Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

WEB-АНАЛІТИКА

Лабораторна робота

Робота №2

Виконав студент гр. ФБ-31мп:

Шевченко С. Ш.

Київ – 2024

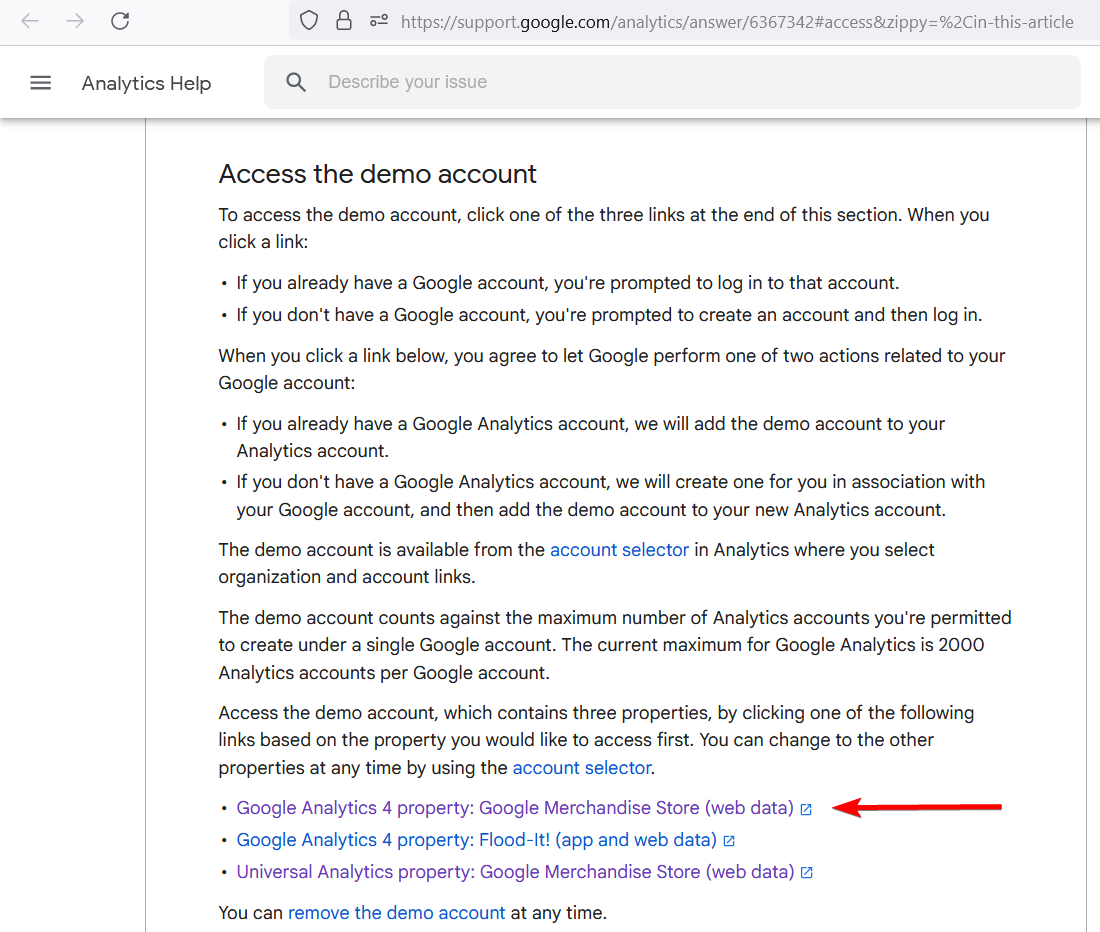
[Репозиторій](https://github.com/dedkuzmich/web-analytics-2024) Github з кодом.

# Завдання:

На основі даних з Google Analytics Demo account вибрати ТРИ різні часові ряди і на їх основі шляхом застосування методів визначення аномалій визначити аномалії в поведінці системи.

# Виконання завдань:

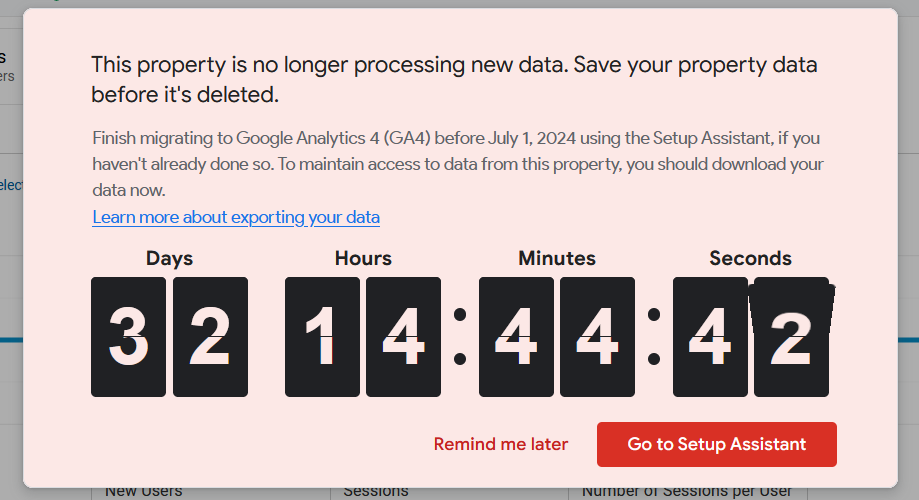
Зареєструємося в [Google Analytics](https://analytics.google.com/analytics/web/provision/?pli=1#/provision). Перейдемо до [селектора](https://support.google.com/analytics/answer/6367342#access&zippy=%2Cin-this-article) демо акаунтів.



