57
        8        66
        9        71
        10       75
        11       76
        12       91
```

Results:

The analysis's findings include information on the five oldest users, a list of users who have never posted a photo, the contest winner and their contact information, the top five most popular hashtags, and a recommendation for the best day to launch an advertising campaign based on user registration trends.