1. PANDAS

התחלנו בייבוא הספריות הנחוצות (Pandas, sqlalchemy והמרכיבים הרלוונטיים שלהם, כולל create_engine). לאחר מכן, טענו למערכת שלושה קבצים נחוצים באמצעות הפונקציות pd.read_table ו-pd.read_json. בהמשך היינו צריכים ליצור קובץ תקציב משותף (joint_budget) על ידי איחוד שני קבצי JSON, שעשינו באמצעות pd.concat באופן הבא:

joint_budget = pd.concat([budget_1, budget_2], ignore_index=True)

לאחר מכן, כדי להכין קובץ כללי לטעינה ב-SQL, יצרנו קובץ ביניים detailed_budget באמצעות מיזוג בין joint_budget ו-department. הקוד נראה כך:

detailed_budget = pd.merge(joint_budget, department, on='department_id', how='outer')

הקובץ הסופי לטעינה ב-SQL הוכן באמצעות פונקציית groupby של Pandas. חיברנו את התקציב וביצענו קיבוץ לפי שני עמודות ("department_id" ו-"department_name"), בדומה ל-SQL. הקוד היה:

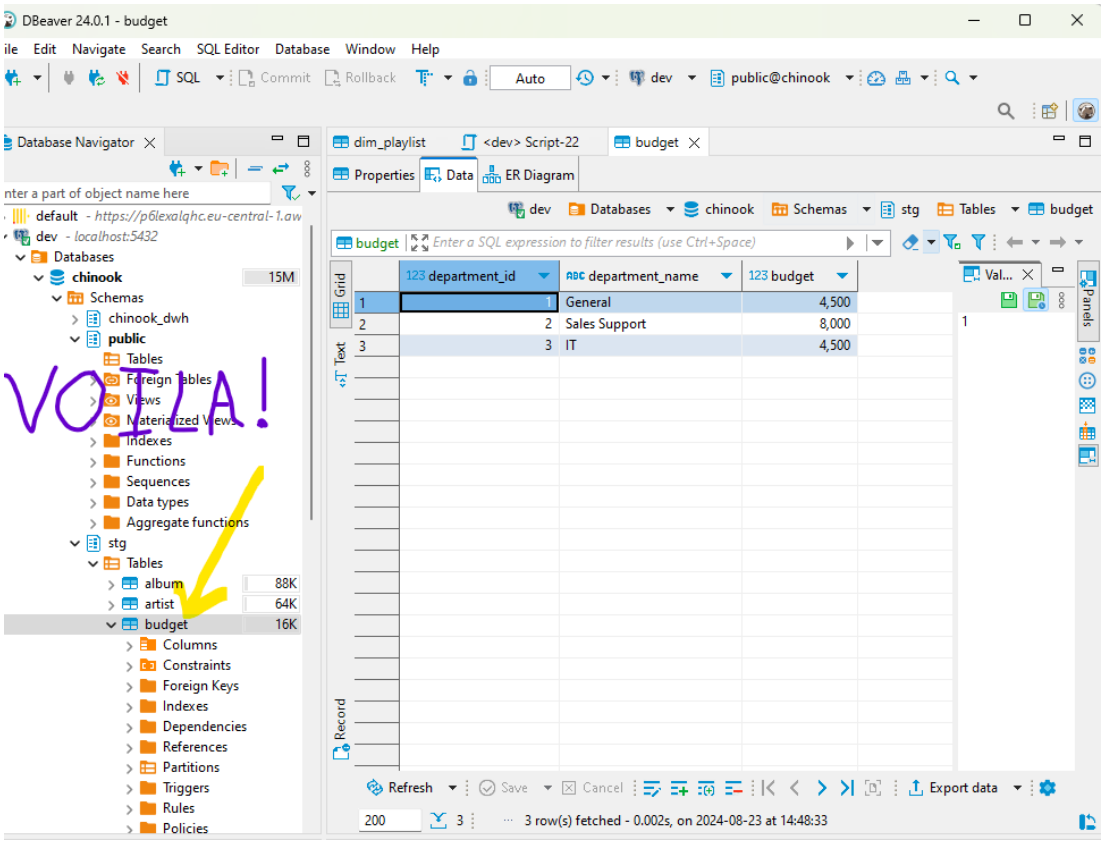
total_budget = detailed_budget.groupby(["department_id", "department_name"])["budget"].sum().reset_index(name='budget')

השלב האחרון היה להכין DataFrame לטעינה למסד הנתונים chinook. ביצענו את זה כך:

engine = create_engine('postgresql://postgres:postgres@localhost/chinook')

total_budget.to_sql(name='budget', con=engine, schema='stg', if_exists='replace', index=False)

כתוצאה מהטעינה המוצלחת, הטבלה budget הופיעה במסד הנתונים chinook, כפי שנראה בצילום המסך למטה.



