מיני פרויקט בבסיסי נתונים

פרויקט: בסיס נתונים עבור בנק

היחידה הנבחרת: מידע ומשכורות של עובדים

מגישים:

אלי ארזי 336301304

אלעד רדומסקי 318645850

***תוכן עניינים:***

*תאור המערכת...................................................3*

..............................................................4ERD

..............................................................4DSD

יצירת טבלאות..................................................5

.............................................6-7Data generator

DESC............................................................8-9

גיבוי ושחזור......................................................10

***תיאור המערכת:***

בפרויקט הזה אנו בונים בסיס נתונים עבור אגף השכר וכוח אדם בבנק.

לבנק ישנם מספר סניפים ולכל סניף עובדים, העובדים מחולקים למנהלים ופקידים ולכן יש יחס של ירושה, לכל סניף יש 10 מחלקות פנימיות.

המערכת פועלת כך:

ישנו עובד, לעובד יש שתי ישויות שיורשות ממנו, עובד יכול להיות מנהל או פקיד, לכל עובד יש משכורת, כמו כן כל עובד עובד בסניף ולכל סניף יש מחלקות (הלוואות, ייעוץ השקעות...).

דוגמא להמחשה:

עובד מסוג פקיד: יעקב

עובד מסוג פקיד: יצחק

עובד מסוג מנהל: אברהם

עובד מסוג מנהל : משה

עובד מסוג מנהל: אהרון

שכר של יעקב 8k

שכר של יצחק 8k

שכר של אברהם 15k

שכר של משה: 15k

שכר של אהרון: 13k

יעקב עובד בסניף: ירוחם כפקיד במחלקת ייעוץ השקעות

משה עובד בסניף: ירוחם כמנהל הסניף

יצחק עובד בסניף: תל אביב כפקיד במחלקת הלוואות

אברהם עובד בסניף: תל אביב, כמנהל הסניף

אהרון עובד בסניף: תל אביב, כמנהל מחלקת הלוואות

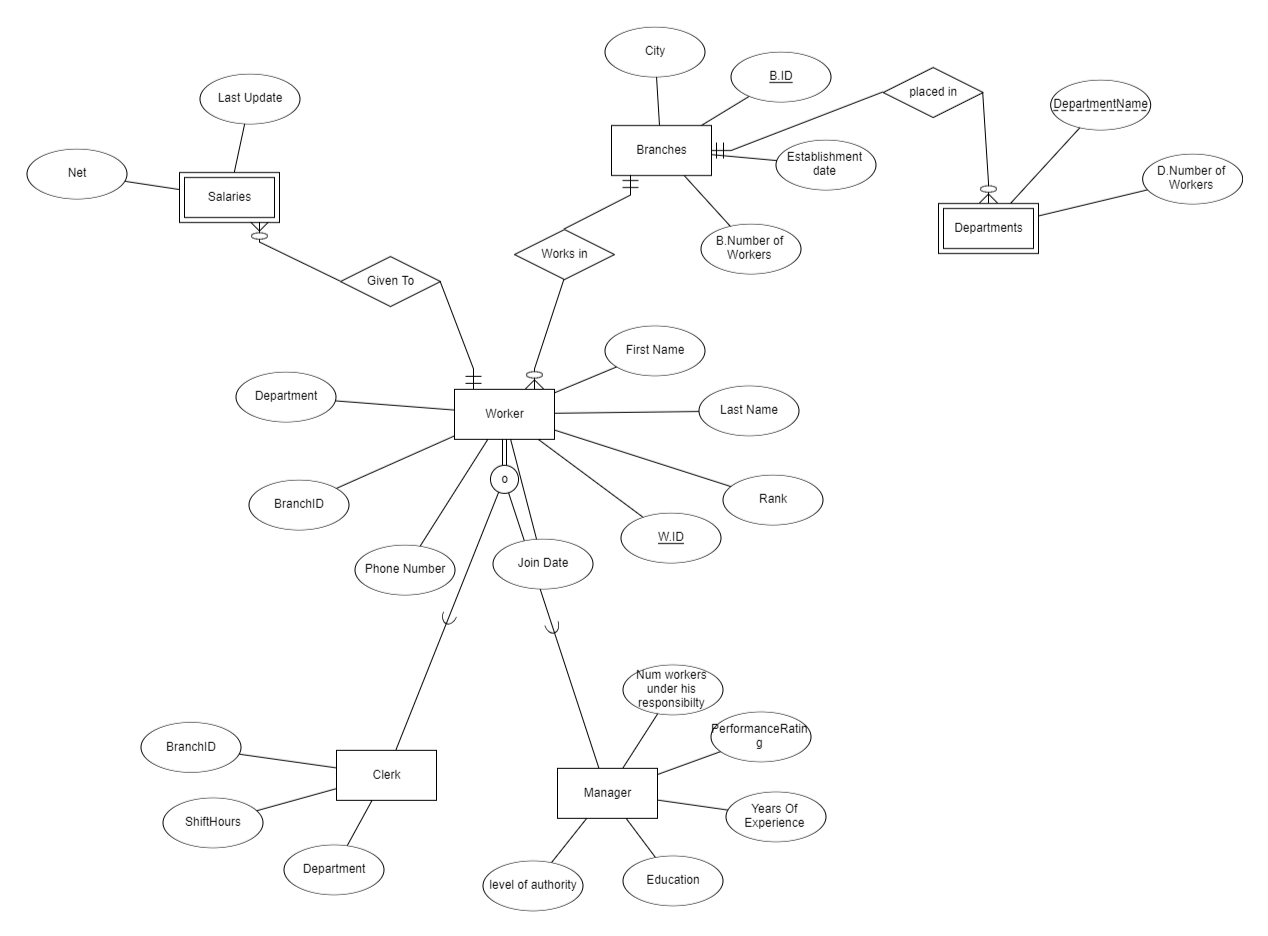
מנהל סניף תל אביב: אברהם

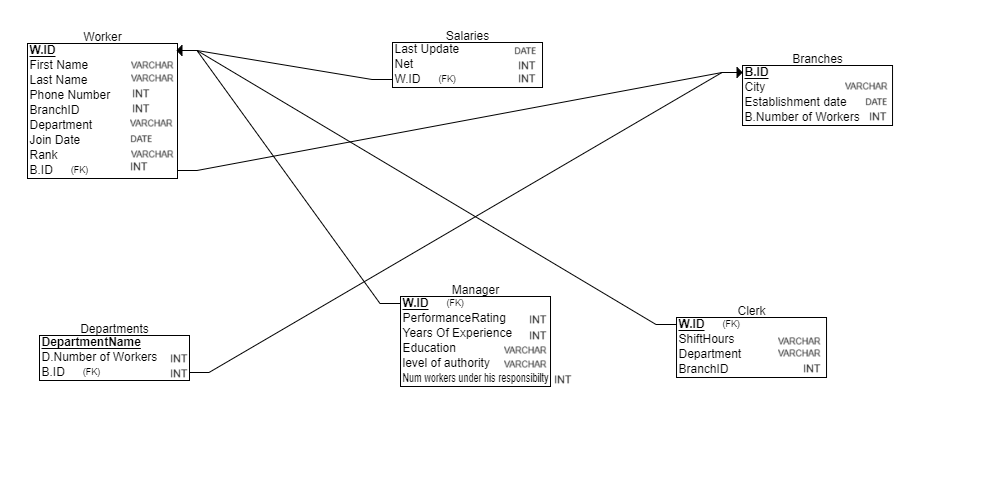
מנהל סניף ירוחם: משה

