**פרויקט מסכם בקורס Big Data בר אילן**

**שמות מבצעי הפרויקט:**

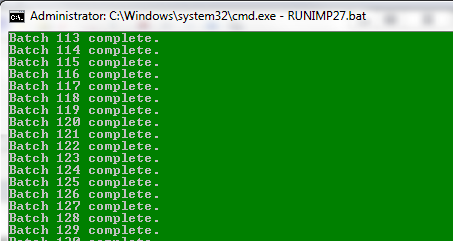
**שם סטודנט א. איתי שרעבי ת.ז. 209246610**

**שם סטודנט ב. אבי רווח ת.ז. 209020361**

**שם סטודנט ג. עינת שאלתיאל ת.ז. 313427361**

**הסטודנט המופיע כאן כסטודנט א' צריך להעלות את קובץ פתרון הפרויקט למודל (רק הוא)**

בפרויקט זה נחקור נתוני טיסות משנים שונות, באמצעות Neo4J ו MongoDB.

1. ניתן לבצע את הפרויקט בקבוצות של עד 3 סטודנטים לקבוצה. את התשובות יש לכתוב במסמך זה בצבע אדום. נסו גם להסביר/לנמק מדוע בחרתם לכתוב שאילתה בצורה שבחרתם – על מנת שגם אם ביצעתם טעות, עדיין יהיה אפשר לתת ניקוד חלקי.
2. נתחיל ב **Neo4J**:
3. התקנת תוכנות וספריות נדרשות:
   1. בספריה שקיבלתם יש ספריה פנימית בשם python27 המכילה קובץ התקנה – יש לבצע את ההתקנה. בסיומה, אמורים לקבל התקנה חדשה של python 2.7 בספריה c:\python27 *(לא לשנות מיקום כי אני מסתמך עליו בקבצים אחרים בהמשך)*
   2. בספריה שקיבלתם, יש לפתוח command line (cmd) ולהריץ את הקובץ runinst27
      1. *קובץ זה יתקין עבורכם מספר ספריות נדרשות עבור הפרויקט*
4. פתיחת בסיס נתונים ב Neo4J:
   1. להריץ Neo4J desktop
   2. לפתוח בו בסיס נתונים חדש (לוחצים על Add Graph)
   3. לבחור Create a Local Graph
   4. לתת לו שם – Graph, וסיסמא 12345 (חובה את אותה סיסמא כי הסקריפט שמעלה את הנתונים מסתמך על זה)
   5. ללחוץ על כפתורStart ואז על Manage ואז להריץ את בסיסי הנתונים ולפתוח ממשק באמצעות הכפתור "Open Browser"
5. העלאת הנתונים ל Neo4J:
   1. בספריה שקיבלתם, יש לפתוח command line (cmd) ולהריץ את הקובץ runimp27
      1. *קובץ זה יעלה עבורכם את נתוני הפרויקט, זה ייקח קצת זמן ותוכלו לראות שהוא סופר בזמן ההעלאה כמו בתצלום המסך הבא:*
      2. *ייתכן שכשתנסו להריץ את הקובץ תקבלו שגיאה:*

*httplib.BadStatusLine: No status line received –*

*the server has closed the connection*

*במקרה זה יש פשוט להריץ את Neo4Jמחדש ושוב להריץ את runimp27*

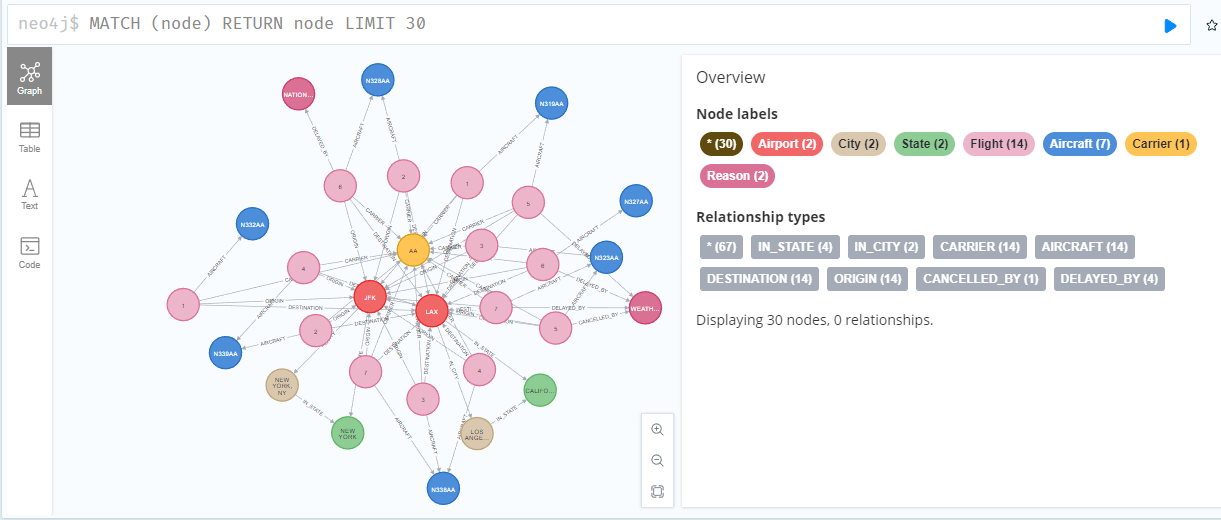
1. על מנת להכיר את הנתונים טוב יותר, בצעו את השאילתה:

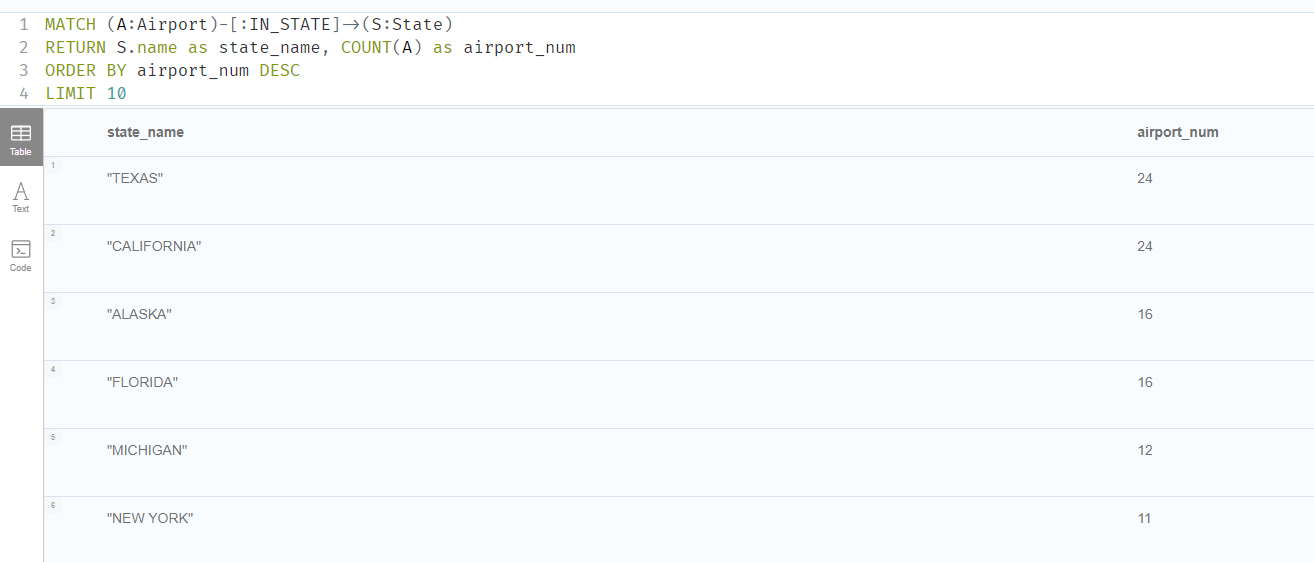
MATCH (node)

RETURN node

LIMIT 30

אשר תחזיר לכם נתונים של טיסות המכילים: שדות תעופה שונים, ערים, ארצות, מספרי טיסות, סוג/סימול המטוס, חברת התעופה וסיבות לאיחור בטיסה.



1. שאילתות לביצוע:
   1. הציגו טבלה של 10 הארצות/מדינות אשר בהן הכי הרבה שדות תעופה. יש להציג את שם הארץ ואת מספר שדות התעופה, ממוינים מהגדול לקטן לפי מספר שדות התעופה.

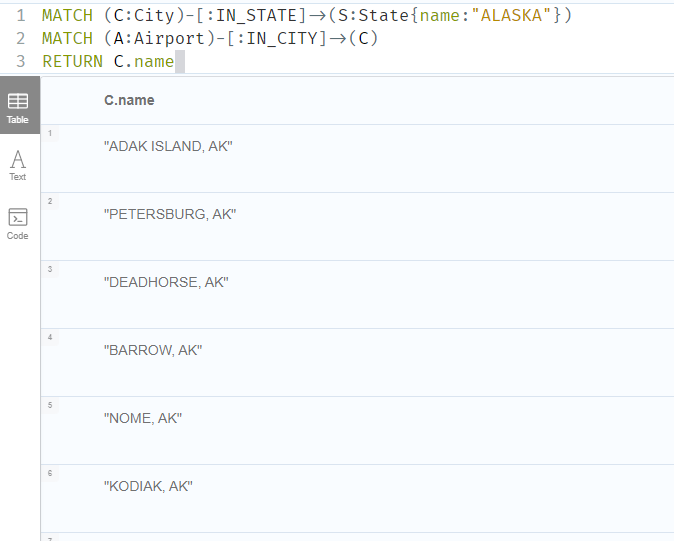
MATCH (A:Airport)-[:IN\_STATE]->(S:State)

RETURN S.name as state\_name, COUNT(A) as airport\_num

ORDER BY airport\_num DESC

LIMIT 10

* 1. הציגו את הערים במדינת Alaska אשר יש בהן שדות תעופה

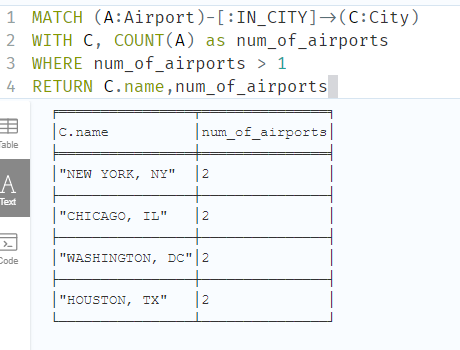


MATCH (C:City)- [:IN\_STATE]🡪(S:STATE{name:"ALASKA"})

MATCH (A:Airport)-[:IN\_CITY)🡪(C)

RETURN C.name

* 1. הציגו ערים אשר בהן יש יותר משדה תעופה אחד



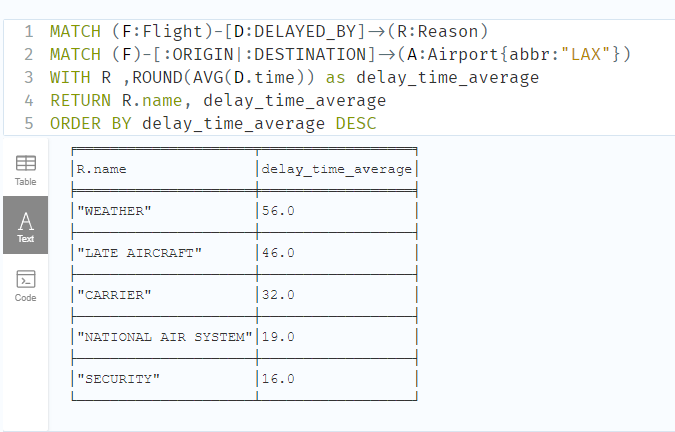
MATCH (A:Airport)-[:IN\_CITY]🡪(C:City)

WITH C, COUNT(A) as num\_of\_airports

WHERE num\_of\_airports > 1

RETURN C.name, num\_of\_airports

* 1. הציגו טבלה המפרטת את הסיבות המובילות לעיכובים בטיסות בשדה התעופה LAX, ממוינת לפי ממוצע זמן העיכוב (מהעיכוב הכבד ביותר לקל) עבור כל סיבה.



