מגישים: בנימין רוסין, יוחאי בניטה

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 כGB31.
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.
2. **searchDate** - תאריך החיפוש באתר Expedia .
מאפשר ניתוח קשר בין זמן החיפוש למחיר הכרטיס.
3. **flightDate** - תאריך הטיסה.
מאפשר לנתח כיצד מחירי הכרטיסים משתנים ביחס לתאריך הטיסה ותאריך ההזמנה.
4. **startingAirport / destinationAirport** - קודים של שדות התעופה המוצא והיעד (IATA).
מסייע בניתוח מסלולים בין שדות תעופה.
5. **totalTravelDistance** - המרחק הכולל של הטיסה במיילים.
עשוי להיות קשור למחיר הטיסה ולמשך הטיסה.

מחירים ומחלקות:

6. **totalFare** - המחיר הכולל של הכרטיס כולל מיסים ועמלות (בדולרים).
7. **isNonStop** - האם מדובר בטיסה ישירה ללא עצירות ביניים.(Boolean)
עשוי להשפיע על נוחות הנוסע ועל המחיר.

זמני טיסות:

11. **segmentsDepartureTimeRaw / segmentsArrivalTimeRaw** - זמני המראה ונחיתה לכל קטע טיסה.
מאפשר לנתח שעות פופולריות (ואת ההשפעה של השעה על המחיר) ולחשב זמני טיסות בפועל.
12. **travelDuration** – משך הטיסה בפורמט שעות ודקות.

מאפייני נוסעים ומטוסים:

14. **seatsRemaining** - מספר המושבים שנותרו בטיסה.
עשוי להשפיע על המחיר עקב ביקוש והיצע.

הכנסת הדאטה לDUCKD:

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 (של מיקומי שדות התעופה) 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.

2. 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.

3. 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.

4. 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.

5. כמה טיסות מתקיימות לכל קו טיסה (ישירות ולא ישירות) והאם יש קשר בין מספר הטיסות למחיר הטיסה?

```
duckdb_conn.execute("""
    CREATE TABLE query5 AS
    SELECT
        f.startingAirport,
        f.destinationAirport,
        f.isNonStop,
        COUNT(*) as num_flights,
        ROUND(AVG(totalFare), 2) as avg_fare,
        ROUND(MIN(totalFare), 2) as min_fare,
        ROUND(MAX(totalFare), 2) as max_fare,
        a1.lat as start_lat,
        a1.lon as start_lon,
        a2.lat as dest_lat,
        a2.lon as dest_lon
    FROM main f
    JOIN airportsLocation a1 ON f.startingAirport = a1.airport
    JOIN airportsLocation a2 ON f.destinationAirport = a2.airport
    GROUP BY f.startingAirport, f.destinationAirport, f.isNonStop,
             a1.lat, a1.lon, a2.lat, a2.lon
    ORDER BY f.startingAirport, f.destinationAirport, f.isNonStop
DESC;
""")
```

מטבלה זו נוכל לראות את מספר הטיסות לכל קו (מוצא ויעד), בהתייחס לטיסות ישירות ולא ישירות, בנוסף יוצגו בה הקאורדינטות של מיקום כל שדה תעופה וכן המחיר הממוצע לכל קו.

מורכב מאוד להסיק תובנות על סמך ההתבוננות בטבלה, בויזואליזציה שתוצג בהמשך יהיה קל יותר לראות את ההשפעות של מספר הטיסות על המחיר. באופן כללי ניתן לראות כי בד״כ, ככל שיש פחות היצע – המחיר עולה.

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, היצע וביקוש, וסוגי טיסות (ישירה או עם עצירות).

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

הכנסת הדאטה לSQLITE:

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.

2. 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.

3. סינון (Filtering):

- **מחיקת נתונים לא רלוונטיים:** לדוגמה, ב - Query 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.

4. 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.

5. 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 SQLITE:

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 11 עמודות 450 שורות. כל שורה תציג את מספר הטיסות לכל קו בטיסה ישירה ולא ישירה וכן את המחיר הממוצע, המקסימלי והמינימלי לאותו קו. מטרתה לבדוק את הקשר בין כמות הטיסות למחיר הטיסה (היצע וביקוש).
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.