במילים אחרות ישנם שתי סניפים תל אביב וירוחם, מנהל סניף תל אביב הוא אברהם ומנהל סניף ירוחם הוא משה, לסניף תל אביב במחלקת הלוואות אהרון הוא המנהל, ויצחק עובד כפקיד במחלקה זו, בסניף ירוחם יעקב עובד כפקיד במחלקת ייעוץ השקעות.

המערכת שומרת על מאגר העובדים והשכר שלהם.

**ERD**



**DSD**

**Create Tables**

CREATE TABLE `branches` (  
 `BranchID` int(11) NOT NULL AUTO\_INCREMENT,  
 `City` varchar(20) NOT NULL,  
 `EstablishmentDate` date NOT NULL,  
 `NumberOfWorkers` int(11) NOT NULL,  
 PRIMARY KEY (`BranchID`)  
)

CREATE TABLE `clerk` (  
 `WorkerID` int(11) NOT NULL,  
 `Department` varchar(20) NOT NULL,  
 `BranchID` int(11) NOT NULL,  
 `ShiftHours` int(11) NOT NULL,  
 KEY `WorkerID` (`WorkerID`),  
 KEY `BranchID` (`BranchID`),  
 KEY `Department` (`Department`),  
 CONSTRAINT `clerk\_ibfk\_1` FOREIGN KEY (`WorkerID`) REFERENCES `worker` (`WorkerID`),  
 CONSTRAINT `clerk\_ibfk\_2` FOREIGN KEY (`BranchID`) REFERENCES `branches` (`BranchID`),  
 CONSTRAINT `clerk\_ibfk\_3` FOREIGN KEY (`Department`) REFERENCES `departments` (`DepartmentName`)

(

CREATE TABLE `departments` (  
 `DepartmentName` varchar(20) NOT NULL,  
 `BranchID` int(11) NOT NULL,  
 `NumberOfWorkers` int(11) NOT NULL,  
 PRIMARY KEY (`DepartmentName`,`BranchID`),  
 KEY `BranchID` (`BranchID`),  
 CONSTRAINT `departments\_ibfk\_1` FOREIGN KEY (`BranchID`) REFERENCES `branches` (`BranchID`) ON UPDATE CASCADE  
)

CREATE TABLE `manager` (  
 `WorkerID` int(11) NOT NULL,  
 `NumWorkersUHR` int(11) NOT NULL,  
 `PreformanceRating` int(11) NOT NULL,  
 `YearsOfExperience` int(11) NOT NULL,  
 `Education` varchar(50) NOT NULL,  
 `LevelOfAuthority` varchar(50) NOT NULL,  
 PRIMARY KEY (`WorkerID`),  
 CONSTRAINT `manager\_ibfk\_1` FOREIGN KEY (`WorkerID`) REFERENCES `worker` (`WorkerID`)

