1-ljv-VodafoneZiggo-exploration

May 10, 2020

1 VodafoneZiggo Case

Author: Leonardo Vida

Email: lleonardovida@gmail.com

1.1 Introduction

This notebook shows how the datasets provided by the VodafoneZiggo (*VZ*) case were analysed and the processes used to gather the insights presented in the main document.

The VZ case provided us two tables belonging to the StackOverflow (*platform*) public dataset. These two tables are `posts` and `users` and are hosted on Big Query. The aim of this case is to provide the company with interesting insights on concerning these tables; in other words, we will use these table to create useful descriptions of: - User behavior on the platform. - Trends on the platform.

1.1.1 Import statements

Import general libraries

[2]:
```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import missingno as msno
from pandas_profiling import ProfileReport
import seaborn as sns
#from IPython.display import HTML, display
```

Import libraries related to Big Query

[3]:
```
import pandas_gbq
from google.cloud import bigquery
from google_auth_oauthlib import flow

# Load magic commands (%%biguery) from gc library
%load_ext google.cloud.bigquery
```

1.1.2 Setup connection with Big Query

Enable BigQuery API for the project from the console Setting up the redirect to the application from 8080 as I am using Jupyter notebook

[4]:
```
# From Google API documentation
launch_browser = True

appflow = flow.InstalledAppFlow.from_client_secrets_file(
   'conf/client_secret_718593840256-l3tnvboe9bsp00ar26q44j9tat1sruiu.apps.
 ↪googleusercontent.com.json',
    scopes=['https://www.googleapis.com/auth/bigquery'])

if launch_browser:
    appflow.run_local_server()
else:
    appflow.run_console()

credentials = appflow.credentials
```

```
Please visit this URL to authorize this application: https://accounts.google.com
/o/oauth2/auth?response_type=code&client_id=718593840256-l3tnvboe9bsp00ar26q44j9
tat1sruiu.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8080%
2F&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fbigquery&state=GA6304yjMaegOz
SHQO2aJYQ1ij7NQ8&access_type=offline
```

[5]:
```
# Create client
# Substitute with VZ project name
project = 'vz-assignments'
client = bigquery.Client(project=project, credentials=credentials)
```

2 Data description

The Big Query interface is used to read the schema of the StackOverflow dataset. The names and the type of the variables are used to infer meaning to the columns of each table. As an additional help, a post on StackOverflow[1] is also used to add additional information for columns that might have not clear meaninng [https://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede]

2.1 Table `Posts`

2.1.1 Connect and explore table

Explore the high-level features of the table `posts`

[6]:
```
ref_table_posts = 'vz-assignments.stackoverflow2015.posts'
table_posts = client.get_table(ref_table_posts)  # Make an API request.
print(
```

```
    "Table '{}.{}.{}'.".format(table_posts.project,
                               table_posts.dataset_id,
                               table_posts.table_id)
)
print("\n")
print("Table has {} rows".format(table_posts.num_rows))
print("Table 'posts' schema: ")
table_posts.schema
```

```
Table 'vz-assignments.stackoverflow2015.posts'.


Table has 5594392 rows
Table 'posts' schema:
```

[6]:
```
[SchemaField('id', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('title', 'STRING', 'NULLABLE', None, ()),
 SchemaField('accepted_answer_id', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('answer_count', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('comment_count', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('community_owned_date', 'TIMESTAMP', 'NULLABLE', None, ()),
 SchemaField('creation_date', 'TIMESTAMP', 'NULLABLE', None, ()),
 SchemaField('favorite_count', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('last_activity_date', 'TIMESTAMP', 'NULLABLE', None, ()),
 SchemaField('last_edit_date', 'TIMESTAMP', 'NULLABLE', None, ()),
 SchemaField('last_editor_display_name', 'STRING', 'NULLABLE', None, ()),
 SchemaField('last_editor_user_id', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('owner_display_name', 'STRING', 'NULLABLE', None, ()),
 SchemaField('owner_user_id', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('parent_id', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('post_type_id', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('score', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('tags', 'STRING', 'NULLABLE', None, ()),
 SchemaField('view_count', 'INTEGER', 'NULLABLE', None, ())]
```

Preview the first five lines of the "posts_answers" table, to understand the content of the table better.

[40]:
```
client.list_rows(table_posts, max_results=3).to_dataframe()
```

[40]:

	id	title	\
0	34016263	Real alternative for Google Feed API	
1	28321638	Strongly Typed RedirectToAction (Futures) usin…	
2	34054617	Magento 2 installing language packs (nl_NL tra…	

	accepted_answer_id	answer_count	comment_count	community_owned_date	\
0	NaN	0	13	None	
1	NaN	0	4	None	

```
2          34055069.0             1              0                  None

                     creation_date  favorite_count  \
0 2015-12-01 08:55:53.743000+00:00               9
1 2015-02-04 12:38:37.197000+00:00               3
2 2015-12-02 22:50:38.273000+00:00               3

                last_activity_date                   last_edit_date  \
0 2015-12-04 11:52:25.647000+00:00 2015-12-04 11:52:25.647000+00:00
1 2015-02-04 12:38:37.197000+00:00                               NaT
2 2015-12-17 08:06:23.237000+00:00 2015-12-11 03:52:32.467000+00:00

  last_editor_display_name  last_editor_user_id owner_display_name  \
0                     None            4671020.0               None
1                     None                  NaN               None
2                     None            1364007.0               None

   owner_user_id parent_id  post_type_id  score  \
0        4671020      None             1     15
1         826568      None             1      4
2        3215647      None             1      2

                                              tags  view_count
0            javascript|php|json|rss|google-feed-api        4240
1  c#|asynchronous|asp.net-mvc-5|asp.net-mvc-futures         425
2                      magento|magento2|magento-2.0        2938
```

From the schema printed above we can infer that this table provides us information concerning posts created on StackOverflow during 2015. At first glance, we can already notice few things: - Only the `title` is present, while the text of the question is not; - The `creation_date` might be useful to extract time-related information concerning the distribution of posts over hours/weekdays/months. - `Answer_count`, `comment_count` and the presence of an `accepted_answer_id` can provide a proxy of difficulty for the question posted. - The `favourite`, `score`, `view_count`, can be used to extract insights concerning which posts were the most appreciated ones. - Finally, the `tags` feature can be used to categorize posts and can provide interesting analyses when used in combination with all the above points.