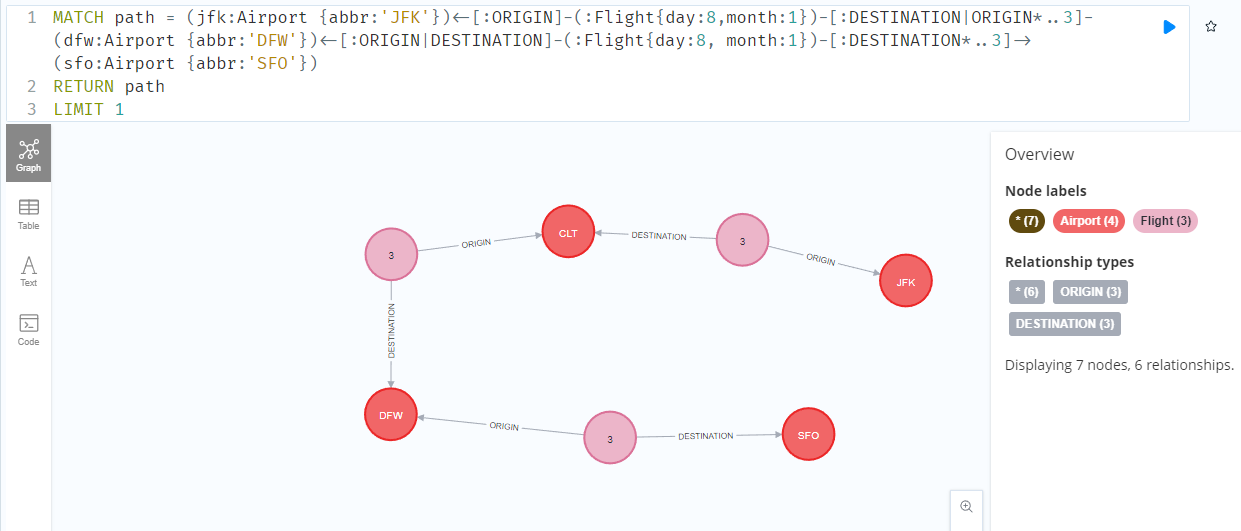
MATCH (F:Flight)-[D:DELAYED\_BY]->(R:Reason)

MATCH (F)-[:ORIGIN|:DESTINATION]->(A:Airport{abbr:"LAX"})

WITH R ,ROUND(AVG(D.time)) as delay\_time\_average

RETURN R.name, delay\_time\_average

ORDER BY delay\_time\_average DESC

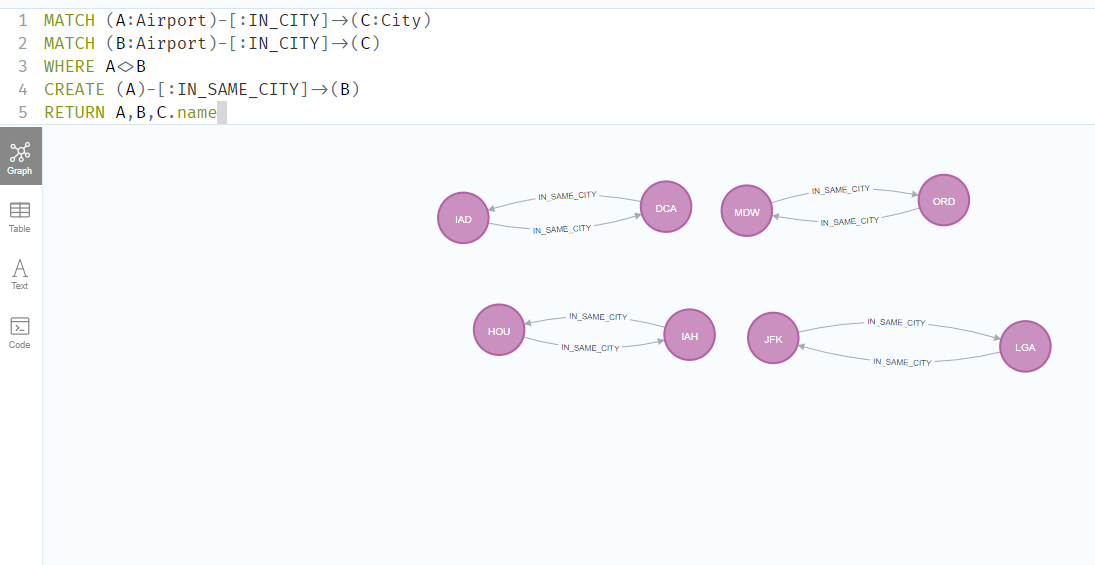
* 1. הצג מסלול (ויזואלית) בין שדה התעופה JFK לשדה התעופה SFO, אשר לפחות עובר בשדה התעופה DFW בתאריך ה 8 בינואר (עשוי בדרך לעבור גם בשדות נוספים).

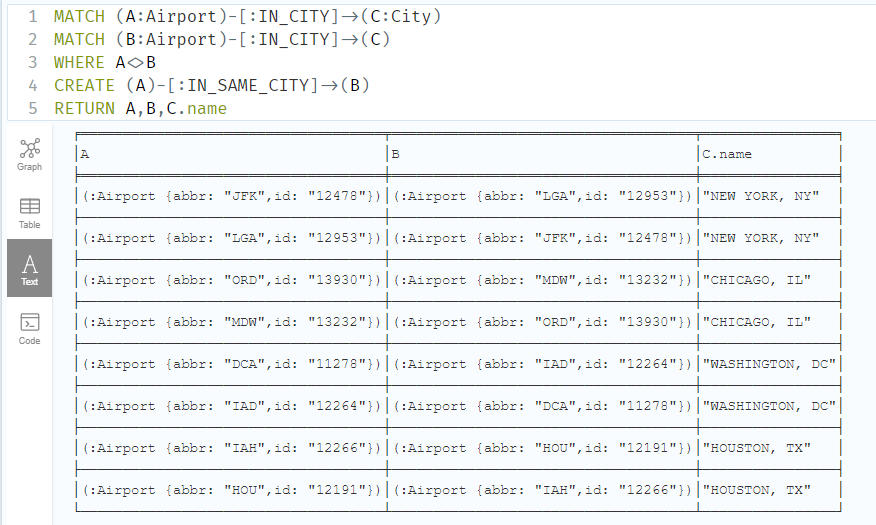
MATCH path = (jfk:Airport {abbr:'JFK'})<-[:ORIGIN]-(:Flight{day:8,month:1})-[:DESTINATION|ORIGIN\*..3]-(dfw:Airport {abbr:'DFW'})<-[:ORIGIN|DESTINATION]-(:Flight{day:8, month:1})-[:DESTINATION\*..3]->(sfo:Airport {abbr:'SFO'})