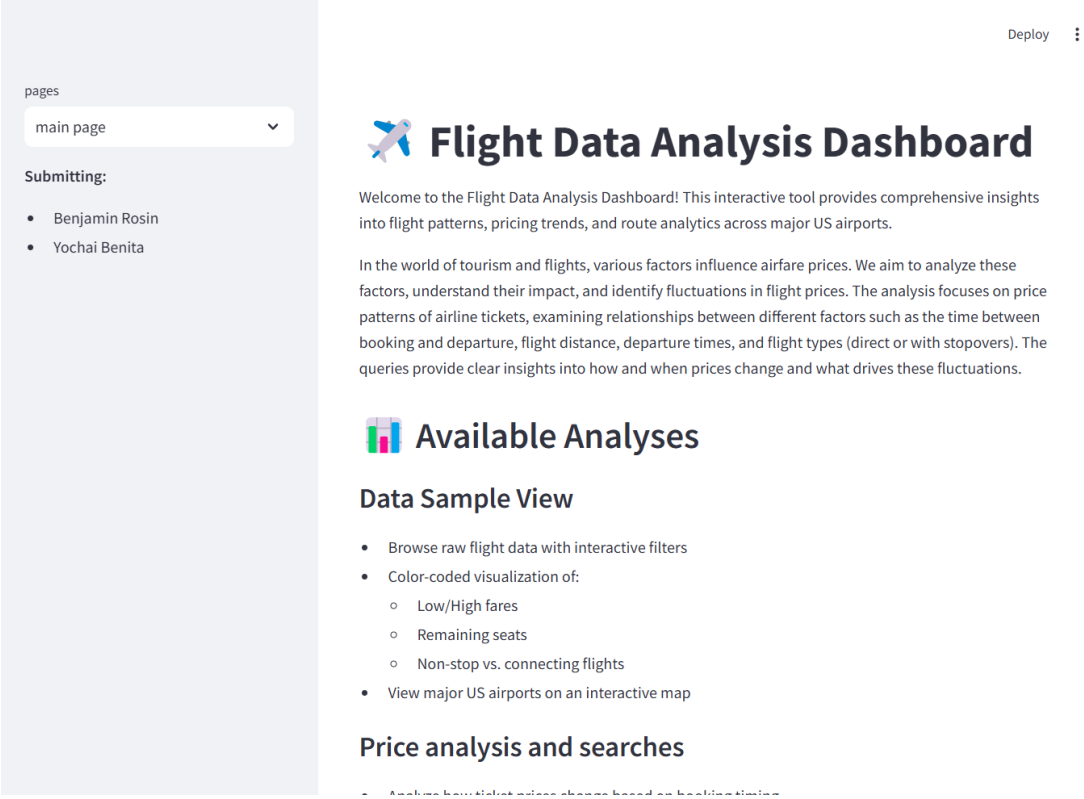
מגישים: בנימין רוסין, יוחאי בניטה

תיאור סכמתי של הדשבורד והסברים על הויזואליזציות:

הדשבורד מורכב מ-7 עמודים עם סרגל ניווט בצד:

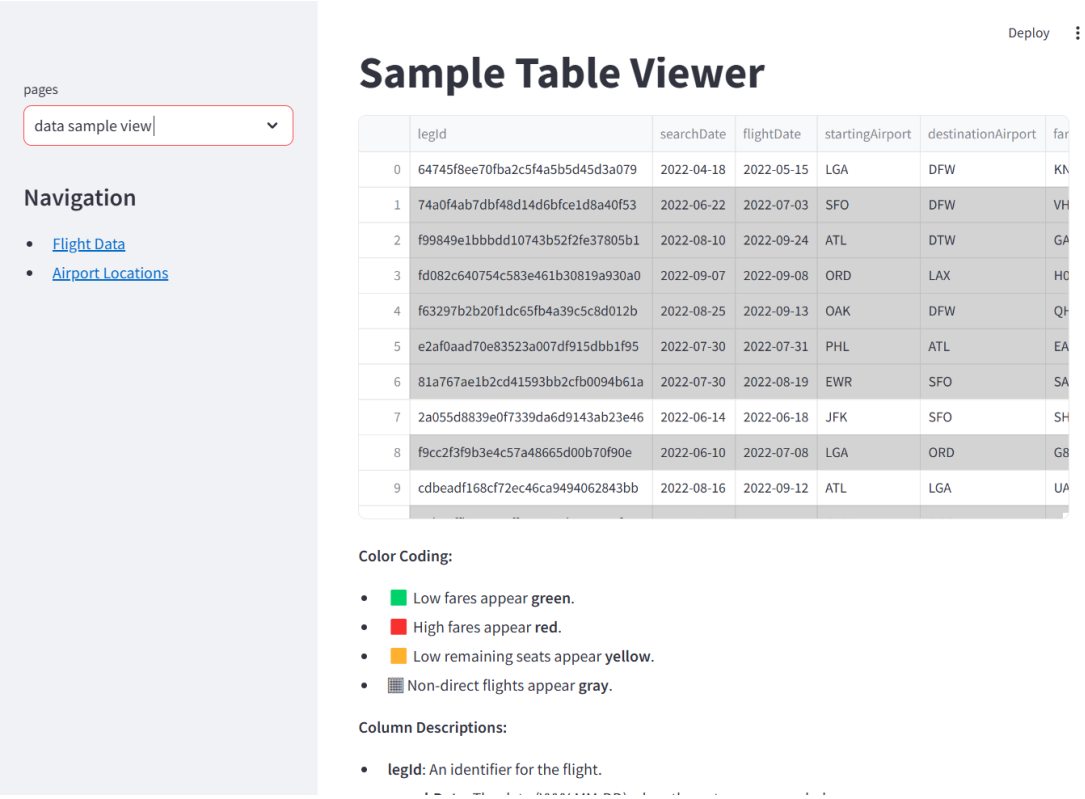
1. Main Page

- דף נחיתה עם תיאור הפרויקט והעבודה שביצענו, הסיפור שנרצה לספר והסבר על הדאשבורד.

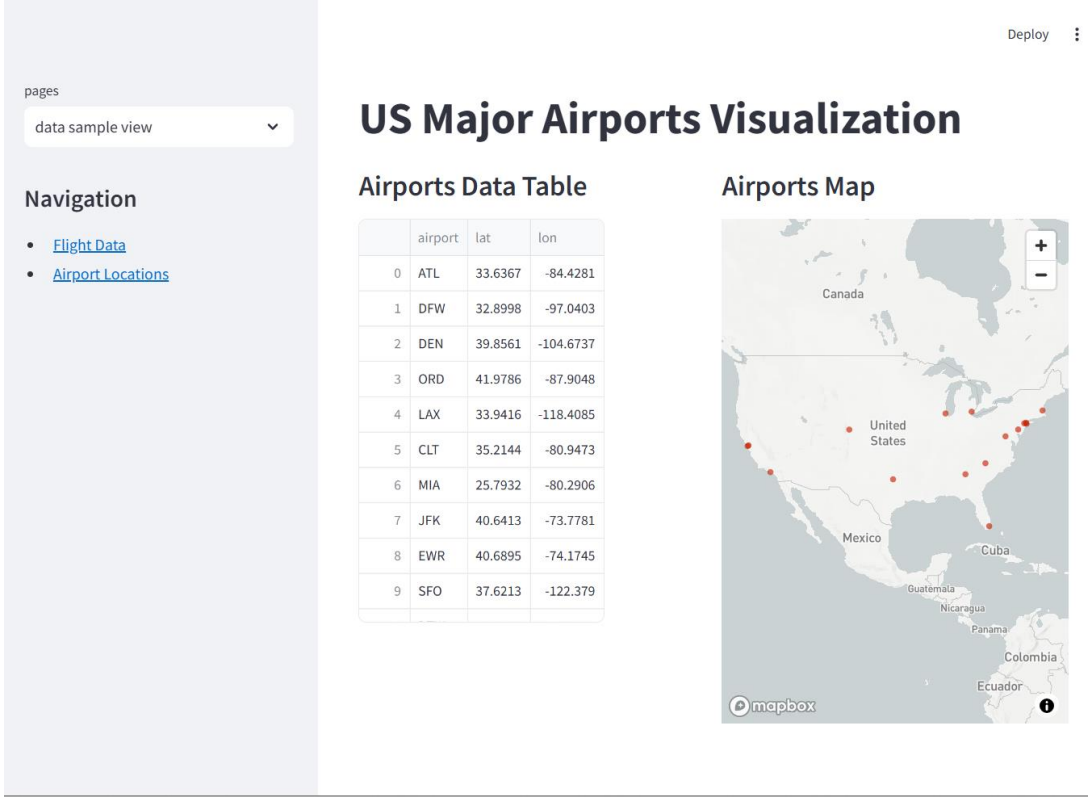


2. Data Sample

- טבלת נתוני הטיסות, בת 500 שורות כאשר היא מייצגת מדגם של הדאטה סט.
- מפת שדות התעופה בארה"ב.