We investigate the content of the table more in depth to check for missing values and unexpected values.

[116]:
```
# Select at random rows from the complete table
query_posts = (
    """
    SELECT *
    FROM `vz-assignments.stackoverflow2015.posts`
    ORDER BY RAND()
    LIMIT 5000
    """
```

```
)
# Save to pandas df
df_posts = client.query(query_posts, project="vz-assignments").to_dataframe()
```

[59]:
```
df_posts.head(5)
```

[59]:

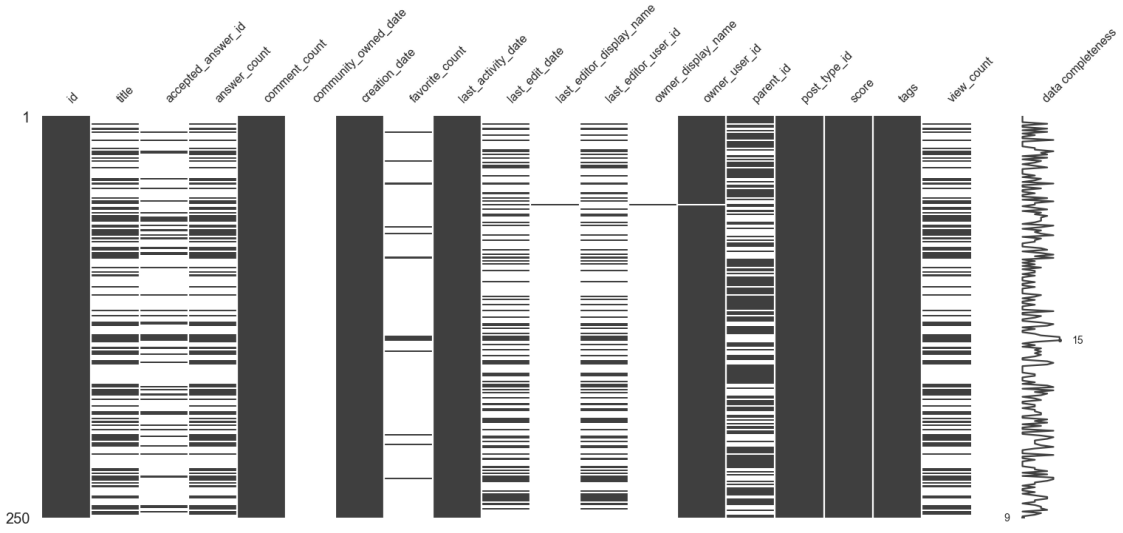
```
        id                                          title  accepted_answer_id  \
0  32785818                                           None                 NaN
1  29677242                                           None                 NaN
2  29830576                                           None                 NaN
3  30131510  How do I get the full name of the Xml Node          30137374.0
4  33293583                                           None                 NaN

   answer_count  comment_count community_owned_date  \
0           NaN              0                  NaT
1           NaN              1                  NaT
2           NaN              0                  NaT
3           1.0              4                  NaT
4           NaN              0                  NaT

                   creation_date  favorite_count  \
0 2015-09-25 15:38:30.537000+00:00             NaN
1 2015-04-16 14:08:05.300000+00:00             NaN
2 2015-04-23 17:34:18.923000+00:00             NaN
3 2015-05-08 19:31:33.427000+00:00             NaN
4 2015-10-23 01:46:09.590000+00:00             NaN

               last_activity_date last_edit_date last_editor_display_name  \
0 2015-09-25 15:38:30.537000+00:00            NaT                    None
1 2015-04-16 14:08:05.300000+00:00            NaT                    None
2 2015-04-23 17:34:18.923000+00:00            NaT                    None
3 2015-05-09 07:42:42.470000+00:00            NaT                    None
4 2015-10-23 01:46:09.590000+00:00            NaT                    None

   last_editor_user_id owner_display_name  owner_user_id   parent_id  \
0                  NaN               None      2621557.0  32755059.0
1                  NaN               None        11702.0  29676770.0
2                  NaN               None       293594.0  29829422.0
3                  NaN               None      4812504.0         NaN
4                  NaN               None      2647530.0  18726480.0

   post_type_id  score                          tags  view_count
0             2      0                                      NaN
1             2      2                                      NaN
2             2      2                                      NaN
3             1      0  c#|xml|xmlnode|xmltextreader        69.0
4             2      0                                      NaN
```

As it visible in the snippet of the table above, there are many missing values as marked by `None`, `NaN` and `NaT`. Further, it also seems that many posts do not contain any value concerning `tags` (i.e. were not tagged). The missing values need to be addressed further.

We quickly use the `pandas_profiling` and the `missingno` packages to create a profile report and further explore the table.