2. DBT

בשלב ההכנה, התחלנו ביצירת תיקיית chinook_dbt באמצעות הפקודה dbt run. לאחר מכן, ערכנו את קבצי ה-schema הנחוצים בתיקיות facts ו-dimensions. בקובץ ה-schema שבתיקיית dimensions, ציינו את המאפיינים של קבצי fact ואת המקורות המתאימים במסד הנתונים chinook. ביצענו פעולות דומות עבור קבצי dimension בקובץ ה-schema שבתיקיית fact.

לאחר מכן, התחלנו ליצור את קבצי ה-SQL המתאימים. יצרנו את dim_playlist, dim_customer, dim_employee, dim_track, fact_invoice, fact_invoiceline. עם זאת, עדיין לא יצרנו את dim_currency, שאינו קשור ל-API של currency, ונעשה זאת במסגרת המשימה הבאה.

בהתאם לכך, עדיין לא טענו את תיקיית chinook_dbt למסד הנתונים chinook ונחזור לכך רק לאחר השלמת המשימה השלישית.

3. API CURRENCIES

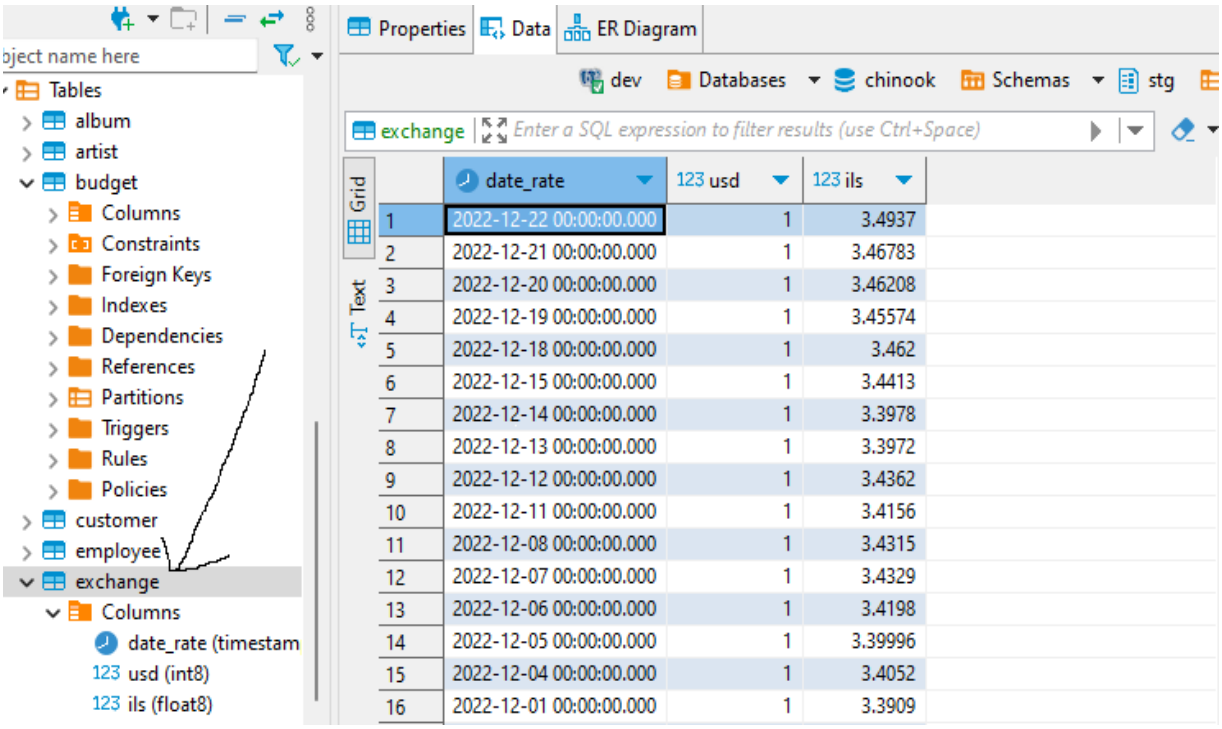
3.1. כדי להתחיל בעבודה, היינו זקוקים לקוד API. נכנסנו לאתר.