Серед 3 демо акаунтів оберемо:

Universal Analytics property: Google Merchandise Store (web data)

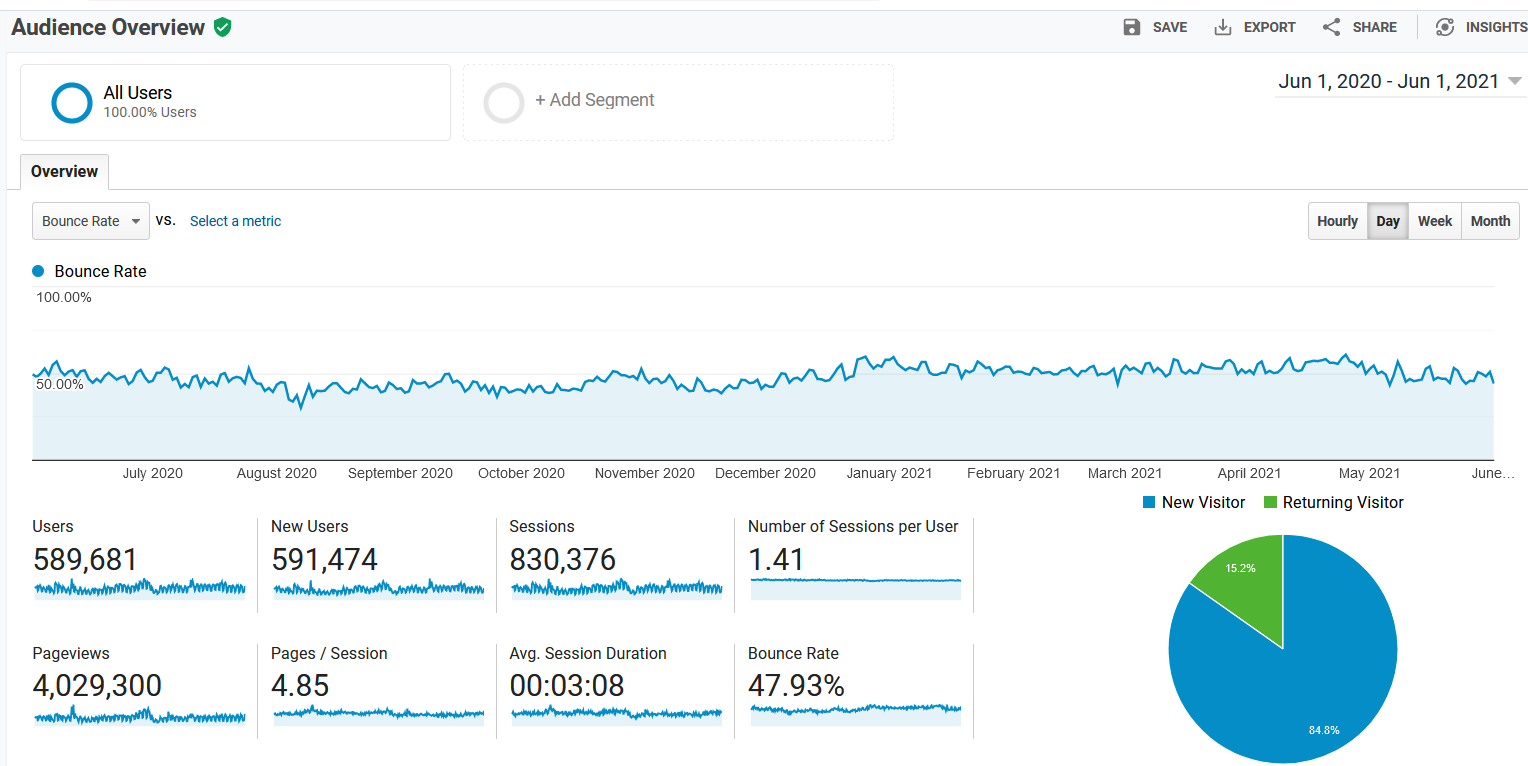
Проблема більш сучасних GA4 в тому, що з них не можна експортувати дані. Приблизно через місяць "legacy" демо акаунт буде видалено :(



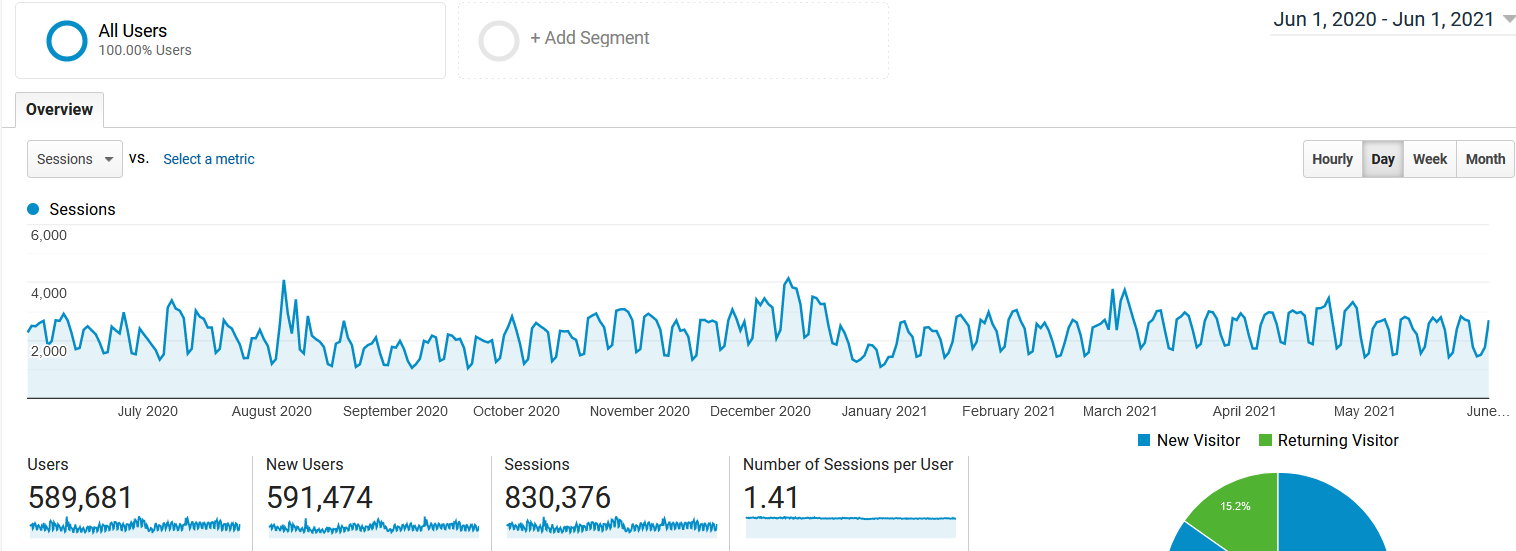
Тепер оберемо 3 часові ряди в часовому проміжку з 1 червня 2020 до 1 червня 2021.