[158]:
```
# White is missing, black is present
msno.matrix(df_posts.sample(250), labels=True)
```

[158]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x126028f40>
```



[70]:
```
profile = df_posts.profile_report(
     title="Report on posts table",
     correlations={
         "pearson": {"calculate": False},
         "spearman": {"calculate": False},
         "kendall": {"calculate": False},
         "phi_k": {"calculate": False},
         "cramers": {"calculate": False},
     },
    missing_diagrams={
        'heatmap': True,
        'dendrogram': True,
    }
 )

#profile.to_widgets()
profile.to_notebook_iframe()
```

```
# Save the file
profile.to_file(output_file="report_posts.html")
```

```
HBox(children=(FloatProgress(value=0.0, description='Summarize dataset', max=26.0, style=Progre
```

```
HBox(children=(FloatProgress(value=0.0, description='Generate report structure', max=1.0, style
```

```
HBox(children=(FloatProgress(value=0.0, description='Render HTML', max=1.0, style=ProgressStyle
```

```
<IPython.core.display.HTML object>
```

```
HBox(children=(FloatProgress(value=0.0, description='Export report to file', max=1.0, style=Pro
```

Clearly, the table `posts` contains many missing values. The variable most relevant to our analysis with many missing values are the following:

- `favorite_count`
- `view_count`
- `answer_count`

2.2 Table `users`

We carry out a similar analysis to the one carried out for the table `posts`

[69]:
```
ref_table_users = 'vz-assignments.stackoverflow2015.users'
table_users = client.get_table(ref_table_users)  # Make an API request.
print(
    "Table '{}.{}.{}'.".format(table_users.project,
                               table_users.dataset_id,
                               table_users.table_id)
)
print("\n")
print("Table has {} rows".format(table_users.num_rows))
print("Table 'users' schema: ")
table_users.schema
```

```
Table 'vz-assignments.stackoverflow2015.users'.
```

```
Table has 10097978 rows
Table 'users' schema:
```

[69]:
```
[SchemaField('id', 'INTEGER', 'NULLABLE', None, ()),
 SchemaField('display_name', 'STRING', 'NULLABLE', None, ()),
 SchemaField('age', 'STRING', 'NULLABLE', None, ()),
 SchemaField('location', 'STRING', 'NULLABLE', None, ())]
```

Let us preview the first five lines of the `users` table

[70]:
```
client.list_rows(table_users, max_results=3).to_dataframe()
```

[70]:

	id	display_name	age	location
0	431965	Damien		Netherlands
1	5503898	Ravers		Portugal
2	8823852	Vishnu Baliga		Kochi, Kerala, India

Create a profile report to quickly explore the table

Clearly, the table `users` is a much smaller table containing only few variables associated to the users. However, two main potential insights are already noticeable:

- The number of rows of `users` is ~2X than the ones of `posts`. This might mean that these users are the sum of the users over the years *up to* 2015. We will not be able to check this hypothesis because we lack a variable referring the creation date of the account, but we can use this assumption to check how many of these users were active by connecting the `posts.last_editor_user_id` and `posts.owner_user_id` with the `users.id`.
- The `location` variable is available and can be used to group users by country/region.

Let us check once more for missing values.

[71]:
```
# Select at random rows from the complete table
query_users = (
    """
    SELECT *
    FROM `vz-assignments.stackoverflow2015.users`
    ORDER BY RAND()
    LIMIT 5000
    """
)
# Save to pandas df
df_users = client.query(query_users, project="vz-assignments").to_dataframe()
```

[74]:
```
profile = ProfileReport(df_users, title='Report on users table')
#profile.to_widgets()
profile.to_notebook_iframe()

# Save the file
profile.to_file(output_file="report_users.html")
```

```
HBox(children=(FloatProgress(value=0.0, description='Summarize dataset', max=11.0, style=Progre

HBox(children=(FloatProgress(value=0.0, description='Generate report structure', max=1.0, style

HBox(children=(FloatProgress(value=0.0, description='Render HTML', max=1.0, style=ProgressStyle

