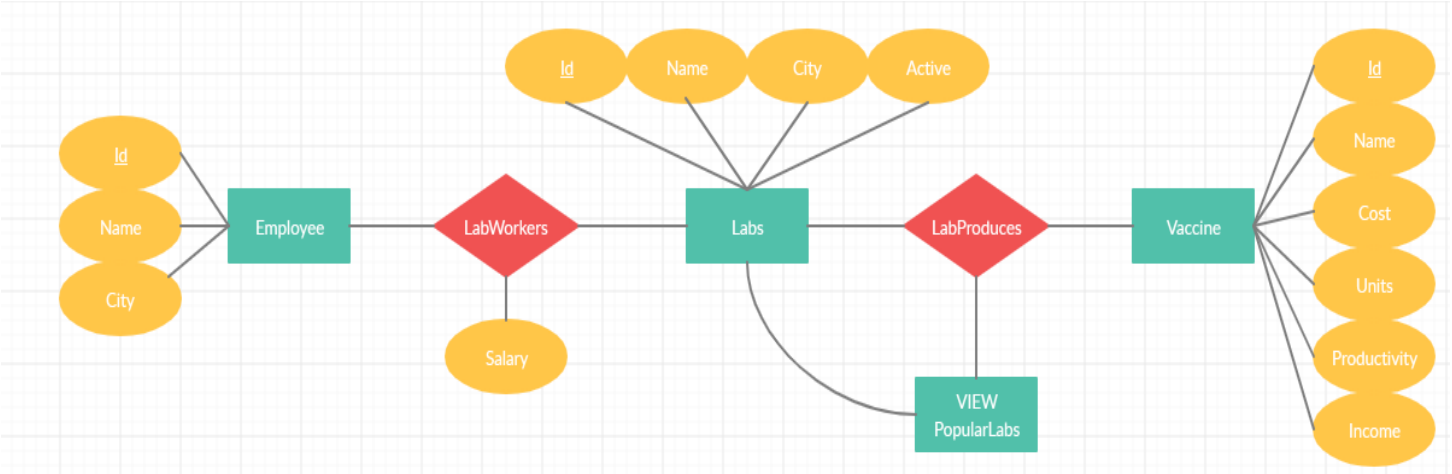
מגיש: אביר שקד 204596597

תרשים ERD (לא פורמאלי):

(לא הצלחתי למצוא איך לייצג view אז ניסיתי לתאר כמה שאפשר זהו ERD לא פורמאלי)



טבלאות(והשאילתה אשר יוצרת אותן):

טבלה Labs:

```
CREATE TABLE Labs (
Id int PRIMARY KEY ,
Name text NOT NULL ,
City text NOT NULL ,
Active boolean NOT NULL ,
CHECK(id > 0))
```

Id	Name	City	Active

כאשר Id הינו מפתח ראשי (כלומר אינו null) והוא מספר גדול מ0.
Name,City הינם מסוג text , active בוליאני (ושלושתם אינם null).

טבלה Employees:

```
CREATE TABLE Employees
Id int PRIMARY KEY ,"
Name text NOT NULL ,"
City text NOT NULL ,"
CHECK(id > 0))
```

Id	Name	City

כאשר Id הינו מפתח ראשי (כלומר אינו null) והוא מספר גדול מ0.
City הינם מסוג text (ואינם null).

טבלה Vaccines:

```
CREATE TABLE Vaccines (
Id int PRIMARY KEY ,
Name text NOT NULL ,
Cost int NOT NULL ,
Units int NOT NULL ,
Productivity int NOT NULL ,
Income int DEFAULT(0) ,
CHECK(id > 0 AND Cost >= 0 AND Units >= 0 AND Productivity >= 0 AND
Productivity <= 100 AND Income >= 0))
```

Id	Name	Cost	Units	Productivity	Income

כאשר Id הינו מפתח ראשי (כלומר אינו null) והוא מספר גדול מ0.
Name הינו Text ואינו null , Cost,Units,Productivity מסוג Int ואינם null.
וכן Cost,Units אינם שליליים , ו Productivity אינו שלילי וקטן שווה ל 100.
ו Income מסוג Int בעל ערך דיפולטי 0 , וכן אי שלילי (ולכן אינו null).

