REPORT ON DAILY ENGAGEMENT UDACITY DATASET

By/ Mostafa Noaman Ahmed

1. Introduction

Udacity is a really famous platform for different learning fields especially the technical topics which are related to programming in general, and “Data Science” in particular, we are here trying to analyze the daily engagement dataset to got the knowledge about the most important topics and to know each user interests to meet their needs.

Business Problems We are Trying to Solve

- The most nanodegrees that users are interested in.
- The preference of each user per the nanodegrees.
- The different and common users for the same nanodegrees.
- The studying statistics related to each user along the year.

Main Goal

- Create an analytical framework to understand
    - User interests
    - Type of Topics that most users prefer

About Dataset

The dataset we’ve consisted of records of users’ daily engagement to different Udacity Nanodegrees. It has various columns which describe the daily engagement such as each user account key, course title, total minutes spent, number of lessons

completed, number of projects completed, etc. Such a dataset is very useful for Udacity company and the content creators works in it, on which different analysis can be done to get insights about the users and also the popular courses which attracts users

This is what the dataset looks like:

	utc_date	acct	registration_date	subscription_start	course_key	sibling_key	course_title	has_visited	total_minutes_visited	lessons_completed	projects_completed
0	2014-11-05	448	2014-08-05	2014-11-05	ud359-nd	ud359	Intro to Data Science	0.0	0.0	0.0	0.0
1	2014-11-05	448	2014-08-05	2014-11-05	ud120-nd	ud120	Intro to Machine Learning	0.0	0.0	0.0	0.0
2	2014-11-05	448	2014-08-05	2014-11-05	ud651-nd	ud651	Data Analysis with R	0.0	0.0	0.0	0.0
3	2014-11-05	448	2014-08-05	2014-11-05	ud507-nd	ud507	Data Visualization and D3.js	0.0	0.0	0.0	0.0
4	2014-11-05	448	2014-08-05	2014-11-05	ud651	ud651	Data Analysis with R	0.0	0.0	0.0	0.0

Table1- DataSet

It has lots of rows and columns, which is 2309239 rows spread across 12 columns. Here is the list of columns this dataset has:

```
data.columns
```

```
Index(['utc_date', 'acct', 'registration_date', 'subscription_start',
       'course_key', 'sibling_key', 'course_title', 'has_visited',
       'total_minutes_visited', 'lessons_completed', 'projects_completed',
       'account_key'],
      dtype='object')
```

Table 2 - Columns in Dataset

2. EDA & Business Implication

EDA stands for exploratory data analysis where we explore our data and grab insights from it. EDA helps us in getting knowledge in form of various plots and diagrams where we can easily understand the data and its features.

Analysis Of the Data

	count	mean	std	min	25%	50%	75%	max
acct	2309239.0	449.644054	321.561123	0.0	201.0	394.0	596.0	1305.000000
has_visited	2309239.0	0.020122	0.140417	0.0	0.0	0.0	0.0	1.000000
total_minutes_visited	2309239.0	1.448495	15.255230	0.0	0.0	0.0	0.0	737.106143
lessons_completed	2309239.0	0.007599	0.112818	0.0	0.0	0.0	0.0	16.000000
projects_completed	2309239.0	0.000397	0.019923	0.0	0.0	0.0	0.0	1.000000

Table 3- Checking of the Skewness in Data

Observation:

acct: account key for each user value is in 0 - 1305 range. As Mean is greater than Median data is rightly skewed.

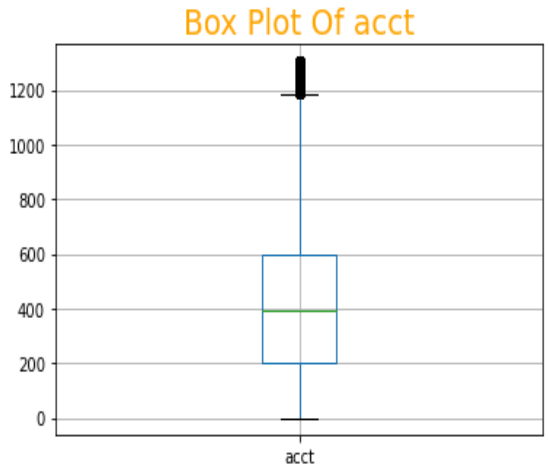
has_visited: the number of visits is in 0 – 1 range. As Mean ~ Median it's almost Normal Distributed.

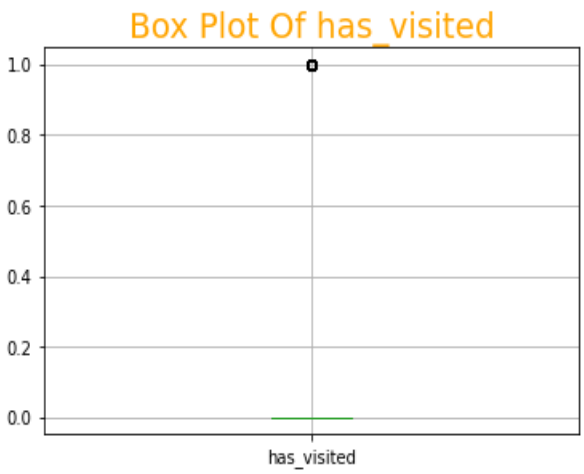
total_minutes_visited: the number of visits is in 0 – 737 range. As Mean is greater than Median data is rightly skewed.

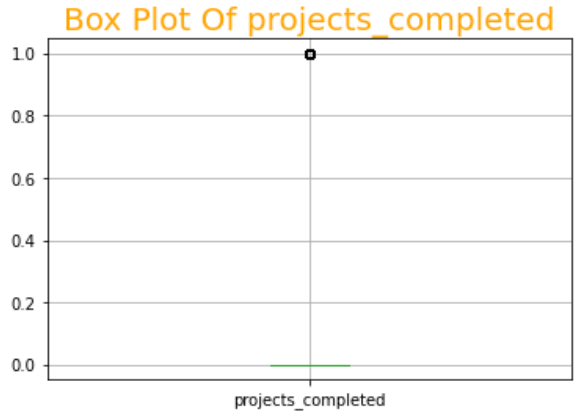
lessons_completed: the number of competed lessons is in 0 – 16 range. As Mean ~ Median it's almost Normal Distributed.

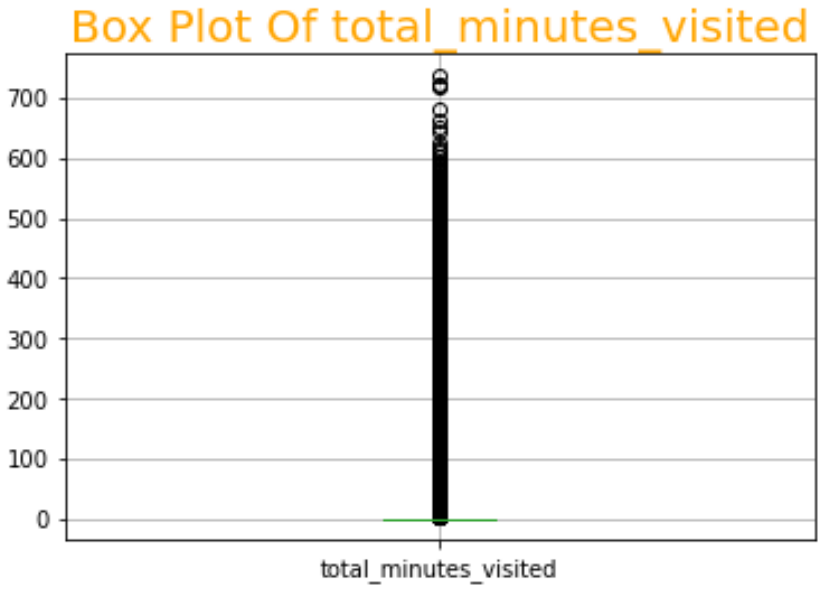
projects_completed: the number of competed projects is in 0 – 1 range. As Mean ~ Median it's almost Normal Distributed.