Інтервал = 1 день, тобто один день = одна точка.

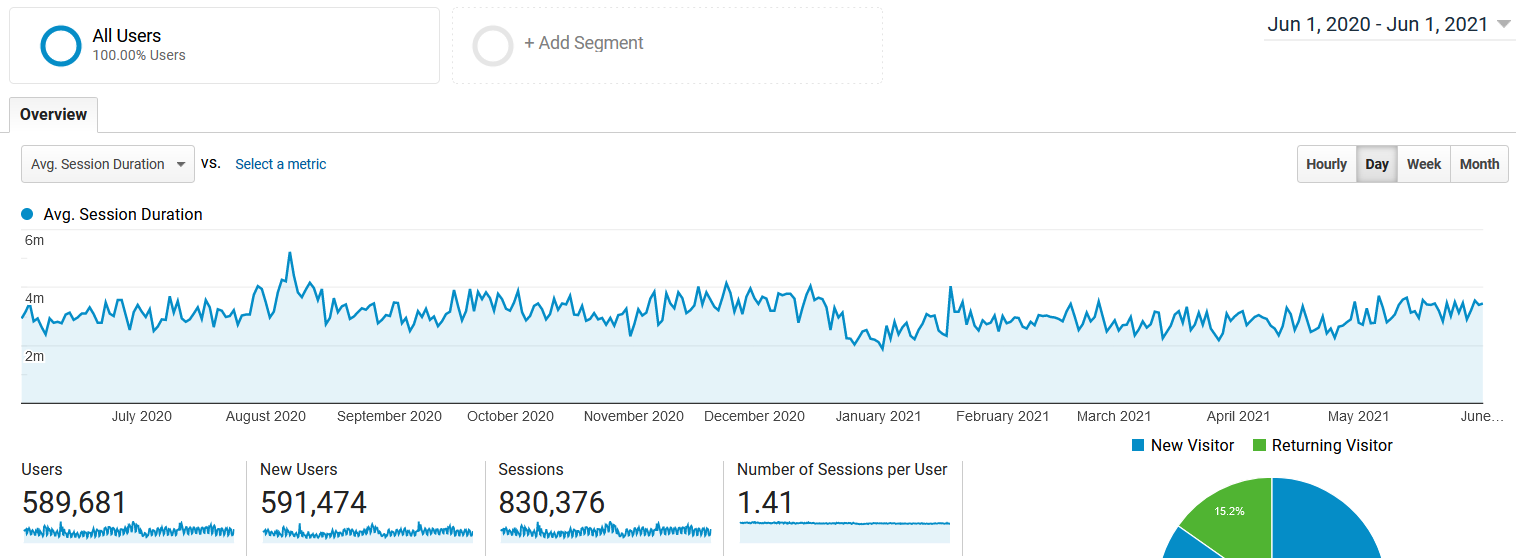
1) Bounce Rate (Показник відмов):



2) Sessions (Сеанси):



3) Avg. Session Duration (Середня тривалість сеансу):



## Створення датасетів з часових рядів

Кожен часовий ряд містить фічу "Day Index", яку можна звести до типу datetime64.

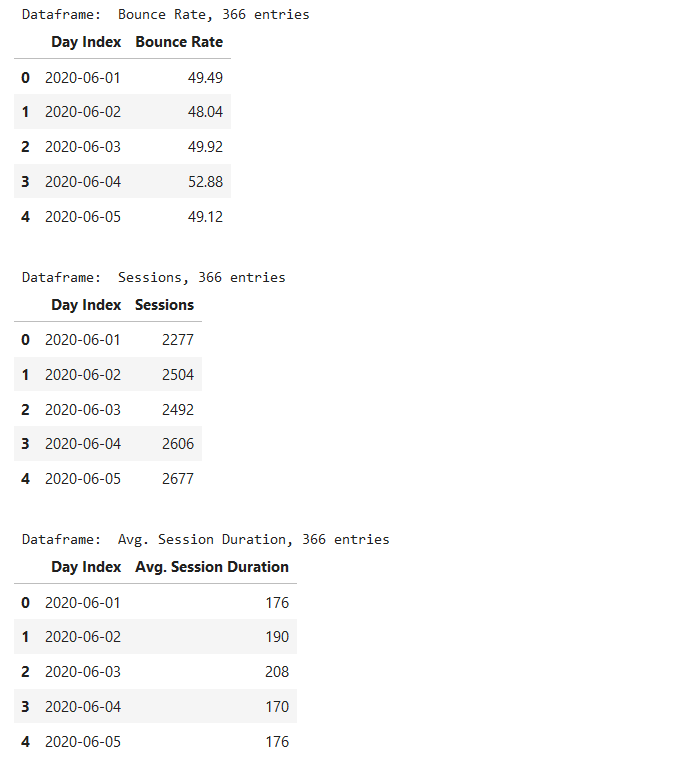
Часовий ряд Bounce Rate містить однойменну фічу, яку можна звести до типу float, прибравши "%" в кінці.

Часовий ряд Sessions містить однойменну фічу, яку можна звести до типу int, прибравши "," між тисячами і сотнями.

Часовий ряд Avg. Session Duration містить однойменну фічу, яку можна звести до типу int, просто перевівши тривалість (год:хв:сек) в секунди.