<IPython.core.display.HTML object>

HBox(children=(FloatProgress(value=0.0, description='Export report to file', max=1.0, style=Pro
```

It seems that many values in `location` do not contain any value.

[79]:
```
# Select at random rows from the complete table
query_null = (
    """
    SELECT
        SUM(CASE WHEN location IS NULL OR location = "" THEN 1 ELSE 0 END)␣
 ↪count_nulls
    FROM `vz-assignments.stackoverflow2015.users`
    """
)
# Save to pandas df
df_null = client.query(query_null, project="vz-assignments").to_dataframe()
```

[80]:
```
df_null
```

[80]:
```
   count_nulls
0      7600342
```

We see that for over 7.6 Million record there is no `location` specified.

2.3 Summary of data description

Given the preliminary insights we discovered above and the aim of the VZ case, we can formulate some questions that will be used in the next step of the analysis:

1. **Researching user behavior on the platform**
   - *Insights on question-related behavior*:
     - What is the percentage of questions that received an answer?
     - How are questions distributed across the months, weeks, day of the week?

  - How are questions distributed within the hours the day?
- *Insights on users*:
  - What are the most common location of the users?
  - Which were the most active users in 2015?

2. **Researching trends on the platform**
   - *Insights on questions-related trends*:
     - What are the most common tags?
     - What are the most common tags within the Data Science/Engineering community?
     - What are the tags with the highest/least number of view_counts?

3 Researching user behavior on the platform

3.1 Insights on question-related behavior

What is the percentage of questions that received an answer?

[7]:
```
query = (
    """
    SELECT
        COUNT(*) AS Number_of_Questions,
        ROUND(100 * SUM(IF(answer_count > 0, 1, 0)) / COUNT(*), 1) AS␣
↪Percent_Questions_with_Answers
    FROM `vz-assignments.stackoverflow2015.posts`
    """
)
df_answered = client.query(query, project="vz-assignments").to_dataframe()
```

[8]:
```
df_answered
```

[8]:
```
   Number_of_Questions  Percent_Questions_with_Answers
0              5594392                            33.9
```

[9]:
```
data = df_answered
data['Percent_Questions_with_Answers'] = data['Percent_Questions_with_Answers']␣
↪/ 100 * data['Number_of_Questions']
```

[10]:
```
data = data.rename(columns={"Percent_Questions_with_Answers":␣
↪"Questions_with_Answers",
                    "Number_of_Questions": "Questions_without_Answers"})
data['Questions_without_Answers'] = data['Questions_without_Answers'] -␣
↪data['Questions_with_Answers']
data
```

[10]:
```
   Questions_without_Answers  Questions_with_Answers
0               3697893.112             1896498.888
```

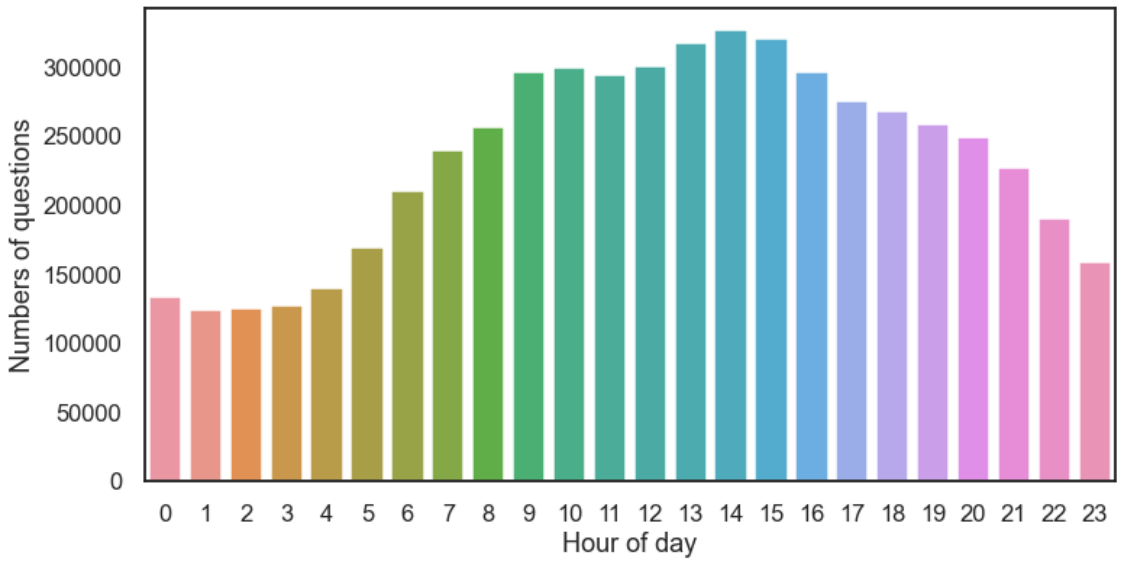
As we do not know when an answer was posted, we will continue analysing the `posts` interested in investigating the time distribution of posts.

Which hour of the day has most questions posted?

[20]:
```
query = """
    SELECT
        Hour_of_day,
        COUNT(1) AS Number_Questions,
    FROM (
        SELECT id AS question_id,
        EXTRACT(HOUR FROM creation_date) AS Hour_of_day,
      FROM `vz-assignments.stackoverflow2015.posts`
      GROUP BY question_id, Hour_of_day
      )
    GROUP BY Hour_of_day
    ORDER BY Hour_of_day;
    """
df_hourday = client.query(query, project="vz-assignments").to_dataframe()
```

[24]:
```
sns.set(style="white", context="talk")
plt.figure(figsize=(12, 6))
test = sns.barplot(x="Hour_of_day", y="Number_Questions", data=df_hourday)
test.set_ylabel("Numbers of questions")
test.set_xlabel("Hour of day")
```

[24]:
```
Text(0.5, 0, 'Hour of day')
```



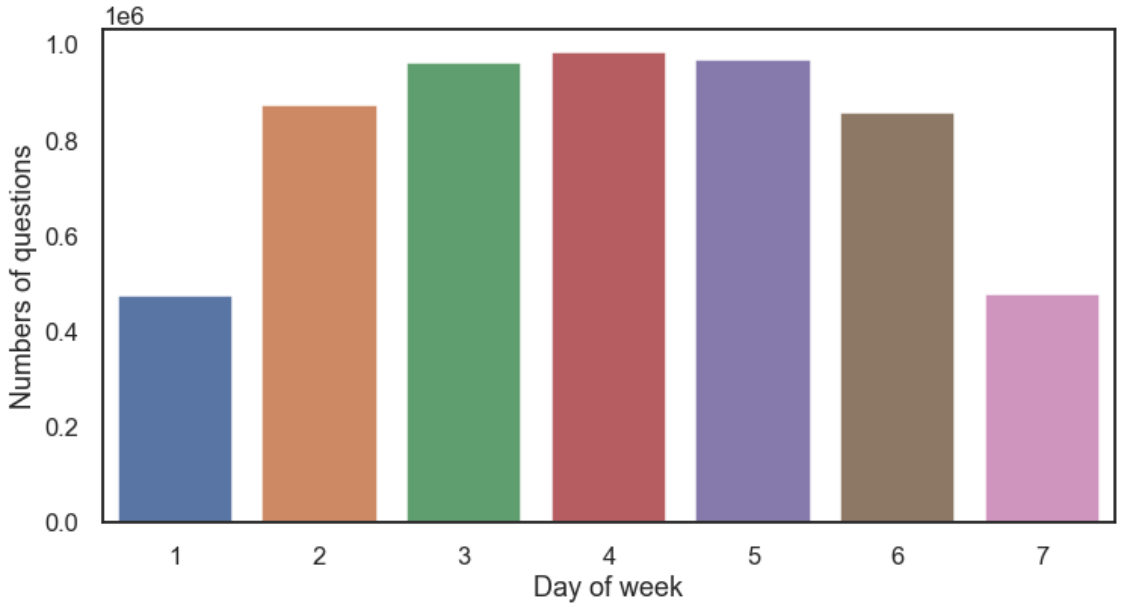
Which day of the week has most questions posted?

[11]:
```
query = """
    SELECT
```

