תשובות חלק א'

חלק א' שאלה 3: STREAMING

נהניתי מאוד מהשאלה המעניינת.

כשמגיע אליי ערך חדש, הדבר הראשון שאני עושה זה **בדיקת תקינות** —אם הערך לא תקין (למשל חסר, שלילי או מחוץ לטווח הגיוני), אני פשוט **זורקת אותו ולא ממשיכה איתו בכלל**.

אם הערך תקין, אני בודקת האם ה־ timestampשלו כבר קיים אצלי.
אם זו הפעם הראשונה שהוא מופיע, אני מוסיפה אותו למילון של כל הנתונים (dictAll) ושמה לו flag = True.
בנוסף, אני מוציאה מה־ timestampאת **השעה העגולה** (למשל 08:00:00) ומשתמשת בה כמפתח במילון אחר (calculateAverage) שבו אני שומרת לכל שעה את **הסכום והכמות של הערכים התקפים**.
אם זו פעם ראשונה לשעה הזו, אני פותחת לה מקום במילון.
אחר כך אני מעדכנת את הסכום והכמות של אותה שעה.

אם ה timestampכבר הופיע קודם : אני בודקת את ה־ flagשלו:
אם הוא היה ,Trueזה אומר שזו **כפילות** : :ולכן אני מורידה את הערך הקודם מהממוצע של אותה שעה (כלומר מחסרת אותו מהסכום ומהכמות), ומעדכנת את ה־ flagל־.False
אם הוא כבר היה ,Falseכלומר הופיע יותר מפעמיים : אני פשוט **מתעלמת ממנו לגמרי**.

את **חישוב הממוצע השעתי** אני לא שומרת במפורש, אלא עושה רק כשצריך, דרך פונקציה פשוטה שמחלקת את הסכום במספר הערכים התקפים.

בגישה הזו אני מצליחה לשמור על מערכת **יעילה, מדויקת ונקייה מכפילויות** ,שמתאימה לעבודה עם נתונים שזורמים בזמן אמת, בלי לבזבז זיכרון מיותר.

הרחבה: מה קורה כשיש עבודה במקביל

אם המערכת שלי עובדת בצורה מקבילית – כלומר, יש כמה תהליכים שפועלים בו-זמנית ומקבלים נתונים במקביל – חשוב מאוד לדאוג שכולם **ידברו אחד עם השני כמו שצריך**.

בלי סנכרון, יכולות לקרות התנגשות בין תהליכים. למשל: שני תהליכים מנסים באותו רגע לעדכן את אותו ,timestamp וכל אחד חושב שהוא היחיד — וזה עלול להוביל לבלגן בנתונים ולחישובים שגויים.

כדי למנוע את זה, אני דואגת להגדיר חלקים מסוימים בקוד כ"אזורים מוגנים" — כאלה שרק תהליך אחד יכול להיכנס אליהם בכל רגע.
זה מבטיח שכל שינוי שאני עושה במילונים שלי (כמו dictAllאו calculateAverage) מתבצע בצורה בטוחה, בלי שתהליך אחר יתנגש איתו באותו זמן.

המטרה היא לשמור על **דיוק ועקביות** גם כשהמערכת מקבלת הרבה נתונים במקביל, ואפילו כשיש עומסים.
הגישה הזו מאפשרת להפעיל את המערכת בצורה חזקה, גמישה ומוכנה להתרחב — בלי לפגוע באמינות של הנתונים או בפשטות של החישוב.

חלק ב' שאלה 3: PARQUET יתרונות

בהתחלה, עבדתי עם קובץ ה־ CSV, ששומר את כל הנתונים בשורות. חילקתי את הקובץ לחלקים קטנים יותר לפי זמן (כמו שעות או ימים), חישבתי ממוצעים שעתיים לכל חלק, וחיברתי את התוצאות לקובץ סופי אחד, בדיוק כמו בדוגמה מהחלק ב.'
מאוחר יותר, כשהתחלתי לעבוד עם קובץ Parquet, ראיתי את ההבדל Parquet. שומר את הנתונים בעמודות במקום בשורות, כך שיכולתי לטעון רק את העמודות הספציפיות שהייתי צריכה) למשל timestamp ו־ value), מבלי לבזבז זיכרון או זמן על העמודות האחרות. זה הפך את כל התהליך להרבה יותר מהיר ויעיל.
בנוסף, בקבצי Parquet יש מידע מובנה על סוגי הנתונים (schema), כך שלא הייתי צריכה לבדוק או להגדיר את סוגי הנתונים בכל פעם – הקובץ כבר יודע ש־ timestamp הוא תאריך ו־ value הוא מספר. זה חסך לי זמן ועזר למנוע טעויות, במיוחד כשעובדים עם כמויות גדולות של נתונים או בעיבוד בזמן אמת.(stream processing)

בקיצור Parquet: קטן יותר, מהיר יותר, נוח יותר לעבודה, ופשוט בחירה טובה יותר כשמדובר בביג דאטה וניתוח מתקדם.