Uni-variate Analysis - Box Plot









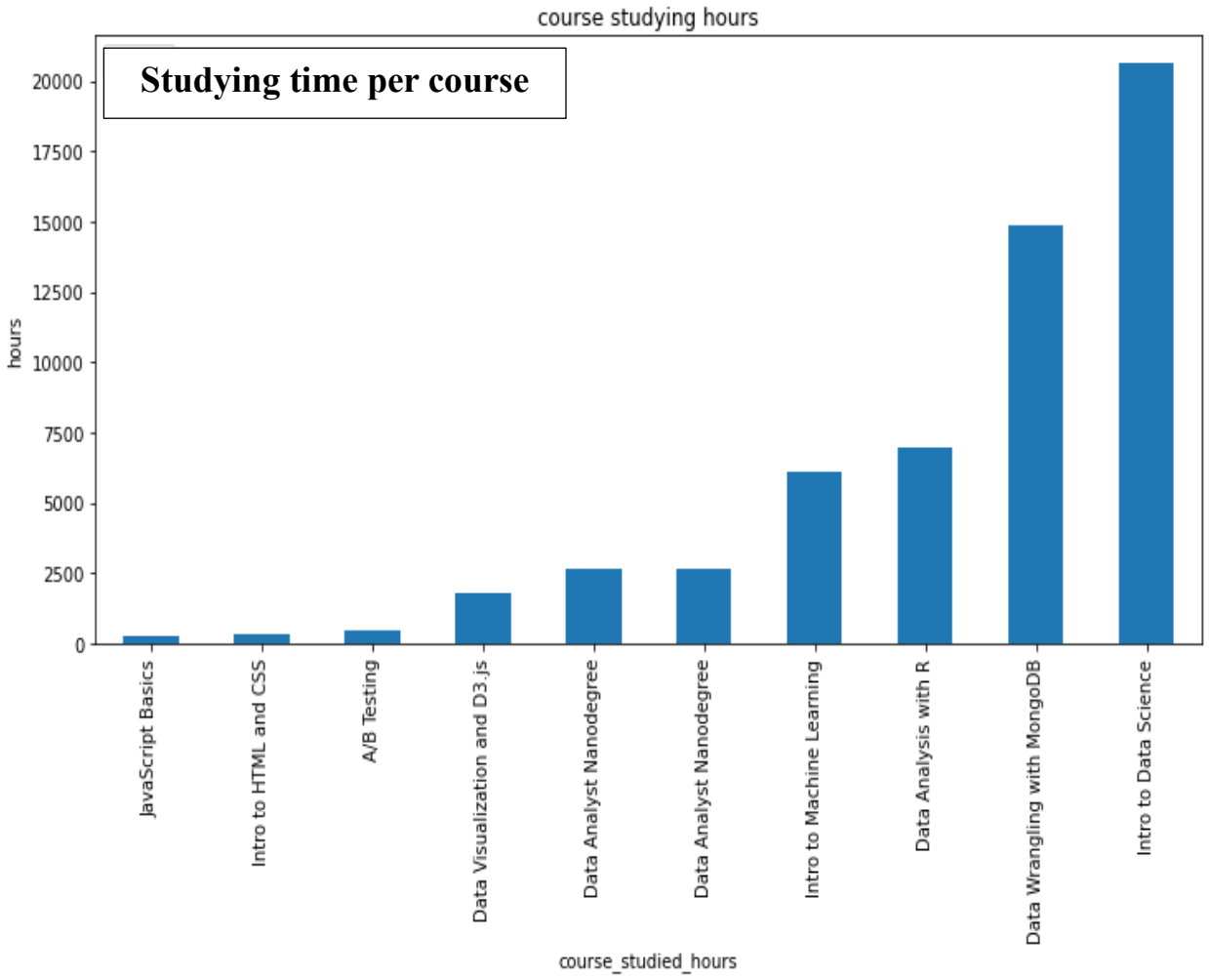
Plot 1- Boxplot

Observation:

Most of features are having outliers**.**

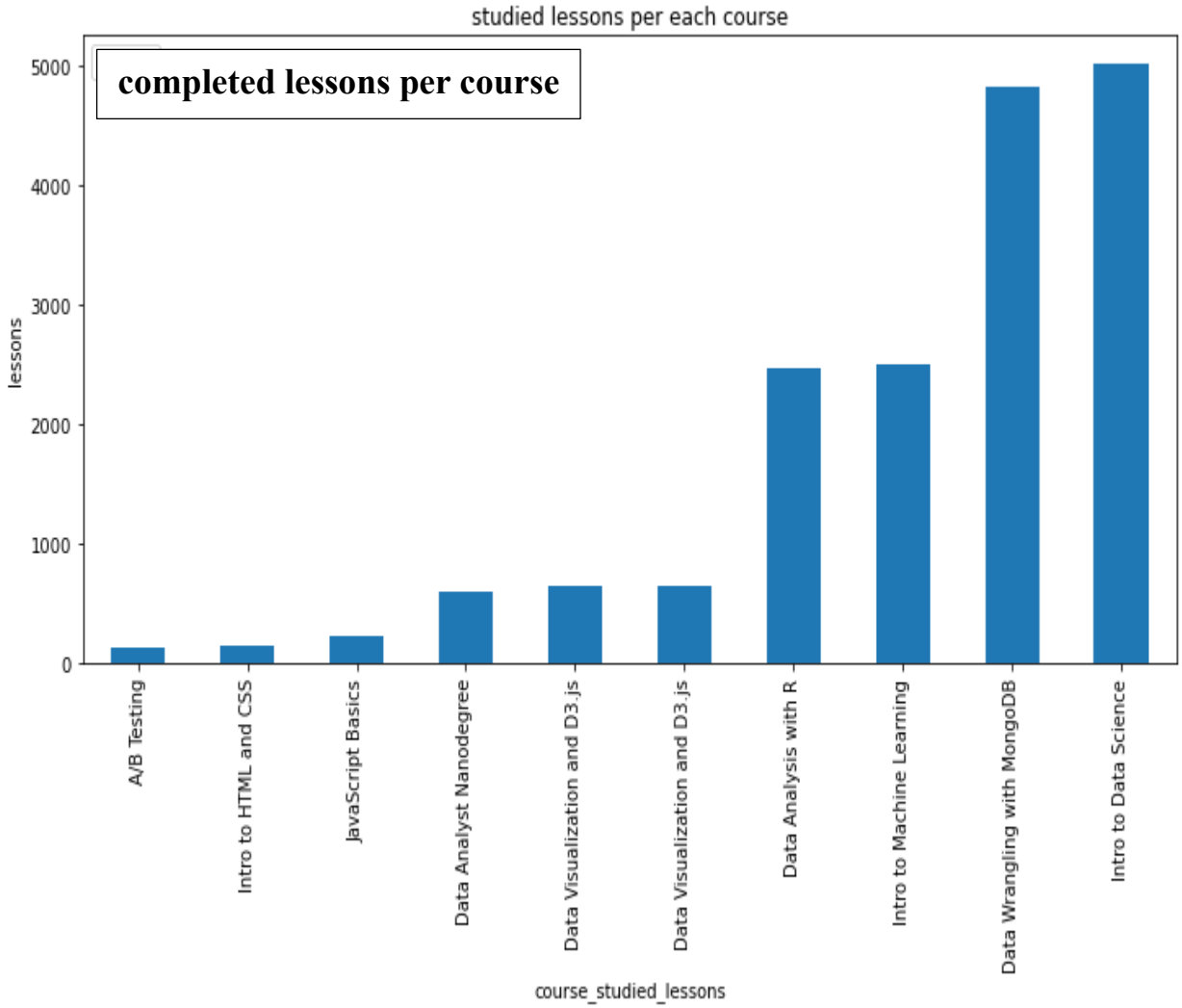
Uni-Variate Analysis of Important relations – Bar Plot

- Analysis of time spent for studying every course:



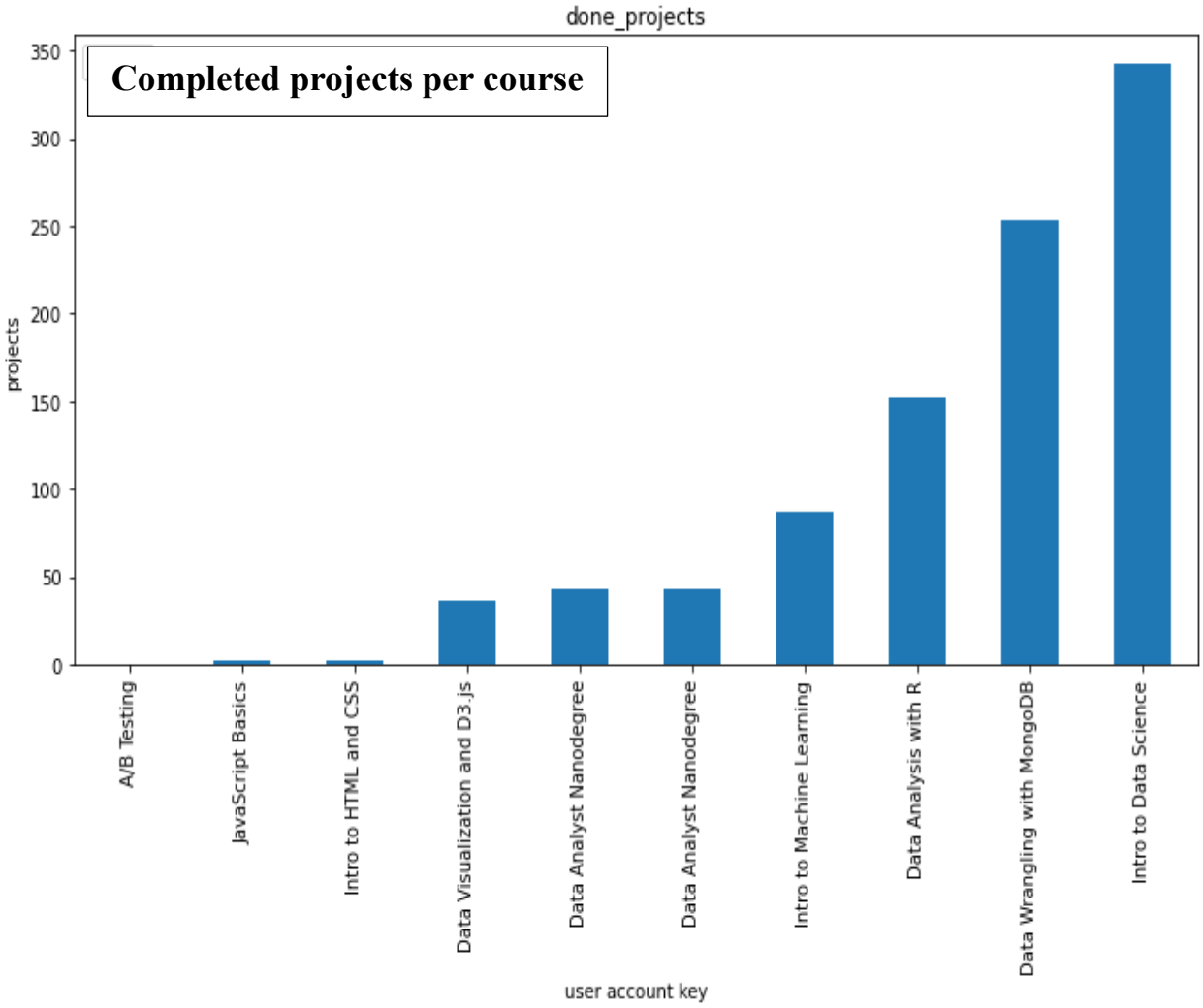
According to studying hours, “Intro to Data Science” is the most studied course.

- Analysis of lessons completed for every course:



According to lessons done, “Intro to Data Science” is the most studied course.

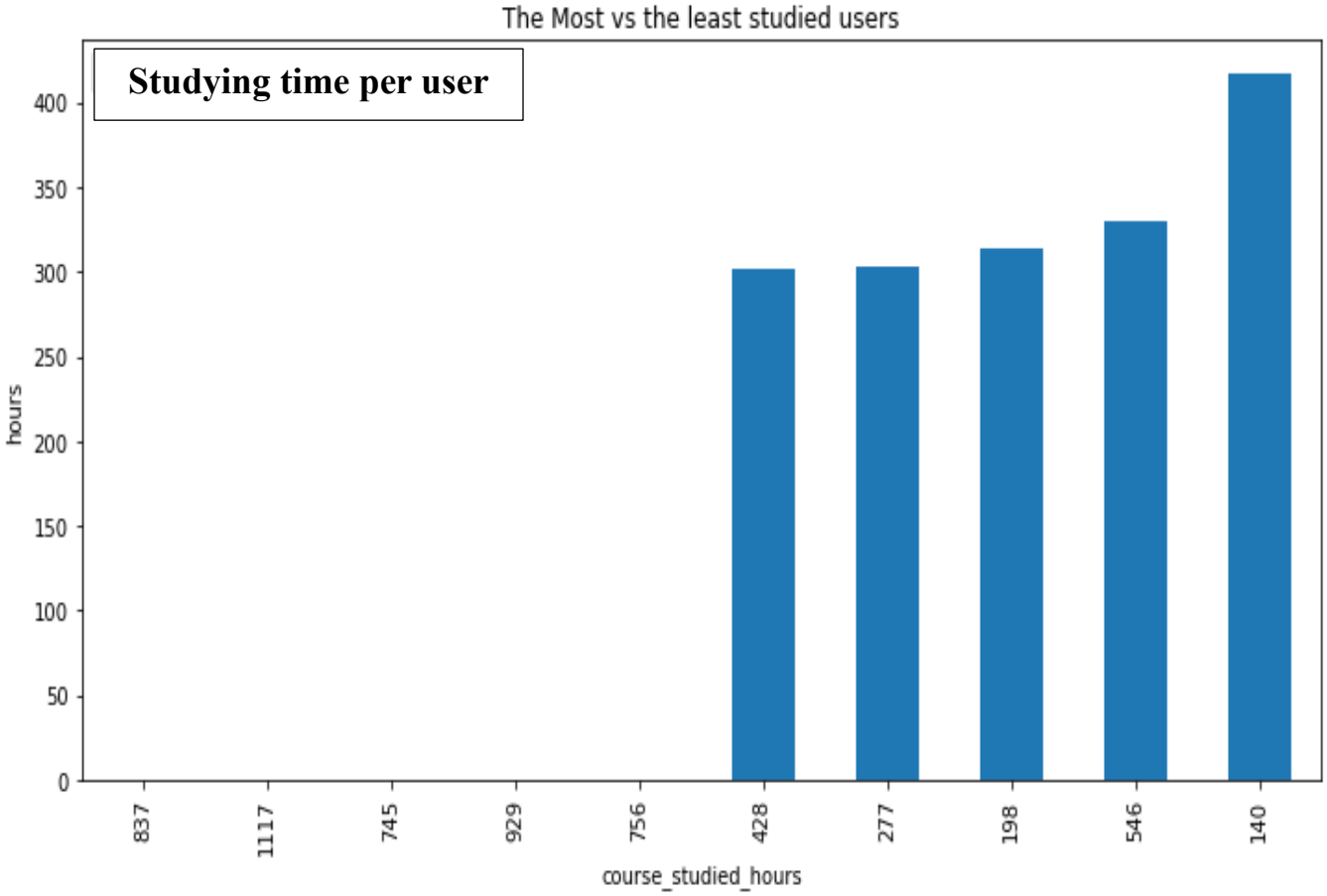
- Analysis of projects completed for every course:



According to projects done, “Intro to Data Science” is the most studied course.

