**פרויקט- אנליזה של ביג דאטה**

מאגר מידע: <https://www.kaggle.com/datasets/dilwong/flightprices/data>

מאגר המידע מכיל 27 עמודות, כ82 מיליון שורות בנפח כולל של כ31GB.  
כל שורה במאגר מתאר חיפוש טיסה באתר Expedia, עבור כל טיסה מתועד ID יחודי לטיסה, תאריך חיפוש, תאריך הטיסה, מוצא, יעד, זמן טיסה, מרחק טיסה, ימי טיסה, מידע אודות תעריף ותוספות לכרטיס, האם הטיסה ישירה, תעריף בסיס, מחיר מלא, מס' המושבים שנותרו בטיסה זו, חברת תעופה, מחלקה, סוג מטוס ועוד.

בנוסף, לצורך מימוש אחת השאילתות הוספנו קובץ CSV שיצרנו, הקובץ מכיל 3 עמודות (code, lat, lon) המכיל את המיקום של כל שדה תעופה.

**פירוט עמודות חשובות:**

1. **legId** - מזהה ייחודי לכל טיסה.

משמש לזיהוי ולמעקב אחרי טיסות מסוימות.

1. **searchDate** - תאריך החיפוש באתר Expedia .

מאפשר ניתוח קשר בין זמן החיפוש למחיר הכרטיס.

1. **flightDate** - תאריך הטיסה.

מאפשר לנתח כיצד מחירי הכרטיסים משתנים ביחס לתאריך הטיסה ותאריך ההזמנה.

1. **startingAirport / destinationAirport**  - קודים של שדות התעופה המוצא והיעד (IATA).

מסייע בניתוח מסלולים בין שדות תעופה.

1. **totalTravelDistance** - המרחק הכולל של הטיסה במיילים.

עשוי להיות קשור למחיר הטיסה ולמשך הטיסה.

**מחירים ומחלקות:**

1. - **totalFare** המחיר הכולל של הכרטיס כולל מיסים ועמלות (בדולרים).
2. - **isNonStop**האם מדובר בטיסה ישירה ללא עצירות ביניים (Boolean).

עשוי להשפיע על נוחות הנוסע ועל המחיר.

**זמני טיסות:**

1. **segmentsDepartureTimeRaw / segmentsArrivalTimeRaw** - זמני המראה ונחיתה לכל קטע טיסה.

מאפשר לנתח שעות פופולריות (ואת ההשפעה של השעה על המחיר) ולחשב זמני טיסות בפועל.

1. – **travelDuration**משך הטיסה בפורמט שעות ודקות.

**מאפייני נוסעים ומטוסים:**

1. - **seatsRemaining** מספר המושבים שנותרו בטיסה.

עשוי להשפיע על המחיר עקב ביקוש והיצע.

על מנת להכניס את הדאטה לDUCKDB השתמשנו ביכולות של DUCKDB, מפאת גודל הדאטה, הקובץ נוצר רק אם הוא לא קיים, שכן מדובר בהמון זמן ריצה.

הקוד להלן:

if not file\_exists("database.duckdb"):  
 duckdb\_conn = duckdb.connect("database.duckdb")  
 duckdb\_conn.execute("CREATE TABLE main AS SELECT \* FROM read\_csv\_auto('itineraries.csv')")  
else:  
 duckdb\_conn = duckdb.connect("database.duckdb")

בנוסף, על מנת להכניס את הקובץ השני לDUCKDB ביצענו:

duckdb\_conn.execute("CREATE TABLE airportsLocation AS SELECT \* FROM read\_csv\_auto('airportsLocation.csv')")

השאלות שנשאל על הדאטה והשאילתות SQL שנבצע עבורן:

1. כיצד משתנים מחירי הטיסות ביחס לזמן החיפוש מול מועד הטיסה?

duckdb\_conn.execute("""duckdb\_conn.execute("""  
 CREATE TABLE query1 AS  
 WITH time\_ranges AS (  
 SELECT \*,  
 (flightDate - searchDate) as days\_until\_flight  
 FROM main  
 )  
 SELECT   
 days\_until\_flight as days\_before\_flight,  
 ROUND(AVG(totalFare), 2) as avg\_fare,  
 COUNT(\*) as number\_of\_searches,  
 ROUND(MIN(totalFare), 2) as min\_fare,  
 ROUND(MAX(totalFare), 2) as max\_fare  
 FROM time\_ranges  
 GROUP BY days\_until\_flight  
 ORDER BY days\_until\_flight  
""")

מהטבלה שקיבלנו, ניתן לראות כי יש באופן כללי יש ירידת מחירים קלה במחיר הממוצע כאשר מזמינים יותר מ45 יום לפני הטיסה, לאחר מכן המחיר עולה באופן רציף עד למועד הטיסה.