RETURN path

LIMIT 1

* 1. יצרו קשתות חדשות בין שדות תעופה השייכים לאותה עיר עם שם הקשר: IN\_SAME\_CITY





MATCH (A:Airport)-[:IN\_CITY]->(C:City)

MATCH (B:Airport)-[:IN\_CITY]->(C)

WHERE A<>B

CREATE (A)-[:IN\_SAME\_CITY]->(B)

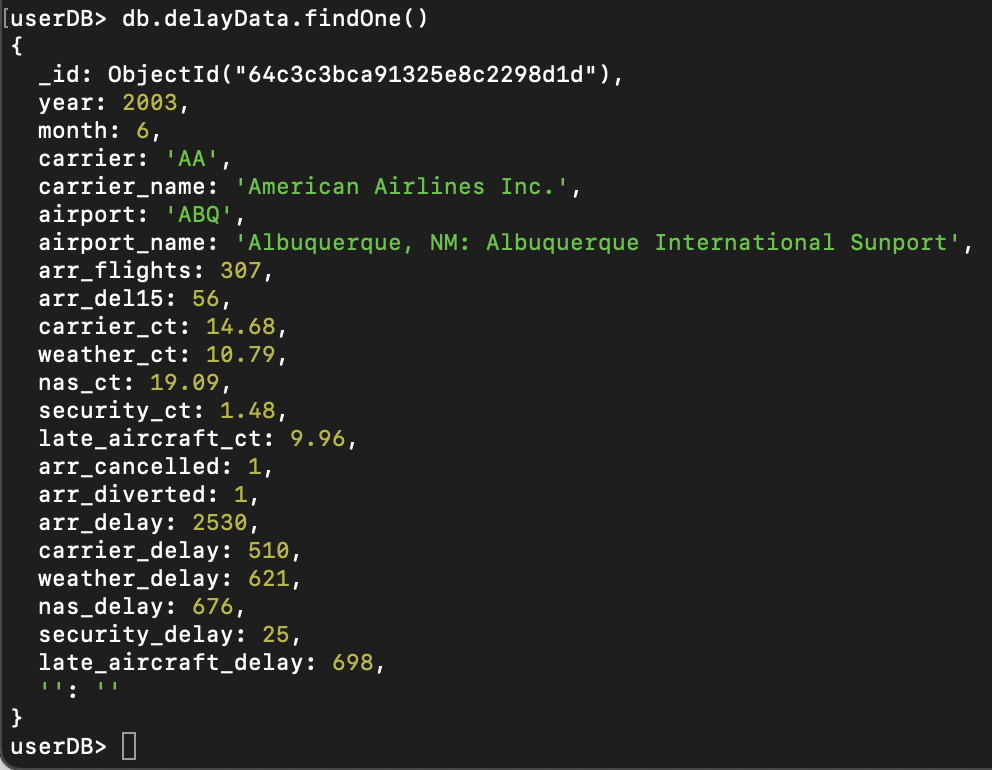
RETURN A,B,C.name

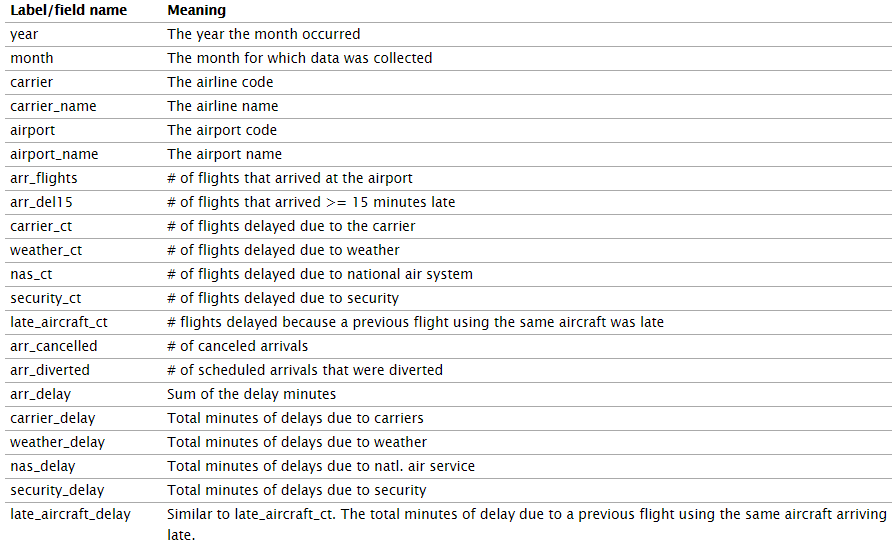
1. כעת נעבור ל **MongoDB**:
2. בספריה תוכלו למצוא קובץ בשם mongoAirlineDelay.csv המכיל מדגם של נתונים משנת 1987 ועד 2014
3. יש להעלות mongodb כפי שלמדנו באמצעות הפקודה mongod –dbpath=c:\tmp (ניתן לשנות את שם הספריה לספריה שיש לכם בה מקום)
4. כעת נייבא את הנתונים אל mongo באמצעות הפקודה:

mongoimport -d userDB -c delayData --type csv --file mongoAirlineDelay.csv --headerline

