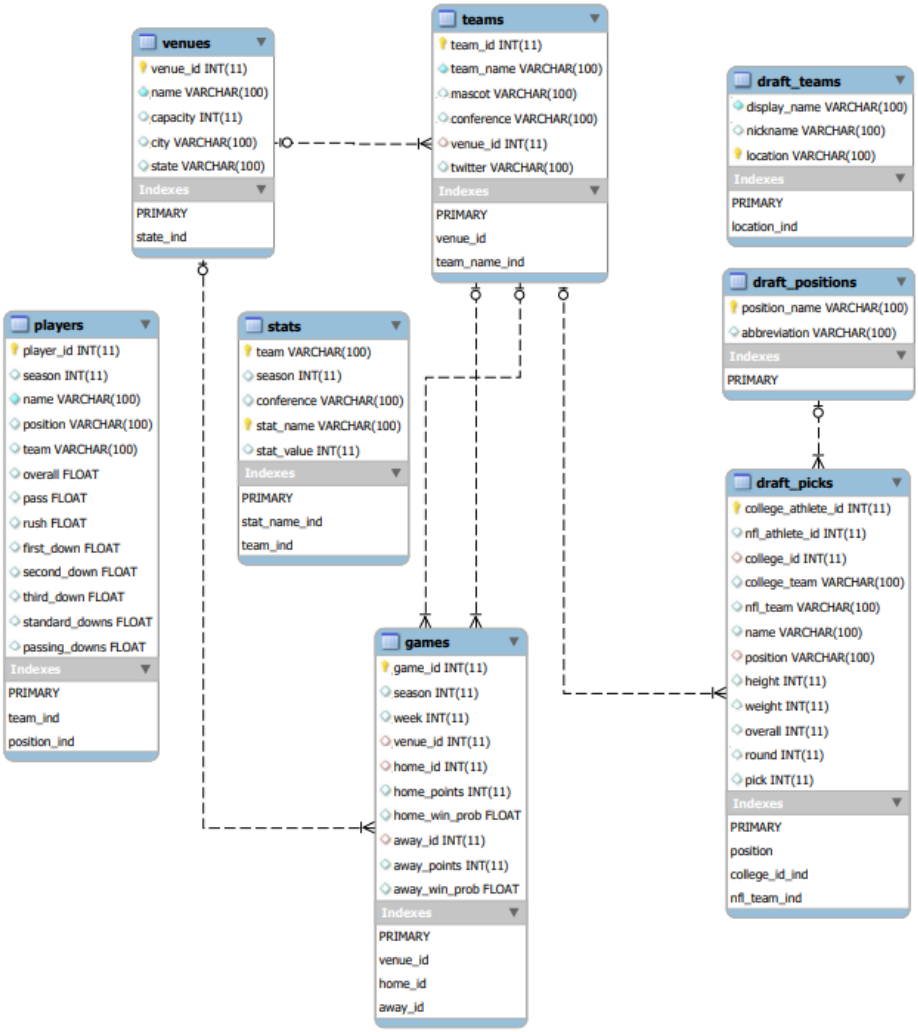
SOFTWARE DOCUMENTATION

Liron Cohen 207481268, Ofer Tlusty 311396303

DB Scheme Structure



הסבר על בחירת עיצוב ה-DB:

מבנה ה-DB שלנו בנוי כך שישנן שתי קבוצות ראשיות של טבלאות: **טבלאות על ליגת ה-NFL** (תכונה מעתה "הליגה הראשית") **וטבלאות הנוגעות לתהליך בחירת השחקנים** (התהליך יכונה לעיתים ה-draft, והליגה תכונה "ליגת הקולג'ים"). בכל קבוצה ישנן טבלאות מרכזיות וכן חיבור בין שתי הקבוצות כפי שיתואר כעת.

בכל קבוצת טבלאות ישנן טבלאות מרכזיות (בליגה הראשית: games, teams ו-players, בליגת הקולג'ים: draft_picks ו-draft_teams) כאשר הטבלאות הנוספות מהוות מידע משלים אודות הטבלאות המרכזיות. כמובן שישנם קשרים גם בין שתי קבוצות הטבלאות ועל כך בסעיף "תיאור הטבלאות" המופיע מטה.

נציין כי חלק מהטבלאות אינן מחוברות אחת לשניה באופן המוצג בדיאגרמה שכן אינן מחוברות באמצעות מפתחות זרים אלא באמצעות אינדקסים כפי שיוסבר בסעיף DB optimization בהמשך.