```
        Day_of_Week,
        COUNT(1) AS Number_Questions,
    FROM (
        SELECT id AS question_id,
        EXTRACT(DAYOFWEEK FROM creation_date) AS day_of_week,
      FROM `vz-assignments.stackoverflow2015.posts`
      GROUP BY question_id, day_of_week
      )
    GROUP BY Day_of_Week
    ORDER BY Day_of_Week;
    """
df_dayweek = client.query(query, project="vz-assignments").to_dataframe()
```

[25]:
```
sns.set(style="white", context="talk")
plt.figure(figsize=(12, 6))
test = sns.barplot(x="Day_of_Week", y="Number_Questions", data=df_dayweek)
test.set_ylabel("Numbers of questions")
test.set_xlabel("Day of week")
```

[25]: Text(0.5, 0, 'Day of week')

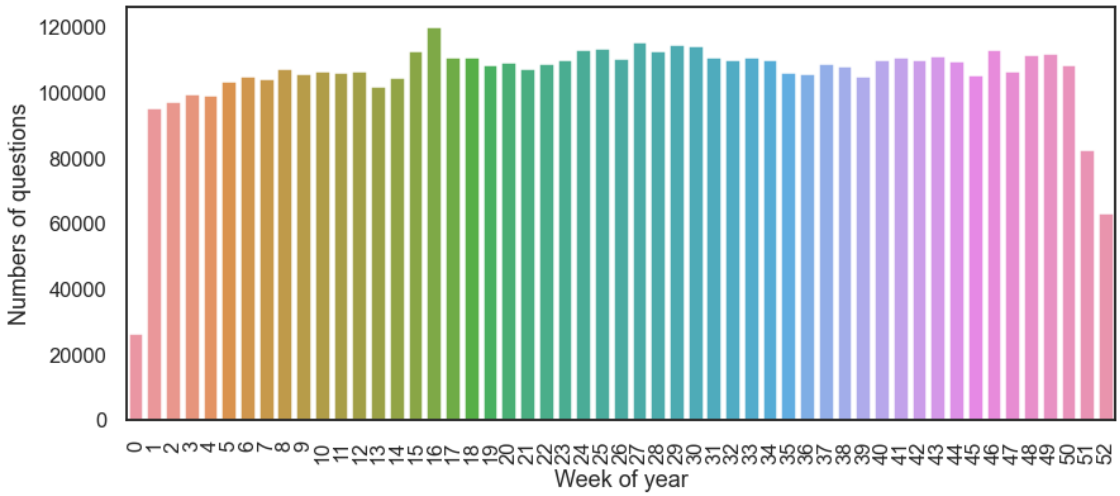


Which week has most questions posted?

[26]:
```
query = """
    SELECT
        Week,
        COUNT(1) AS Number_Questions,
```

```
    FROM (
        SELECT id AS question_id,
        EXTRACT(WEEK FROM creation_date) AS Week,
      FROM `vz-assignments.stackoverflow2015.posts`
      GROUP BY question_id, Week
      )
    GROUP BY Week
    ORDER BY Week;
    """
df_week = client.query(query, project="vz-assignments").to_dataframe()
```

[44]:
```
sns.set(style="white", context="talk")
plt.figure(figsize=(14, 6))

chart = sns.barplot(x="Week",
                    y="Number_Questions",
                    data=df_week,
                    dodge=False)
chart.set_ylabel("Numbers of questions")
chart.set_xlabel("Week of year")
chart.set_xticklabels(chart.get_xticklabels(), rotation=90)

plt.show()
```

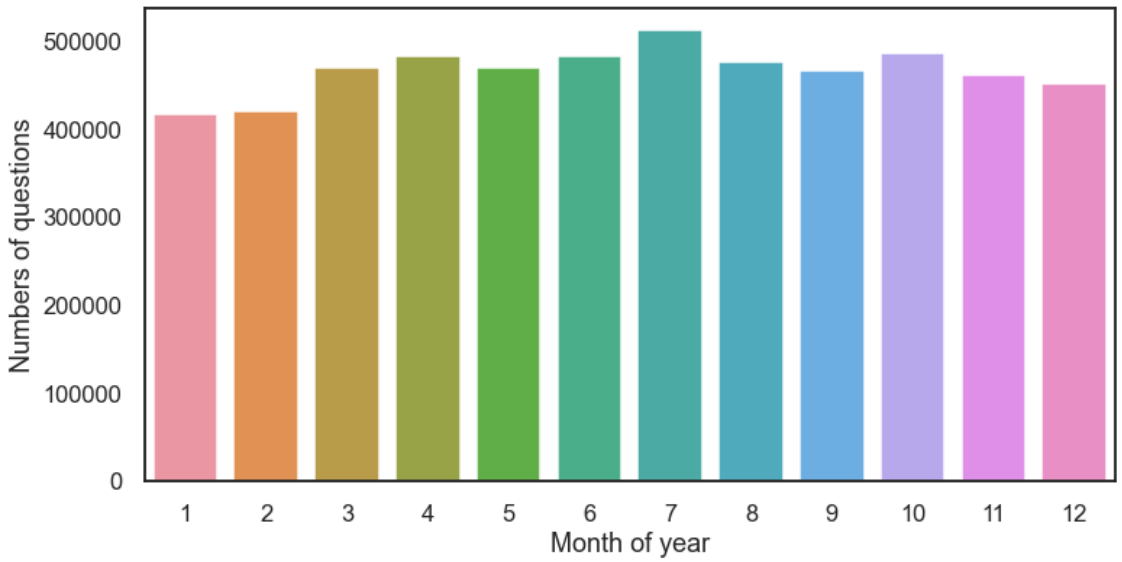


Which month has most questions posted?

[45]:
```
query = """
    SELECT
        Month,
        COUNT(1) AS Number_Questions,
```