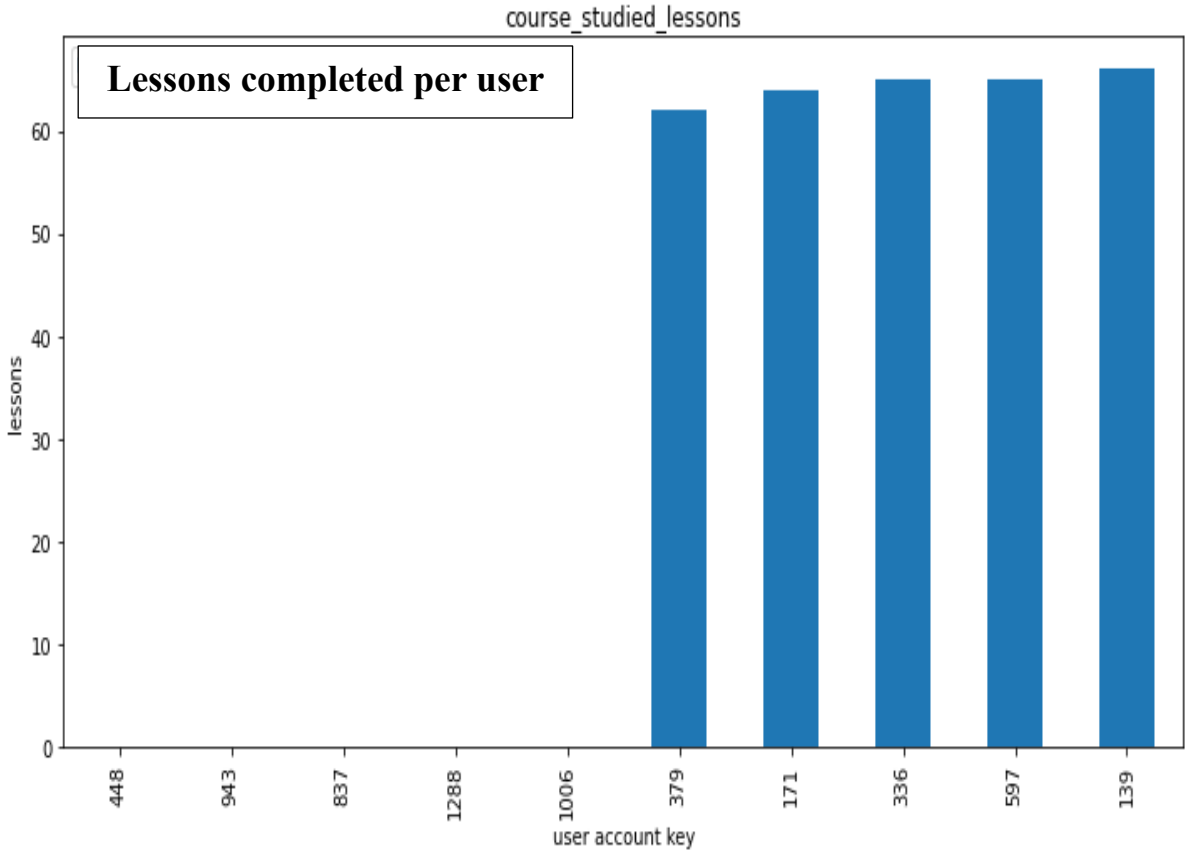
- “Intro to Data Science” course is the most popular course for all users.

- Analysis of time spent for studying per user:



According to studying hours, user whose account key equals to 140 is the most user studied.

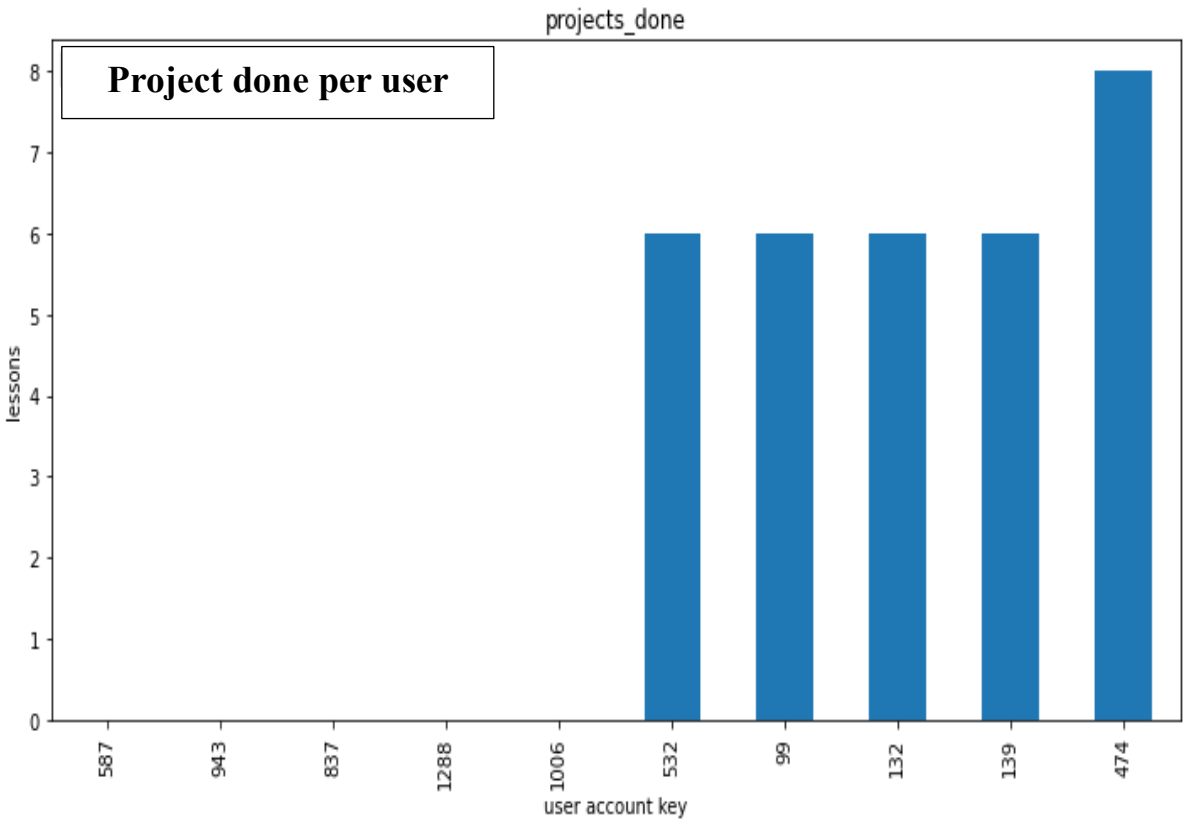
- Analysis of lessons completed for every course:



Plot 2- Barplot

According to lessons done, user whose account key equals to 139 is the most user done lessons.

- Analysis of projects completed for every course:



According to projects done, user whose account key equals to 474 is the most user applied projects.

- Users whose account key equals 140, 139 and 474 are special users for Udacity platform, which allows them to take more discounts on courses they studying.

3. User progress per each year

To get the details of each student in order to meet their needs it's good to calculate the studied hours , lessons and projects per each course of the courses we have in the "course title".

The code below collects each student data into a dictionary and creates a nested dictionary contains each year as a key and the corresponding data in detailed about every user.

Metadata

```
# to get the time spent per user for each of the courses across each year
def users_metadata(df, cols):
    total_yrs = {}
    lst_of_all_cols = []    # will be used in extraction of values from the df using regex

    for yr in pd.DatetimeIndex(df[cols[0]]).year.unique():
        total_all = {}
        print(f"\nfor year {yr}:")
        for acct in df[cols[1]].unique():
            courses_hrs = {}
            courses_lesn = {}
            courses_proj = {}
            total_hrs = 0
            for course in df[cols[2]].unique():
                c_hrs = course+'_hrs'
                c_lsns = course+'_lesson'
                c_proj = course+'_proj'

                mins = df[cols[3]][(pd.DatetimeIndex(df[cols[0]]).year==yr) & (df[cols[1]]==acct) & (df[cols[2]]==course)].sum()
                lessons = df[cols[4]][(pd.DatetimeIndex(df[cols[0]]).year==yr) & (df[cols[1]]==acct) & (df[cols[2]]==course)].sum
                projects = df[cols[5]][(pd.DatetimeIndex(df[cols[0]]).year==yr) & (df[cols[1]]==acct) & (df[cols[2]]==course)].su
                courses_hrs[c_hrs] = mins/60
                courses_lesn[c_lsns] = lessons
                courses_proj[c_proj] = projects
                total_hrs += mins/60

            total_all[acct] = {cols[1]:acct, cols[0]:yr, "total_study_hrs":total_hrs, cols[4]:lessons, cols[5]:projects, 'c_hrs'
            #print(total_all[acct])
            print(f"\tuser {acct} done...")
        total_yrs[yr] = total_all

    lst_of_all_cols.extend((cols[1], cols[0], "total_study_hrs", cols[4], cols[5]))
    lst_of_cols = [va for va in list(courses_hrs) + list(courses_lesn) + list(courses_proj)]
    lst_of_all_cols = lst_of_all_cols + lst_of_cols

    return total_yrs, lst_of_all_cols
```

Then we convert the output from a dictionary to a dataframe:

```
users_study_df = pd.DataFrame.from_dict(metadata, orient='index').T
```

```
users_study_df
```

	2014	2015
448	{'account_key': 448, 'subscription_start': 201...	{'account_key': 448, 'subscription_start': 201...
44	{'account_key': 44, 'subscription_start': 2014...	{'account_key': 44, 'subscription_start': 2015...
258	{'account_key': 258, 'subscription_start': 201...	{'account_key': 258, 'subscription_start': 201...
366	{'account_key': 366, 'subscription_start': 201...	{'account_key': 366, 'subscription_start': 201...
587	{'account_key': 587, 'subscription_start': 201...	{'account_key': 587, 'subscription_start': 201...
...	...	...
990	{'account_key': 990, 'subscription_start': 201...	{'account_key': 990, 'subscription_start': 201...
686	{'account_key': 686, 'subscription_start': 201...	{'account_key': 686, 'subscription_start': 201...
1067	{'account_key': 1067, 'subscription_start': 20...	{'account_key': 1067, 'subscription_start': 20...
754	{'account_key': 754, 'subscription_start': 201...	{'account_key': 754, 'subscription_start': 201...
854	{'account_key': 854, 'subscription_start': 201...	{'account_key': 854, 'subscription_start': 201...