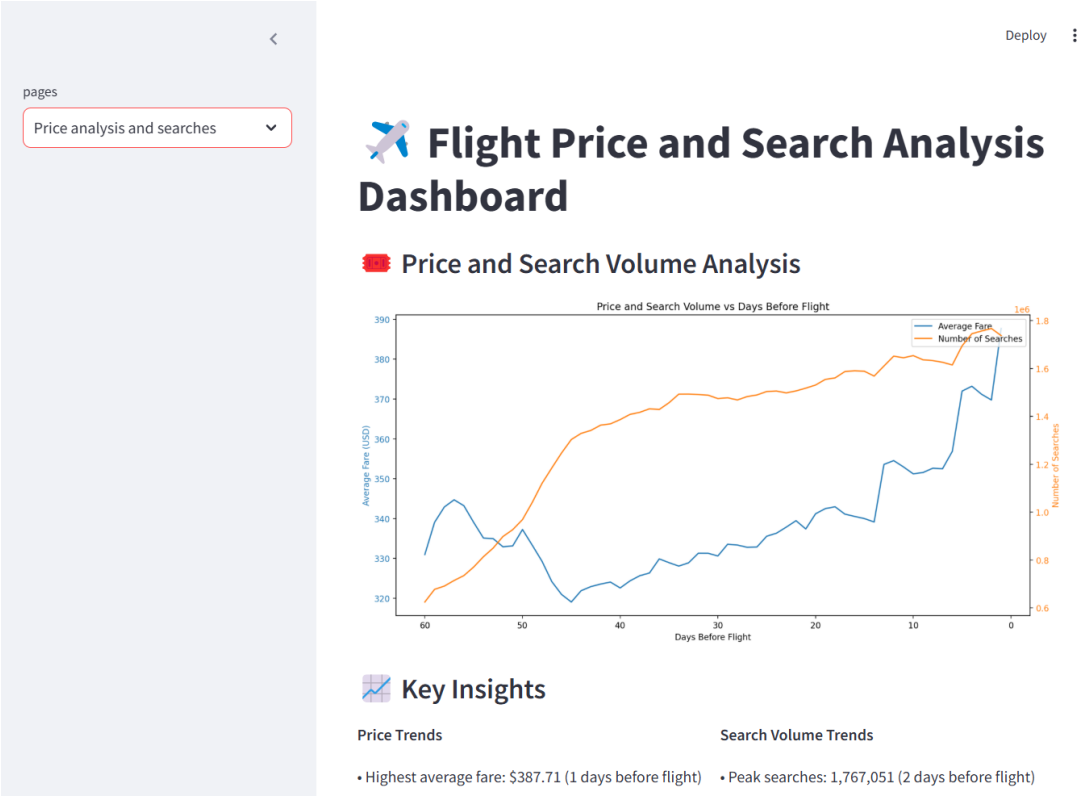
	legId	searchDate	flightDate	startingAirport	destinationAirport	far
0	64745f8ee70fba2c5f4a5b5d45d3a079	2022-04-18	2022-05-15	LGA	DFW	KN
1	74a0f4ab7dbf48d14d6bfce1d8a40f53	2022-06-22	2022-07-03	SFO	DFW	VH
2	f99849e1bbbdd10743b52f2fe37805b1	2022-08-10	2022-09-24	ATL	DTW	GA
3	fd082c640754c583e461b30819a930a0	2022-09-07	2022-09-08	ORD	LAX	HO
4	f63297b2b20f1dc65fb4a39c5c8d012b	2022-08-25	2022-09-13	OAK	DFW	QH
5	e2af0aad70e83523a007df915dbb1f95	2022-07-30	2022-07-31	PHL	ATL	EA
6	81a767ae1b2cd41593bb2cfb0094b61a	2022-07-30	2022-08-19	EWR	SFO	SA
7	2a055d8839e0f7339da6d9143ab23e46	2022-06-14	2022-06-18	JFK	SFO	SH
8	f9cc2f3f9b3e4c57a48665d00b70f90e	2022-06-10	2022-07-08	LGA	ORD	G8
9	cdbeadf168cf72ec46ca9494062843bb	2022-08-16	2022-09-12	ATL	LGA	UA