# Parse .csv input files  
dfs = []  
*for* df\_name *in* ["Bounce Rate", "Sessions", "Avg. Session Duration"]:  
 df = pd.read\_csv(f"input/{df\_name}.csv")  
 df.name = df\_name  
 df["Day Index"] = pd.to\_datetime(df["Day Index"], format = "%m/%d/%y")  
  
 *if* df.name == "Bounce Rate":  
 # 13.37% => 13.37  
 df[df.name] = df[df.name].str.replace("%", "").astype(float)  
 *elif* df.name == "Sessions":  
 # "1,337" => 1337  
 df[df.name] = df[df.name].str.replace(",", "").astype(int)  
 *elif* df.name == "Avg. Session Duration":  
 # 00:02:17 => 137 (seconds)  
 df[df.name] = pd.to\_timedelta(df[df.name]).dt.total\_seconds().astype(int)  
  
 dfs.append(df)  
 df.to\_csv(f"parsed/{df.name}.csv", index = *False*)  
 print(f"\nDataframe: {df.name}, {len(df)} entries")  
 display(df.head())

Отримаємо такі датасети:



## Детектування аномалій

Функція, яка відображає аномалії на площину разом із часовим рядом:

# Function that will show anomalies  
*def* save\_and\_show(*method*, *df*, *df\_anomalies*, *df\_average* = *None*):  
 print(f"\n{len(*df\_anomalies*)} anomalies in {*df*.name}")  
 *df\_anomalies*.to\_csv(f"anomalies/{*method*}/{*df*.name}.csv", index = *False*)  
  
 # Plot df, anomalies and average if provided  
 plt.figure(figsize = (10, 5))  
 plt.plot(*df*["Day Index"], *df*[*df*.name], label = *df*.name, color = "blue", alpha = 0.5)  
 plt.scatter(*df\_anomalies*["Day Index"], *df\_anomalies*[*df*.name], color = "green", label = "Anomalies", marker = "\*")  
 *if df\_average is not None*:  
 plt.plot(*df\_average*["Day Index"], *df\_average*["Moving Average"], label = "Moving Average", color = "red")  
 plt.title(f"{*method*}: {*df*.name}")  
 plt.xlabel("Day Index")  
 plt.ylabel(*df*.name)  
 plt.grid(*True*)  
 plt.tight\_layout()  
 plt.legend()  
 plt.show()

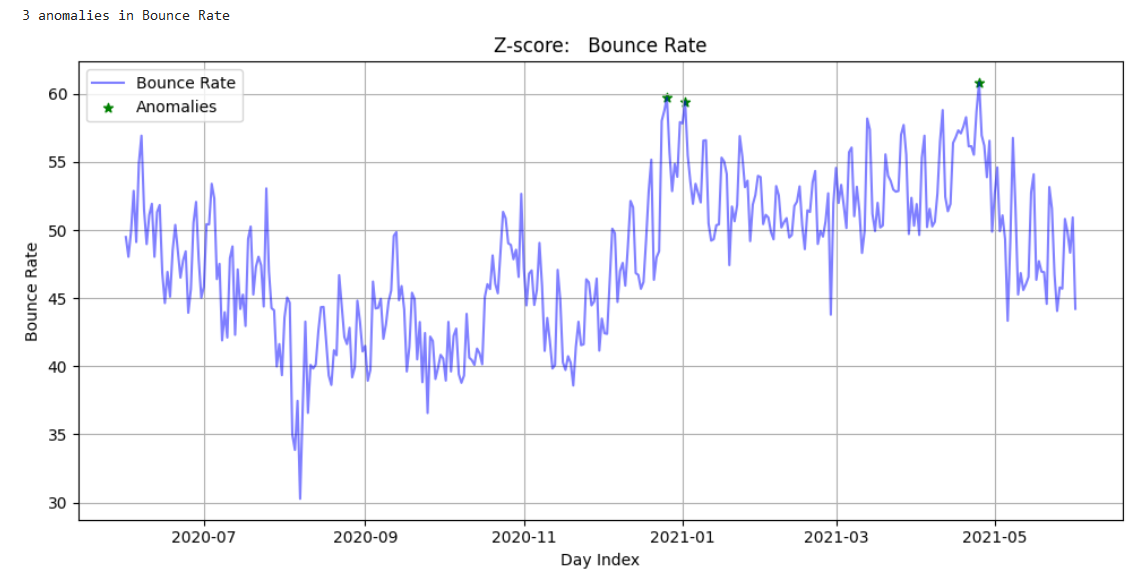
Аномалії детектувалися такими методами:

* Z-score
* Local Outlier Factor (LOF)
* Moving average
* Interquartile Range (IQR)

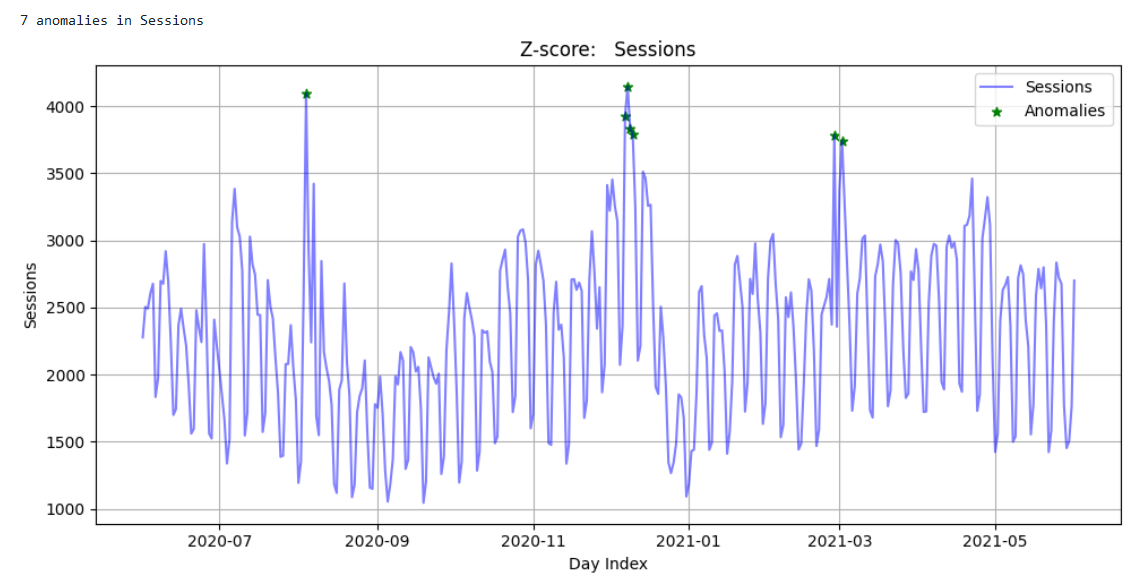
**Z-score** використовує середнє значення та стандартне відхилення набору даних для виявлення аномалій. Значення, які відхиляються від середнього більше, ніж на задану кількість стандартних відхилень, вважаються аномаліями. Цей метод підходить для часових рядів, які приблизно відповідають нормальному розподілу.