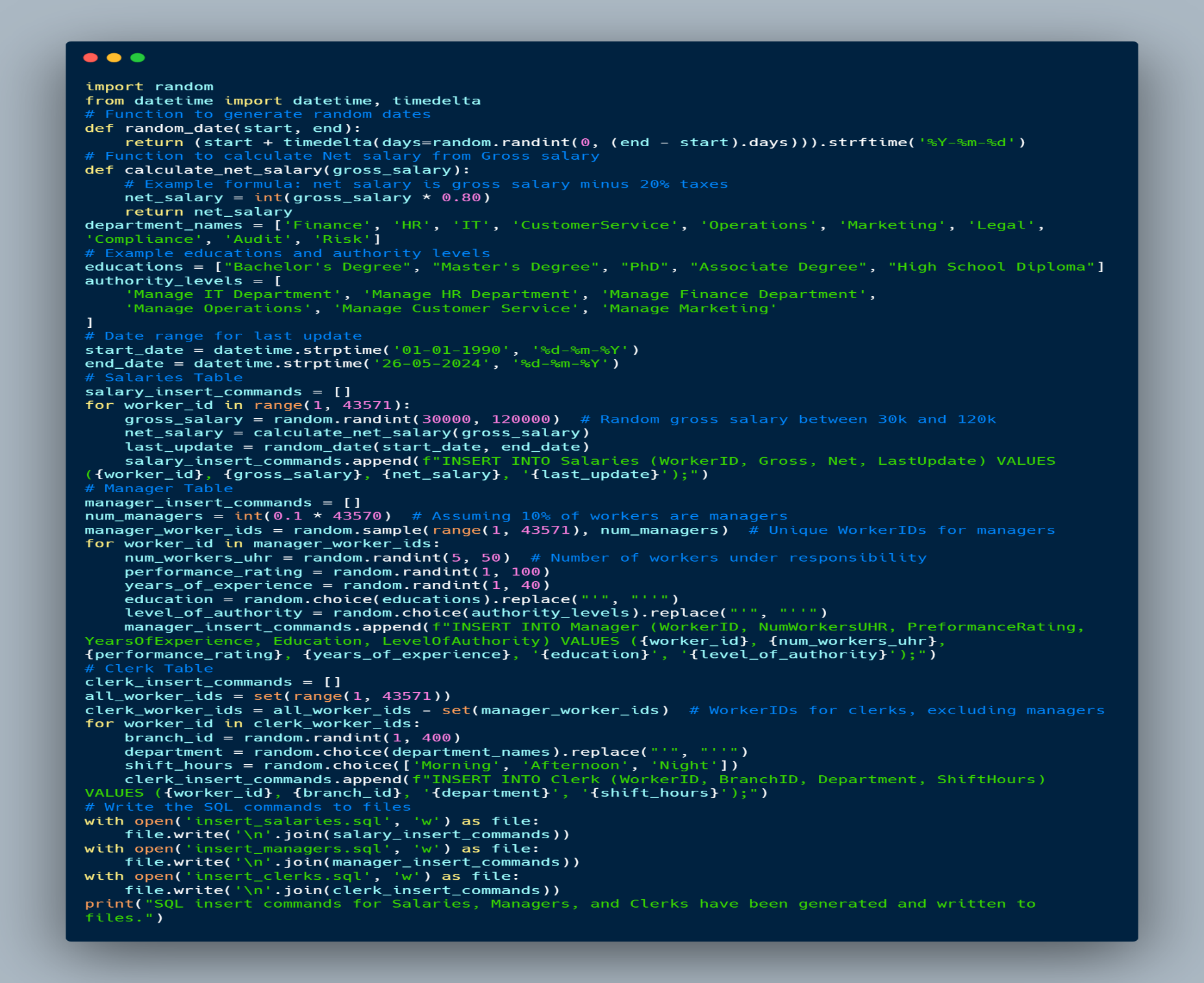
)