	airport	lat	lon
0	ATL	33.6367	-84.4281
1	DFW	32.8998	-97.0403
2	DEN	39.8561	-104.6737
3	ORD	41.9786	-87.9048
4	LAX	33.9416	-118.4085
5	CLT	35.2144	-80.9473
6	MIA	25.7932	-80.2906
7	JFK	40.6413	-73.7781
8	EWR	40.6895	-74.1745
9	SFO	37.6213	-122.379

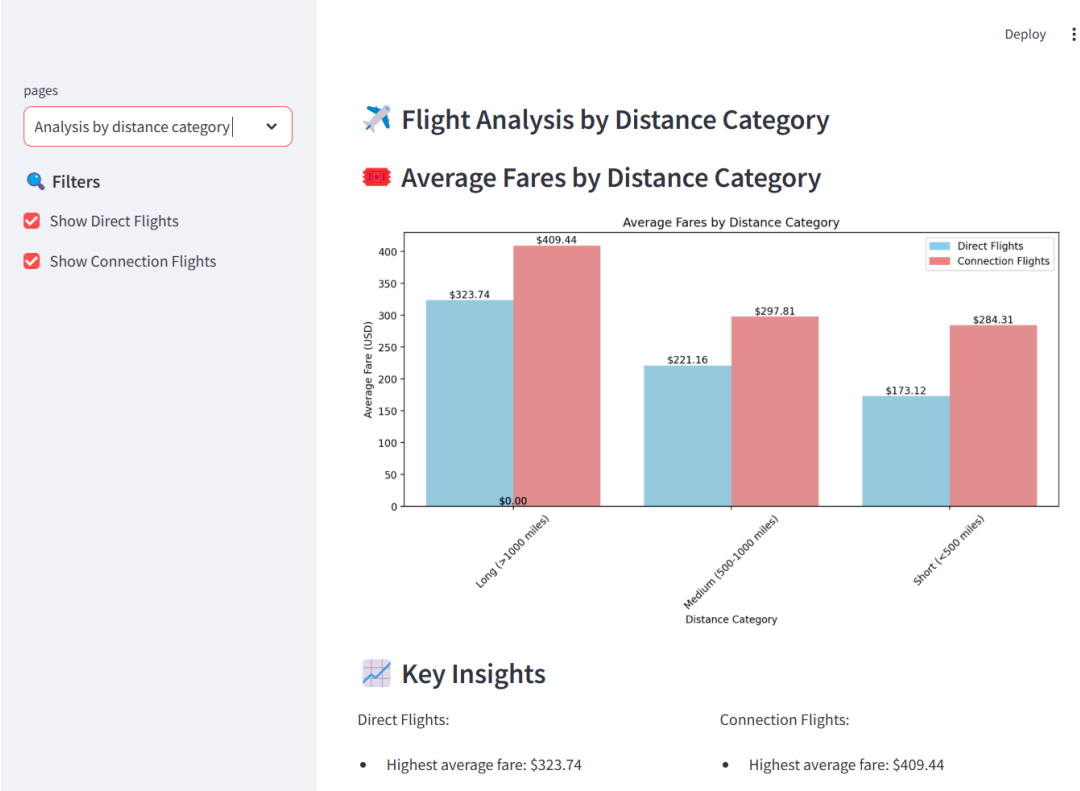
3. Price Analysis and Searches

- גרף המציג את מחיר הכרטיס ומספר החיפושים ביחס לזמן שלפני הטיסה.



