**סעיף ב:**  
**3:**  
**כדי לדעת את הממוצעים השעתיים עבור נתונים שמגיעים אליי ב-stream, אשנה את הקוד שכתבתי כך ש:**  
  
**במקום לעבור על כל הקובץ שקיבלתי ולאחסן את כל הנתונים (אחרי סינון הנתונים השגויים) ב-DataFrame על ידי ספריית pandas, אקבל נתון-נתון בזמן אמת ועבור כל נתון כבר אבצע את הפעולות הדרושות.**

**אכין מראש אוסף מסוג מילון (Dictionary) שבו יתאחסנו הנתונים שמגיעים מה-stream, כך שה-key יהיה התאריך והשעה (מעוגל לשעה הקרובה), וה-value יהיה מילון נוסף שיכיל את הסכום הנוכחי של הערכים ואת כמות האיברים בשעה זו.**

**עבור כל שורה שמגיעה מה-stream, קודם אבדוק שהיא תקינה:**

* **האם ה-timestamp תקין (בפורמט DD/MM/YYYY HH:MM).**
* **האם ה-value הוא מספר.**
* **האם ה-timestamp כבר קיים (כדי למנוע כפילויות).**  
  **אם השורה תקינה, אעגל את השעה שלה לשעה הקרובה (למשל, 12:30 יהפוך ל-12:00), ואז אכניס אותה לתא המתאים במילון:**
* **אם השעה כבר קיימת במילון, אעדכן את הסכום (אוסיף את הערך החדש) ואת כמות האיברים (אוסיף 1).**
* **אם השעה לא קיימת, אצור מפתח חדש עם הסכום והכמות ההתחלתיים.**  
    
  **בסופו של דבר, אקבל מילון המכיל עבור כל תאריך ושעה את הממוצע שלו, אותו אוכל להדפיס או לשמור לקובץ.**

**:4**

**יעילות באחסון וקריאה של קבצי Parquet לעומת CSV:**

בעת קריאה מקובץ CSV, הנתונים נקראים שורה אחר שורה. כלומר, גם אם יש עמודות שאינן רלוונטיות לעיבוד, המערכת עדיין קוראת את כל העמודות בכל שורה על מנת לגשת לנתון מסוים.  
לעומת זאת, קבצי Parquet שומרים את הנתונים בצורה עמודתית, כך שכל עמודה נשמרת בנפרד. לכן, ניתן לגשת רק לעמודות הדרושות מבלי לעבור על כלל הנתונים בכל שורה, מה שמייעל את תהליך הקריאה והעיבוד – במיוחד בקבצים גדולים.