טבלה LabWorkers:

```
CREATE TABLE LabWorkers (
Employee_Id int REFERENCES Employees(Id) ON DELETE CASCADE ,
Lab_Id int REFERENCES Labs(Id) ON DELETE CASCADE ,
Salary int NOT NULL ,
UNIQUE(Employee_Id,Lab_Id) ,
CHECK(Salary >= 0))
```

Employee_Id	Lab_Id	Salary

כאשר Employee_Id,Lab_Id הינם uniqu (כלומר מפתחות), ושניהם מפתחות זרים בהתאם מהטבלאות Employees,Labs (המוגדרים להתעדכן במחיקה).
בחרתי שלא להגדיר אותם כ PrimaryKeys מפני שאנו לא רוצים לאפיין כניסה בטבלה הנל באמצעות הצמד, לפעמים נרצה את המידע עבור Lab_Id כלשהו ולפעמים לפי Employee_Id כלשהו , לכן הם מוגדרים כ Unique מפני שלא נרצה כפילות של עובד שעובד במעבדה בשכר שונה, אך נרצה לאפשר למעבדה עובדים שונים, ולעובד לעבוד במעבדות שונות.
Salary הינו Int , אינו NULL ואי שלילי.

טבלה LabProduces:

```
CREATE TABLE LabProduces (
Lab_Id int REFERENCES Labs(Id) ON DELETE CASCADE ,
Vaccine_Id int REFERENCES Vaccines(Id) ON DELETE CASCADE ,
UNIQUE(Lab_Id,Vaccine_Id))
```

Lab_Id	Vaccine_Id

כאשרVaccine_Id,Lab_Id הינם uniqu (כלומר מפתחות), ושניהם מפתחות זרים בהתאם מהטבלאות Vaccines,Labs (המוגדרים להתעדכן במחיקה).
הם לא מוגדרים כPrimaryKeys מסיבה דומה למוזכרת לעיל.

View PopularLabs:

```
CREATE VIEW PopularLabs AS
SELECT L.Id AS Lab_Id
FROM Labs AS L " +
WHERE L.Id NOT IN (SELECT P.Lab_Id FROM LabProduces AS P, Vaccines AS V
WHERE L.Id = P.Lab_Id AND P.Vaccine_Id = V.Id AND V.Productivity <= 20)
```

Lab_Id

כאן נקבל view אשר מציג את כל הId מ Labs אשר מקיימים כי , הId אינו נמצא בטבלה של המעבדות אשר מייצרות תרופה בעלת Productivity קטן מ20 (לכן בפרט אם המעבדה לא מייצרת אף תרופה היא תופיע גם כן).

מתודות API:

הערות לגבי אופן פירוט המתודות: בכל המתודות נעשה שימוש במתודת עזר

אשר מטפלות בשגיאה ומחזירה את ערך החזרה בהתאם ובמתודה אשר משחררת את המשאבים, אך אתעלם מהן כאן מפני שזהו פרט מימוש פנימי.

בשאילתות כאשר יופיעו ?, אנו נוסיף להם ערכים באצמעות setters של ה statement ולכן לא אציין זאת מחדש בכל מתודה, כמו כן כאשר נפעל בהתאם לשדות של טיפוס מסויים נקבל אותם בעזרת ה getters של הטיפוס, ולכן גם כאן לא אציין זאת מחדש בכל מתודה.

כמו כן את הקלט ניתן לראות לפי תיאור שם המתודה ולכן לא אציין זאת.

(*) הוספתי את השאילות למסד הנתונים אשר הן החלקי המרכזי בכל מתודה על מנת להסביר את המימוש, ניתן מהשאילתה עצמה להבין את אופן הפעולה על הטבלאות, וניסיתי להסביר במילים את הלוגיקה העומדת מאחורי השאילתה.

מתודות יצירת ניקוי ומחיקת הטבלה: createTables מפורטת ליד כל טבלה בהתאמה.