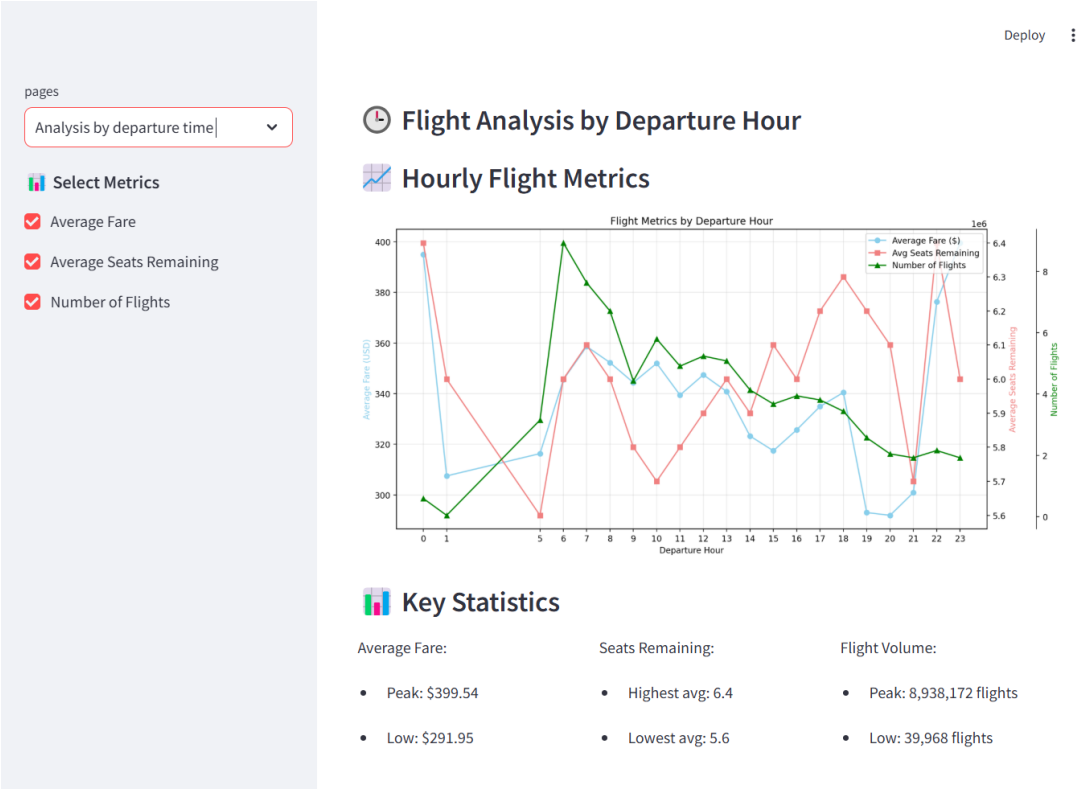
4. Analysis by Distance Category

- גרף אינטראקטיבי שמציג השוואת מחירים ממוצעים לפי קטגוריות מרחק.
- קיימת אפשרות לסנן בין טיסות ישירות וטיסות קונקשיין.
- מוצגת סטטיסטיקה לכל סוג טיסה.