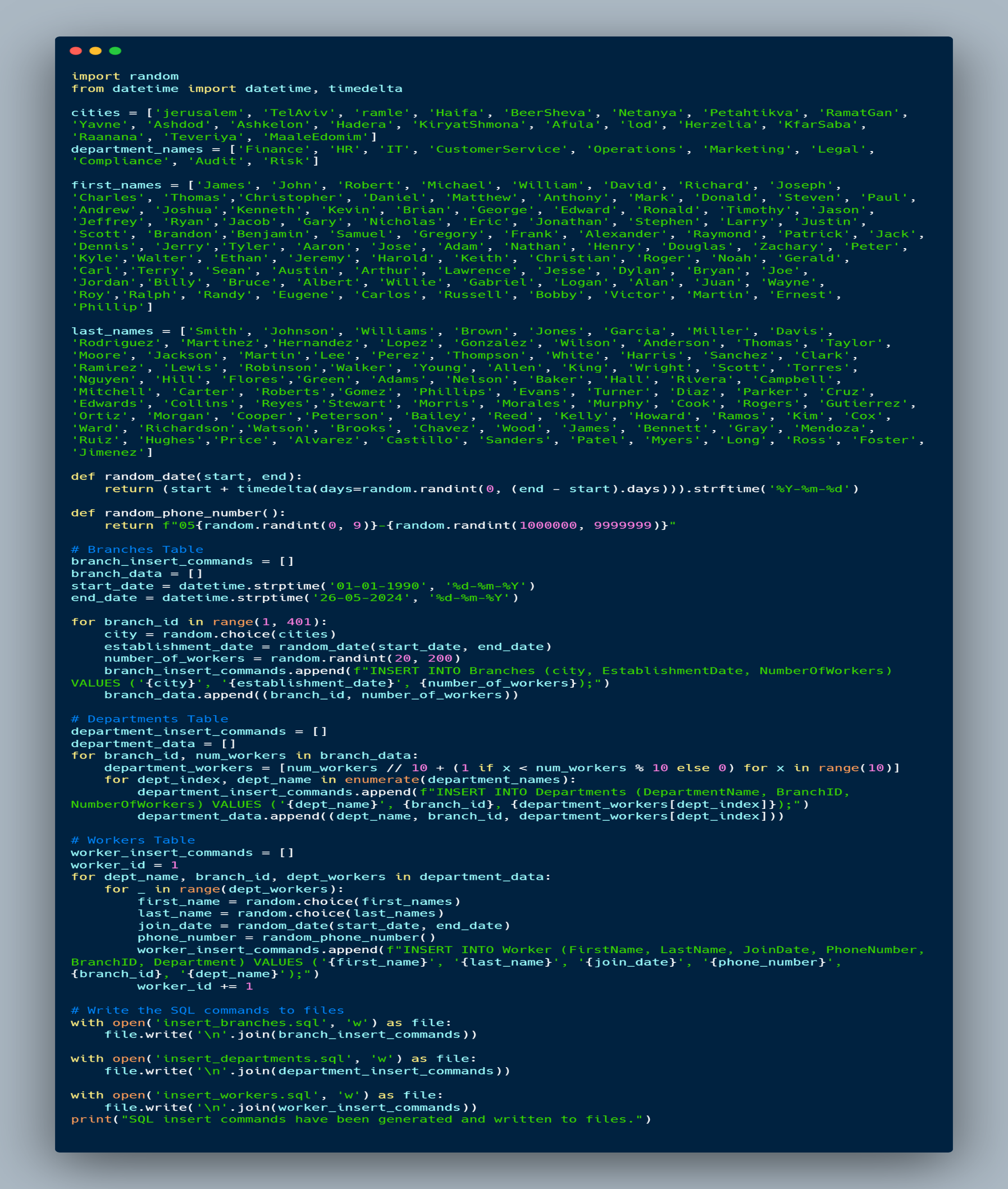
CREATE TABLE `salaries` (  
 `WorkerID` int(11) NOT NULL,  
 `LastUpdate` date NOT NULL,  
 `Net` int(11) NOT NULL,  
 PRIMARY KEY (`WorkerID`),  
 CONSTRAINT `salaries\_ibfk\_1` FOREIGN KEY (`WorkerID`) REFERENCES `worker` (`WorkerID`) ON UPDATE NO ACTION  
)

CREATE TABLE `worker` (  
 `WorkerID` int(11) NOT NULL AUTO\_INCREMENT,  
 `FirstName` varchar(20) NOT NULL,  
 `LastName` varchar(20) NOT NULL,  
 `JoinDate` date NOT NULL,  
 `PhoneNumber` int(11) NOT NULL,  
 `BranchID` int(11) NOT NULL,  
 `Department` varchar(20) NOT NULL,  
 `Rank` varchar(20) NOT NULL DEFAULT 'Junior',  
 PRIMARY KEY (`WorkerID`)  
)

**Python script-data generator**

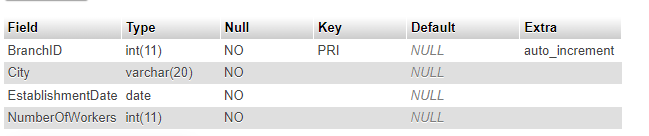
**Salaries, Manager and Clerk Tables:**



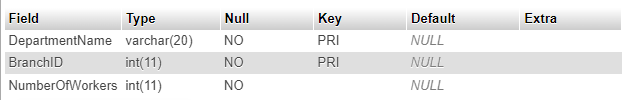
**Branches, Departments and Workers Tables:**

**DESC commands:**

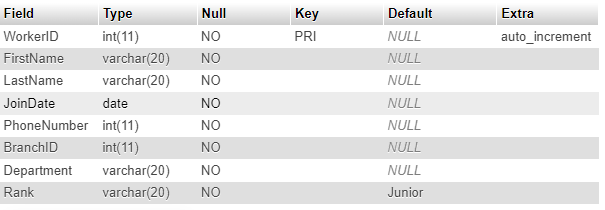
DESC branches;



DESC departments;



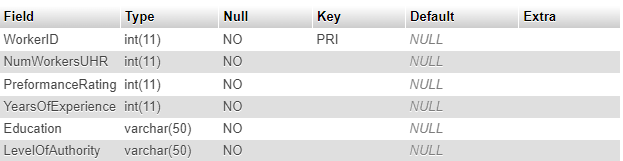
DESC worker;



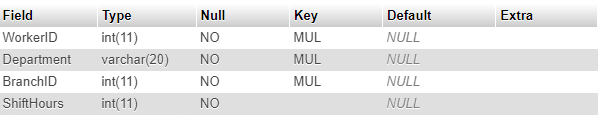
DESC salaries;