1164 rows × 2 columns

Then we pass the created dataframe “users_study_df” to the following function to clean this messy form of data and returns the important user information for us

Regex for cleaning data

```
def regex_df(df, new_cols):
    lst_of_dfs = []

    for c in df.columns:
        df_lst = []
        #df_new = pd.DataFrame(columns=new_cols)

        for row in df[c]:
            spare = []

            for col in new_cols:
                to_extract = re.compile(f"'{col}': [0-9]+((.[0-9])?)*")   # (?s:.*?)   # .[0-9]?
                #to_extract = re.match(col, str(row))
                compared = to_extract.search(str(row))
                if compared!=None:
                    spare.append(str(compared.group()[len(col)+4:]))
                else:
                    spare.append(None)

            df_lst.append(spare)

        df_new = pd.DataFrame(df_lst, columns=new_cols)
        df_new.to_csv(f'daily_engagement_{c}.csv')
        print(f"daily_engagement_{c} has been saved....")
        lst_of_dfs.append(df_new)
    return lst_of_dfs
```

We now saved our cleaned data for each year which describes all important information about each user.

Actually there's a little problem, which is the csv files that are created for students contains all of them for both years, and that's not true, it's not a real problem as the data filled for those students that didn't achieve any progress during the year as zeros for all columns, but if we remove them we'll have a more cleaned data, so it's optional, and I'm going to remove it and save my data once again without any overwriting

Data after cleaning:

```
df_2014.drop(neglect_2014.index, inplace=True)
```

Data for 2014

```
df_2014
```

	Unnamed: 0	account_key	subscription_start	total_study_hrs	lessons_completed	projects_completed	Intro to Data Science_hrs	Intro to HTML and CSS_hrs	Data Analyst Nanodegree_hrs	Data Visualization and D3.js_hrs
0	0	448	2014	7.865358	0.0	0.0	1.630967	0.764402	3.406065	0.152922
1	1	44	2014	46.394157	12.0	0.0	22.170880	0.000000	2.638167	0.000000
2	2	258	2014	99.500500	11.0	1.0	22.683016	0.000000	5.058335	9.445516
3	3	366	2014	102.581688	11.0	0.0	33.869980	0.000000	2.968251	3.994800
4	4	587	2014	17.879348	0.0	0.0	16.004608	0.000000	1.617224	0.000000
...	...	...	...	...	...	...	...	...	...	...
405	405	987	2014	1.060631	0.0	0.0	0.000000	0.000000	0.217330	0.000000
406	406	917	2014	0.612932	0.0	0.0	0.506507	0.000000	0.106425	0.000000
407	407	474	2014	74.349377	11.0	1.0	15.806123	2.393722	3.264829	7.740320
408	408	634	2014	52.854911	11.0	1.0	20.995900	0.000000	1.178572	0.000000
409	409	319	2014	18.552699	1.0	0.0	10.557275	0.000000	3.717940	0.151121