במסגרת בניית ה-DB הקפדנו לעמוד בעקרונות אשר נלמדו בקורס:

1) חיבור בין טבלאות ייעשה ע"י מפתחות ראשיים, מפתחות זרים או לפחות על ידי שדה מאונדקס
2) נמנענו משמירת מידע כפול בין מספר טבלאות שלא לצורך.
3) חלוקת המידע בין הטבלאות נבחר בצורה שתהיה נוחה ואיטואיטיבית יחסית.
4) הגדרנו שדות משמעותיים כ-NOT NULL על מנת למנוע בעייתיות בתשובות השאילתות.
5) עדכון ושינוי ה-DB יכול להתבצע בצורה נוחה ויעילה – למשל הוספה של קבוצה חדשה בליגה הראשית תבוצע בטבלה יחידה, הוספת סטטיסטיקה מסוימת על שחקן תבוצע בטבלת players בלבד בעוד סטיסטיקה מסוימת על קבוצה תתבצע בטבלת ה-stats.
6) יעילות ביחס לשאילתות המבוצעות על ה-DB – בחירת המפתחות, האינדקסים ומבנה ה-DB עצמו נעשה מתוך מחשבה על השאילתות אשר בחרנו להפעיל באתר/אפליקציה שבנינו כדי לייעל את ריצת השאילתות.

<u>תיאור הטבלאות:</u>

- teams:
  - <u>הסבר כללי</u>: הטבלה כוללת מידע על הקבוצות בליגה הראשית (בשונה מהטבלה draft_team עליה נרחיב בהמשך) וכוללת מידע על שם הקבוצה וה-conference אליה היא משוייכת, שם הקמיע וחשבון הטוויטר של הקבוצה וכן מזהה האצטדיון של הקבוצה.
  - <u>הגדרה</u>: teams(team_id,team_name,mascot,conference,venue_id,twitter)
  - <u>מפתחות</u>:
    - team_id :primary key
    - venue_id :foreign key
  - <u>אינדקסים</u>:
    - team_id (primary key) *[בשימוש בשאילתות 1, 2 ו-6]*
    - venue_id (foreign key) *[בשימוש בשאילתא 6]*
    - team_name – משמש כשדה החיבור בין טבלה זו לטבלה player (שכן לא קיים בה שדה של team_id וראו DB Optimization בהמשך המסמך) *[בשימוש בשאילתות 2 ו-6]*
- games:
  - <u>הסבר כללי</u>: הטבלה כוללת מידע על המשחקים בליגה הראשית וכוללת פרטים על המשחק עצמו (מזהה, עונה ושבוע), על האצטדיון בו התקיים המשחק וכן פרטים על שתי הקבוצות ששיחקו במשחק (מזהה הקבוצה, נקודות במשחק, הסתברות לניצחון).
  - <u>הגדרה</u>:

games(game_id,season,week,venue_id,home_id,home_points,home_post_win_prob,away_id,away_points,away_post_win_prob)

  - <u>מפתחות</u>:
    - game_id :primary key
    - away_id ,home_id ,venue_id :foreign key
  - <u>אינדקסים</u>:
    - game_id (primary key)
    - away_id ,home_id ,venue_id (foreign key) *[בשימוש בשאילתא 1]*

- **players**:
  - הסבר כללי: הטבלה כוללת מידע על השחקנים המשחקים בליגה הראשית וכוללת פרטים על השחקן עצמו (מזהה, שם, תפקיד, שיוך לקבוצה) וכן סטטיסטיקות אודותיו (וראו בהגדרה המופיעה מטה).
  - הגדרה:

    players(player_id,season,name,position,team,overall,pass,rush,first_down,second_down, third_down,standard_downs,passing_downs)

  - מפתחות:
    - primary key: player_id
  - אינדקסים:
    - (primary key) player_id
    - team – משמש כשדה החיבור בין טבלה זו לטבלה teams (שכן לא קיים בטבלה player שדה של team_id וראו DB Optimization בהמשך המסמך) *[בשימוש בשאילתא 2]*
    - Position – משמש לאופטימיזציה של שאילתא 2 בה אנו מבצעים שליפה על בסיס קלט מהמשתמש ובוחרים שורות לפי שדה זה. *[בשימוש בשאילתא 2]*

- **venues**:
  - הסבר כללי: הטבלה כוללת מידע על האצטדיונים של קבוצות וכוללת מידע על שם המקום, הקיבולת שלו ומיקומו.
  - הגדרה: venues(venue_id, name, capacity, city, state)
  - מפתחות:
    - primary key: venue_id
  - אינדקסים:
    - (primary key) venue_id *[בשימוש בשאילתות 1 ו-6]*
    - State – משמש לאופטימיזציה של שאילתא 4 בה אנו מבצעים אגרגציה לפי שדה זה. *[בשימוש בשאילתא 4]*