# Z-score  
*for* df *in* dfs:  
 df\_zscore = df[[df.name]].apply(zscore)  
 threshold = 2  
 df\_anomalies = df[(df\_zscore > threshold).any(axis = 1)]  
 save\_and\_show("Z-score", df, df\_anomalies)

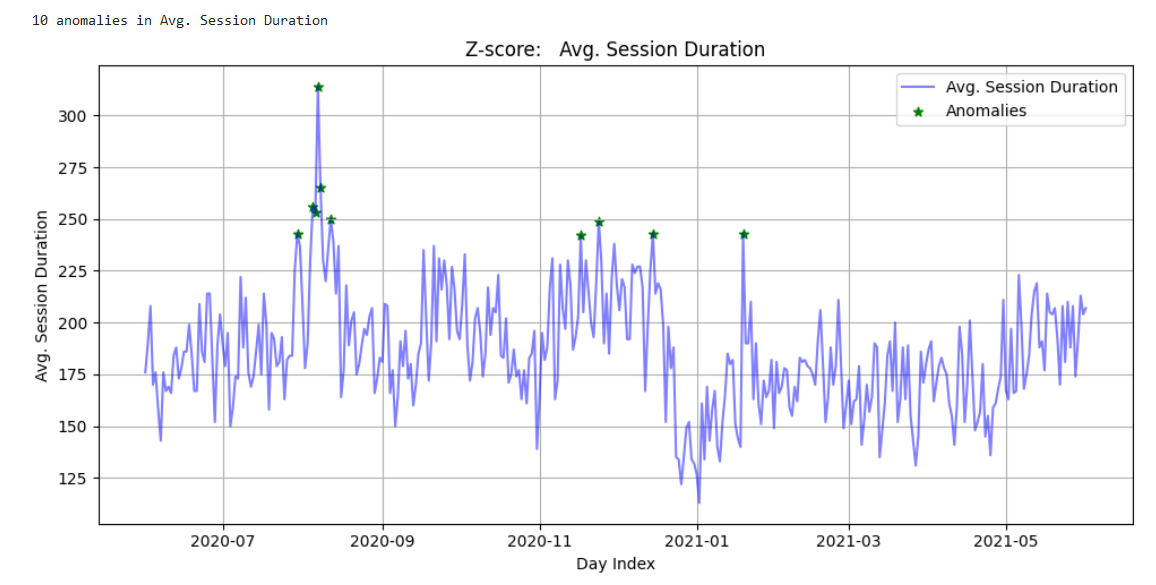
1) Bounce Rate: 3 аномалії



2) Sessions: 7 аномалій



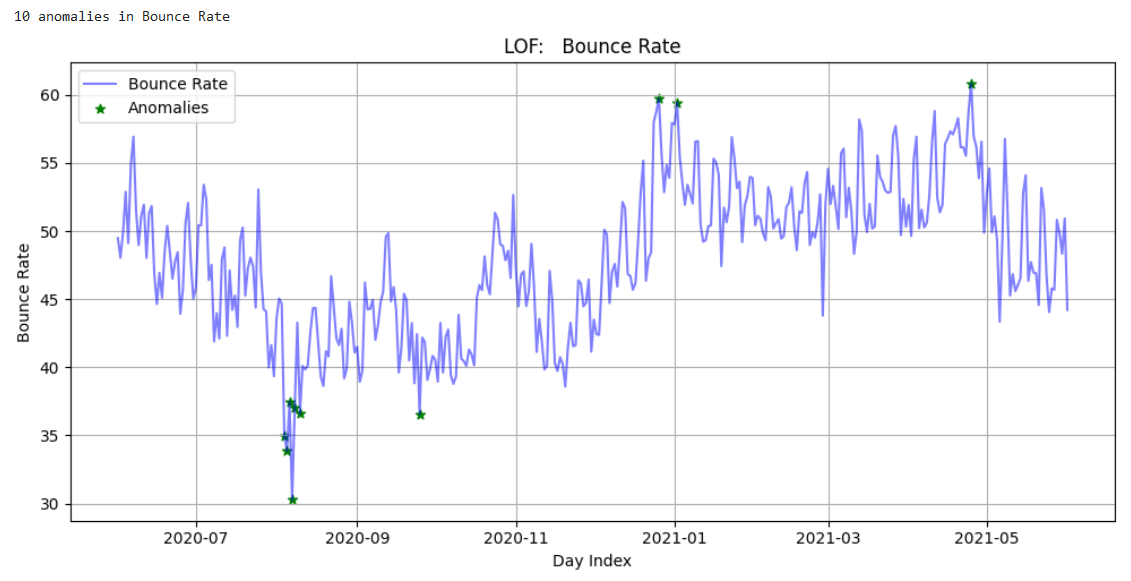
3) Avg. Session Duration: 10 аномалій



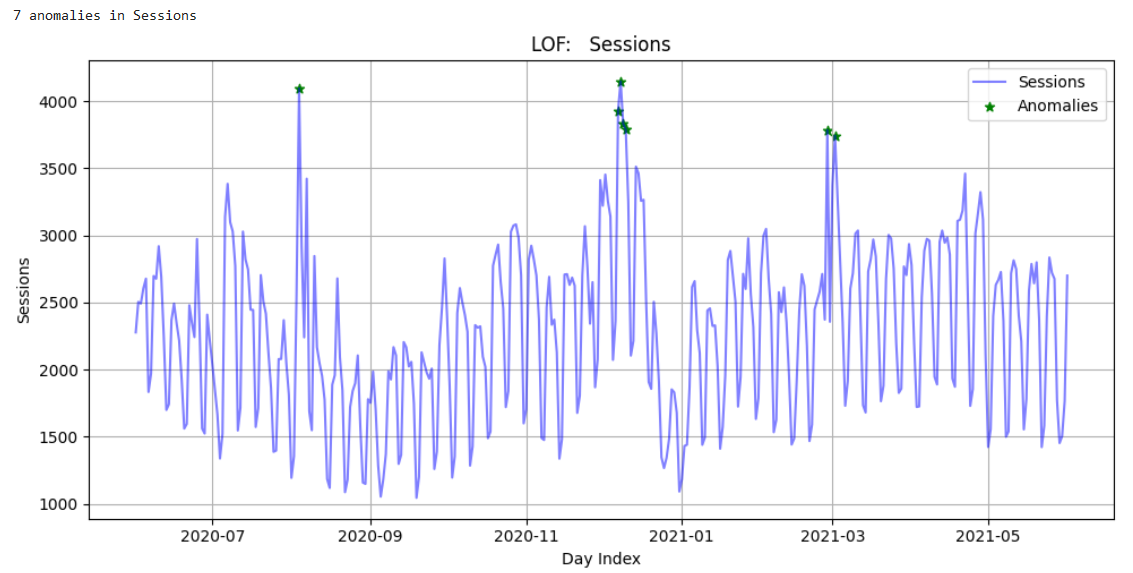
**Local Outlier Factor** обчислює фактор локальної аномалії для кожної точки даних шляхом порівняння локальної густини цієї точки з густинами її сусідів. Точка даних з низькою густиною відносно сусідів вважається аномалією. LOF добре працює з локальними аномаліями та не залежить від припущень про розподіл даних.

# Local Outlier Factor (LOF)  
*for* df *in* dfs:  
 X = df[[df.name]].values.reshape(-1, 1)  
 lof = LocalOutlierFactor(n\_neighbors = 19)  
 anomaly\_scores = lof.fit\_predict(X)  
 df\_anomalies = df[anomaly\_scores == -1]  
 save\_and\_show("LOF", df, df\_anomalies)