410 rows × 33 columns

```
df_2015.drop(neglect_2015.index, inplace=True)
```

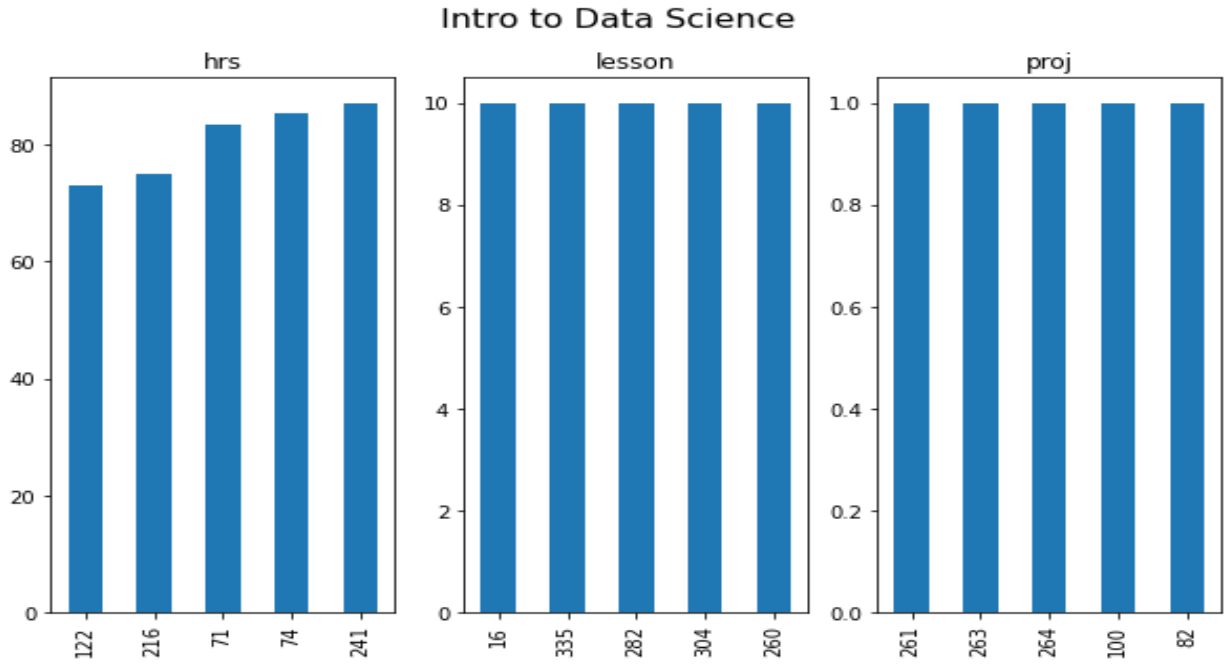
Data for 2015

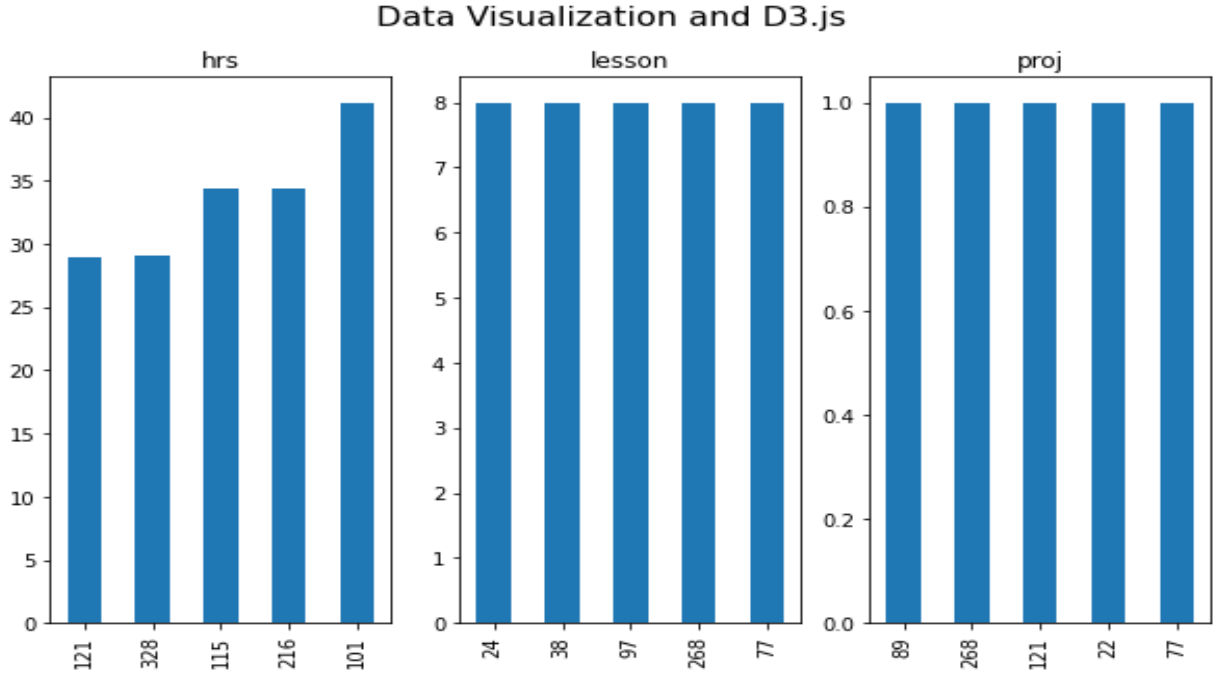
```
df_2015
```

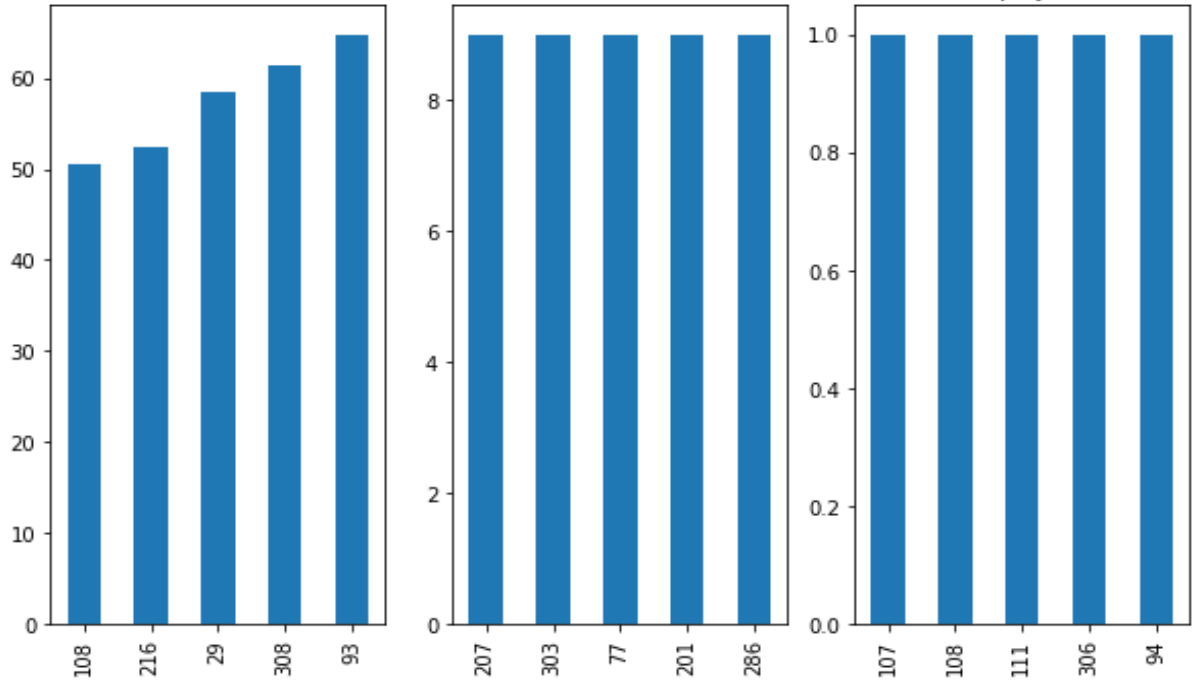
	Unnamed: 0	account_key	subscription_start	total_study_hrs	lessons_completed	projects_completed	Intro to Data Science_hrs	Intro to HTML and CSS_hrs	Data Analyst Nanodegree_hrs	Data Visualization and D3.js_hrs
0	0	448	2015	13.087179	0.0	0.0	0.052843	0.332655	10.152174	1.096989
1	1	44	2015	24.786225	0.0	1.0	0.000000	0.000000	1.376196	0.045869
3	3	366	2015	3.230153	0.0	0.0	1.925213	0.000000	0.305574	0.000000
4	4	587	2015	41.099905	5.0	0.0	28.539121	0.000000	1.250587	0.000000
6	6	412	2015	3.528850	0.0	0.0	0.137447	0.000000	0.407035	0.000000
...	...	...	...	...	...	...	...	...	...	...
1159	1159	990	2015	0.836528	0.0	0.0	0.000000	0.000000	0.836528	0.000000
1160	1160	686	2015	0.160202	0.0	0.0	0.000000	0.000000	0.160202	0.000000
1161	1161	1067	2015	0.495629	0.0	0.0	0.000000	0.000000	0.495629	0.000000
1162	1162	754	2015	6.888770	0.0	0.0	0.000000	0.000000	0.000000	0.000000
1163	1163	854	2015	0.367953	0.0	0.0	0.000000	0.000000	0.367953	0.000000