1. האם משתלם יותר לטוס בטיסה ישירה או בטיסה עם עצירות (קונקשן) בהתחשב במרחקי טיסה שונים?

duckdb\_conn.execute("""  
CREATE TABLE query2 AS  
SELECT   
 CASE   
 WHEN totalTravelDistance < 500 THEN 'Short (<500 miles)'  
 WHEN totalTravelDistance < 1000 THEN 'Medium (500-1000 miles)'  
 ELSE 'Long (>1000 miles)'  
 END as distance\_category,  
 ROUND(AVG(CASE WHEN isNonStop THEN totalFare END), 2) as direct\_avg\_fare,  
 ROUND(AVG(CASE WHEN NOT isNonStop THEN totalFare END), 2) as connection\_avg\_fare,  
 COUNT(CASE WHEN isNonStop THEN 1 END) as direct\_flights\_count,  
 COUNT(CASE WHEN NOT isNonStop THEN 1 END) as connection\_flights\_count  
FROM main  
WHERE totalTravelDistance IS NOT NULL  
GROUP BY 1  
ORDER BY distance\_category;  
""")

מהטבלה שקיבלנו, ניתן לראות באופן מפתיע שמחירן הממוצע של טיסות ישירות בכל הקטגוריות יהיה זול יותר מטיסות לא ישירות.

1. איך שעת ההמראה משפיעה על המחיר, הזמינות ואופי הטיסה (ישירה\לא ישירה)?

duckdb\_conn.execute("""  
 CREATE TABLE query3 AS  
 WITH flight\_times AS (  
 SELECT \*,  
 CAST(SUBSTR(segmentsDepartureTimeRaw, 12, 2) AS INTEGER) as departure\_hour  
 FROM main  
 WHERE segmentsDepartureTimeRaw IS NOT NULL  
 )  
 SELECT   
 departure\_hour,  
 ROUND(AVG(totalFare), 2) as avg\_fare,  
 ROUND(AVG(seatsRemaining), 1) as avg\_seats\_remaining,  
 COUNT(\*) as number\_of\_flights,  
 ROUND(MIN(totalFare), 2) as min\_fare,  
 ROUND(MAX(totalFare), 2) as max\_fare,  
 SUM(isNonStop::INTEGER) \* 100.0 / COUNT(\*) as nonstop\_percentage  
 FROM flight\_times  
 GROUP BY departure\_hour  
 ORDER BY departure\_hour  
""")

מהטבלה שקיבלנו, נראה כי אין דפוס אחיד לגמרי למחיר הממוצע לטיסה ביחס לשעה, אך ניתן לראות שבשעות 6-10 בבוקר וכן בשעות 22-00 בלילה מחירי הטיסות יקרים יותר. לאחר השעה 10 בבוקר המחירים יורדים (כמעט תמיד. במקומות בהן אין ירידה, העלייה היא לא משמעותית) באופן רציף עד לשעה 22 ומשם מתחילה עלייה משמעותית.

בנוסף נראה כי אחוז הטיסות ללא עצירה עולה באופן רציף (כמעט) עד לשעה 20 בערב שבה 50 אחוז מהטיסות הן ישירות. לאחר מכן אחוז הטיסות הישירות יורד משמעותית כך שבשעות הלילה יש הכי מעט טיסות ישירות.

אין הרבה פער במספר המושבים הזמינים לאורך שעות היום, מדובר בפער של פחות מכיסא בממוצע לאורך כל שעות היום. כמו כן אין חוקיות מסויימת.

1. מהם ימי השבוע היקרים והזולים ביותר לטיסה והאם יש הבדל בין טיסות ישירות לטיסות עם עצירות?

duckdb\_conn.execute("""  
 CREATE TABLE query4 AS  
 WITH day\_info AS (  
 SELECT   
 totalFare,  
 STRFTIME(flightDate::DATE, '%A') as flight\_day,  
 isNonStop  
 FROM main  
 )  
 SELECT   
 flight\_day,  
 ROUND(AVG(totalFare), 2) as avg\_fare,  
 ROUND(AVG(CASE WHEN isNonStop THEN totalFare END), 2) as avg\_nonstop\_fare,  
 ROUND(AVG(CASE WHEN NOT isNonStop THEN totalFare END), 2) as avg\_connection\_fare,  
 COUNT(\*) as number\_of\_flights  
 FROM day\_info  
 GROUP BY flight\_day  
 ORDER BY CASE flight\_day  
 WHEN 'Sunday' THEN 0  
 WHEN 'Monday' THEN 1   
 WHEN 'Tuesday' THEN 2  
 WHEN 'Wednesday' THEN 3  
 WHEN 'Thursday' THEN 4  
 WHEN 'Friday' THEN 5  
 WHEN 'Saturday' THEN 6  
 END  
""")