מתודת clearTables:

```
DELETE FROM X
```

נעשו באופן דומה כאשר X הוא שם הטבלה אותה נרצה לנקות ובעצם נמחוק בה את כל הטאפלים

נעשה זאת עבור כל טבלה מפני שהview יתנקה בהתאם לטבלאות עליהן הוא מוגדר.

מתודת delteTables:

```
DROP VIEW IF EXISTS PopularLabs
```

עבור ה view

```
DROP TABLE IF EXISTS X
```

נעשו באופן דומה כאשר X הוא שם הטבלה אותה נרצה למחוק, נשים לב שנמחק תכילה את הview

לאחר מכן נמחק את כל היחסים (טבלאות אשר המפתחות בהן הם מפתחות זרים מטבלאות אחרות) ולאחר מכן את הטבלאות האחרות.

ReturnValue addLab(Lab lab):

```
INSERT INTO Labs VALUES (? ,? ,? ,?)
```

במתודה זו מתבצעת הכנסה של טאפל לLabs בהתאם לשדות של lab.

ערכי החזרה:OK במקרה של הצלחה, BAD_PARAMS במקרה של פרמטרים לא חוקיים בהתאם לטבלה Labs , ALLREADY_EXISTS במקרה שהId (שהוא המפתח הראשי אשר בעזרתו ניתן לזהות טאפל) כבר קיים, ERROR במקרה של שגיאה של המסד.

Lab getLabProfile(Integer labID):

```
SELECT * FROM Labs where Id = ?
```

במתודה זו מתבצעת השמה מערך של טאפל ב Labs vמתאים למפתח Id = labID (בגלל שId הינו מפתח ראשי ניתן בעזרתו לזהות טאפל), אל Lab אשר שדותיו יהיו ערכי הטאפל בהתאמה בעזרת ה setters של Lab וה getters של ResultSet.

ערכי החזרה:Lab המכיל את שדות הטאפל המתאים אם המפתח קיים , Lab.badLab אחרת.

ReturnValue deleteLab (Lab lab):

```
DELETE FROM Labs where Id = ?
```

במתודה זו אנו מוחקים מהטבלה Labs טאפלים אשר הId שלהם שווה לשדה id של lab.
מכיוון שId הינו מפתח ראשי, ניתן בעזרתו לזהות את המעבדה ולכן נמחוק עלפיו שורה **יחידה**.
(*) לפי אופן הגדרת הטבלאות האחרות בעת המחיקה כל טבלה אחרת המכילה מפתח זר השייך ל Id המל תתעדכן בהתאם.

ערכי החזרה: OK במקרה של הצלחה, NOT_EXISTS אם המעבדה המתאימה אינה קיימת (כלומר לא נמחקו שורות), ERROR במקרה של שגיאה של המסד.

ReturnValue addEmployee (Employee employee):

```
INSERT INTO Employees VALUES (? ,? ,?)
```

במתודה זו מתבצעת הכנסה של טאפל אל Employees בהתאם לשדות של employee.

ערכי החזרה:OK במקרה של הצלחה, BAD_PARAMS במקרה של פרמטרים לא חוקיים בהתאם לטבלה Employees, ALLREADY_EXISTS במקרה שהId (שהוא המפתח הראשי אשר בעזרתו ניתן לזהות טאפל) כבר קיים, ERROR במקרה של שגיאה של המסד.

Employee getEmployeeProfile (Integer EmployeeID):

```
SELECT * FROM Employees where Id = ?
```

במתודה זו מתבצעת השמה מערך של טאפל ב Employees vמתאים למפתח Id = EmployeeID (בגלל שId הינו מפתח ראשי ניתן בעזרתו לזהות טאפל), אל Employee אשר שדותיו יהיו ערכי הטאפל בהתאמה בעזרת ה setters של Lab וה getters של ResultSet.

ערכי החזרה:Employee המכיל את שדות הטאפל המתאים אם המפתח קיים , Employee.badEmployee אחרת.

ReturnValue deleteEmployee (Employee employee):

```
DELETE FROM Employees where Id = ?
```

במתודה זו אנו מוחקים מהטבלה Employees טאפלים אשר הId שלהם שווה לשדה id של employee.
מכיוון שId הינו מפתח ראשי, ניתן בעזרתו לזהות את העובד ולכן נמחוק עלפיו שורה **יחידה**.
(*) לפי אופן הגדרת הטבלאות האחרות בעת המחיקה כל טבלה אחרת המכילה מפתח זר השייך ל Id הנל תתעדכן בהתאם.