וקיבלנו קוד API בחינם. כדי לקבל את המידע הרלוונטי, העתקנו מידע מסדרת Time Series FX (Daily) עבור המטבעות USD ו-ILS. בנוסף, היינו צריכים לקבל נתונים על שערי המטבעות בטווח התאריכים המצוינים בטבלת fact_invoice (מ-2018-01-01 עד 2022-12-22).

לאחר קבלת הנתונים הללו, התחלנו לעבוד ב-Python. היינו צריכים ליצור פונקציה המבוססת על הפרמטרים הבאים: start_date, end_date, base_currency (USD), target_currency (ILS) ו-api_key.

בהתבסס על הנתונים הללו, פיתחנו את הפונקציה המתאימה. לאחר מכן, היה צורך להמיר את הנתונים לפורמט טבלה ולייבא אותם למסד הנתונים chinook באמצעות קוד של create_engine.

לאחר טעינה מוצלחת, טבלת exchange הופיעה במסד הנתונים chinook, כפי שניתן לראות בצילום המסך המצורף. יש לשים לב לכך שבימים מסוימים שער המטבע לא צויין, וזה קורה בגלל שהבורסה לא פעילה בסופי שבוע. ניתן לתקן זאת מאוחר יותר ב-Power BI באמצעות פונקציית "fill down" (מילוי ערכי null על בסיס היום הקודם).



	date_rate	123 usd	123 ils
1	2022-12-22 00:00:00.000	1	3.4937
2	2022-12-21 00:00:00.000	1	3.46783
3	2022-12-20 00:00:00.000	1	3.46208
4	2022-12-19 00:00:00.000	1	3.45574
5	2022-12-18 00:00:00.000	1	3.462
6	2022-12-15 00:00:00.000	1	3.4413
7	2022-12-14 00:00:00.000	1	3.3978
8	2022-12-13 00:00:00.000	1	3.3972
9	2022-12-12 00:00:00.000	1	3.4362
10	2022-12-11 00:00:00.000	1	3.4156
11	2022-12-08 00:00:00.000	1	3.4315
12	2022-12-07 00:00:00.000	1	3.4329
13	2022-12-06 00:00:00.000	1	3.4198
14	2022-12-05 00:00:00.000	1	3.39996
15	2022-12-04 00:00:00.000	1	3.4052
16	2022-12-01 00:00:00.000	1	3.3909

4. POWER BI

החלטנו לבנות את המודל שלנו על בסיס שני דפים:

בעמוד הראשון הצגנו נתוני מכירות, כולל מדדים לפי חודשים ושנים, וכן חמשת הרוכשים המובילים.

בעמוד השני ריכזנו את כל המידע על רצועות, פלייליסטים ואלבומים, כולל מתאם בין אורך השיר לנפח מכירותיו.

כמסנן כללי השתמשנו בסלייסר לפי תאריך ובסלייסר לפי ז'אנר. עקב מאפייני בסיס הנתונים SQL שעליו מבוסס המודל שלנו, המסנן הכללי לפי תאריך פועל על ויזואליזציות הקשורות למכירות (למשל, מדדי נאמנות לקוחות), בעוד שהמסנן הכללי לפי ז'אנר פועל על ויזואליזציות הקשורות למידע על רצועות, פלייליסטים ונפחים (למשל, חמשת האלבומים המובילים לפי נפח השירים).

מסננים כלליים אלו מיושמים באמצעות סלייסרים מסונכרנים. בנוסף, כדי להקל על המעבר בין הדפים, הוספנו כפתור מיוחד.

4.1. השתמשנו במדדים באמצעות RANKX כדי לחשב קבוצות ואלבומים, והגדרנו תנאי שהמדד יעבוד רק על חמשת הקבוצות המובילות. ככלי ויזואליזציה השתמשנו בגרף עמודות מקובצות (stacked column chart) עם ציר X (שם הקבוצה) וציר Y (פורמולה DAX לחמשת האלבומים המובילים).

4.2. השתמשנו ב-measures דרך RANKX, שחישב את הקבוצות והשירים, והגדרנו תנאי ב-return שיתפקד רק על חמשת המוזיקאים המובילים. כאמצעי ויזואליזציה השתמשנו ב-stacked column chart (ציר ה-X – שם הקבוצה, ציר ה-Y – נוסחת DAX עבור חמשת השירים המובילים).