```
    FROM (
        SELECT id AS question_id,
        EXTRACT(MONTH FROM creation_date) AS Month,
      FROM `vz-assignments.stackoverflow2015.posts`
      GROUP BY question_id, Month
      )
    GROUP BY Month
    ORDER BY Month;
    """
df_month = client.query(query, project="vz-assignments").to_dataframe()
```

[48]:
```
sns.set(style="white", context="talk")
plt.figure(figsize=(12, 6))

chart = sns.barplot(x="Month",
                    y="Number_Questions",
                    data=df_month,
                    dodge=False)
chart.set_ylabel("Numbers of questions")
chart.set_xlabel("Month of year")
#chart.set_xticklabels(chart.get_xticklabels(), rotation=90)

plt.show()
```



From the graphs above, we can state that: - Most questions are posted in the interval of time from 10-16, which makes sense when compared to working day (9-17). - Most questions are also posted during the week, which also validates the theory that people ask questions during their working-times. - Concerning the months and week of posting, there is less seasonality in the trend,

for exception of the last week of December and the firsts of January, where the majority of the users on StackOverflow seem to be taking holidays. - Interestingly, one can see a peak around week 15-17, which could be related to the end of the first quarter in many big organizations.

3.2 Insights on users

What are the most common locations of users?

[96]:
```
query = """
    SELECT location as Location,
        SUM(CASE WHEN location IS NULL OR location = "" THEN 0 ELSE 1 END) AS␣
 ↪Count
    FROM `vz-assignments.stackoverflow2015.users`
    GROUP BY Location
    ORDER BY Count DESC;
    """
df_location_gen = client.query(query, project="vz-assignments").to_dataframe()
```

[96]:

	Location	Count
0	India	57800
1	Bangalore, Karnataka, India	41231
2	Germany	28234
3	Hyderabad, Telangana, India	24238
4	Pune, Maharashtra, India	23084
…	…	…
173326	Jamhuri, Nairobi, Kenya	1
173327	tunder bay, ontario, canada.	1
173328	Handwara J&K, India	1
173329	Lyon, Franciaország	1
173330		0

[173331 rows x 2 columns]

[98]:
```
df_location_gen.head(10)
```

[98]:

	Location	Count
0	India	57800
1	Bangalore, Karnataka, India	41231
2	Germany	28234
3	Hyderabad, Telangana, India	24238
4	Pune, Maharashtra, India	23084
5	Chennai, Tamil Nadu, India	21337
6	China	20185
7	London, United Kingdom	19830
8	United States	17404
9	France	16674
10	Mumbai, Maharashtra, India	14912
11	Paris, France	14874

```
12                          USA  13947
13  Bengaluru, Karnataka, India  13708
14                  Philippines  13430
15               United Kingdom  12923
16                    Indonesia  12213
17                       Canada  11623
18                    Singapore  11604
19                  Delhi, India  11372
```

As we see, there are many overlaps concerning users that live in India. We therefore use the top locations and fetch them individually, although the method is not very efficient (it could be done with parameters)

[112]:
```
q_in = """
    SELECT
        SUM(CASE WHEN location IS NULL OR location = "" THEN 0 ELSE 1 END) AS␣
 ↪Count
    FROM `vz-assignments.stackoverflow2015.users`
    WHERE Location LIKE "%India%"
    ORDER BY Count DESC;
    """

q_de = """
    SELECT
        SUM(CASE WHEN location IS NULL OR location = "" THEN 0 ELSE 1 END) AS␣
 ↪Count
    FROM `vz-assignments.stackoverflow2015.users`
    WHERE Location LIKE "%Germany%"
    ORDER BY Count DESC;
    """

q_ch = """
    SELECT
        SUM(CASE WHEN location IS NULL OR location = "" THEN 0 ELSE 1 END) AS␣
 ↪Count
    FROM `vz-assignments.stackoverflow2015.users`
    WHERE Location LIKE "%China%"
    ORDER BY Count DESC;
    """

q_us = """
    SELECT
        SUM(CASE WHEN location IS NULL OR location = "" THEN 0 ELSE 1 END) AS␣
 ↪Count
    FROM `vz-assignments.stackoverflow2015.users`
    WHERE Location LIKE "%United States%"
    ORDER BY Count DESC;
```

```
    """

df_loc_in = client.query(q_in, project="vz-assignments").to_dataframe()
df_loc_ch = client.query(q_ch, project="vz-assignments").to_dataframe()
df_loc_us = client.query(q_us, project="vz-assignments").to_dataframe()
df_loc_de = client.query(q_de, project="vz-assignments").to_dataframe()
```

[113]:
```
print("Indian users: ", df_loc_in.iloc[0]["Count"], "\n",
      "German users: ", df_loc_de.iloc[0]["Count"], "\n",
      "Chinese users: ", df_loc_ch.iloc[0]["Count"], "\n",
     "USA users: ", df_loc_us.iloc[0]["Count"])
```