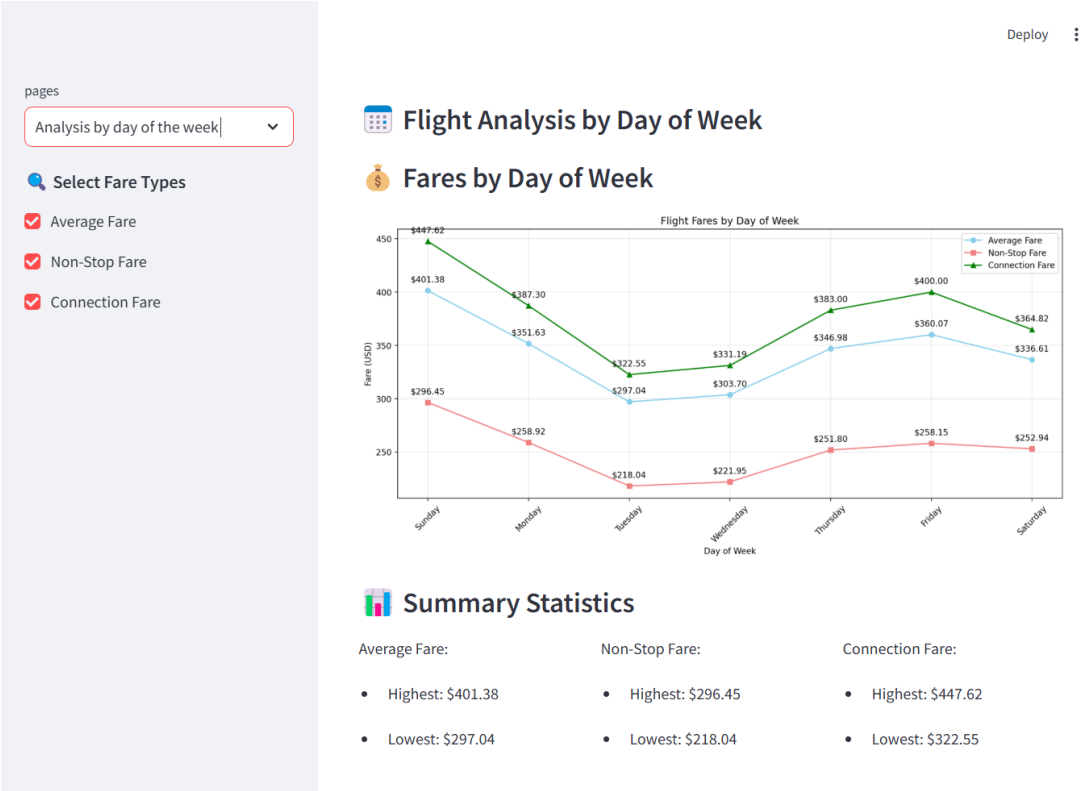
5. Analysis by Departure Time

- גרף אינטראקטיבי המציג נתונים שונים ביחס לשעת ההמראה, כגון מחיר ממוצע, מושבים פנויים בממוצע ומספר טיסות.
- מוצגות סטטיסטיקות לכל מדד המוצג בגרף.



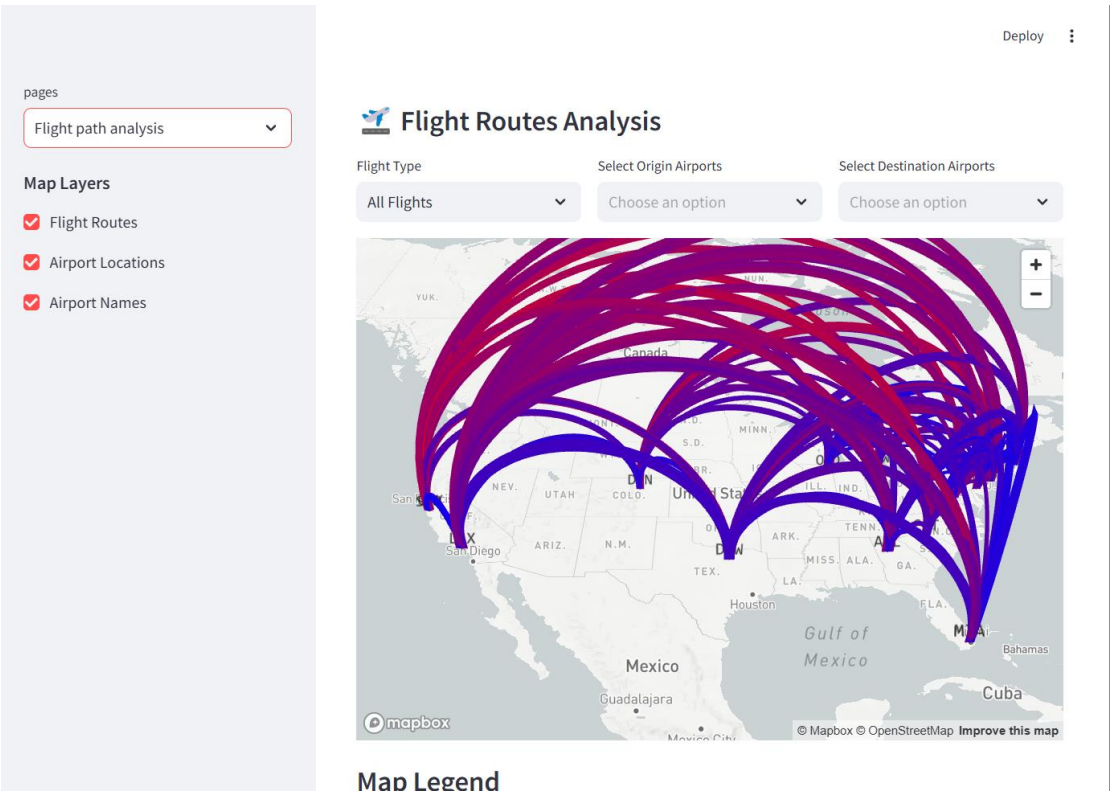
6. Analysis by Day of the Week

- גרף אינטראקטיבי המציג השוואת מחירים לפי ימי השבוע.
- מאפשר לסנן ולהציג מחיר ממוצע כללי, מחירים עבור טיסות ישירות וכן מחירים עבור טיסות לא ישירות.
- מוצגות סטטיסטיקות לכל מדד המוצג בגרף.