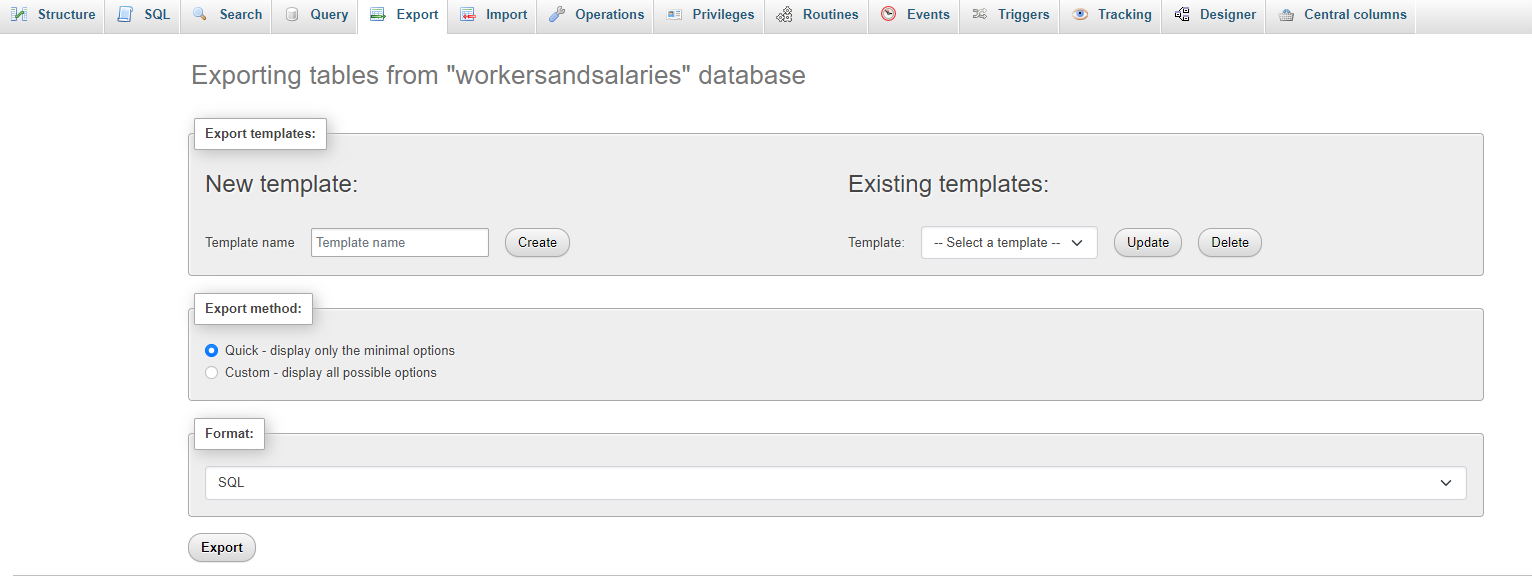
DESC manager;



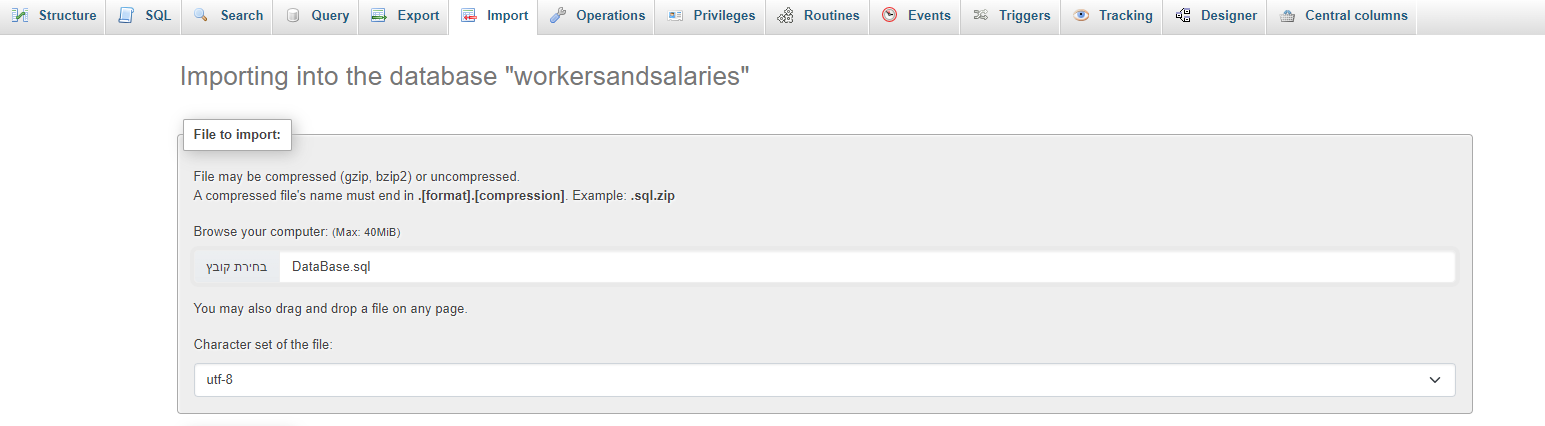
DESC clerk;



