# SOFTWARE documentation

Liron Cohen 207481268, Ofer Tlusty 311396303

## **DB Scheme Structure**

להוסיף שרטוט של הסכמה עם העמודות, האינדקסים והמפתחות (הראשיים והזרים)

להוסיף חרטוט כלשהו על למה בחרנו לעצב אותו ככה

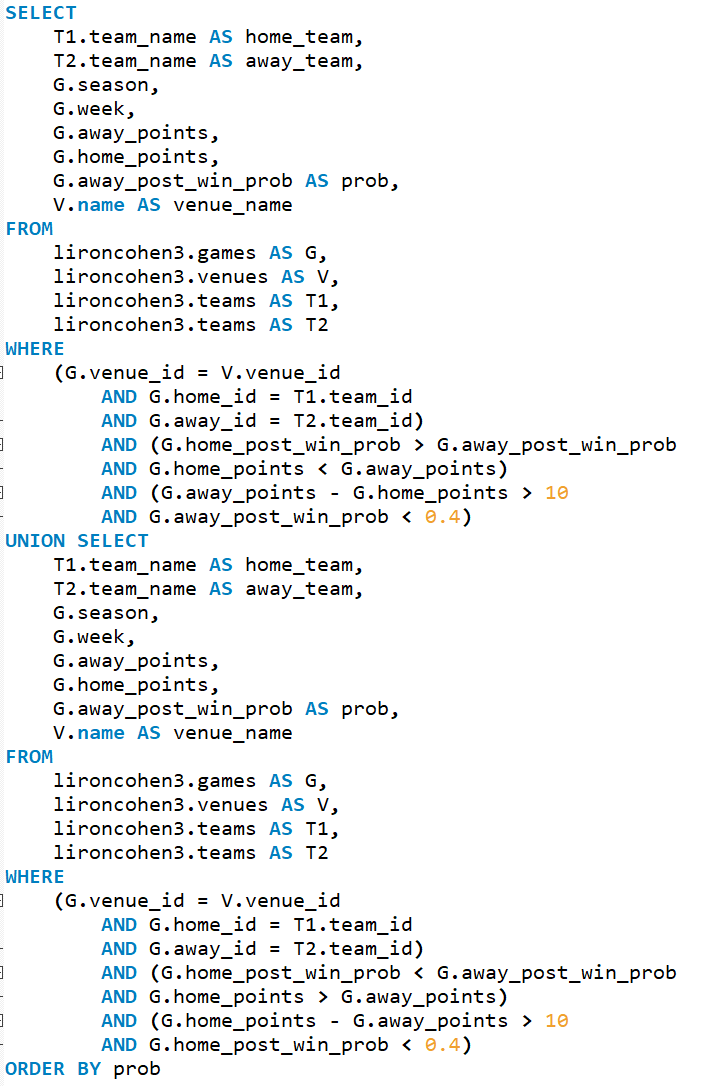
## **DB Optimizations**

להוסיף הסבר על האינדקסים שלנו ועוד אופטימיזציות (אולי במבנה ה-DB? אולי במפתחות?)

## **Queries**

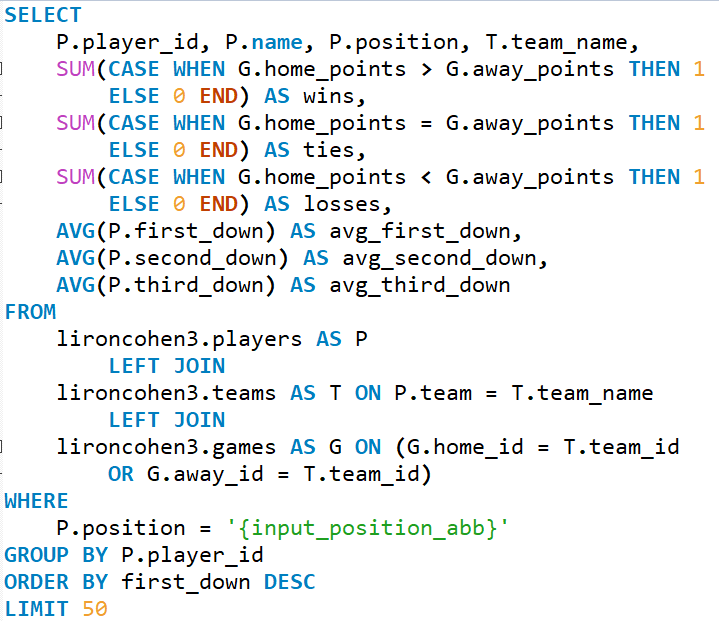
להלן פירוט על כל אחת מהשאילתות שכתבנו:

1. **get\_teams\_that\_won\_against\_odds**

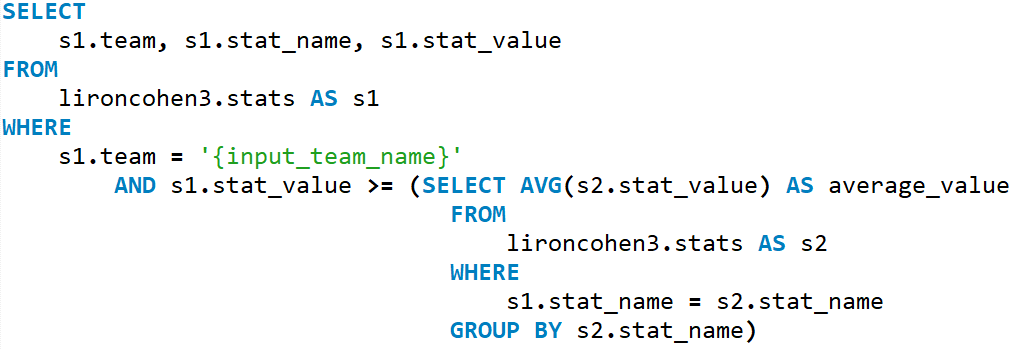


* קבוצות שניצלחו "כנגד כל הסיכויים": השאילתא מחזירה את שמות הקבוצות, תוצאת המשחק הסופית, את העונה והשבוע בו התקיים המשחק, את האידטדיום בו התרחש המשחר וכן את ההסתברות שניתנה עבור הקבוצה המנצחת כך שיתקיים התנאי הבא: נבחר להציג רק את הקבוצות אשר ההסתברות שלהם לנצח הייתה קטנה מ-40% ובכל זאת ניצחו ביותר מ-10 נקודות!
* האופטימיזציות שביצענו עוזרות להרצת שאילתא זו ע"י כך שיצרנו בטבלה games אינדקסים על מזהי הקבוצות המשחקות (home\_id ו-away\_id) אשר מייעלים את ביצוע ה-join בין טבלה זו לטבלת ה-teams בצורה משמעותית.
* עיצוב ה-DB שלנו תומך בשאילתא זו בכך שהחיבורים בין הטבלאות מבוצעים בין מפתחות בלבד (בין אם ראשיים או זרים) וכולל גם שימוש במזהים שאונדקסו על מנת לשפר את יעילות השאילתא (כמוסבר מעלה).

1. **get\_top\_position\_players\_stats**

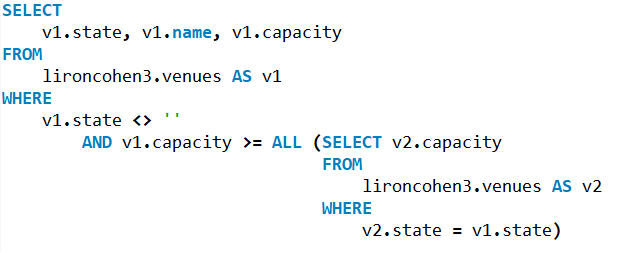


* שאילתא זו מקבלת שם של תפקיד בפוטבול האמריקאי (QB, CB, LT וכו') ומחזירה את מזהי השחקן עצמו (מספר שחקן, שמו ותפקידו), את שם הקבוצה וסטטיסטיקה אודותיה (מספר ניצחונות, תיקו והפסדים) וכן סטטיסטיקה אישית על השחקן (ממוצע להצלחת בהתקדמות בניסיון הראשון, השני והשלישי). את התוצאות נמיין בסדר יורד לפי ממוצע הניסיון הראשון ונחזיר את 50 התוצאות הראשונות.
* האופטימיזציות שביצענו עוזרות להרצת שאילתא זו ע"י כך שיצרנו אינדקס בטבלת teams על השדה team\_name וכן אינדקס נוסף בטבלת players על השדה team – נבחין כי בטבלת ה-players לא מופיע השדה team\_id למולו היינו רוצים לבצע את החיבור בין הטבלאות (כך במקור).   
  כמו כן, יצרנו שני אינדקסים בטבלת games על השדות home\_id ו-away\_id אשר מאפשרים לבצע את החיבור בין הטבלאות teams ו-games בצורה יעילה יותר.
* עיצוב ה-DB שלנו תומך בשאילתא זו בכך שהחיבורים בין הטבלאות מבוצעים בין מפתחות (בין אם ראשיים או זרים) או מזהים שאונדקסו, וכולל גם שימוש במזהים שאונדקסו על מנת לשפר את יעילות השאילתא (כמוסבר מעלה).