מהטבלה שקיבלנו, ניתן לראות כי הימים שלישי ורביעי הם הזולים ביותר לטיסה, ולאחר מכן מחיר הטיסה הממוצע עולה לאורך סוף השבוע (עם ירידה קלה בשבת) עד ליום ראשון שהוא היום היקר ביותר לטיסה. לאחר יום ראשון המחירים יורדים.

קיימת קורולציה כמעט מלאה בין תנודות המחיר הממוצע של טיסות לא ישירות לטיסות ישירות ביחס ליום בשבוע אך טיסה לא ישירה תמיד תהיה יקרה יותר.

1. מהם הנתיבים (מוצא ויעד) של טיסות להם היו שינויי מחיר קיצוניים (מעל 20%) כלפי מעלה וכלפי מטה?

duckdb\_conn.execute("""  
 CREATE TABLE query5 AS  
 WITH price\_volatility AS (  
 SELECT   
 startingAirport,  
 destinationAirport,  
 flightDate,  
 AVG(totalFare) as avg\_fare,  
 LAG(AVG(totalFare)) OVER (  
 PARTITION BY startingAirport, destinationAirport  
 ORDER BY flightDate  
 ) as prev\_day\_fare  
 FROM main  
 GROUP BY startingAirport, destinationAirport, flightDate  
 )  
 SELECT   
 flightDate,  
 startingAirport,  
 destinationAirport,  
 avg\_fare,  
 prev\_day\_fare,  
 ((avg\_fare - prev\_day\_fare) / prev\_day\_fare \* 100) as daily\_change\_percent  
 FROM price\_volatility  
 WHERE ((avg\_fare - prev\_day\_fare) / prev\_day\_fare \* 100) > 20  
 OR ((avg\_fare - prev\_day\_fare) / prev\_day\_fare \* 100) < -20  
 ORDER BY daily\_change\_percent DESC;  
""")

מטבלה זו נוכל לראות באיזה קווי טיסה התרחשו שינויי מחיר קיצוניים של מעל 20 אחוז, כאשר יש טיסות שמחירן השתנה ב200 אחוז! ניתן לראות כי הטיסות שמחירן קפץ למעלה במעל 80 אחוז התרחשו בעיקר בחודשים אוקטובר ונובמבר 2022

**הסיפור שנרצה לספר:**

בעולם התיירות והטיסות, קיימים גורמים שונים המשפיעים על מחירי הטיסה. נרצה לנתח אותם ולהבין מהם הגורמים לכך ולזהות תנודות במחירי הטיסות.

הניתוח מתמקד בדפוסי המחירים של כרטיסי טיסה, תוך בחינה של קשרים בין גורמים שונים כמו הזמן בין ההזמנה עד הטיסה, מרחק הטיסה, זמני יציאה, וסוגי טיסות (ישירה או עם עצירות).

השאילתות מספקות תובנות ברורות שמראות איך ומתי המחירים משתנים ומהם הגורמים שמניעים את השינויים הללו.

כדי ליצור טבלה לדוגמא בת 500 שורות, נשתמש בשאילתה הבאה שדוגמת את הטבלה הגדולה:

duckdb\_conn.execute("CREATE TABLE sample AS SELECT \* FROM main USING SAMPLE 500;")

כדי להעלות את הדאטה הקטן לתוך SQLITE, נשתמש בפקודות הבאות:

נתקין את התוסף של SQLite, לאחר מכן נתחבר לדאטה בייס ונקרא לו sqliteDB. נבצע העתקה של הטבלאות בעזרת שאילתה שמתחילה ב – CREATE TABLE ולבסוף נקבל את הדאטה הקטן שלנו בSQLITE.

duckdb\_conn.execute("INSTALL sqlite;")  
duckdb\_conn.execute("LOAD sqlite;")  
duckdb\_conn.execute("ATTACH 'database.sqlite' AS sqliteDB (TYPE SQLITE);")  
  
duckdb\_conn.execute("CREATE TABLE sqliteDB.sample AS SELECT \* FROM sample")  
duckdb\_conn.execute("CREATE TABLE sqliteDB.airportsLocation AS SELECT \* FROM airportsLocation")  
for i in range(1, 6):  
 duckdb\_conn.execute(f"CREATE TABLE sqliteDB.query{i} AS SELECT \* FROM query{i}")