**גיבוי בסיס הנתונים:**

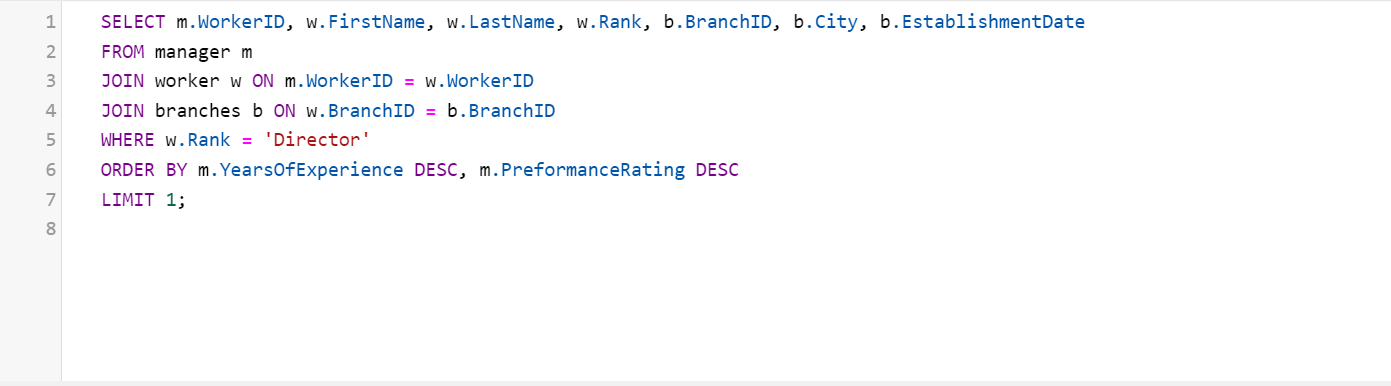
****

**שחזור בסיס הנתונים:**

****

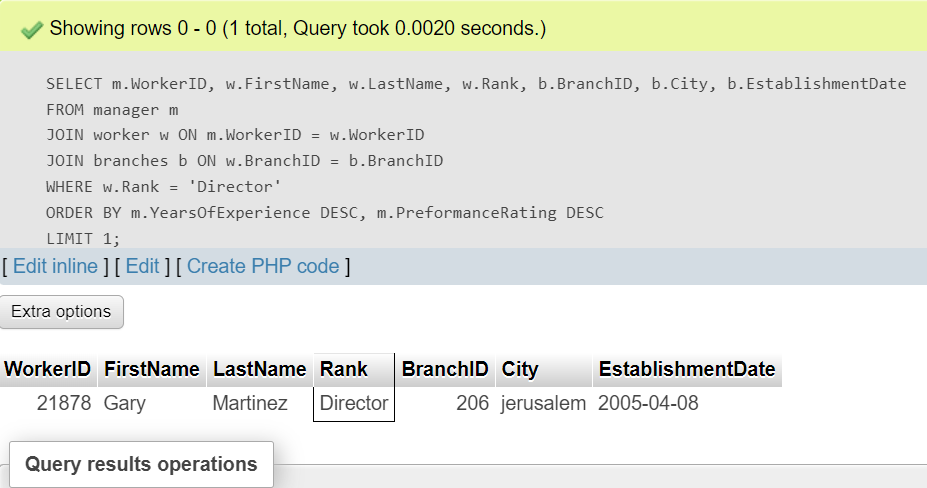
**שלב 2**

**שאילתת select 1:**

שאילתה שמחזירה את המנהל עם הנסיון והדירוג הכי גבוהים ופולטת מידע על הסניף שלו.

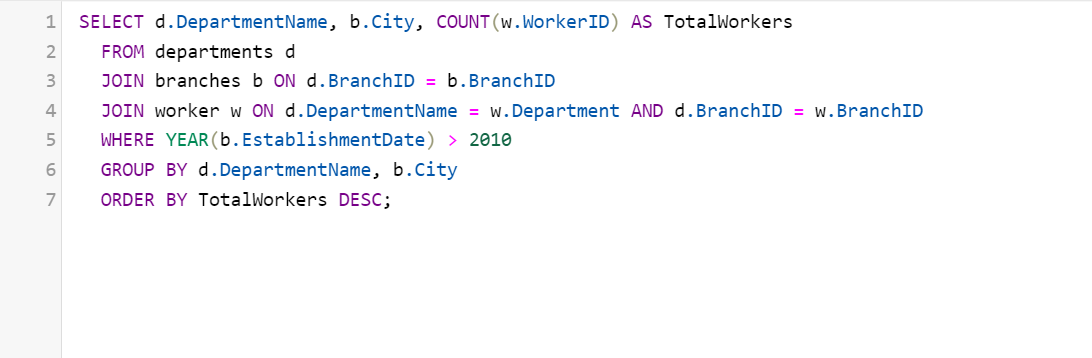
השאילתה מצרפת את הטבלה manager עם worker עי התאמת עמודת workerID, ואז מצרפת את worker עם branches בעזרת branchID, ואז מסננת לפי דרגה (rank), ומסדרת את שנות הניסיון בסדר יורד, במקרה של שוויון השאילתה מסדרת לפי דירוג (performance rating).

כלומר השאילתה הזאת מחפשת את פרטי המנהל עם מס הכי גבוה של שנות נסיון עם הדירוג הגבוה ביותר ועם הדרגה הכי גבוהה (director) השאילתה מחזירה את הפרטים של אותו מנהל כולל מידע על המחלקה בו הוא עובד.