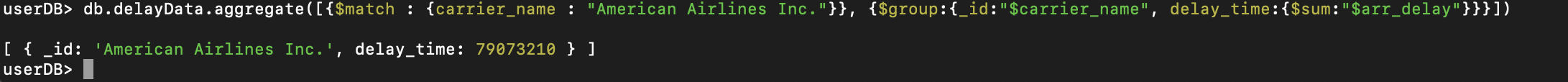
1. בזמן ייבוא הנתונים תוכלו לקבל חיווי כדוגמת: 
2. אחרי כן, יש כמובן לפתוח חלון cmd נוסף ולהעלות mongo.exe על מנת לכתוב פקודות.
3. על מנת לוודא שהנתונים הוכנסו כמו שצריך, עברו לחלון ה mongo ובחרו בבסיס הנתונים באמצעות הפקודה: use userDB . לאחר מכן כתבו db.delayData.findOne() אשר אמורה להציג לכם את המסמך הראשון ב collection שלנו.

Output:



1. לנוחיותכם, מובאת רשימת השדות:
2. שאילתות לביצוע:
   1. מצאו את מספר הדקות המצטבר אשר חברת American Airlines איחרה בה לאורך כל התקופה

db.delayData.aggregate([{$match : {carrier\_name : "American Airlines Inc."}}, {$group:{\_id:"$carrier\_name", delay\_time:{$sum:"$arr\_delay"}}}])



* 1. מצאו את מספר הדקות המצטבר של האיחור **עבור כל אחת** מחברות התעופה

db.delayData.aggregate([{$group:{\_id:"$carrier\_name", delay\_time:{$sum:"$arr\_delay"}}}])

A screen shot of a computer program

Description automatically generated

* 1. מצאו את שדה התעופה אשר בו היו הכי הרבה הסטות מסלול (arr\_diverted)

db.delayData.aggregate([{$group:{\_id:"$airport\_name", diverted\_time:{$sum:"$arr\_diverted"}}},{$sort:{diverted\_time:-1}},{$limit: 1}])

A screen shot of a computer

Description automatically generated

* 1. מצאו את שדה התעופה אשר בו היו הכי הרבה ביטולי טיסות בשנת 2012

db.delayData.aggregate([{$match:{"year":2012}},{$group:{\_id:"$airport\_name",cancellations:{$sum:"$arr\_cancelled"}}},{$sort:{cancellations: -1}},{$limit:1}])

A screen shot of a computer

Description automatically generated

* 1. עבור **כל אחת** מחברות התעופה, מצאו את קטגוריית העיכוב התורמת את העיכוב הכי קטן ומצא את שדה התעופה שבו יש לחברה זו הכי הרבה עיכובים בגיל סיבה/קטגוריה זו

var results = [];

db.delayData.aggregate([

{

$group: {

\_id: "$carrier\_name",

carrier\_delay\_sum: { $sum: "$carrier\_delay" },

weather\_delay\_sum: { $sum: "$weather\_delay" },

nas\_delay\_sum: { $sum: "$nas\_delay" },

security\_delay\_sum: { $sum: "$security\_delay" },

late\_aircraft\_delay\_sum: { $sum: "$late\_aircraft\_delay" }

}

},

{

$project: {

carrier\_name: "$\_id",

sums: [

{ sum: "$carrier\_delay\_sum", reason: "carrier\_delay" },

{ sum: "$weather\_delay\_sum", reason: "weather\_delay" },

{ sum: "$nas\_delay\_sum", reason: "nas\_delay" },

{ sum: "$security\_delay\_sum", reason: "security\_delay" },

{ sum: "$late\_aircraft\_delay\_sum", reason: "late\_aircraft\_delay" }

]

}

},

{

$addFields: {

minDelay: { $min: "$sums.sum" }

}

}

]).forEach(function (cur\_airline\_doc) {

var carrier\_name = cur\_airline\_doc.carrier\_name;

var minres = cur\_airline\_doc.sums.find(function (sum) {

return sum.sum === cur\_airline\_doc.minDelay;

}).reason;

var aggPipeline = [

{ $match: { "carrier\_name": carrier\_name } },

{

$group: {

\_id: "$airport\_name",

res\_sum: { $sum: "$" + minres }

}

},

{ $sort: { res\_sum: -1 } },

{ $limit: 1 }

];

var aggResult = db.delayData.aggregate(aggPipeline).toArray();

if (aggResult.length > 0) {

var topAirport = aggResult[0];

results.push({

carrier\_name: carrier\_name,

delay\_reason: minres,

top\_airport: topAirport.\_id,

res\_sum: topAirport.res\_sum

});

}

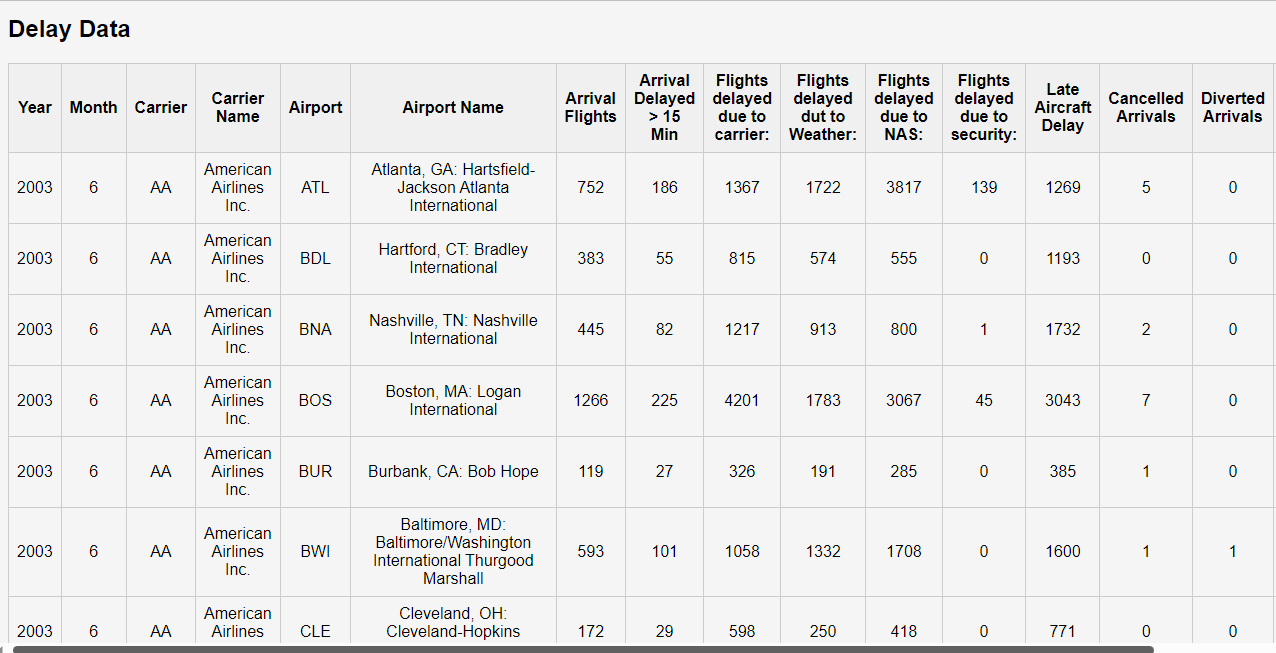
});