ערכי החזרה: OK במקרה של הצלחה, NOT_EXISTS אם העובד המתאים אינו קיים (כלומר לא נמחקו שורות), ERROR במקרה של שגיאה של המסד.

ReturnValue addVaccine (Vaccine vaccine):

```
INSERT INTO Vaccines VALUES (? ,? ,? ,? ,?)
```

במתודה זו מתבצעת הכנסה של טאפל אל Vaccines בהתאם לשדות של vaccine.

ערכי החזרה:OK במקרה של הצלחה, BAD_PARAMS במקרה של פרמטרים לא חוקיים בהתאם לטבלה Vaccines, ALLREADY_EXISTS במקרה שהId (שהוא המפתח הראשי אשר בעזרתו ניתן לזהות טאפל) כבר קיים, ERROR במקרה של שגיאה של המסד.

Vaccine getVaccineProfile (Integer VaccineID):

```
SELECT * FROM Vaccines where Id = ?
```

במתודה זו מתבצעת השמה מערך של טאפל ב Vaccines vמתאים למפתחId = VaccineID (בגלל שId הינו מפתח ראשי ניתן בעזרתו לזהות טאפל), אל Vaccine אשר שדותיו יהיו ערכי הטאפל בהתאמה בעזרת ה setters של Vaccine וה getters של ResultSet.

ערכי החזרה:Vaccine המכיל את שדות הטאפל המתאים אם המפתח קיים , Vaccine.bad Vaccine אחרת.

ReturnValue deleteVaccine (Vaccine vaccine):

```
DELETE FROM Vaccines where Id = ?
```

במתודה זו אנו מוחקים מהטבלה Vaccines טאפלים אשר הId שלהם שווה לשדה id של vaccine.
מכיוון שId הינו מפתח ראשי, ניתן בעזרתו לזהות את החיסון ולכן נמחוק עלפיו שורה **יחידה**.
(*) לפי אופן הגדרת הטבלאות האחרות בעת המחיקה כל טבלה אחרת המכילה מפתח זר השייך ל Id הנל תתעדכן בהתאם.

ערכי החזרה: OK במקרה של הצלחה, NOT_EXISTS אם החיסון המתאים אינו קיים (כלומר לא נמחקו שורות), ERROR במקרה של שגיאה של המסד.

ReturnValue employeeJoinLab(Integer employeeID, Integer labID, Integer salary):

```
INSERT INTO LabWorkers VALUES (? ,? ,?)
```

במתודה זו מתבצעת הכנסה של טאפל אל LabWorkers בהתאם לemployeeID ו labID ו salary.
כאשר לפי אופן הגדרתם כדי שההכנסה תצליח על labID ו employeeID להיות מפתחות זרים מLabs ו Employees בהתאמה, וכן הזוג הנל ייחודי (כלומר לא יכול להופיע יותר מפעם אחת). (*) נשים לב שמאילוצים אלו גם נובע כי הם בהכרח חיוביים ממש.

ערכי החזרה:OK במקרה של הצלחה, BAD_PARAMS במקרה של פרמטרים לא חוקיים בהתאם כפי שצויין לעיל ולפי הגדרת הטבלה LabWorkers, ALLREADY_EXISTS בקרה שזוג המפתחות הייחודיים לעיל כבר קיימים בטבלה, ERROR במקרה של שגיאה של המסד.

ReturnValue emplyeeLeftLab(Integer labID, Integer employeeID):

```
DELETE FROM LabWorkers where Employee_Id = ? AND Lab_Id = ?

        "
```

במתודה זו מתבצעת מחיקה מהטבלה LabWorkes של הטאפל אשר מכיל Emplyee_Id = employeeID וכן Lab_Id = labID.

כאשר באופן דומה לעיל נבצע השמות לstatement.