**הרצה על בסיס הנתונים:** 

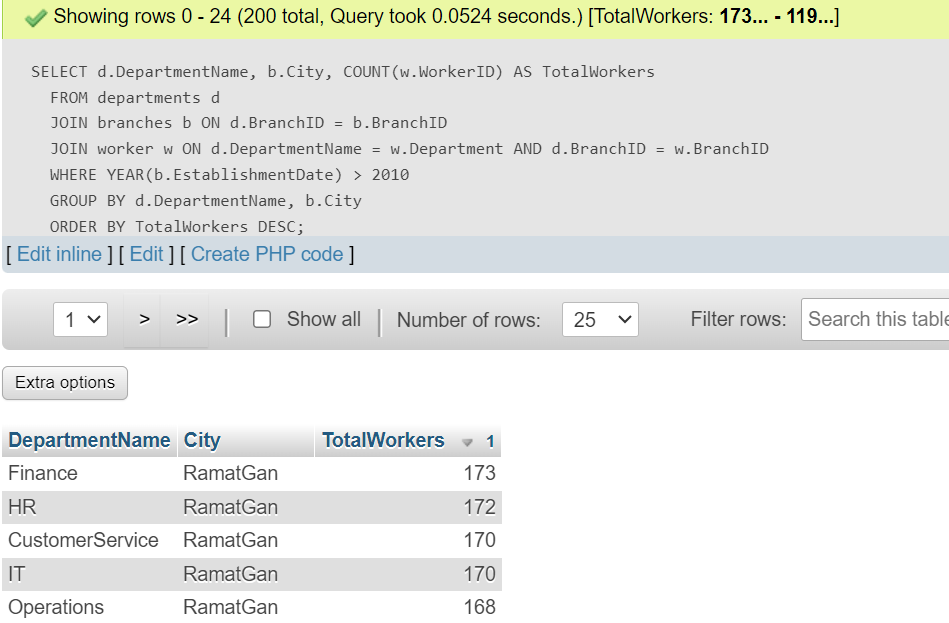
**שאילתת select 2:**

שאילתה המחזירה את מספר העובדים הכולל בכל מחלקה ואת העיר בו היא ממוקמת - עבור סניפים שהוקמו לאחר 2010.



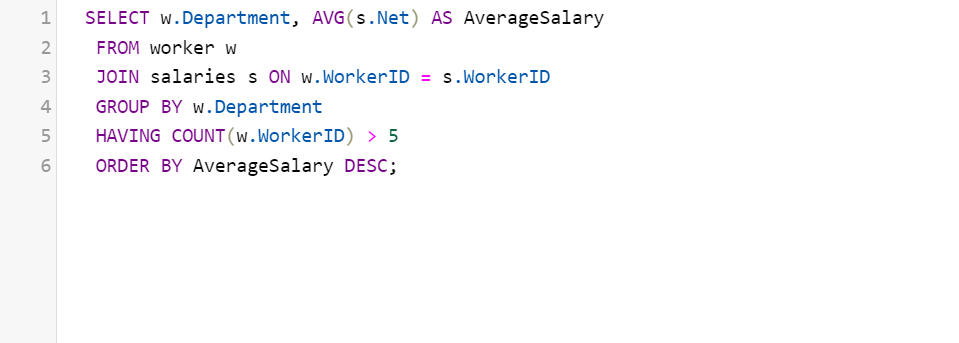
השאילתה לוקחת שם מחלקה, עיר, וסופרת כמה עובדים יש בכל מחלקה וקוראת לסכום TotalWorkers, השאילתה מצרפת את טבלת הסניפים עם טבלת המחלקות על בסיס departmentName ו branchID, ואז מצרפת את טבלת המחלקות עם טבלת העובדים על בסיס departmentName ו branchID, לאחר מכן יש פילטר שלוקח רק את הסניפים שנוסדו לאחר 2010, מקבצת את שם הסניף לעיר ומסדרת בסדר יורד על בסיס מס העובדים.

**הרצה על בסיס הנתונים:**



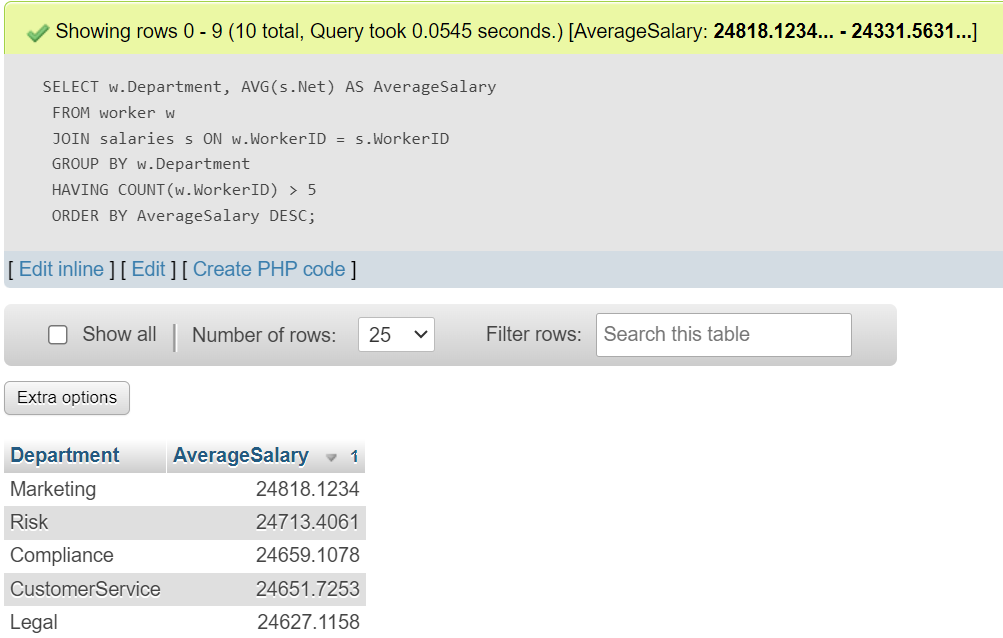
**שאילתת select 3:**

שאילתה שמוצאת את השכר הממוצע לכל מחלקה, עבור מחלקות עם יותר מ5 עובדים.