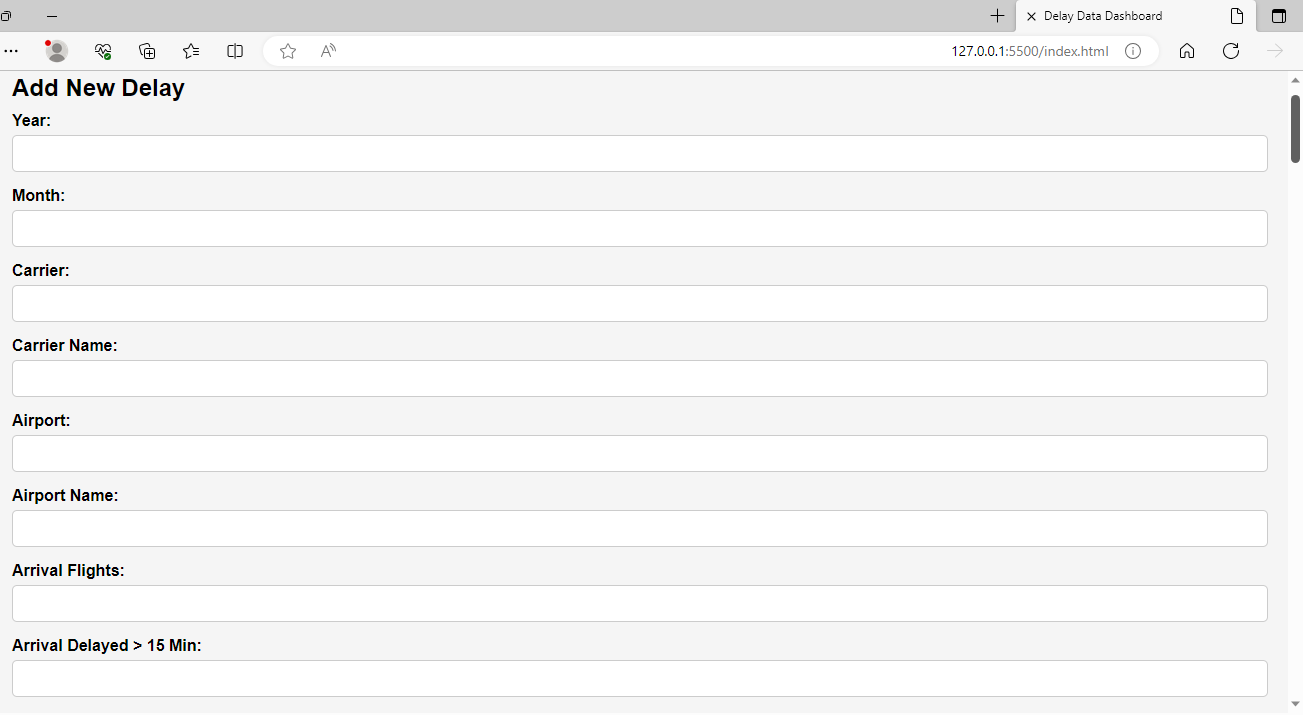
printjson(results);

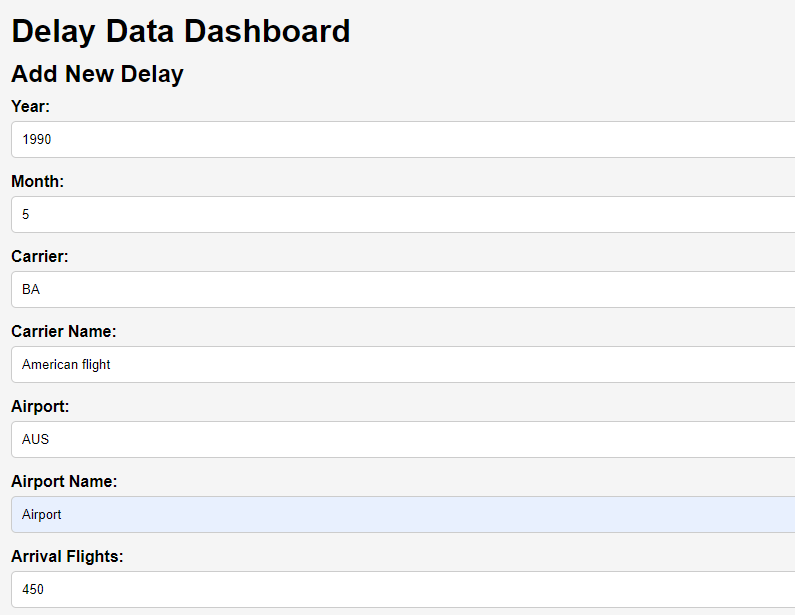
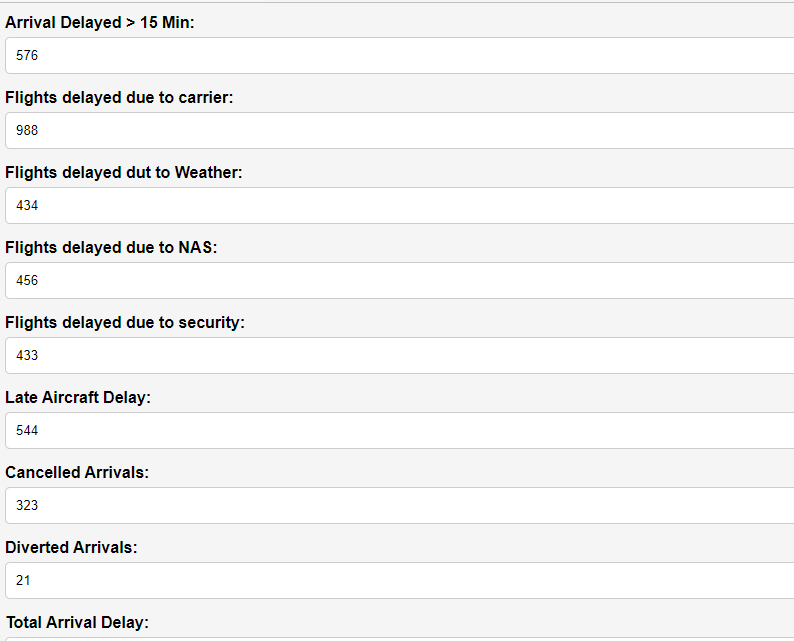
1. ממשו אתר ב Node.js (כפי שראינו בשיעור האחרון) המאפשר:

לינק לגיטהאב: <https://github.com/itaish421/Final_Project_Big_Data.git>

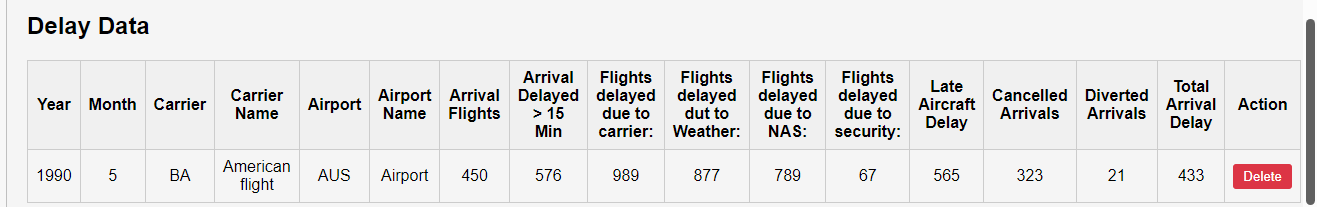
* 1. לצפות ברשימת כל הטיסות בטבלה ב HTML



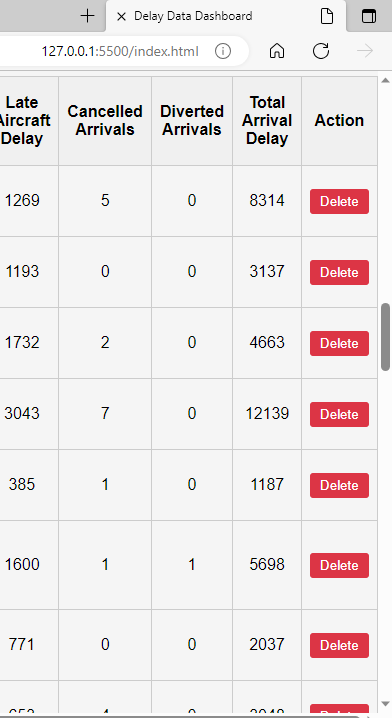
* 1. הממשק יאפשר למחוק טיסות קיימות מבסיס הנתונים ולהוסיף טיסות חדשות

**הוספת טיסה חדשה**

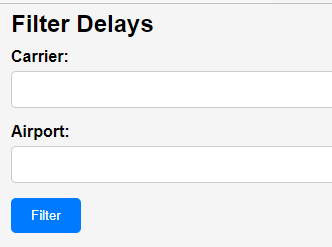
**כעת, נוכל לראות את פרטי הטיסה שהוספנו לאחר שלחצנו על כפתור add delay**