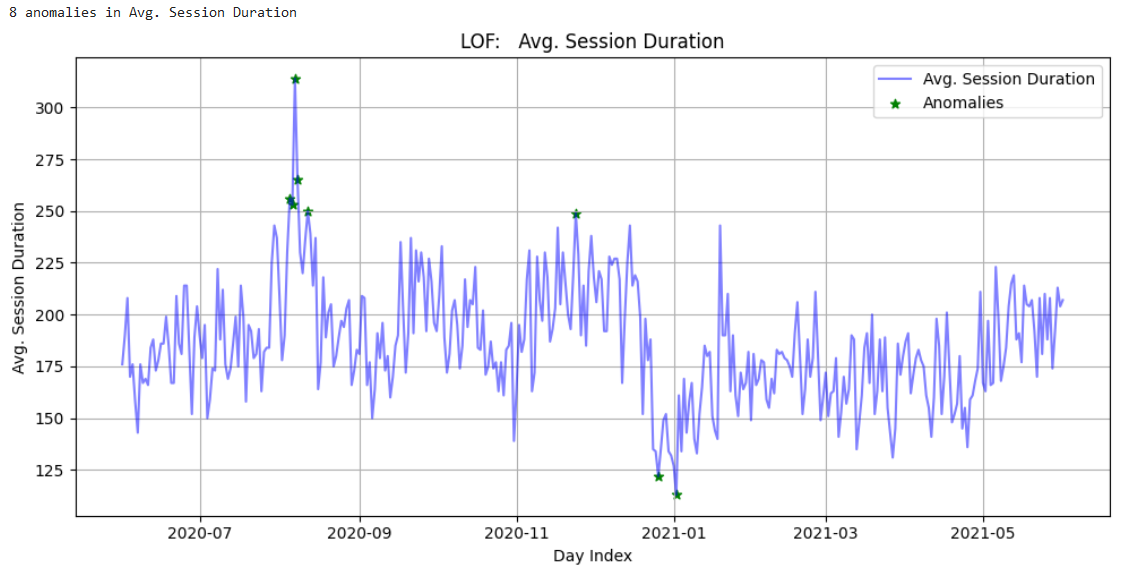
1) Bounce Rate: 10 аномалій



2) Sessions: 7 аномалій



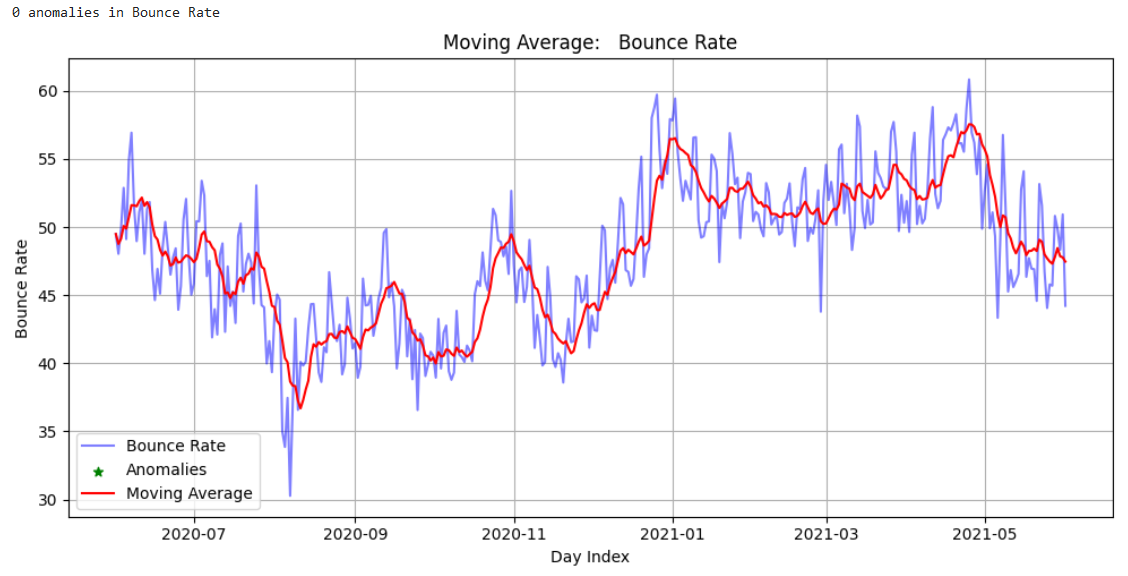
3) Avg. Session Duration: 8 аномалій



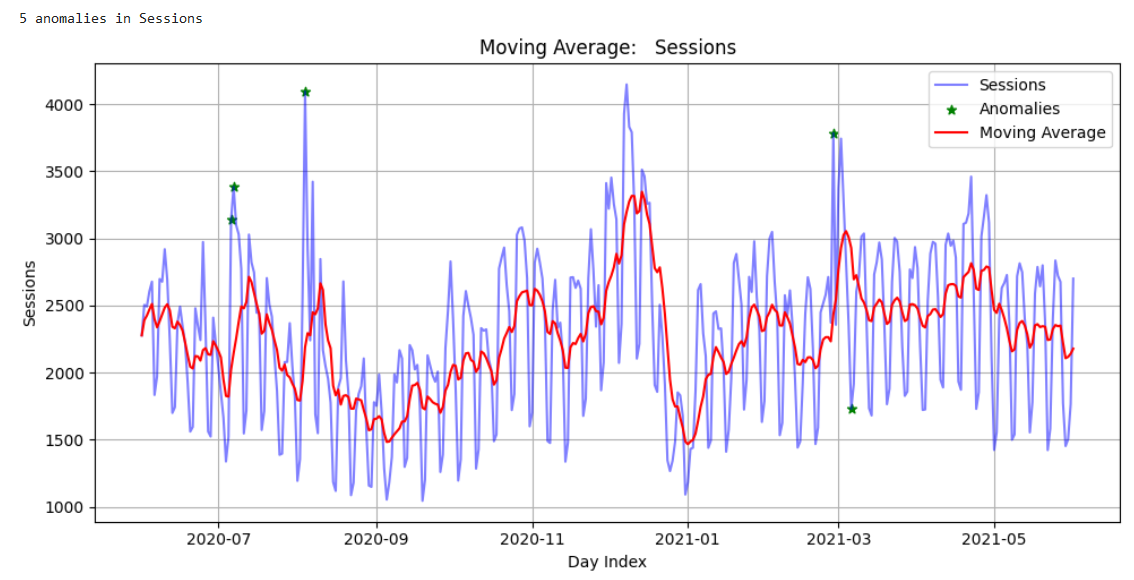
**Moving average** вважає аномаліями значення, які значно відхиляються від середнього арифметичного попередніх значень у часовому ряді.

# Moving Average  
*for* df *in* dfs:  
 df\_average = df.copy()  
  
 # Calculate moving average  
 window\_size = 8  
 df\_average["Moving Average"] = df\_average[df.name].rolling(window = window\_size, min\_periods = 1).mean()  
  
 # Find anomalies based on moving average  
 threshold = 3  
 deviation = abs(df\_average[df.name] - df\_average["Moving Average"])  
 std\_threshold = threshold \* df\_average["Moving Average"].std()  
 df\_anomalies = df\_average[deviation > std\_threshold]  
 save\_and\_show("Moving Average", df, df\_anomalies, df\_average)