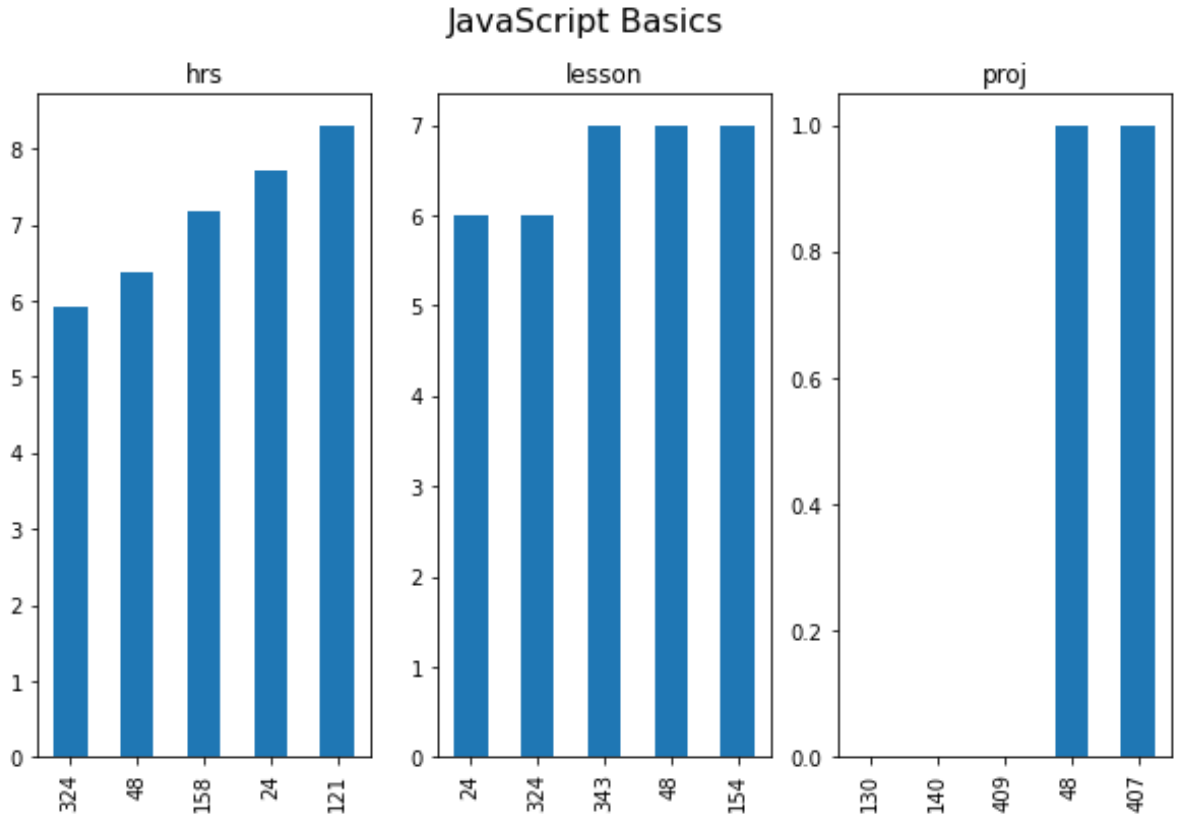
891 rows × 33 columns

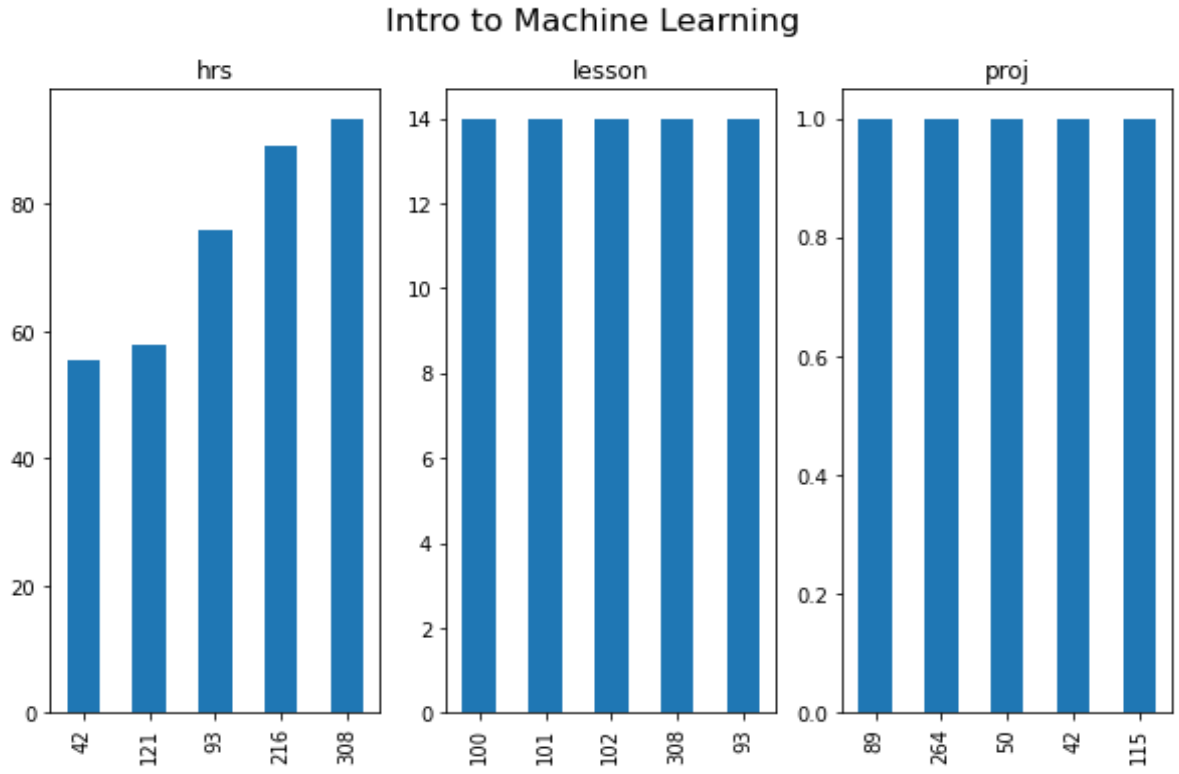
Uni-Variate Analysis of Important relations for year 2014 – Bar Plot