1. **get\_team\_stats\_better\_than\_average**

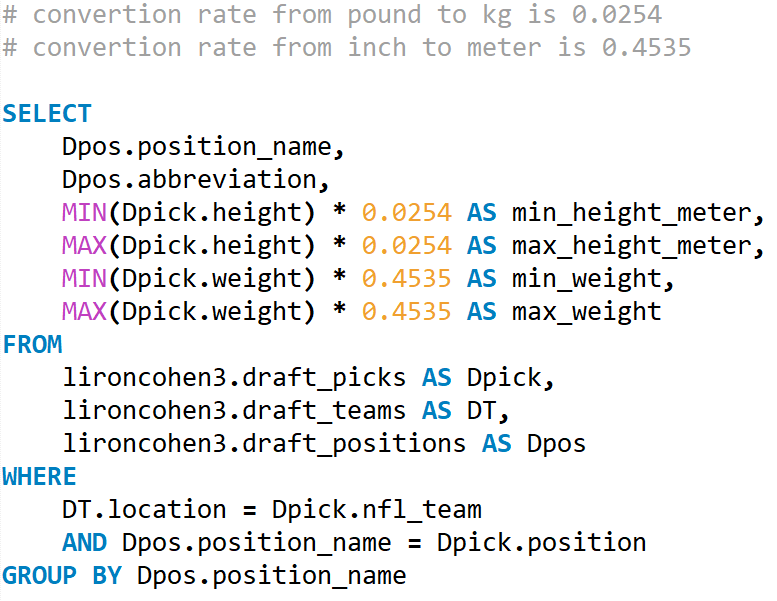
* שאילתא זו מקבלת שם של קבוצה (לדוגמה Miami) ומחזירה את שמות וערכי הסטטיסטיקות בהן הקבוצה מעל ממוצע הסטטיסטיקה. בשאילתא נעשה שימוש ב-subquery המחזיר את ממוצע הסטטיסטיקה על פני כל ערכי הקבוצות, והחלק הראשי בודק האם ערך הסטטיסטיקה של הקבוצה גדול מהממוצע.
* האופטימיזציות שביצענו עוזרות להרצת שאילתא זו ע"י כך שיצרנו בטבלה stats אינדקס על שם הסטטיסטיקה (דבר העוזר לחישוב הממוצע במהירות) וכן אינדקס על שם הקבוצה על מנת למצוא את הסטטיסטיקות של הקבוצה במהירות.
* עיצוב ה-DB שלנו תומך בשאילתא זו בכך שבטבלת הסטטיסטיקות מופיעות העמודות הנחוצות לשאילתא בטבלה אחת (ללא צורך ב-joins) - שם הקבוצה, שם הסטטיסטיקה, ערך הסטטיסטיקה.

1. **get\_max\_capacity\_venue\_per\_state**



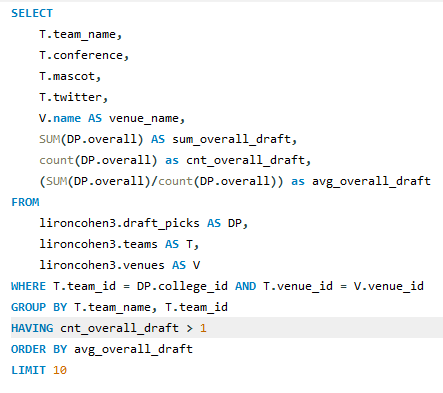
* שאילתא זו מחזירה את השם והגודל של ה-venue הגדול ביותר בכל state (מבחינת capacity). בשאילתא נעשה שימוש ב-subquery המחזיר את כל גדלי ה-venues ב-state וכן מתבצע וידוא שהערך המוחזר גדול או שווה לכל הערכים המתקבלים לאותו ה-state.
* האופטימיזציות שביצענו עוזרות להרצת שאילתא זו ע"י כך שיצרנו בטבלה venues אינדקס על   
  ה-state וכך ניתן להחזיר את הערכים המתאימים ל-state במהירות.
* עיצוב ה-DB שלנו תומך בשאילתא זו בכך שבטבלת ה-venues מופיעות העמודות הנחוצות לשאילתא בטבלה אחת (ללא צורך ב-joins) - שם ה-state, שם ה-venue, גודל ה-venue.