ערכי החזרה: OK בהצלחה,NOT_EXISTS אם הטאפל אשר מכיל את המפחות הייחודיים הנמל אינו נמצא בטבלה, ERROR בשגיאה של המסד.

ReturnValue labProduceVaccine(Integer vaccineID, Integer labID):

```
INSERT INTO LabProduces VALUES (? ,? )
```

במתודה זו מתבצעת הכנסה של טאפל אל LabProduces בהתאם לemployeeID ו vaccineID .
כאשר לפי אופן הגדרתם כדי שההכנסה תצליח עליהם להיות מפתחות זרים מLabs ו Vaccines בהתאמה, וכן הזוג הנל ייחודי (כלומר לא יכול להופיע יותר מפעם אחת). (*) נשים לב שמאילוצים אלו גם נובע כי הם בהכרח חיוביים ממש.

ערכי החזרה:OK במקרה של הצלחה, BAD_PARAMS במקרה של פרמטרים לא חוקיים בהתאם כפי שצויין לעיל ולפי הגדרת הטבלה LabProduces , ALLREADY_EXISTS בקרה שהזוג הנל כבר קיים בטבלה, ERROR במקרה של שגיאה של המסד.

ReturnValue labStoppedProducingVaccine(Integer labID, Integer vaccineID):

```
DELETE FROM LabProduces WHERE Lab_Id = ? AND Vaccine_Id = ?
```

במתודה זו מתבצעת מחיקה מהטבלה LabProduces של הטאפל אשר מכיל Vaccine_Id = vaccineID וכן Lab_Id = labID.

כאשר באופן דומה לעיל נבצע השמות לstatement.

ערכי החזרה: OK בהצלחה,NOT_EXISTS אם הזוג הנל אינו קיים בטבלה, ERROR בשגיאה של המסד.

ReturnValue vaccineSold(Integer vaccineID, Integer amount):

```
UPDATE Vaccines
SET Units = Units - ?, Cost = Cost * 2 ,
Productivity = LEAST(Productivity + 15, 100) ,
Income = Income + Cost * ?
WHERE Id = ?
```

במתודה זו מתבצע עדכון לטבלה Vaccine כך שכעת הטאפל אשר מזוהה עם הvaccineID=Id

יכיל ב Units מספר הקטן כעת בamount , ב Cost מספר הכפול ב2, בProductivity מספר הגדול ב15 עד למקסימום של 100, וב Income מספר הגדול כעת בCost*amount.

ערכי החזרה:OK במקרה של הצלחה, BAD_PARAM במקרה של הפרת דרישות הטבלה, כלומר Units יהיה קטן מ0 אם תתבצע, NOT_EXISTS במקרה והתרופה הנל אינה קיימת, ERROR בשגיאה של המסד.

ReturnValue vaccineProduced(Integer vaccineID, Integer amount):

```
SELECT NULL WHERE ? >= 0
```

כאשר הערך הינו amount ולכן נקבל ערך חזרה אם amount חיובי ממש, אחרת לא נקבל ערך החזרה מהשאילתה.

```
UPDATE Vaccines
SET Units = Units + ? ,Cost = Cost / 2 ,
Productivity = GREATEST(Productivity - 15, 0)
WHERE Id = ?
```

מתבצע עדכון לטבלה Vaccine כך שכעת הטאפל אשר מזוהה עם הvaccineID=Id

יכיל ב Units מספר הגדול כעת בamount , ב Cost מספר הקטן פי 2, בProductivity מספר הקטן ב15 עד למינימום של 0.

ערכי החזרה:OK במקרה של הצלחה, BAD_PARAM ש amount קטן מ0 אשר בדקנו בשאילה הראשונה במתודה, NOT_EXISTS במקרה והתרופה הנל אינה קיימת, ERROR בשגיאה

Boolean isLabPopular (Integer labID):

```
SELECT * FROM PopularLabs WHERE Lab_Id = ?
```

במתודה זו אנו נבדוק האם המעבדה בעלת labID מופיע בטבלה PopularLabs אשר הוגדרה והוסברה לעיל. (כלומר המעבדה פופולרית לפי דרישות התרגיל).

ערכי החזרה:true במקרה שהמעבדה שייכת false אחרת.