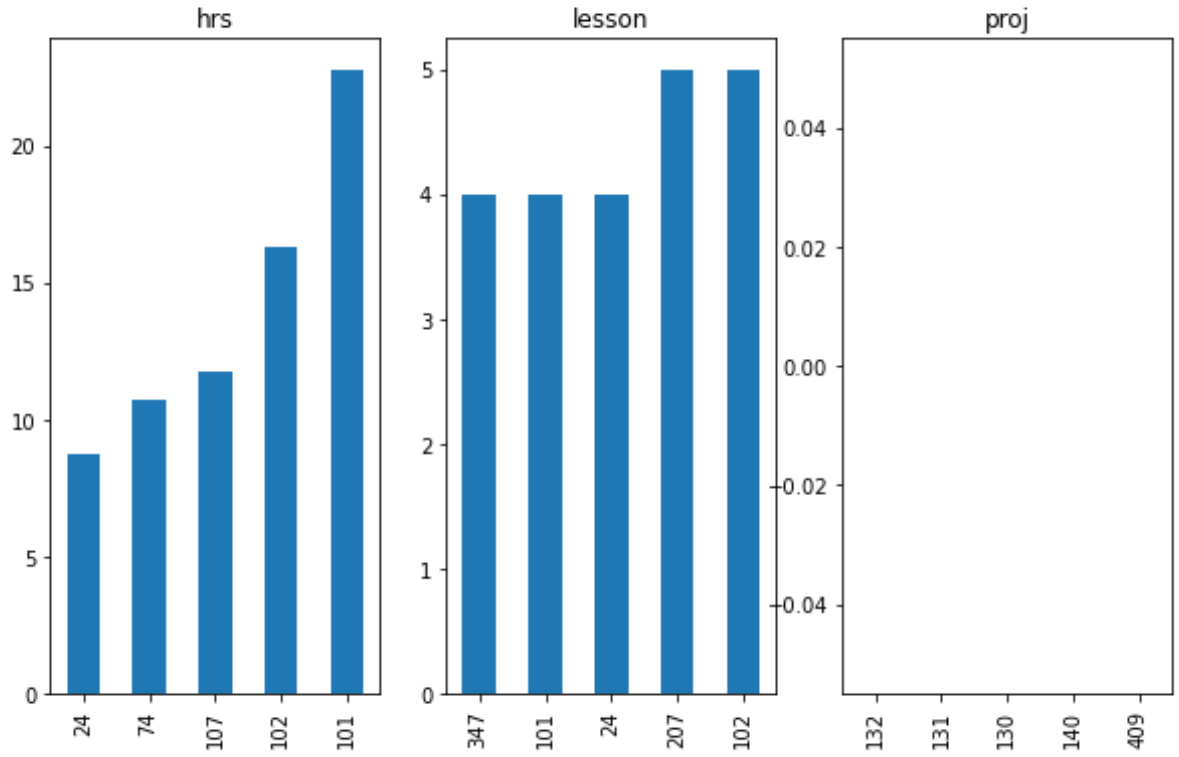


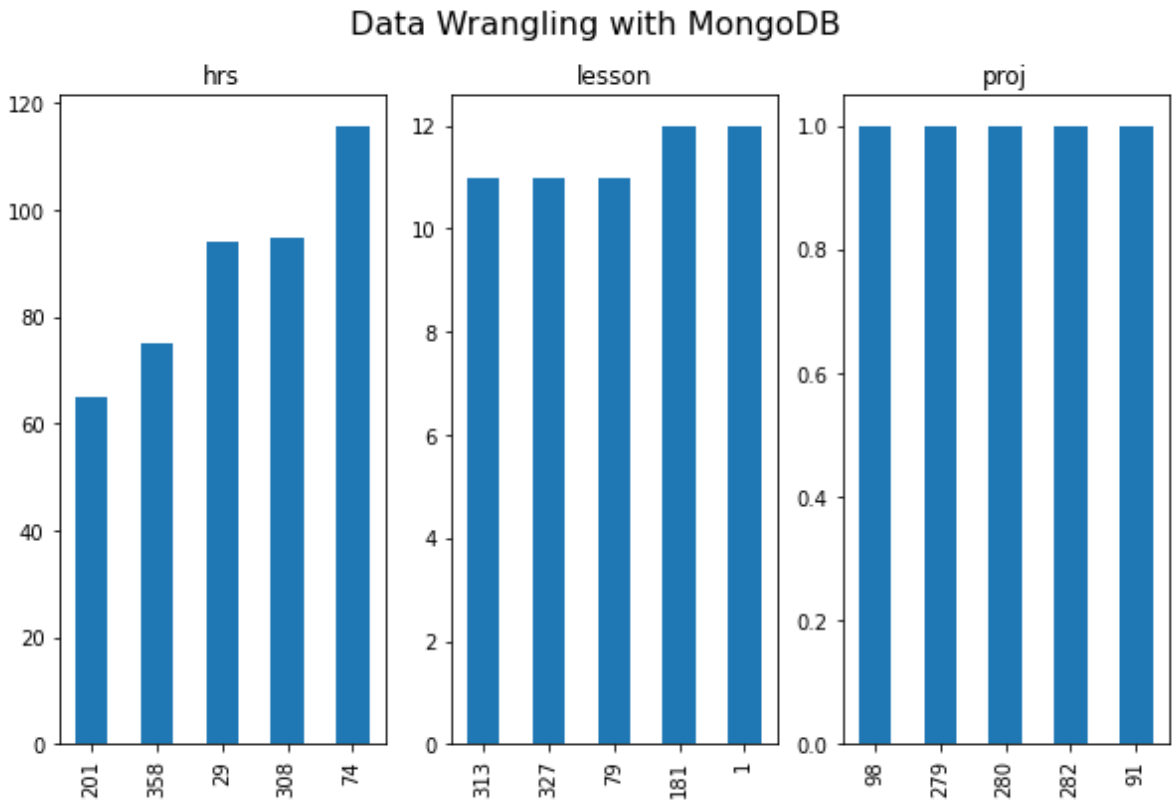


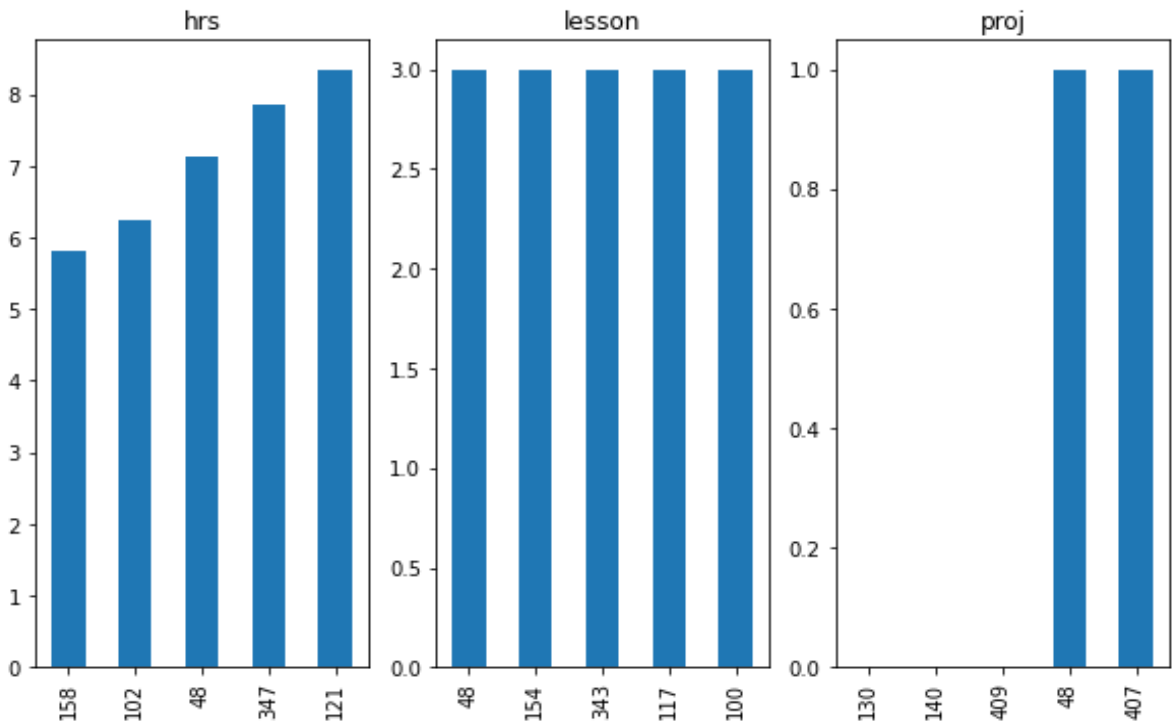

Data Analysis with R
hrs
lesson
proj
0
10
20
30
40
50
60
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0
108
216
29
308
93
207
303
77
201
286
107
108
111
306
94

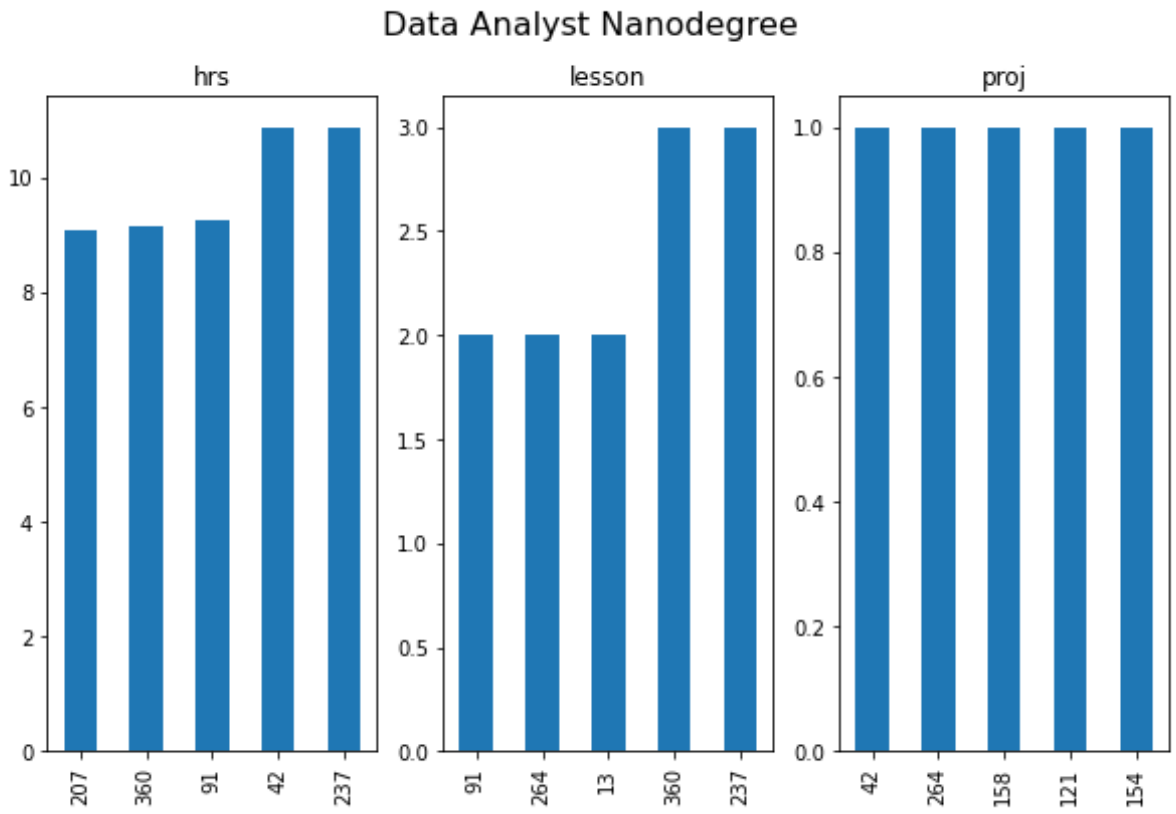

JavaScript Basics
hrs
lesson
proj
0
1
2
3
4
5
6
7
8
0
1
2
3
4
5
6
7
0.0
0.2
0.4
0.6
0.8
1.0
324
48
158
24
121
24
324
343
48
154
130
140
409
48
407


Intro to Machine Learning
hrs
lesson
proj
0
20
40
60
80
0
2
4
6
8
10
12
14
0.0
0.2
0.4
0.6
0.8
1.0
42
121
93
216
308
100
101
102
308
93
89
264
50
42
115


A/B Testing
hrs
lesson
proj
0
5
10
15
20
0
1
2
3
4
5
-0.04
-0.02
0.00
0.02
0.04
24
74
107
102
101
347
101
24
207
102
132
131
130
140
409


Data Wrangling with MongoDB
hrs
lesson
proj
120
100
80
60
40
20
0
12
10
8
6
4
2
0
1.0
0.8
0.6
0.4
0.2
0.0
201
358
29
308
74
313
327
79
181
1
98
279
280
282
91


Intro to HTML and CSS
hrs
lesson
proj
8
7
6
5
4
3
2
1
0
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.0
0.8
0.6
0.4
0.2
0.0
158
102
48
347
121
48
154
343
117
100
130
140
409
48
407



Final Words:

We now contains a cleaned dataset for the user progress per year which can help us for clustering our users into categories if we applied a ML model for that.