פרויקט סיום

מגישים: מיכאל נימקובסקי, אפרים שוחט, אמיר עליוה

פייתון - PANDAS

1. קראנו את הקבצים באמצעות python pandas:

```
raw_dep = pd.read_table ('C://python_data//raw-department.txt', sep='-')
dep_bud_1 = pd.read_json ('C://python_data//raw-department-budget.txt', lines=True)
dep_bud_2 = pd.read_json('C://python_data//raw-department-budget2.txt', orient='records')
```

2. ייצרנו 2 dataframeים אחד עבור כל טבלה כך ש:

a. dataframe אחד מורכב מ-union של שני קבצי ה-json (שניהם מהווים חלק מאותה הטבלה)
b. ה-dataframe השני מורכב מהקובץ בפורמט ה-delimiter.

```
dep_bud_union = pd.concat([dep_bud_1, dep_bud_2],ignore_index = True)
raw_dep = pd.read_table ('C://python_data//raw-department.txt', sep='-')
```

3. ביצענו join בין שתי הטבלאות וסכמנו את התקציב עבור כל מחלקה:

```
DepBudgetDetailed = pd.merge(dep_bud_union, raw_dep, on='department_id', how='outer')
TotalDepBudget = DepBudgetDetailed.groupby(["department_id","department_name"])["budget"].sum().reset_index(name='budget')
print(TotalDepBudget)
```

4. התחברנו ל-postgres ל-chinook והזרמנו אליו את הטבלה לסכמת stg:

```
engine = create_engine('postgresql://postgres:postgres@localhost/chinook')
TotalDepBudget.to_sql(name='department_budget', con=engine, schema='stg', if_exists='replace', index=False)
```

DBT

הערות:

א. בתיקיית DBT הוספנו את כל טבלאות ה-dimensions וטבלאות ה-facts יחד עם schemaה שלהם.

ב. לכל טבלה הוספנו שדה שמכיל את זמן הריצה של ה-dbt וכן השתמשו ב-source בfrom.

1. Dim_playlist

הבאנו לטבלת Dim_playlist את כלל השדות מ-playlisttrack חיברנו אליה את טבלת playlist והבאנו ממנה את כלל השדות.

```
select  pl.playlistid as playlist_id
        ,pl.name as playlist_name
        ,pl.last_update as last_update
        ,plt.trackid as track_id
        ,plt.last_update as track_last_update
        ,'{{ run_started_at.strftime ("%Y-%m-%d %H:%M:%S") }}'::timestamp as dbt_time
from {{source('stg','playlist')}} as pl
    left join {{source('stg','playlisttrack')}} plt on pl.playlistid = plt.playlistid
where 1=1
 {% if is_incremental() %}
 and plt.last_update::timestamp > (select max(pl.last_update) from {{this}})
 {% endif %}
```

- נסו לחשוב, האם יש משהו שדורש התייחסות עקב כך שמדובר בחיבור של שתי טבלאות שונות עם שני תאריכי עדכון שונים? (מבחינת אופי הבאת המידע והתעדכנותו, Materialization וכו')

מבחינת אופי הבאת הנתונים, השתמשנו בif is_incremental בשביל להוסיף לאחר כל עדכון רק את הטראקים המעודכנים מטבלת playlisttrack שנוספו, כדי לא לשלוף בכל פעם מחדש את כל הטבלה.

2. Dim_customer

- הבאנו לטבלה את כלל השדות מ-customer והוספנו שדה שבו יהיה הדומיין מכתובת המייל. על פי בקשת הלקוח השדות של שם פרטי ושם משפחה הם בפורמט של אות ראשונה גדולה והשאר קטנות.

```
SELECT  customer.customerid
    ,INITCAP(customer.firstname) as first_name
    ,INITCAP(customer.lastname) as last_name
    ,customer.company
    ,customer.address
    ,customer.city
    ,customer.state
    ,customer.country
    ,customer.postalcode
    ,customer.phone
    ,customer.fax
    ,customer.email
    ,left(substring(email, position('@' in email) + 1),position('.' in substring(email, position('@' in email) + 1))-1) as domain
    ,customer.supportrepid
    ,customer.last_update
    ,'{{ run_started_at.strftime ("%Y-%m-%d %H:%M:%S")}}'::timestamp as dbt_time
FROM {{source('stg','customer')}}
```