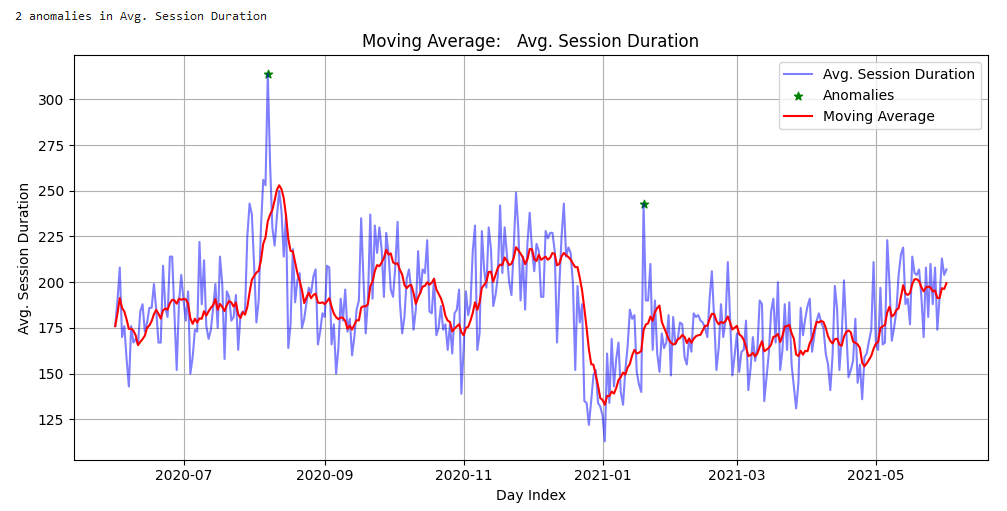
1) Bounce Rate: 0 аномалій



2) Sessions: 5 аномалій



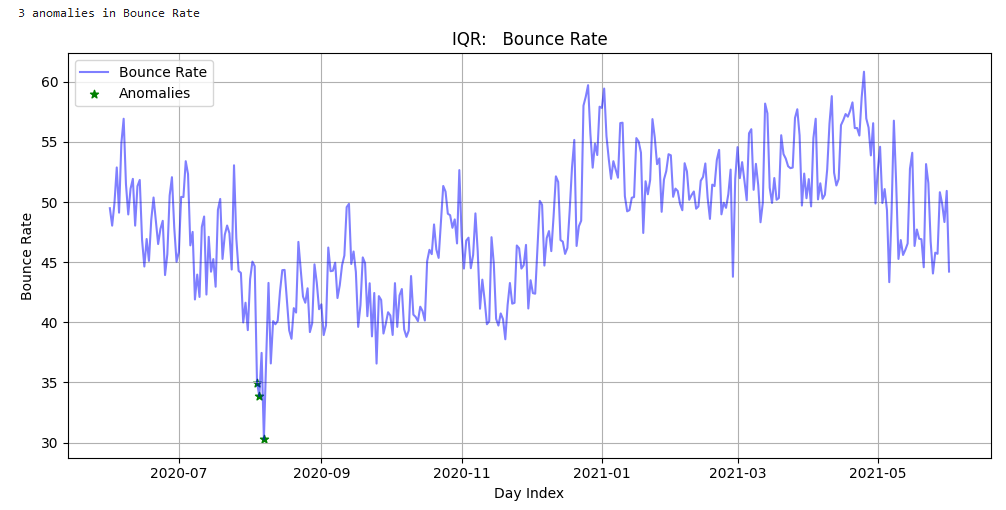
3) Avg. Session Duration: 2 аномалії



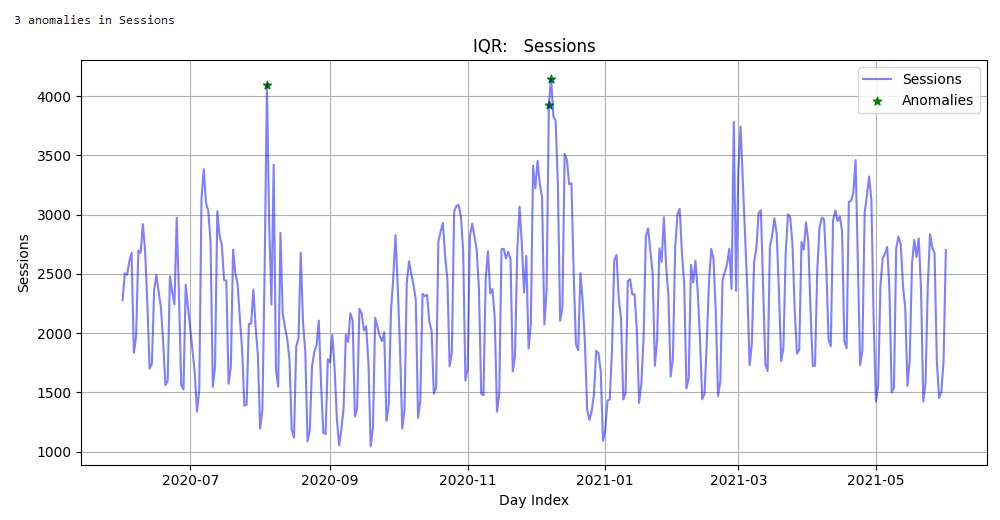
**Interquartile Range** базується на квартилях розподілу даних. Аномаліями вважаються значення, які виходять за межі діапазону, визначеного нижнім та верхнім квартилями +\- деякий множник міжквартильного діапазону (альфа). Цей метод є стійким до викидів та не вимагає припущень про розподіл даних, що робить його актуальним для асиметричних розподілів.

# Interquartile Range (IQR)  
*for* df *in* dfs:  
 # Get 1st and 3rd quartiles of the data  
 q1 = df[df.name].quantile(0.25)  
