עליך לכתוב מערכת המורכבת משלושה מודולים לפחות.
מודול ראשון שולח הודעה לתור מסוג RABBITMQ המכילה מיקום של קובץ, סוג הקובץ (CSV או JSON), ושם טבלה אליה יש לטעון את הקבצים. פורמט הקובץ יהיה JSON תקני או CSV תקני (הקבצים מצורפים במייל).
מודול שני יאזין לתור וברגע קבלת ההודעה יטען את הקובץ אל הטבלה המתאימה בבסיס נתונים מסוג SQLITE (ניתן להניח שבסיס הנתונים נמצא בתיקיית הפרוייקט – אין צורך לשלוח את מיקומו או שמו).
לאחר טעינת הקובץ לטבלה המערכת שולחת הודעה לתור נוסף על סיום טעינה
מודול שלישי יאזין לתור הנוסף ובקבלת ההודעה יציג בגרף שמתעדכן בזמן אמת את הנתונים הבאים :
1. סך כל המכירות לחודש בכל שנה
2. כמות הלקוחות הפעילים באותו חודש (לקוח פעיל בחודש הינו לקוח שביצע רכישה אחת לפחות)

*את קובץ ה JSON יש לפרסר ולטעון כטבלה ולא כ JSON.
* קובצי ה CSV וה JSON מצורפים במייל ומכילים את כל ה DATA הנדרש לתרגיל.
*את קבצי ה CSV וה JSON יש לטעון לאותה הטבלה.

הרכיב שדוגם את בסיס הנתונים ומייצר את הגרף צריך להציג גרף בזמן אמת (כמעט זמן אמת – ניתן לספוג DELAY של עד כשניה)

הנחיות:
1. הרכיב שמייצר את הגרף יכול להשאר בריצה לעד. הרכיב במודול השני יכול להשאר בריצה לעד.
2. לטובת בדיקת המערכת יש לבצע טעינות אל בסיס הנתונים במספר איטרציות (לפי שמות הקבצים המצורפים) ולראות את השינויים בזמן אמת לאחר הטעינה.
3. יש לפתח את הפתרון ב PYTHON3 (ניתן להשתמש בכל ספריה תקנית של PYTHON)
4. הפתרון צריך להתאים לכל מערכת הפעלה (LINUX WINDOWS MAC)
5. בסיס הנתונים יהיה מסוג SQLITE , ניתן להוריד מ https://www.sqlite.org
6. יש להעלות את הפרוייקט אל GITHUB ולשלוח חזרה לינק אל הפרוייקט
7. התקשורת אל האפליקציה תתבצע על ידי RABBITMQ ניתן להתקין מ https://www.rabbitmq.com
8. על הקוד להיות
   - יעיל
   - קריא
   - מסודר
   - מגובה ב TEST'ים
   - עם ארכיטקטורה מתאימה כך שיהיה ניתן להוסיף לו קומפוננטות חדשות בקלות
   - בעל יכולת התאוששות מנפילות

9. מצורף class dbHandler ניתן להשתמש בו ולשנותו במידת הצורך (ייתכן ויש בו באג מובנה)

בהצלחה