3. Dim_employee

הבאנו לטבלת dimension את כלל השדות מ-employee, חיברנו את טבלת department_budget והוספנו ממנה את שם המחלקה ואת תקציב המחלקה, הוספנו שדה ובו מספר השנים בהם העובד כבר מועסק. בנוסף, הוספנו שדה שיכיל את הדומיין מכתובת המייל. בשאלת הבונוס, הוספנו שדה אשר מציע האם העובד הוא מנהל (is_manager).

```
SELECT   emp.*
     ,db.department_name
     ,db.budget
     ,EXTRACT(YEAR FROM age(CURRENT_DATE, emp.hiredate)) AS years_in_company
     ,left(substring(email, position('@' in email) + 1),position('.' in substring(email, position('@' in email) + 1))-1) as domain
     ,case when emp.employeeid in (select distinct reportsto from stg.employee) then 1
        else 0
          end as is_manager
     ,'{{ run_started_at.strftime ("%Y-%m-%d %H:%M:%S") }}'::timestamp as dbt_time
FROM {{source('stg','employee')}} emp
   JOIN {{source('stg','department_budget')}} db on emp.departmentid = db.department_id
```

4. Dim_track

הבאנו לטבלה את כלל השדות מtrack. בנוסף חיברנו את טבלאות album, artist, mediatype, genre והבאנו מהן את כלל השדות, הפכנו את השדה של משך השיר במילישניות למשך השיר בשניות ועיגלנו כלפי מעלה. בשאלת הבונוס, הוספנו שדה שמציג את משך השיר בפורמט MI:SS (עמודת duration).

```
SELECT  track.trackid as trackid
     ,track.name as track
     ,album.title as album
     ,artist.name as artist
     ,genre.name as genre
     ,mediatype.name as mediatype
     ,track.composer as composer
     ,to_char((milliseconds / 1000) * INTERVAL '1 second', 'MI:SS') as duration
     ,round(track.milliseconds/ 1000,0) AS seconds
     ,track.unitprice
     ,'{{ run_started_at.strftime ("%Y-%m-%d %H:%M:%S")}}'::timestamp as dbt_time
FROM {{source('stg','track')}} track
   join {{source('stg','album')}} album on track.albumid = album.albumid
   join {{source('stg','artist')}} artist on album.artistid = artist.artistid
   join {{source('stg','genre')}} genre on track.genreid = genre.genreid
   join {{source('stg','mediatype')}} mediatype on track.mediatypeid = mediatype.mediatypeid
```

- אילו שדות אין צורך להביא מטבלת track?

לאחר בניית הדאשבורד חזרנו לסעיף זה וראינו שלאנליזה שביצענו לא היה צורך בשדות composer, unitprice, mediatype אך הדבר תלוי בתובנות אליהם רוצים להגיע.

5. Fact_invoice

הכנו טבלת fact ל-invoice והבאנו ממנה את כלל השדות בנוסף הפכו את הטבלה כך שתהיה
Materialized – Incremental

```
{{ config(materialized='incremental',unique_key='invoiceid') }}

SELECT  invoice.*
      ,'{{ run_started_at.strftime ("%Y-%m-%d %H:%M:%S")}}'::timestamp as dbt_time
FROM {{source('stg','invoice')}}
WHERE 1=1
 {% if is_incremental() %}
 and last_update::timestamp > (select max(last_update) from {{this}})
 {% endif %}
```

- האם צריך להביא מ-invoice את השדות של הכתובת?

לצורך ביצוע הניתוח על הנתונים שמתקבלים מהטבלאות ראינו כי לא היה לנו שימוש בשדות billingaddress, billingcity, billingcountry, billingpostalcode, billingstate לכן הבאת שדות אלו לא היוותה תרומה משמעותית לביצוע האנליזה. עם זאת, במידה והיינו בוחרים לעשות את האנליזה בדשבורד על נושאים אחרים שדות אלו יכולים היו לשמש אותנו.