1. **get\_min\_max\_weights\_heights\_per\_draft\_position**



* שאילתא זו מחזירה עבור שחקנים המיועדים לבחירה (drafts) את הגבהים והמשקלים המינימליים והמקסימליים (קצוות הטווח) בחלוקה לפי תפקיד במשחק (ושמו המקוצר). נציין כי הקבועים בשאילתא הינם קבועי ההמרה בין יחידות המידה האמריקאיות לאלו הישראליות/אירופאיות.
* האופטימיזציות שביצענו עוזרות להרצת שאילתא זו ע"י כך שיצרנו בטבלה draft\_teams אינדקס על השדה location (המשמש כשם הקבוצה ולא כמיקום כפי שהשם אולי מרמז) וכן בטבלה draft\_picks על השדה nfl\_team (שם הקבוצה בליגה, ולא ברמת הקולג'ים) אשר מהווה גורם החיבור היחיד בין הטבלאות.
* עיצוב ה-DB שלנו תומך בשאילתא זו בכך שהחיבורים בין הטבלאות מבוצעים בין שדות אשר אונדקסו על מנת לשפר את יעילות השאילתא (כמוסבר מעלה).

1. **get\_most\_picked\_college\_teams\_in\_draft**



* שאילתא זו מחזירה עבור כל קבוצה פרטים אודותיה (שם הקבוצה, ה-conference שלה, שם הקמיע, חשבון הטוויטר שלה והאצטדיון שלה) וכן פרטים אודות מחזור הבחירה (draft) האחרון:
  + מספר השחקנים אשר בחרה – cnt\_overall\_draft
  + סכום מיקומי השחקנים אשר בחרה – sum\_overall\_draft
  + ממוצע מיקומי השחקנים אשר בחרה – avg\_overall\_draft

מהתוצאות שהתקבלו סיננו את הקבוצות שבחרו שחקן בודד בלבד שכן מדד הממוצע מטה את התוצאות עבור מקרים מסוג זה. לאחר מכן סידרנו את התוצאות לפי ממוצע המיקומים בסדר יורד – **כלומר קיבלנו מדד לאיזה קבוצות קיבלו שחקנים מבוקשים יותר**.

* האופטימיזציות שביצענו עוזרות להרצת שאילתא זו ע"י כך שיצרנו בטבלה draft\_picks אינדקס על השדה college\_id אשר לחבר בין טבלה זו לטבלה teams.
* עיצוב ה-DB שלנו תומך בשאילתא זו בכך שהחיבורים בין הטבלאות מבוצעים בין מפתחות (בין אם ראשיים או זרים) או מזהים שאונדקסו, וכולל גם שימוש במזהים שאונדקסו על מנת לשפר את יעילות השאילתא (כמוסבר מעלה).

## **Code Structure**

הקוד מחולק ל-4 חלקים:

1. **create db**

אחראי על בניית הסכמה של ה-DB - הטבלאות, העמודות, המפתחות והאינדקסים.

* DBCreator - שם המחלקה. הבנאי מתחבר ל-DB לפי פרטי ההתחברות.
* create\_all\_tables - מתודה שיוצרת את כל הטבלאות (ריקות) לפי הסכמה שצוינה לעיל.
* create\_all\_indexes - מתודה שיוצרת את כל האינדקסים הנדרשים לאופטימיזציות עבור   
  ה-queries.

1. **get data from api**

אחראי על הבאת הנתונים מתוך ה-API.

* מכיל מתודות סטטיות שנקראות מתוך החלק השלישי של הקוד (insert data into db).
* כל מתודה מכילה בקשת get ל-API עם ה-URL המתאים וה-headers הדרושים (כפי שמפורט בחלק ה-API), מקבלת את התשובה, מפרסרת אותה לפורמט JSON ומחזירה את הנתונים שהתקבלו.
* דוגמאות למתודות - get\_games, get\_stats, get\_teams וכו'.

1. **insert data into db**

אחראי על קבלת הנתונים מתוך החלק השני של הקוד (get data from api) והכנסתם ל-DB בהתאם לעמודות והתנאים הנדרשים ב-DB design.