 q3 = df[df.name].quantile(0.75)  
 iqr = q3 - q1  
  
 # Define the upper and lower bounds  
 alpha = 1.2  
 lower\_bound = q1 - (alpha \* iqr)  
 upper\_bound = q3 + (alpha \* iqr)  
 df\_anomalies = df[(df[df.name] < lower\_bound) | (df[df.name] > upper\_bound)]  
 save\_and\_show("IQR", df, df\_anomalies)

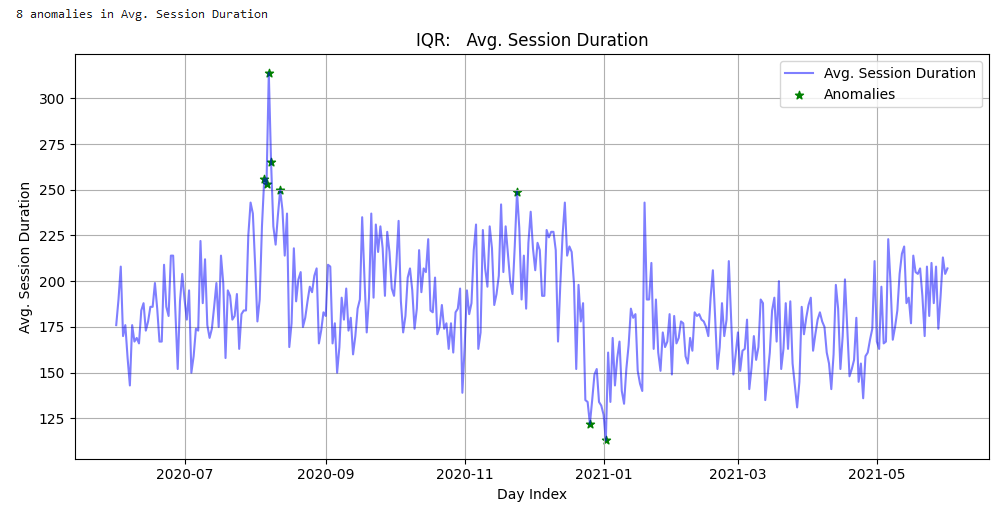
1) Bounce Rate: 3 аномалії



2) Sessions: 3 аномалії



3) Avg. Session Duration: 8 аномалій



Можна побачити, що методи Z-score та LOF виявили майже однакову к-сть аномалій.

На мою думку, найбільш точний метод - це IQR.

Всі аномалії збережено у файли .csv у папках відповідно до методу детектування.