4.3. השתמשנו ב-measures דרך RANKX, שחישב את הז'אנרים והשירים, והגדרנו תנאי ב-return שיתפקד רק על חמשת הז'אנרים המובילים. כאמצעי ויזואליזציה השתמשנו ב-stacked column chart (ציר ה-X – שם הז'אנר, ציר ה-Y – נוסחת DAX עבור חמשת השירים המובילים).

4.4. כדי לקבוע את הפלייליסט הגדול והקטן ביותר, השתמשנו ב-RANKX. סיכמנו את שם הפלייליסט, וחישבנו את מספר השירים בסדר יורד (לפלייליסט הקטן ביותר) ובסדר עולה (לפלייליסט הגדול ביותר), והגדרנו תנאי ב-return שיתפקד רק על הפלייליסט הגדול ביותר או הקטן ביותר.

כדי לקבוע את ממוצע מספר השירים בפלייליסט, השתמשנו בנוסחת DAX הבאה: Average Tracks per Playlist = AVERAGEX (VALUES (dim_playlist[playlist_name]), CALCULATE ([COUNT(dim_playlist[trackid])).

ככלי ויזואליזציה עבור הפלייליסט הגדול והקטן ביותר השתמשנו ב-multi-row card (בשדות ציינו את שם הפלייליסט ואת נוסחת DAX לפלייליסט הגדול/קטן ביותר). לעומת זאת, לציון ממוצע מספר השירים בפלייליסטים השתמשנו בכרטיס (card) ובשדה הזנו את נוסחת DAX Average Tracks per Playlist.

4.5. התחלנו ביצירת measures עם סכומים כלליים בדולרים ובשקלים. עשינו זאת באמצעות SUMX, שבהם הכפלנו שקלים ודולרים בסכום הכולל של רכישות המוצרים. ככלי ויזואליזציה השתמשנו במטריצה (matrix) ובחרנו בעמודות את שמות הלקוחות ובערכים את הסכומים הכלליים בדולרים ובשקלים. כדי להציג רק את חמשת הלקוחות המובילים, השתמשנו הפעם במסנן במודל זה. הגדרנו את המסנן כך: בסוג המסנן בחרנו ב-Top N, ציינו את Top 5 ובסינון את הסכום בשקלים.

4.6. השתמשנו בכלי ויזואליזציה של line chart. לציר X יצרנו עמודה חדשה ב-fact_invoice המייצגת את סוף החודש, והשתמשנו בהיררכיה של חודש ושנה. לציר Y השתמשנו בסכום הכולל של המכירות ב-fact_invoice. כאגדה השתמשנו בתחילת השנה (גם עמודה חדשה ב-fact_invoice).

4.7. כדי לענות על שאלה זו החלטנו להשתמש בכלי ויזואליזציה של bubble chart. כן, הגענו למסקנה באמצעות יצירת bubble chart. לציר X השתמשנו בשניות, לציר Y ובגודל הצינור מיקמנו נוסחת DAX מיוחדת המחשבת את המכירות לפי שירים, ובאגדה ציינו את שמות השירים. בעקבות המחקר, התגלה כי השירים הנמכרים ביותר הם אלה עם משך זמן של 282-300 שניות.

4.8 – 4.9. החלטנו להשתמש בכלי ויזואליזציה של matrix, ובכותרות העמודות הצבנו את המדינות (מ-fact_invoice ולא מ-dim_customer) ואת הז'אנרים. כדי לקבוע את TOP-5 ו-BOTTOM-5 countries, יצרנו שתי נוסחאות DAX מיוחדות:

מכירות כלליות לפי TOP-5 ו-BOTTOM-5 countries (חישוב לפי מיקום המדינה במכירות הכלליות וסיכום מכירות הז'אנרים);

אחוזי מכירות לפי ז'אנרים – חלוקת מכירות כלליות לפי TOP-5 ו-BOTTOM-5 countries לפי מכירות הז'אנרים.

4.10. הויזואליזציות שביצענו בעצמנו:

Map לפי מדינות ובכמות הערך השתמשנו בסכום הכללי של המכירות;

Card עם סכום המכירות הכולל;

Pie Chart עם חישוב נאמנות הלקוחות (מי רכש פעם אחת, מי רכש בין שתי ארבע רכישות ומי רכש יותר מחמש רכישות). השתמשנו בנוסחאות DAX עם Calculate לפי ספירה נפרדת של מזהה האורח עם סינון לפי ספירת רכישות.