

דוח סיכום, ניתוח והמלצות:

ניתוח מעלות וחסרונות של שיטות החישוב

הפרויקט השווה בין שתי גישות ארכיטקטוניות לחישוב נוסחאות דינמיות על מערך של מיליון רשומות. כל שיטה מנצלת יתרונות שונים של פלטפורמת הביצוע:

1. שיטה א': Python - Pandas Vectorization זוהי שיטת המבוססת על ספריות ממוטבות לביצועים.

מעלות (יתרונות)	חסרונות (חולשות)
מהירות חישוב גבוהה: החישוב בפועל של מיליון רשומות הוא מהיר מאוד בזכות NumPy ו- Pandas (וקטורизציה).	צוואר בקבוק I/O : זמן הריצה הכולל נפגע קשות עקב הצורך להעביר מיליון רשומות על גבי הרשת חזרה לשרת ה-SQL.
גמישות בקוד: תומך בקלות רבה יותר בלוגיקה מורכבת ופונקציות מתמטיות לא סטנדרטיות ש-SQL Server אינו תומך בהן.	חוסר יציבות: הביצועים תלויים בעומס הרשת ובביצועי מחשב הלקוח (הזמנים משתנים מריצה לריצה).
קלות פיתוח: קל יחסית לכתוב, לקרוא ולתחזק את לוגיקת הנוסחאות ב-Python.	תקורה: דורש תקורה גבוהה של זיכרון ב-RAM של הלקוח לצורך טעינת ועיבוד מיליון הרשומות.

2. שיטה ב': דינמי SQL - SQL Stored Procedure זוהי שיטת המבוססת על יכולות הליבה של בסיס הנתונים.

מעלות (יתרונות)	חסרונות (חולשות)
מהירות I/O : ביצועים מיטביים וזמני ריצה קצרים (כפי שהוכח בנתונים). מבטל את תעבורת הרשת כליל.	מורכבות בטיפול סינטקטי: קשה מאוד לבצע המרות סינטקטיות מורכבות (כגון $^{\wedge}$ ל- <code>POWER(base, exp)</code>) בתוך ה-SQL הדינמי.
יעילות ארכיטקטונית: מנצל את מנוע הטרנזקציות הממוטב של ה-DB ואת טכנולוגיית ה- Batch Processing של ה-SQL Server.	מורכבות לוגית: קשה יותר לכתוב לוגיקה מורכבת ומתקדמת (כגון לוגיקה עם לולאות או פונקציות מתמטיות לא סטנדרטיות) באמצעות T-SQL לעומת Python.

שימוש במשאבים: מפעיל עומס ישיר על שרת ה-DB (CPU ו-Disk I/O), מה שעלול להשפיע על מערכות אחרות.	אבטחה וניהול: כל הקוד מאוחסן ומנוהל במקום מרכזי אחד בשרת ה-DB, מה שמקל על ניהול סכמות ואבטחת מידע.
--	---

תוצאות ומסקנה סופית

הצגת טבלת ה-log:

Results				
	targil_id	Full_Formula	method	run_time
4	2	$c * 2$	Python_Pandas_Vectorized	22.6846158504486
5	3	$b - a$	SQL_Dynamic_SP	9.421
6	3	$b - a$	Python_Pandas_Vectorized	20.3861541748047
7	4	$d / 4$	SQL_Dynamic_SP	8.465
8	4	$d / 4$	Python_Pandas_Vectorized	19.8974905014038
9	5	$a * 1.5 - d$	SQL_Dynamic_SP	8.148
10	5	$a * 1.5 - d$	Python_Pandas_Vectorized	25.4603409767151
11	6	$(a + b) * 8$	SQL_Dynamic_SP	8.382
12	6	$(a + b) * 8$	Python_Pandas_Vectorized	24.3315691947937
13	7	$\sqrt{c^2 + d^2}$	SQL_Dynamic_SP	8.771
14	7	$\sqrt{c^2 + d^2}$	Python_Pandas_Vectorized	20.7960424423218
15	8	$\log(\text{ABS}(b)) + c$	SQL_Dynamic_SP	8.849
16	8	$\log(\text{ABS}(b)) + c$	Python_Pandas_Vectorized	20.5135653018951
17	9	$\text{abs}(d - b)$	SQL_Dynamic_SP	8.685
18	9	$\text{abs}(d - b)$	Python_Pandas_Vectorized	25.2919239997864
19	10	$a^3 / 100$	SQL_Dynamic_SP	8.855
20	10	$a^3 / 100$	Python_Pandas_Vectorized	51.2735366821289
21	11	IF (a > 5) THEN b * 2 ELSE b / 2	SQL_Dynamic_SP	9.329
22	11	IF (a > 5) THEN b * 2 ELSE b / 2	Python_Pandas_Vectorized	22.3692562580109
23	12	IF (b < 10) THEN a + 1 ELSE d - 1	SQL_Dynamic_SP	8.711
24	12	IF (b < 10) THEN a + 1 ELSE d - 1	Python_Pandas_Vectorized	33.1352670192719
25	13	IF (a == c) THEN 1 ELSE 0	SQL_Dynamic_SP	8.149
26	13	IF (a == c) THEN 1 ELSE 0	Python_Pandas_Vectorized	33.184650182724

ההמלצה הסופית על השיטה הטובה ביותר לחישוב נוסחאות דינמיות היא: שיטת SQL Stored Procedure - דינמי SQL

1. ניתוח נתוני הביצועים המכריעים

הטבלה הבאה מסכמת את הביצועים הממוצעים של שתי השיטות שנבדקו (זמן כולל = חישוב + שמירה של מיליון רשומות):

דוגמה	זמן SQL SP (שניות)	זמן Python (שניות)	יתרון SQL SP (פי)
נוסחה 5 $(a * 1.5 - d)$	8.148	25.460	3.12
נוסחה 7 $((...)\text{sqrt})$	8.771	20.796	2.37
נוסחה 10 $(a^3 / 100)$	8.855	51.273	5.79
נוסחה 13 $\text{IF } (a == c) \text{ THEN}$ (1)	8.149	33.184	4.07
ממוצע כללי	~8.5 שניות	~28.0 שניות	~3.3 פי

2. סיבת ההמלצה

ה-SQL Stored Procedure הוא המומלץ ביותר כיוון שהוא מבטל את צוואר הבקבוק של הרשת (Network I/O).

1. **יעילות I/O:** ביישומים העובדים עם נפח נתונים גדול (1,000,000 רשומות), זמן העברת הנתונים בין הלקוח (Python) לשרת ה-SQL הוא הגורם המעכב ביותר. ה-SQL SP עוקף זאת על ידי ביצוע כל הפעולות (קריאה, חישוב ושמירה) **בתוך שרת הנתונים (Server-Side)**.
2. **מהירות גולמית:** שיטת ה-SQL SP מנצלת את מנגנון ה-Batch Processing וה-I/O הממוטב של ה-SQL Server, ומספקת ביצועים מהירים **פי 3.3 בממוצע** מהקוד ב-Python, למרות שקוד ה-Python כבר ממוטב לחישוב וקטורי.
3. **יציבות:** הביצועים של ה-SQL SP יציבים יותר (זמן קצר יותר) מכיוון שהם אינם מושפעים מתנודות עומס רשת או עומס על מחשב הלקוח.

מסקנה: עבור משימות קריטיות של חישוב דינמי המוני במערכת תשלומים, יש להעדיף פתרונות **SQL Stored Procedure (Server-Side Native)** לטובת מקסום ביצועים ויציבות.