6. Fact_invoiceline

הכנו טבלת fact ל-invoiceline והבאנו ממנה את כלל השדות בנוסף הפכו את הטבלה כך שתהיה
Materialized – Incremental

```
{{ config(materialized='incremental',unique_key='invoicelineid') }}

SELECT  invoiceline.*
      ,'{{ run_started_at.strftime ("%Y-%m-%d %H:%M:%S")}}'::timestamp as dbt_time
FROM {{source('stg','invoiceline')}}
WHERE 1=1
 {% if is_incremental() %}
 and last_update::timestamp > (select max(last_update) from {{this}})
 {% endif %}
```

API CURRENCIES

שלבי העבודה:

1. מצאנו את התאריכים הרלוונטיים שיש להביא עבורם את המידע. בשביל למצוא את התאריכים הרלוונטיים לקחנו מעמודת invoicedate בטבלת invoice את התאריך המקסימלי והתאריך המינימאלי שכן אלו הם התאריכים בהם התבצעו העסקאות בפועל.

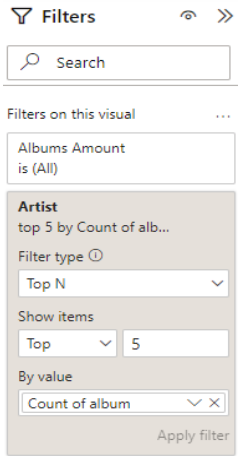
2. הכנו סקריפט פייתון שמתשאל את ה-API ומביא את יחס ההמרה מדולר לשקל עבור כל יום בתקופת הזמן הרלוונטית, הוצאנו טבלה שמכילה עבור כל תאריך רלוונטי את ערך ההמרה מדולר לשקל. גילינו כי חלק מהתאריכים לא מופיעים בAPI שבחרנו, חקרנו את הסיבה ומצאנו כי בישראל נהוג מסחר של 5 ימים, מיום שני עד שישי. כך ששער ההמרה של יום שישי נותר זהה בימים שבת וראשון ולכן הAPI לא מחזיר שער המרה לימים אלו. כדי שכל הימים יופיעו בתרגיל החלטנו לאחר התייעצות כי נוסיף את התאריכים החסרים בdbt ולאחר מכן נעשה להם מילוי אוטומטי כלפי מטה בתוכנת Power BI. בדרך זו הצגנו את השער התקין בכל הימים הרצויים והתוצאה יצאה קוהרנטית.

```
import requests
import pandas as pd
from sqlalchemy import create_engine

start_date = "2018-01-01"
end_date = "2022-12-22"
base_currency = "USD"
target_currency = "ILS"
api_key = "PX6trJMKpoX3Derc2EIZnfkNomjbuJG3"

def get_daily_rates(start_date, end_date, base_currency, target_currency, api_key):
    url = f"https://www.alphavantage.co/query?function=FX_DAILY&from_symbol={base_currency}&to_symbol={target_currency}&outputsize=full&apikey={api_key}"
    response = requests.get(url)
    rates = response.json()['Time Series FX (Daily)']
    filtered_rates = {date: rate for date, rate in rates.items() if start_date <= date <= end_date}
    return filtered_rates

daily_rates = get_daily_rates(start_date, end_date, base_currency, target_currency, api_key)
converted_amounts = [{'date_rate': pd.to_datetime(date), 'usd': 1, 'ils': float(rate['4. close'])} for date, rate in daily_rates.items()]
df = pd.DataFrame(converted_amounts)
```

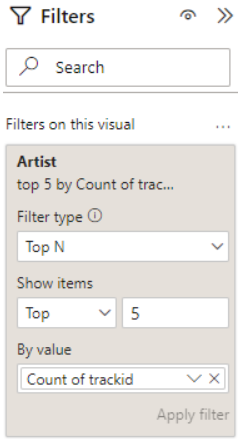
3. הזרמנו את הטבלה הנ"ל לתוך טבלת exchange_rate ל-Database בשם chinook לסכמת stg.

```
engine = create_engine('postgresql://postgres:postgres@localhost/chinook')
df.to_sql(name='exchange_rate', con=engine, schema='stg', if_exists='replace', index=False)
```

1. **הצגנו את ה-top 5 של האמנים עם הכי הרבה אלבומים.**
ייצרנו ויז'ואל של טבלה, השתמשנו בשדה artist מטבלת dim_track בשביל שמות האמנים.
בvalues בשביל למצוא את כמות האלבומים השתמשנו בcount distinct של עמודת album מטבלה זו.
לאחר מכן, בשביל לסנן את חמשת האמנים עם מירב האלבומים השתמשנו בפילטר של TOP N.