* APIDataInserter - שם המחלקה. הבנאי מתחבר ל-DB לפי פרטי ההתחברות.
* fill\_all\_tables - מתודת אם שקוראת לכל פונקציות ה-insert המתאימות וממלאת את כל טבלאות ה-DB.
* כל מתודה נוספת קוראת למתודה המתאימה לה מהחלק השני של הקוד (get data from api) ומקבלת ממנה את הנתונים המלאים שהגיעו מהקריאה ל-API. היא עוברת על כל הערכים הראשיים ב-JSON שהוחזר (שקול לשורה ב-DB), בודקת שהערך המתאים   
  ל-PRIMARY KEY של הטבלה אינו NULL ומפרסרת את הערכים הדרושים מתוך ה-JSON למשתנים. אם אחד הערכים הדרושים לא קיים ב-JSON היא מכניסה למשתנה המתאים None. לאחר מכן, היא יוצרת INSERT INTO query שמכניסה לטבלה את הערכים של השורה לפי הסדר הדרוש ומריצה את ה-query. ערכי None מוכנסים לטבלה כ-NULL אוטומטית.
* דוגמאות למתודות - insert\_draft\_picks\_data, insert\_stats\_data וכו'.

1. **query db**

אחראי על הרצת השאילתות על ה-DB.

* DBQuery - שם המחלקה. הבנאי מתחבר ל-DB לפי פרטי ההתחברות.
* קיימת מתודה לכל שאילתא שצוינה לעיל ובה יוצרים את השאילתא, מריצים אותה ומחזירים את התשובה המתקבלת. שאילתות full text מקבלות את הערך כפרמטר והוא מוכנס לתוך מחרוזת השאילתא.
* דוגמאות למתודות - get\_teams\_that\_won\_against\_odds, get\_top\_position\_players\_stats וכו'.

## **API**

השתמשנו ב-API שכתובתו https://api.collegefootballdata.com ובו יש נתונים על פוטבול אמריקאי. השימוש ב-API מצריך KEY שקיבלנו בהרשמה לאתר. קריאות ל-API מתבצעות באמצעות בקשות GET עם שני headers:

1. accept - פורמט התשובה שאנו רוצים לקבל. בחרנו ב-application/json.
2. authorization - מפתח האימות שקיבלנו.

ב-API ישנם סוגים רבים של נתונים וניתן לפנות אליו גם דרך swagger. בחרנו מתוכו 7 קריאות:

|  |  |  |
| --- | --- | --- |
| קריאה | משמעות הקריאה | מס' רשומות |
| /games?year=2022&seasonType=regular | נתונים על משחקים ששוחקו בשנת 2022 בעונה הרגילה | 3668 |
| /teams | נתונים על קבוצות | 1791 |
| /player/usage?year=2022 | נתונים על שחקנים בשנת 2022 | 2616 |
| /stats/season?year=2022 | נתונים על סטטיסטיקות בשנת 2022 | 4192 |
| /venues | נתונים על מקומות בהם מתרחשים משחקים (stadiums) | 804 |
| /draft/positions | נתונים על תפקידי שחקנים  בדראפט הפוטבול האמריקאי | 28 |
| /draft/picks?year=2022 | נתונים עבור הבחירות בדראפט של שנת 2022 | 262 |
| /draft/teams | נתונים עבור הקבוצות שבחרו בדראפט של שנת 2022 | 32 |

לאחר קבלת התשובות מה-API פרסרנו את הנתונים לתוך json באמצעות json.load והחזרנו אותם למחלקה שיוצרת את הטבלאות ב-DB.

## **General Flow**

משתמשי האפליקציה הם אוהדים אשר רוצים לקבל מידע מעניין על הספורט האהוב עליהם - פוטבול אמריקאי.   
ראשית, יוצרים בסיס נתונים אשר מכיל את הנתונים הדרושים לקבלת המידע באמצעות create\_db.   
לאחר מכן, מביאים את הנתונים מתוך ה-API של collegefootballdata באמצעות get\_data\_from\_api.  
אחר כך, מכניסים את הנתונים לתוך הטבלאות שבבסיס הנתונים באמצעות insert\_data\_into\_db.  
בשלב זה כל המידע המעניין נמצא בתוך בסיס הנתונים ונותר רק לשלוף אותו.   
באמצעות ה-UI האוהדים לוחצים על הכפתורים עם השאילתא המעניינת אותם, אם צריך מכניסים את הטקסט המתאים (עבור שאילתות full text), המחלקה query\_db מבצעת שליפה על ה-DB והאוהדים מקבלים על המסך את תוצאות השאילתא המבוקשת.