7. Flight Path Analysis

- מפה אינטראקטיבית של נתיבי טיסה, כאשר צבעי הקווים מייצגים מחירים (כחול – זול, אדום – יקר) ככל שהצבע יותר קרוב לאדום – המחיר יותר יקר.
- ניתן לסנן את התוכן המוצג לפי סוג הטיסה, מוצא ויעד.
- ניתן לבחור מה יוצג במפה: מיקומי נמלי התעופה, שמות הנמלים, וכן נתיבי הטיסה.
- מוצגות סטטיסטיקות לכל נתיב המוצג בגרף.



הוראות הרצה לדאשבורד:

1. יש להתקין את הספריות הנדרשות להרצת הפרויקט ע״י הפקודה הבאה:
   pip install -r requirements.txt
2. יש להריץ בטרמינל:
   streamlit run ./dashboard.py

GRAPH DB:

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:

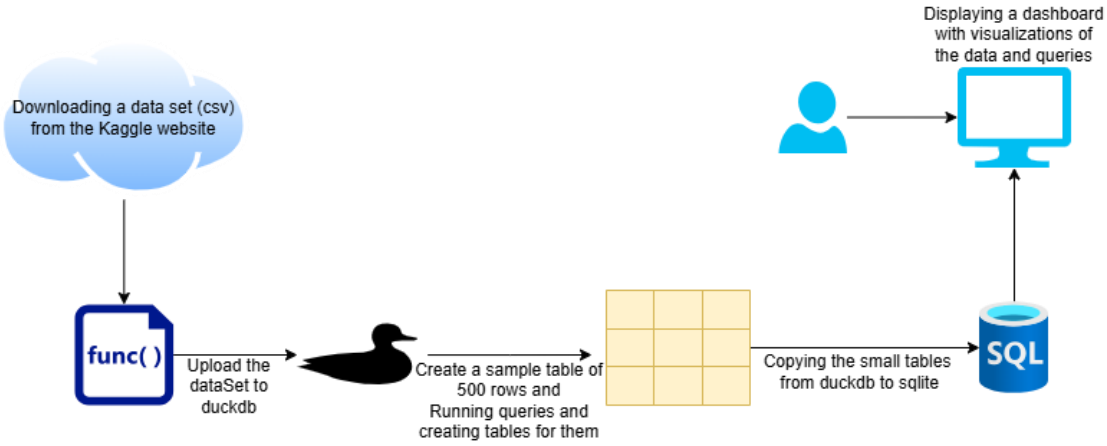
1. תאריך טיסה (Flight Date) - מתי הטיסה אמורה להתקיים, מקושר דרך FLIES_ON.
2. מחיר כולל (Total Fare) - עלות הטיסה, מקושר דרך HAS_PRICE.
3. סוג טיסה (Flight Type) - האם זו טיסה ישירה או עם עצירות, מקושר דרך IS_TYPE.
4. מרחק טיסה (Travel Distance) - כמה מיילים הטיסה מכסה, מקושר דרך TRAVELS.
5. שדה תעופה (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.

מבנה של המערכת:

את קבצי הCSV עם הדאטה נעלה בעזרת יכולות של הDUCKBD לקובץ בפורמט הנ"ל, על גבי הטבלאות הנ"ל נריץ שאילתות שיוצרות מס' טבלאות קטנות, חלק מהטבלאות מהוות 'טעימה' מהדאטה הגדול וחלקן מהוות תשובה לשאילתה ספציפית. לאחר מכן, בעזרת כלים של DUCKDB ניצור קובץ SQLITE ונעתיק אליו את כל הטבלאות הקטנות. בכך סיימנו את השלב הראשון. בשלב שני, בהרצת הקוד, על בסיס המידע שנמצא בקובץ הSQLITE נציג ויזואליזציות מרשימות שמנגישות את המסקנות מהמידע לצרכן.



העבודה על הפרויקט הנ"ל התחלקה באופן שווה בין שנינו לאורך הזמן, כל האלמנטים בפרויקט נכתבו בעבודה משותפת.