- **stats**:
  - הסבר כללי: הטבלה כוללת שמות וערכי סטטיסטיקות הנוגעות לקבוצות השונות בליגה הראשית.
  - הגדרה:

    stats(team,season,conference,stat_name,stat_value)
  - מפתחות:
    - primary key: (team,stat_name)
  - אינדקסים:
    - (primary key) team,stat_name
    - stat_name משמש לאופטימיזציה של שאילתא 3 בה אנו מחשבים ממוצע לכל סטטיסטיקה עבור כל הקבוצות. *[בשימוש בשאילתא 3]*
    - אינקדס מסוג FULLTEXT עבור השדה: team המשמש לאופטימיזציה של שאילתא 3 בה אנו מקבלים קלט מהמשתמש ומעוניינים למצוא את הסטטיסטיקות של הקבוצה במהירות. *[בשימוש בשאילתא 3]*
- **draft_positions**:
  - הסבר כללי: הטבלה כוללת הצמדה בין תפקיד במשחק לבין שמו המקוצר.
  - הגדרה

    draft_positions (position_name,abbreviation)
  - מפתחות:
    - primary key: position_name
  - אינדקסים:
    - (primary key) position_name *[בשימוש בשאילתא 5]*
- **draft_picks**:
  - הסבר כללי: הטבלה כוללת מידע הנוגע לשחקנים אשר השתתפו ב-draft (בחירת שחקנים מליגת הקולג'ים לליגה הראשית). המידע כולל התייחסות לשחקן עצמו (שם, תפקיד, גובה, משקל וכן סטטיסטיקות אודותיו), שיוך קבוצתי (באיזה קולג' שיחק, איזה קבוצה בליגה הראשית בחרה בו).
  - הגדרה:

draft_picks(college_athlete_id,nfl_athlete_id,college_id,college_team,nfl_team,name,position, height,weight,overall,round,pick)

  - מפתחות:
    - primary key: college_athlete_id
    - foreign key: position, college_id
  - אינדקסים:
    - (primary key) college_athlete_id
    - (foreign hey) college_id, position *[בשימוש בשאילתא 6]*
    - nfl_team – משמש כשדה החיבור בין טבלה זו לטבלה draft_teams. *[בשימוש בשאילתא 5]*

- draft_teams:
  - הסבר כללי: הטבלה כוללת מידע על הקבוצות בליגת הקולג'ים וכוללת מידע על שם הקבוצה בליגת הקולג'ים (display_name), כינוי הקבוצה וכן מיקומה אשר בפועל מייצג את שמה כ-nft_team כפי שיוסבר בהמשך הפרויקט (שאילתא 5).
  - הגדרה: draft_teams(display_name,nickname,location)
  - מפתחות:
    - display_name :primary key
  - אינדקסים:
    - display_name (primary key)
    - location - משמש כשדה החיבור בין טבלה זו לטבלה draft_picks *[בשימוש בשאילתא 5]*

DB Optimizations

האופטימיזציות שנעשו בעת עיצוב מבנה ה-DB, במפתחות שנבחרו ובאינדקסים שהתווספו פורטו לעיל.

נציין כי לאור העדר מזהה ייחודי המהווה מפתח ראשי בחלק מהטבלאות, החלטנו להוסיף אינדקסים על שדות טקסטואליים (לרוב שמות הקבוצות) אשר מהווים שדות שיאפשרו חיבור הגיוני בין זוג טבלאות שונות. במידה והיינו משתמשים במפתח זר היינו נדרשים להגדיר את השדות הללו כ"ייחודיים" באמצעות הפעולה UNIQUE ובכך לאבד חלק מהמידע.

Queries

להלן פירוט על כל אחת מהשאילתות שכתבנו:

1. <u>get_teams_that_won_against_odds</u>

```
SELECT T1.team_name AS home_team, T2.team_name AS away_team, G.season, G.week, G.away_points, G.home_points, G.away_win_prob AS prob, V.name AS venue_name
FROM lironcohen3.games AS G, lironcohen3.venues AS V, lironcohen3.teams AS T1, lironcohen3.teams AS T2
WHERE (G.venue_id = V.venue_id AND G.home_id = T1.team_id AND G.away_id = T2.team_id)
        AND (G.home_win_prob > G.away_win_prob AND G.home_points < G.away_points)
        AND (G.away_points - G.home_points > 10 AND G.away_win_prob < 0.4)
UNION SELECT T1.team_name AS home_team, T2.team_name AS away_team, G.season, G.week, G.away_points, G.home_points, G.away_win_prob AS prob, V.name AS venue_name
FROM lironcohen3.games AS G, lironcohen3.venues AS V, lironcohen3.teams AS T1, lironcohen3.teams AS T2
WHERE (G.venue_id = V.venue_id AND G.home_id = T1.team_id AND G.away_id = T2.team_id)
        AND (G.home_win_prob < G.away_win_prob AND G.home_points > G.away_points)
        AND (G.home_points - G.away_points > 10 AND G.home_win_prob < 0.4)
ORDER BY prob
```

- <u>קבוצות שניצלחו "כנגד כל הסיכויים":</u> 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 (במוסבר מעלה).