2. **הצגנו את ה-top 5 של האמנים עם הכי הרבה שירים (tracks).**
ייצרנו ויז'ואל של טבלה, השתמשנו בשדה artist מטבלת dim_track בשביל שמות האמנים.
בvalues בשביל למצוא את כמות השירים השתמשנו בcount של עמודת trackid מטבלה זו.
לאחר מכן, בשביל לסנן את חמשת האמנים עם מירב השירים השתמשנו בפילטר של TOP N.
יש לשים לב כי קיימים שני אמנים עם 92 שירים שנמצאים במקום החמישי, לכן בחרנו להציג 6 אמנים בשביל לתת מקום לשניהם.



3. **הצגנו את ה-top 5 של הז'אנרים עם הכי הרבה <u>אלבומים</u>.**
ייצרנו ויז'ואל של טבלה, השתמשנו בשדה genre מטבלת dim_track בשביל שמות ז'אנרים.
בvalues בשביל למצוא את כמות האלבומים השתמשנו בcount distinct של עמודת album מטבלה זו.
לאחר מכן, בשביל לסנן את חמשת הז'אנרים עם מירב האלבומים השתמשנו בפילטר של TOP N.

4. הצגנו את הפלייליסט עם הכי הרבה שירים, הפלייליסט עם הכי מעט שירים, ואת ממוצע השירים בפלייליסט

א. הפלייליסט עם הכי הרבה שירים, הצגנו דרך multi-row card השתמשנו בסינון של Top N בנוסף, השתמשנו בעמודה של playlist_id כי רצינו להדגיש שקיימים שני פלייליסטים שהם עם אותו השם **(Music)**, וגם עם מספר שירים זהה **(3290)**, השוני היחיד בניהם הוא ה-playlist_id.

ב. הפלייליסט עם הכי מעט שירים, הצגנו דרך multi-row card השתמשנו בסינון של Top N ומצאנו כי ישנם שני פלייליסטים עם מספר השירים הנמוך ביותר ולכן הצגנו את שניהם.

ג. ממוצע השירים בפלייליסט, הצגנו ב-card השתמשנו במדד שיצרנו לכך:

```
Average Tracks per Playlist =
CALCULATE(
   COUNTROWS('dim_playlist'),
   REMOVEFILTERS('dim_playlist'[playlist_id])
) /
COUNTROWS(
   DISTINCT('dim_playlist'[playlist_id])
)
```

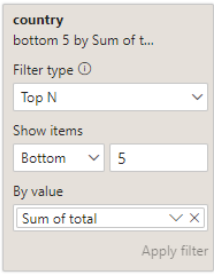
5. מיהם 5 הלקוחות שרכשו בסכומים הגבוהים ביותר? מה הסכום (בדולרים וגם בשקלים)?
ייצרנו ויז'ואל של matrix, השתמשנו בשדה fullname מטבלת dim_customer בשביל שמות הלקוחות. בvalues בשביל למצוא את סה"כ ההכנסות פר לקוח בדולרים השתמשנו במדד total in usd ולחישוב בשקלים השתמשנו במדד total in ils. בנוסף, בשביל לסנן את חמשת הלקוחות שרכשו בסכומים הגבוהים ביותר השתמשנו בפילטר של TOP N. לבסוף, השתמשנו בconditional formatting בעיצוב של data bars על העמודות של ההכנסות בדולרים ובשקלים בשביל להציג באופן ברור את ההתפלגות של ההכנסות מחמשת הלקוחות הרווחים ביותר.

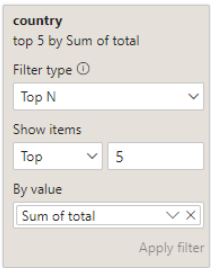
6. הצגנו גרף של סכום מכירות עבור כל חודש בכל שנה.
ייצרנו גרף של clustered column chart, את ציר הx חילקנו לשנים וחודשים ובציר הy עשינו sum לעמודת total מטבלת fact_invoice