Integer getIncomeFromVaccine(Integer vaccineID):

```
SELECT Income FROM Vaccines WHERE Id = ?
```

במתודה זה ניגש לטבלה Vaccines וניקח את הטאפל אשר מזוהה ע"י vaccineID=Id, וניקח ממנו את ערך הIncome.

ערכי החזרה: Income המתאים לטאפל הנל במקרה שהתרופה קיימת, 0 אחרת.

Integer getTotalNumberOfWorkingVaccines():

```
SELECT SUM(Units) AS Sum_Units FROM Vaccines WHERE Productivity > 20
```

במתודה זו ניגש לטבלה Vaccines , נבחר את כל הטאפלים אשר מקיימים כי הProductivity שלהם גדול מ20, ולאחר מכן נסכום את כל הUnits של הטאפלים המתאימים.

ערכי החזרה: מספר החיסונים שעובדים לפי ההגדרה לעיל.

Integer getTotalWages(Integer labID):

```
SELECT SUM(W.Salary) AS Sum_Wages FROM LabWorkers AS W, Labs AS L
WHERE W.Lab_Id = ? AND W.Lab_Id = L.Id and L.Active = true AND
(SELECT COUNT(Employee_Id) FROM LabWorkers AS W2
WHERE W2.Lab_Id =W.Lab_Id) > 1
```

במתודה זו אנו סוכמים את כל הsalary מהמכפלה הקרטזית בין LabWorkers ל Labs אשר כל הטאפלים בה הם כאלה המסכימים על הId של המעבדה, שהיא פעילה וכן שהId שווה ל labID

וכן כי הטאפל מקיים כי מספר העובדים מLabworkers המקיימים כי המעבדה שלהם היא המעבדה עליה אנו מסתכלים בטאפל הנל, גדול מ1.

ערכי החזרה: סכום המשכורות של כל הטאפלים העומדים בדרישה המוזכרת לעיל (כלומר של מעבדות בהן יש יותר מעובד 1), 0 אם אף טאפל אינו עומד בדרישה.

Integer getBestLab():

```
SELECT L.id as Id FROM LabWorkers AS W , Labs AS L, Employees AS E
WHERE L.Id = W.Lab_Id AND W.Employee_Id = E.Id AND E.City = L.City
GROUP BY L.Id ORDER BY COUNT(L.Id) DESC, L.Id ASC LIMIT 1
```

במתודה זו אנו נבחר את ה Labs.Id של הטאפל השייכך למכפלה הקרטזית בין Labs,LabWorkers,Employees , כאשר נפצל אותם לקבוצות לפי הId של המעבדה, וכעת נבחר רק טאפלים אשר מסכימים על כך שהעובד בטאפל והמעבדה בטאפל מתאימים לכך שהעובד עובד במעבדה הנל ושהיא פעילה, (כלומר נקבל קבוצות שבכל קבוצה יש את כל העובדים העובדים במעבדה מסויימת כאשר היא פעילה), נסדר אותם בסדר יורד לפי סכום השורות בקבוצה זו, וסידור משני לפי הId של המעבדה. ונגביל את התוצאה לטאפל יחיד.

ערכי החזרה: מספר הId של המעבדה הטובה ביותר (כלומר בעלת מספר העובדים הגבוה ביותר אשר גרים בעיר של המעבדה) לפי התנאים לעיל, 0 במקרה של שגיאה או שכל העובדים לא עובדים.

String getMostPopularCity():

```
SELECT E.city AS City
FROM LabWorkers AS W ,Employees AS E
WHERE W.Employee_Id = E.Id
GROUP BY E.City ORDER BY COUNT(E.city) DESC, E.City DESC LIMIT 1
```

במתודה זו נבחר את City של הטאפל השייך למכפלה הקרטזית בין LabWorkers,Employees כאשר נפצל אותם לקבוצות לפי העיר, ונבחר רק את הטאפלים אשר מסכימים על כך שהעובד בטאפל עובד במעבדה כלשהי (כלומר קיבלנו קבוצות כך שבכל קבוצה נמצאים כל העובדים של כל המעבדות אשר שייכות לעיר זו), ונמיין אותם לפי סכום הטאפלים בקבוצה בסדר יורד ומיון משני לפי העיר בסדר יורד, ונגביל את התוצרה לטאפל יחיד.