2. <u>get_top_position_players_stats</u>

```
SELECT
    P.name,
    P.position,
    T.team_name,
    SUM(CASE WHEN G.home_points > G.away_points THEN 1 ELSE 0 END) AS wins,
    SUM(CASE WHEN G.home_points = G.away_points THEN 1 ELSE 0 END) AS ties,
    SUM(CASE WHEN G.home_points < G.away_points THEN 1 ELSE 0 END) AS losses,
    AVG(P.first_down) AS avg_first_down,
    AVG(P.second_down) AS avg_second_down,
    AVG(P.third_down) AS avg_third_down
FROM
    lironcohen3.players AS P
        LEFT JOIN
    lironcohen3.teams AS T ON P.team = T.team_name
        LEFT JOIN
    lironcohen3.games AS G ON (G.home_id = T.team_id
        OR G.away_id = T.team_id)
WHERE
    P.position = '{input_position_abb}'
GROUP BY P.player_id
ORDER BY first_down DESC
LIMIT 50
```

- שאילתא זו מקבלת שם של תפקיד בפוטבול האמריקאי (QB, CB, LT וכו') ומחזירה את מזהי השחקן עצמו (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).

3. <u>get_team_stats_better_than_average</u>

```
SELECT
    s1.team, s1.stat_name, s1.stat_value
FROM
    lironcohen3.stats AS s1
WHERE
    s1.team = '{input_team_name}'
        AND s1.stat_value >= (SELECT AVG(s2.stat_value) AS average_value
                              FROM
                                  lironcohen3.stats AS s2
                              WHERE
                                  s1.stat_name = s2.stat_name
                              GROUP BY s2.stat_name)
```

- שאילתא זו הינה שאילתת ה-FULL TEXT אשר בחרנו לממש אשר משתמש באינדקס Full text אשר הוגדר בעת בניית הטבלאות.
- 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.

4. <u>get_max_capacity_venue_per_state</u>

```
SELECT
    v1.state, v1.name, v1.capacity
FROM
    lironcohen3.venues AS v1
WHERE
    v1.state <> ''
        AND v1.capacity >= ALL (SELECT v2.capacity
                                FROM
                                    lironcohen3.venues AS v2
                                WHERE
                                    v2.state = v1.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.

5. <u>get_min_max_weights_heights_per_draft_position</u>

```
# convertion rate from pound to kg is 0.0254
# convertion rate from inch to meter is 0.4535

SELECT
    Dpos.position_name,
    Dpos.abbreviation,
    MIN(Dpick.height) * 0.0254 AS min_height_meter,
    MAX(Dpick.height) * 0.0254 AS max_height_meter,
    MIN(Dpick.weight) * 0.4535 AS min_weight,
    MAX(Dpick.weight) * 0.4535 AS max_weight
FROM
    lironcohen3.draft_picks AS Dpick,
    lironcohen3.draft_teams AS DT,
    lironcohen3.draft_positions AS Dpos
WHERE
    DT.location = Dpick.nfl_team
    AND Dpos.position_name = Dpick.position
GROUP BY Dpos.position_name
```

- 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).

6. get_most_picked_college_teams_in_draft

```sql
SELECT
    T.team_name,
    T.conference,
    T.mascot,
    T.twitter,
    V.name AS venue_name,
    SUM(DP.overall) AS sum_overall_draft,
    count(DP.overall) as cnt_overall_draft,
    (SUM(DP.overall) / count(DP.overall)) as avg_overall_draft
FROM
    lironcohen3.draft_picks AS DP,
    lironcohen3.teams AS T,
    lironcohen3.venues AS V
WHERE
    T.team_id = DP.college_id
        AND T.venue_id = V.venue_id
GROUP BY T.team_name , T.team_id
HAVING cnt_overall_draft > 1
ORDER BY avg_overall_draft
LIMIT 10
```

- 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. <u>create db</u>

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.

2. <u>get data from api</u>

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'.

3. <u>insert data into db</u>

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'.

4. <u>query db</u>

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	1792
/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	30

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.