7. האם קיימת קורלציה בין אורך השיר לבין הרווחים שהוא מניב במכירות?
כדי לבדוק אם קיימת קורלציה בנינו Scatter chart הצבנו בציר ה-x את השניות מטבלת dim_track ובציר ה-y סכמנו את המדד total per track שבנינו בשביל התרגיל.
מצאנו ששירים באורך שבין 150 ל350 שניות הרווח בהם הוא הגבוהה ביותר.

```
total per track = fact_invoiceline[unitprice] * fact_invoiceline[quantity]
```

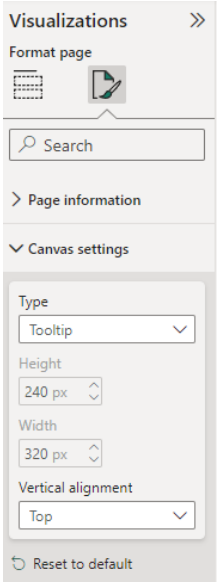
8. **הצגנו רווחים לפי מדינה עבור 5 המדינות עם הכי הרבה רווחים ו-5 המדינות עם הכי מעט רווחים.**
ייצרנו שני ויז'ואלים של Treemap, הקטגוריות הם שמות המדינות והערכים הם סכימה של עמודת total מטבלת fact_invoice השוני בין שני הויז'ואלים הוא בסינון של השל הטבלאות בTOP N, מצורפים שני הסינונים שביצענו הראשון מוצא את 5 התחתונים והשני את 5 העליונים:

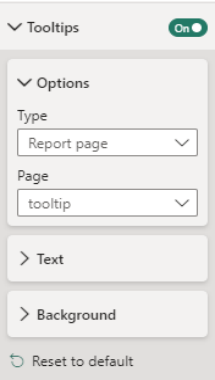


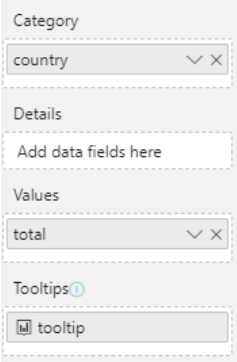


9 **בונוס: בתוך כל מדינה משאלה 8, מהו אחוז הרווחים של כל ז'אנר מתוך כלל הרווחים במדינה?**

בשביל סעיף זה יצרנו page חדש בשם tooltip שמציג Donut chart המחולק לשם ה-genre וב-values השתמשנו בספירה של כמות השירים מטבלת dim_track. לאחר מכן הגדרנו את העמוד כ-tooltip הלכנו לdashboard1 והגדרנו בשתי הויז'ואלים של החלוקה עפ"י מדינות את הtooltip



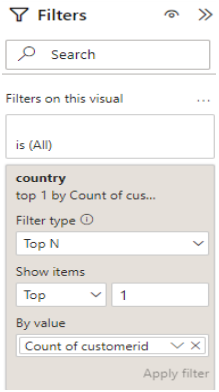




- **פתרו את הבאים והוסיפו לאחר מכן 3 משלכם.**

א. סה"כ סכום המכירות, הצגנו דרך card בראש הדשבורד הראשון את הסכום הכולל של המכירות, לצורך כך השתמשנו בsum של עמודת total מטבלת fact_invoice.

ב. המדינה עם כמות הלקוחות הגדולה ביותר, הצגנו דרך multi-row card את המדינה עשינו ספירה לעמודת customerid ולבסוף השתמשנו בסינון של Top N



ג. הצגנו את ה-top 5 של הז'אנרים עם הכי הרבה שירים.
ייצרנו ויז'ואל של טבלה, השתמשנו בשדה genre מטבלת dim_track בשביל שמות ז'אנרים.
בvalues בשביל למצוא את כמות השירים השתמשנו בcount של עמודת trackid מטבלה זו.
לאחר מכן, בשביל לסנן את חמשת הז'אנרים עם מירב השירים השתמשנו בפילטר של TOP N.

- **שמנו את הגרפים בשני דאשבורדים שונים, והוספנו פילטרים שיחתכו את שניהם**
הוספנו פילטר של תאריך ופילטר של ז'אנר, כדי שהפילטרים יוכלו לחתוך את הגרפים בו זמנית בשני הדשבורדים העתקנו את שני הפילטרים מהדשבורד הראשון והדבקנו בדשבורד השני קיבלנו הודעה האם אנחנו רוצים שהסלייסרים יהיה מסונכרנים עם הויז'ואלים של הדשבורד השני לחצנו סנכרן.