ערכי החזרה: שם ה City של עיר הלידה הטובה ביותר (כלומר בעלת סכום העובדים השונים בכל מעבדה השייכת לעיר הגבוה ביותר) לפי התנאים לעיל, מחרוזת ריקה במקרה שאף עובד אינו עובד.

ArrayList getPopularLabs():

```
SELECT Lab_Id FROM PopularLabs
WHERE Lab_Id IN (SELECT Lab_Id FROM LabProduces)
ORDER BY Lab_Id ASC LIMIT 3
```

במתודה זו נבחר את שלושת או פחות של ה Lab_Id של הטאפלים מתוך PopularLabs כאשר הid הנל נמצא ב LabProduces כלומר המעבדה מייצרת תרופה כלשהי לפחות, מסודר לפי Lab_Id בסדר עולה, ונגביל את התוצאה ל3.

כלומר נמיין את כל מספרי המעבדות אשר פופולריות, מייצרות תרופה כלשהי, לפי סדר עולה ונבחר את 3 הראשונות.

ערכי החזרה: רשימה אשר בה שלושת (או פחות) מספרי המעבדות בהתאם לעיל (שהן פופולריות מייצרות תרופה אחת לפחות ובעלות מספרי זהות הכי נמוכים מבין אלו). או רשימה ריקה אם אין כאלה מעבדות.

ArrayList getMostRatedVaccines():

```
SELECT Id FROM Vaccines
ORDER BY (Productivity + Units - Cost) DESC, Id ASC LIMIT 10
```

במתודה זו נבחר את עשרה או פחות שדות ה Id של טאפלים מתוך Vaccines כאשר נמיין אותם לפי סכום הUnits , Productivity פחות Cost , בסדר יורד , ומיון משני לפי ה Id ונגביל את התוצאה לעשרה טאפלים.

כלומר נמיין את התרופות לפי הערך לעיל במיון משני לפי מספר המעבדה, ונגביל את התוצאה לעשרת הטאפלים הראשונים.

ערכי החזרה: רשימה אשר בה עשרה (או פחות) מספרי התרופות אשר לפי הערך לעיל גבוה ביותר, ובמקרה של שוויון לפי מספר התרופה. או רשימה ריקה אם אין תרופות כלל או במקרה של שגיאה.

:ArrayList getCloseEmployees(Integer employeeID)

```
ELECT E.Id as Id FROM Employees AS E
WHERE ? IN (SELECT E1.Id FROM Employees AS E1) AND E.Id <> ? AND
(SELECT COUNT(W1.Lab_Id) FROM LabWorkers AS W1
WHERE W1.Employee_Id = ?) <= 2 * (SELECT COUNT(W1.Lab_Id)
FROM LabWorkers AS W1 WHERE W1.Employee_Id = E.Id AND
W1.Lab_Id IN
SELECT Lab_Id FROM LabWorkers AS W2 WHERE W2.Employee_Id = ?))
ORDER BY E.Id ASC LIMIT 10
```

במתודה זו נבחר את עשרת או פחות השדות Id של הטאפלים מתוך Employees אשר מקיימים כי סכום המעבדות בהן עובד העובד בעל הeomployeeID ID הינו קטן או שווה לפי 2 מסכום המעבדות בהן עובד העובד הנל (אשר שונה מemployeeID), אשר עובד בהן העובד eomployeeID, נמיין את התוצאה לפי מספר העובד בסדר עולה , ונגביל את התוצאה ל10 טאפלים.

כלומר אנו נקבל כי כל Id בתוצאה שייך לעובד שונה מ eomployeeID , וכן מספר המעבדות בהן העובד בעל הId הנל, הינו לפחות 50% ממספר המעבדות בהן עובד eomployeeID.

ערכי החזרה: רשימה בה עשרה (או פחות) מספרי העובדים אשר עובדים לפחות ב50% מהמעבדות בהן עובד העובד בעל המספר עובד eomployeeID אשר מספרי העובד שלהם הקטנים ביותר.