```
Indian users:  398893
 German users:  57777
 Chinese users:  48209
 USA users:  196465
```

Which users posted the highest number of questions?

[138]:
```
query = """
SELECT
     u.id AS User_id,
     COUNT(1) AS Number_of_posts
 FROM `vz-assignments.stackoverflow2015.users` AS u
 INNER JOIN `vz-assignments.stackoverflow2015.posts` AS p
     ON u.id = p.owner_user_id
 GROUP BY User_id
 ORDER BY Number_of_posts DESC
 LIMIT 50
 """
df_users_posts= client.query(query, project="vz-assignment").to_dataframe()
```

[145]:
```
df_users_posts.head(3)
```

[145]:

	User_id	Number_of_posts
0	1144035	7570
1	548225	4300
2	3732271	3392

[143]:
```
query = """
    SELECT
        COUNT(1) AS Num_Users,
        ROUND(AVG(number_posts)) AS Avg_Num_Posts
    FROM (
        SELECT
            u.id AS user_id,
            COUNT(1) as number_posts
        FROM `vz-assignments.stackoverflow2015.posts` p
```

```
        JOIN `vz-assignments.stackoverflow2015.users` u
            ON u.id = p.owner_user_id
        GROUP BY user_id
        );
    """
df_avg_users_posts = client.query(query, project="vz-assignment").to_dataframe()
```

[144]:
```
df_avg_users_posts
```

[144]:

	Num_Users	Avg_Num_Posts
0	1012870	5.0

Using the query above, we see that among the users that posts questions, 5 questions per year were posted in 2015.

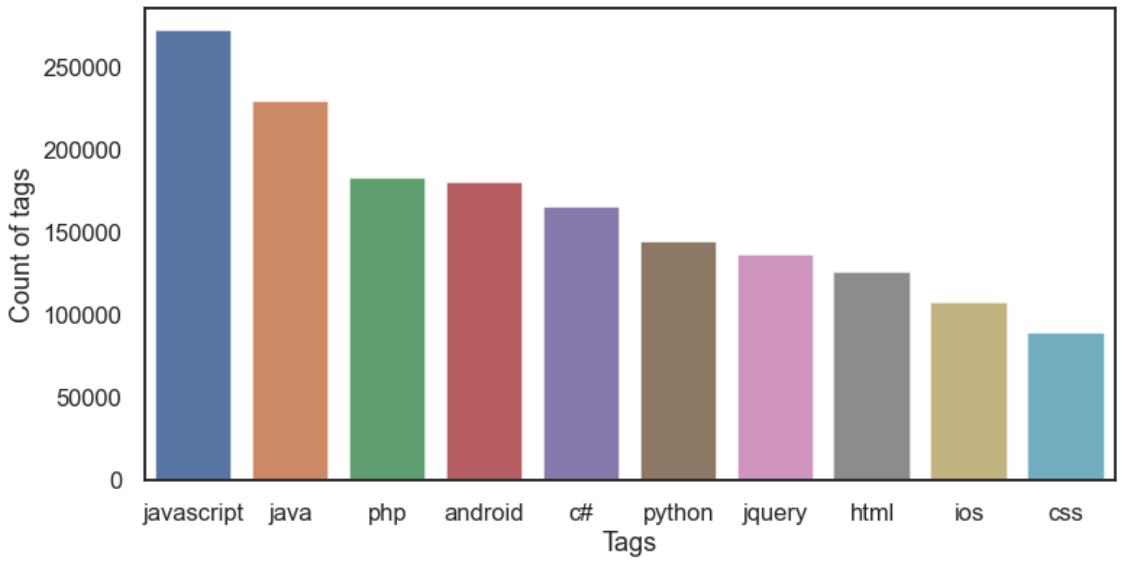
3.3 Researching trends on the platform

3.3.1 Insights on questions-related trends

What are the most common tags?

[150]:
```
query = """
    SELECT
        tag as Tag,
        COUNT(*) AS Count
    FROM (
        SELECT SPLIT(tags, '|') tags
        FROM `vz-assignments.stackoverflow2015.posts`
        ),
        UNNEST(tags) Tag
    GROUP BY Tag
    HAVING Tag != ""
    ORDER BY Count
    DESC LIMIT 10
"""
df_common_tags = client.query(query, project="vz-assignment").to_dataframe()
```

[152]:
```
sns.set(style="white", context="talk")
plt.figure(figsize=(12, 6))

chart = sns.barplot(x="Tag",
                    y="Count",
                    data=df_common_tags,
                    dodge=False)
chart.set_ylabel("Count of tags")
chart.set_xlabel("Tags")
#chart.set_xticklabels(chart.get_xticklabels(), rotation=90)