**מחיקת טיסות**

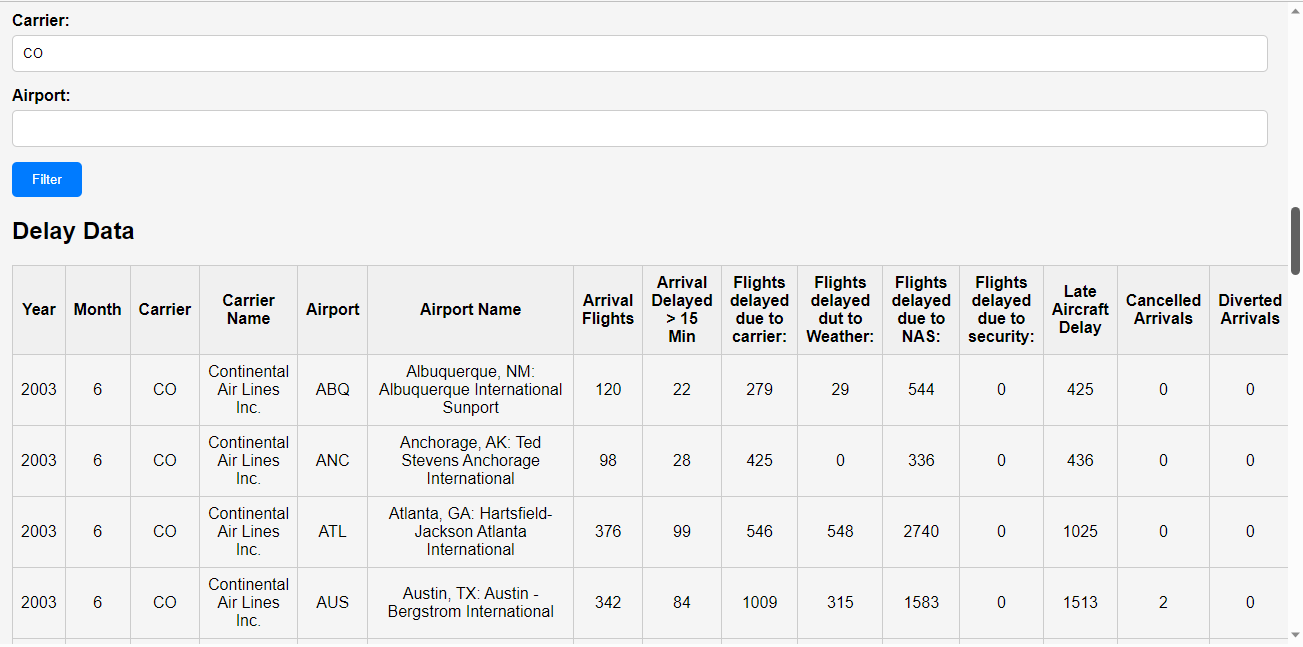


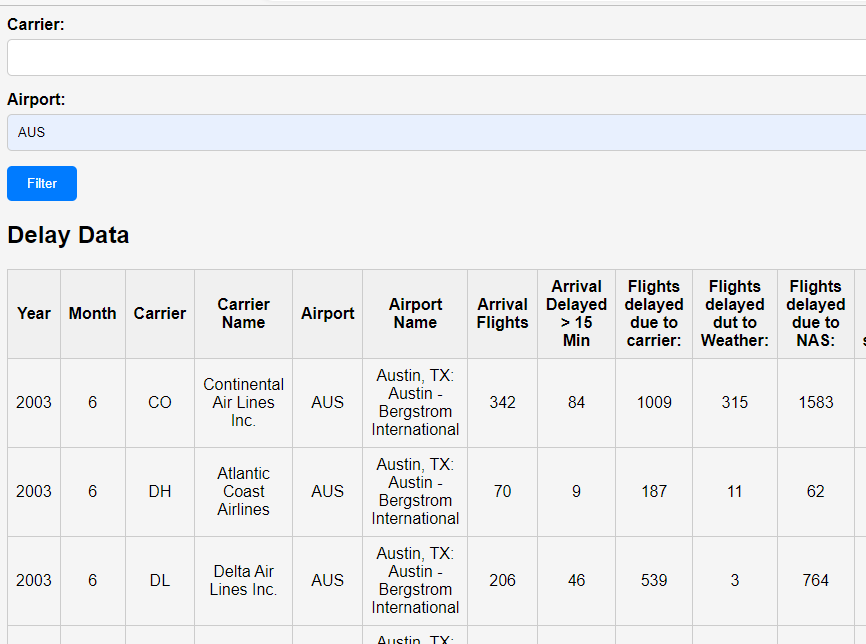
* 1. הממשק יאפשר לפלטר/לסנן טיסות שיש להן:
     1. טיסות של carrier מסוים
     2. טיסות שנחתו בשדה תעופה מסוים

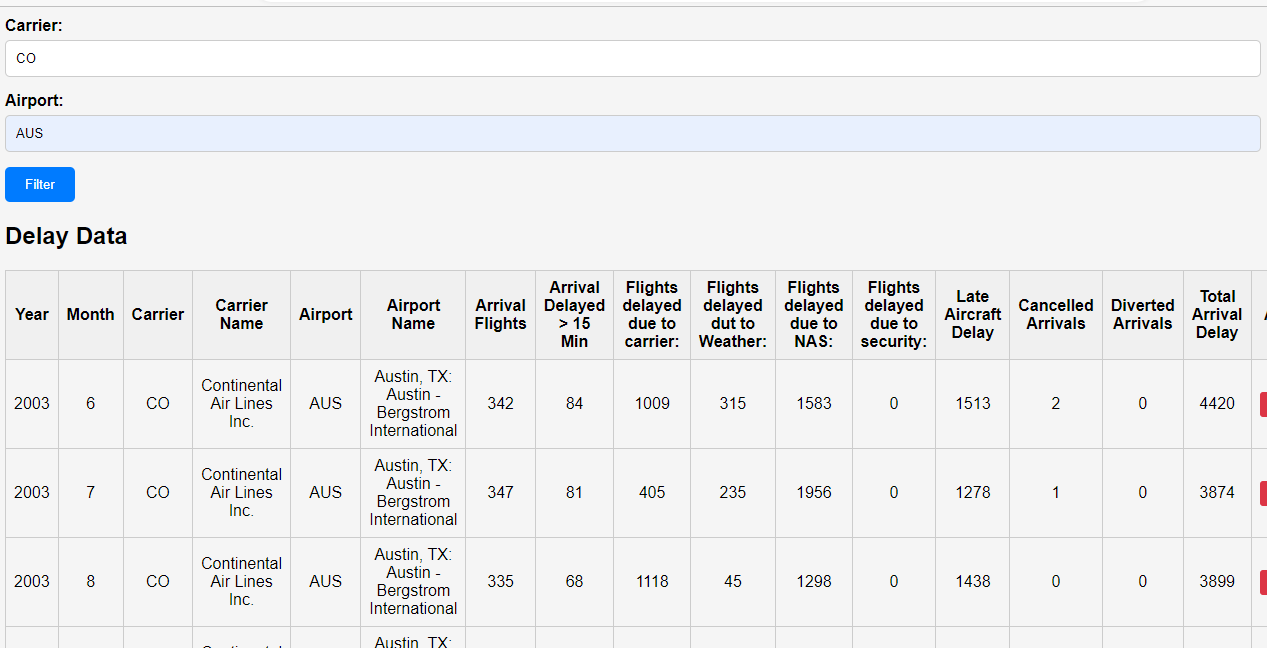


**תוצאות הטבלה אחרי סינון לפי carrier או שדה תעופה או שניהם**

סינון לפי carrier ושדה תעופה בנפרד





סינון לפי שדה תעופה וcarrier ביחד

**בהצלחה בפרויקט ובתקופת הבחינות!**