**שיטות הקטנת הנתונים**

במהלך כל אחת מהשאילתות נעשה שימוש בטכניקות ספציפיות להקטנת הדאטה, תוך שמירה על מידע חיוני לתובנות וניתוחים. הנה פירוט רחב יותר:

1. **קיבוץ (GROUP BY):**

קיבוץ הנתונים לפי קטגוריות נבחרות, כמו:

* **מספר ימים עד הטיסה(Query 1) :** מקבץ את כל הנתונים לפי ערכי days\_until\_flight, כך שבמקום לשמור כל חיפוש כיחידה נפרדת, שומרים סיכומים סטטיסטיים עבור כל מספר ימים.
* **קטגוריות מרחק :(Query 2)** שימוש בקטגוריות "Short/Medium/Long" כדי להפחית את המידע על טיסות ספציפיות ולהתמקד בתבניות כלליות.
* **שעות יציאה :(Query 3)** מעבר משימוש בזמנים מדויקים (שעות ודקות) לשעות כלליות, כדי לצמצם את מספר הקטגוריות ולזהות מגמות רחבות יותר.
* **ימים בשבוע :(Query 4)** קיבוץ לפי יום בשבוע מאפשר לזהות השפעות לפי ימי חול או סוף שבוע, במקום להתמקד בתאריכים ספציפיים.

**רציונל:** הקיבוץ מפחית את מספר השורות ומחליף את המידע הגולמי בערכים סטטיסטיים מרכזיים, כמו ממוצעים, מינימום ומקסימום, שחשובים לקבלת תובנות.

1. **פונקציות אגרגציה :(Aggregations)**

בכל שאילתה מחושבים ערכים כמו:

* **ממוצעים :(AVG)** למשל, מחיר ממוצע של כרטיסי טיסה כדי להבין את עלות הטיסות בכל קטגוריה.
* **ספירות :(COUNT)** למשל, מספר החיפושים או הטיסות בכל קטגוריה כדי למדוד פעילות.
* **ערכים קיצוניים :(MIN/MAX)** כמו מחיר מינימלי ומקסימלי של טיסות, כדי לזהות שונות במחירים.
* **אחוזים :(PERCENTAGE)** למשל, אחוז הטיסות הישירות ב-Query 3.

**רציונל**: פונקציות האגרגריה מאפשרות ריכוז מידע משמעותי במקום שמירה על כל הנתונים הגולמיים, שמכבידים על תהליך הניתוח.

1. **סינון :(Filtering)**

* **מחיקת נתונים לא רלוונטיים:** לדוגמה, בQuery 5 - מסננים רק מקרים שבהם השינוי היומי במחיר היה מעל 20% או מתחת ל--20%.
* **התמקדות בערכים לא ריקים:** ב Query 2 - מתמקדים בטיסות עם מרחקים מוגדרים ומסננים שורות שבהן totalTravelDistance חסר.

**רציונל:** סינון מקטין את הדאטה בכך שהוא מוודא שהניתוח יתבצע רק על מידע משמעותי ורלוונטי, תוך התעלמות מפרטים שלא תורמים להבנה.

1. **חלוקה לקטגוריות :(Bucketing)**

ב - Query 2-המרחקים מחולקים לטווחים קבועים מראש, כמו ."Short/Medium/Long"

ב Query 4 - - ימים בשבוע מחולקים לסדר מוגדר כדי לזהות הבדלים בין ימי חול לסופי שבוע.

**רציונל:** חלוקה לקטגוריות מפשטת את ניתוח הנתונים ומאפשרת זיהוי מגמות ברורות בתוך קבוצות מוגדרות היטב.

1. **פונקציות חלון (Window Functions):**

ב - Query 5 נעשה שימוש ב LAG - כדי לחשב את המחיר הממוצע של יום קודם ולהשוות אותו למחיר הממוצע הנוכחי.

**רציונל:** פונקציות חלון שומרות על הקשר בין שורות ומאפשרות ניתוח השינויים לאורך זמן, תוך שמירה על מבנה נתונים מסודר לצמצום מידע לא רלוונטי.

**מה קיבלנו מכל זה?**

1. **צמצום נפח המידע:** במקום לשמור מיליוני שורות של נתונים גולמיים, התוצאה היא טבלה קטנה יותר וממוקדת יותר.
2. **פשטות בניתוח:** במקום לעבד נתונים מורכבים, יש נתונים מסוכמים שקל יותר לעבוד איתם.
3. **זיהוי מגמות ודפוסים:** המידע המעובד מאפשר לזהות מגמות ברורות ולבצע אופטימיזציות בקלות.
4. **שיפור ביצועים:** פחות נתונים לעיבוד מובילים לזמן תגובה קצר יותר וביצועים טובים יותר עבור שאילתות עתידיות.