plt.show()
```



Count number of posts per user per Tag = "python" As an example, we can retrieve the user who posted the most questions containing the tag "python"

[153]:
```
query = """
 SELECT
     p.owner_user_id AS user_id,
     COUNT(1) AS Number_of_posts
 FROM `vz-assignments.stackoverflow2015.users` AS u
 INNER JOIN `vz-assignments.stackoverflow2015.posts` AS p
     ON u.id = p.owner_user_id
 WHERE p.tags LIKE '%python%'
 GROUP BY p.owner_user_id
 ORDER BY Number_of_posts DESC
 LIMIT 50
 """
df_post_per_tag = client.query(query, project="vz-assignment").to_dataframe()
```

[155]:
```
df_post_per_tag.head(5)
```

[155]:

	user_id	Number_of_posts
0	651174	203
1	308827	136
2	1389110	108
3	578822	102
4	2242044	94

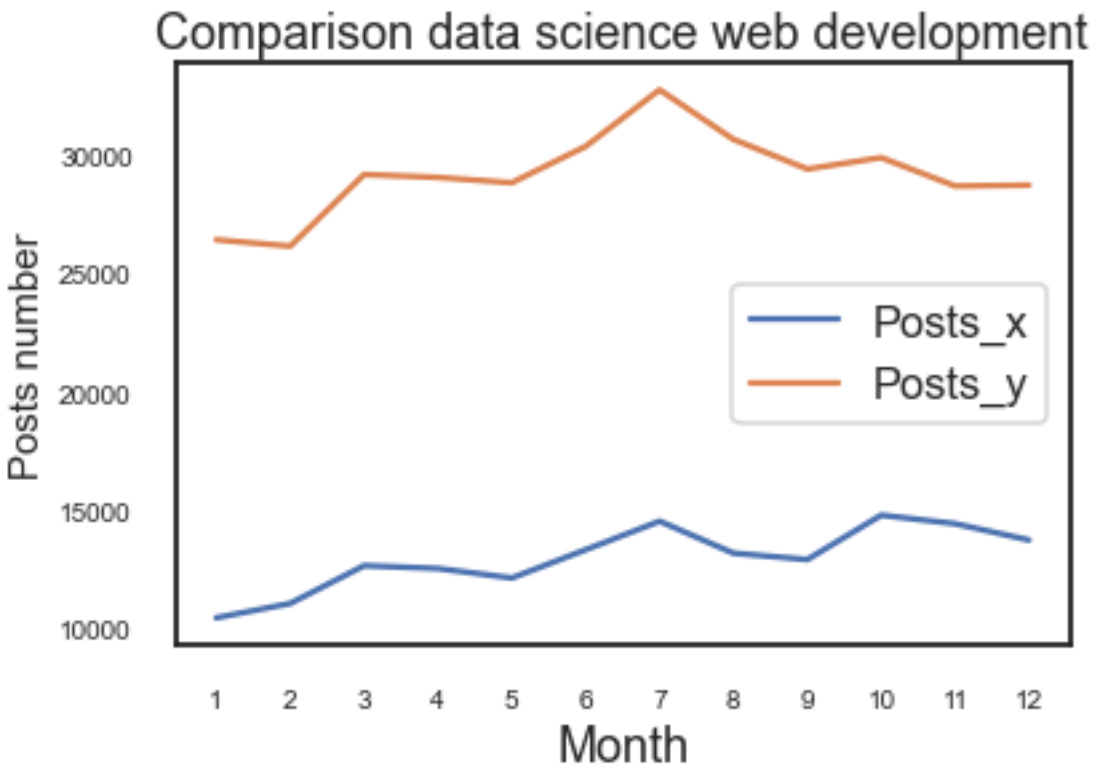
How do tags evolve in the community? Here we compare the evolution of the usage of the tags belonging to data science vs. those belonging to web development

[176]:
```
q_ml = """
    SELECT
        EXTRACT(MONTH FROM creation_date) AS Month,
        COUNT(id) AS Posts
    FROM `vz-assignments.stackoverflow2015.posts`
    WHERE (
        tags like '%python%' OR
        tags like '%R%'
    )
    GROUP BY Month
    ORDER BY Month;
    """
q_de = """
    SELECT
        EXTRACT(MONTH FROM creation_date) AS Month,
        COUNT(id) AS Posts
    FROM `vz-assignments.stackoverflow2015.posts`
    WHERE (
        tags like '%Javascript%' OR
        tags like '%php%' OR
        tags like '%css%' OR
        tags like '%html%'
    )
    GROUP BY Month
    ORDER BY Month;
    """
df_common_dsml = client.query(q_ml, project="vz-assignment").to_dataframe()
df_common_dsde = client.query(q_de, project="vz-assignment").to_dataframe()
```

[177]:
```
dsde = pd.merge(df_common_dsml, df_common_dsde, how='inner', on = 'Month')
```

[181]:
```
dsde.rename(columns = {"Posts_x": "Data_science", "Posts_y": "Web_development"})
dsde = dsde.set_index('Month')
```

[188]:
```
dsde.plot(kind='line')
plt.xlabel('Month')
plt.ylabel('Posts number', fontsize=15)
y_pos=[1,2,3,4,5,6,7,8,9,10,11,12]

plt.xticks(y_pos,fontsize=10)
plt.yticks(fontsize=10)
plt.title('Comparison data science web development')
plt.show()
```



3.4 Recommendation

The company could create a system that, given a user id and a topic, enables their HR department to search whether an interviewee posted questions concerning that topic on StackOverflow. An **extremely** basic MVP of this system takes the form of the following function.

[218]:
```
def interviewee_finder(user_id, topic, client):
    my_query = """
    SELECT
         p.owner_user_id AS user_id,
         COUNT(1) AS Number_of_posts
     FROM `vz-assignments.stackoverflow2015.posts` AS p
     INNER JOIN `vz-assignments.stackoverflow2015.users` AS u
         ON u.id = p.owner_user_id
     WHERE p.tags LIKE @topic AND p.owner_user_id = @user_id
     GROUP BY p.owner_user_id
     LIMIT 5
     """

    job_config = bigquery.QueryJobConfig(
    query_parameters=[
        bigquery.ScalarQueryParameter("topic", "STRING", "python"),
```

```
        bigquery.ScalarQueryParameter("user_id", "STRING", "3545988"),
        ]
    )
    query_job = client.query(query, job_config=job_config)

    results = query_job.to_dataframe()

    return results
```