**הטבלאות שנמצאות בדאטה בייס הקטן:**

1. **query1 –** הטבלה מכילה 5 עמודות ו60 שורות. מטרתה להציג את שינוי מחירי הטיסות ביחס לזמן החיפוש אל מול מועד הטיסה. כלומר, נרצה לראות אם יש קשר בין מועד ההזמנה למחירי הטיסה.
2. **query2 –** הטבלה מכילה 5 עמודות ו3 שורות, מטרתה לנתח כדאיות של טיסה ישירה מול טיסה לא ישירה תוך התייחסות לטווח הטיסה.
3. **query3 –** הטבלה מכילה 7 עמודות ו21 שורות, שורה עבור כל שעה עגולה שתועדה בה טיסה. מטרתה לנתח שעות שיותר יקר או יותר זול לטוס בהן, שעות בהן יש זמינות מושבים גבוהה יותר וכן את השעות בהן יש יותר טיסות ישירות.
4. **query4 –** הטבלה מכילה 5 עמודות ו7 שורות, שורה עבור כל יום בשבוע. מטרתה לנתח את הימים הזולים והיקרים ביותר באותו שבוע באופן ממוצע וכן בהתייחסות לטיסות ישירות או לא ישירות.
5. **query5 –** הטבלה מכילה 6 עמודות ו14194 שורות. מטרתה לנתח טיסות בהן היה שינוי מחיר ממוצע של מעל 20 אחוז (למעלה או למטה).
6. **Sample –** הטבלה מכילה 27 עמודות ו500 שורות. מטרתה להוות מעין "מדגם" של הטבלה הגדולה.
7. **airportsLocation –** הטבלה מכילה 3 עמודות ו16 שורות. מכיל מידע גולמי על מיקום של כל שדה תעופה בדאטה.

**\*\*\*\*\*כל מה שקשור לויזואליזציות, תיאור סכמתי של הדשבורד\*\*\*\***

**GRAPH DB:**

תמונה שמכילה עיגול, צילום מסך, תרשים, טקסט

התיאור נוצר באופן אוטומטי

סכמת הגרף מציגה את מבנה הדאטה בייס שעליו אנו עובדים.

במרכז הסכמה נמצא הצומת המרכזי Flight (טיסה) שמקושר ל-8 צמתים המקיפים אותו, כל אחד מייצג מאפיין שונה של הטיסה.

הצמתים המקושרים לטיסה הם:

1. תאריך טיסה (Flight Date) - מתי הטיסה אמורה להתקיים, מקושר דרך FLIES\_ON.

2. מחיר כולל (Total Fare) - עלות הטיסה, מקושר דרך HAS\_PRICE.

3. סוג טיסה (Flight Type) - האם זו טיסה ישירה או עם עצירות, מקושר דרך IS\_TYPE.

4. מרחק טיסה (Travel Distance) - כמה מיילים הטיסה מכסה, מקושר דרך TRAVELS.

5. שדה תעופה (Airport) - קוד שדה התעופה, מקושר דרך DEPARTS\_FROM/ARRIVES\_AT כי כל טיסה יוצאת משדה תעופה אחד ונוחתת בשדה תעופה אחר.

6. מושבים (Seats) - כמה מושבים פנויים בטיסה, מקושר דרך HAS\_SEATS.

7. זמן המראה (Departure Time) - מתי הטיסה ממריאה, מקושר דרך DEPARTS\_AT.

8. תאריך חיפוש (Search Date) - מתי בוצע החיפוש של הטיסה, מקושר דרך SEARCHED\_ON.

כל הקשרים בין הצומת המרכזי לשאר הצמתים (nodes) מסומנים בחיצים המראים את כיוון הקשר, ולכל קשר יש תווית המתארת את סוג הקשר (למשל FLIES\_ON, HAS\_PRICE וכו').

הסכמה הזו מאפשרת לעקוב אחר מחירי טיסות, לנתח זמינות מושבים, להשוות בין סוגי טיסות שונים ולזהות מגמות במחירים לאורך זמן.

\*\*\*\***צילומי מסך של הדאשבורד\*\*\*\***

**\*\*\*26. סיכום תרגיל\*\*\*\***

**\*\*\*אחריות של כל אחד מאיתנו\*\*\*\***

**\*\*\*\*רידמי\*\*\***