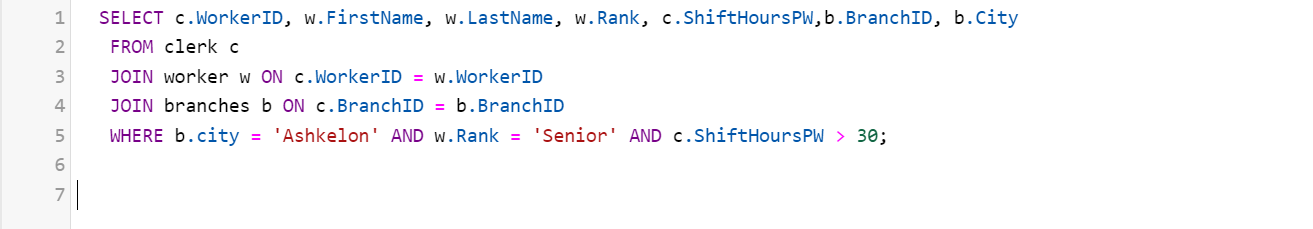
השאילתה לוקחת את עמודת המחלקות מטבלת עובד, ומחשבת את הממוצע מטבלת משכורת וקוראת לזה average salary, לאחר מכן יש איחוד בין טבלת עובד לטבלת משכורת על בסיס workerID, מקבצים על בסיס מחלקה, ישנו פילטר שלוקח רק את הקבוצות עם יותר מ5 עובדים לבסוף מסדרים הכל בסדר יורד על בסיס המשכורת הממוצעת.

**הרצה על בסיס הנתונים:**



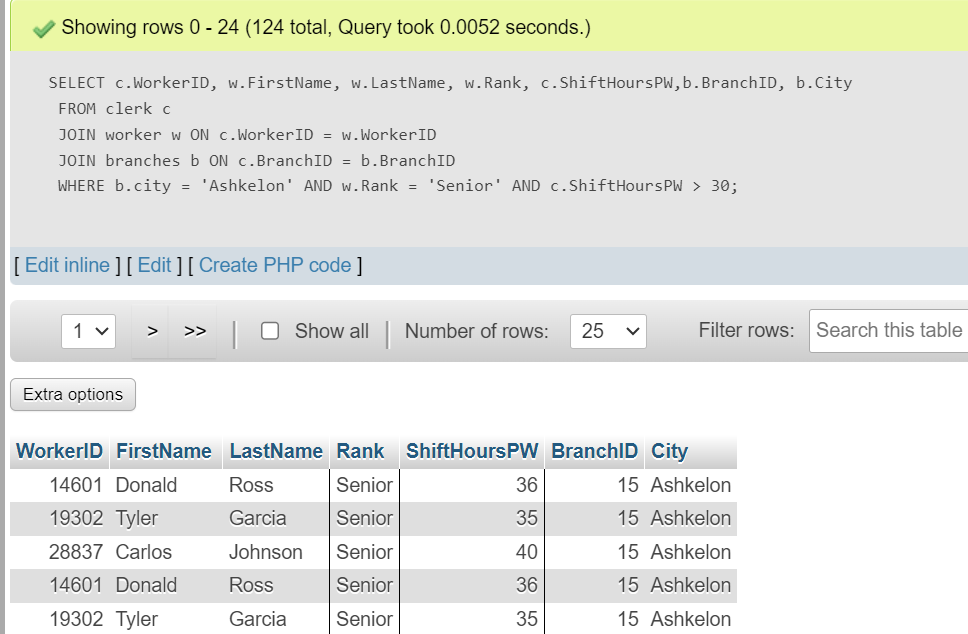
**שאילתת select 4:**

שאילתה שמחפשת ומחזירה את הפרטים של כל הפקידים בדרגת Senior שעובדים באשקלון ויש להם משמרות של יותר מ30 שעות בשבוע.



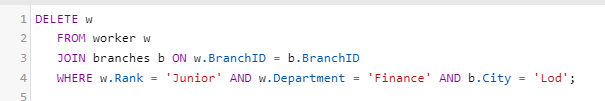
לוקחים את עמודת תז של עובד מטבלת עובד פקיד, לוקחים שם שם משפחה ודרגה של עובד מטבלת עובד, לוקחים את שעות משמרת מטבלת עובד פקיד, את הסניף ואת העיר מטבלת סניפים, לאחר מכן מצרפים את טבלת עובד פקיד עם טבלת עובד על בסיס workerID , ומצרפים את הטבלה של עובד פקיד עם טבלת סניפים על בסיס BranchID, לאחר מכן יש פילטר שלוקח רק את הערכים שהעיר היא אשקלון, שהדרגה היא senior ועם משמרת של יותר מ30 שעות בשבוע.

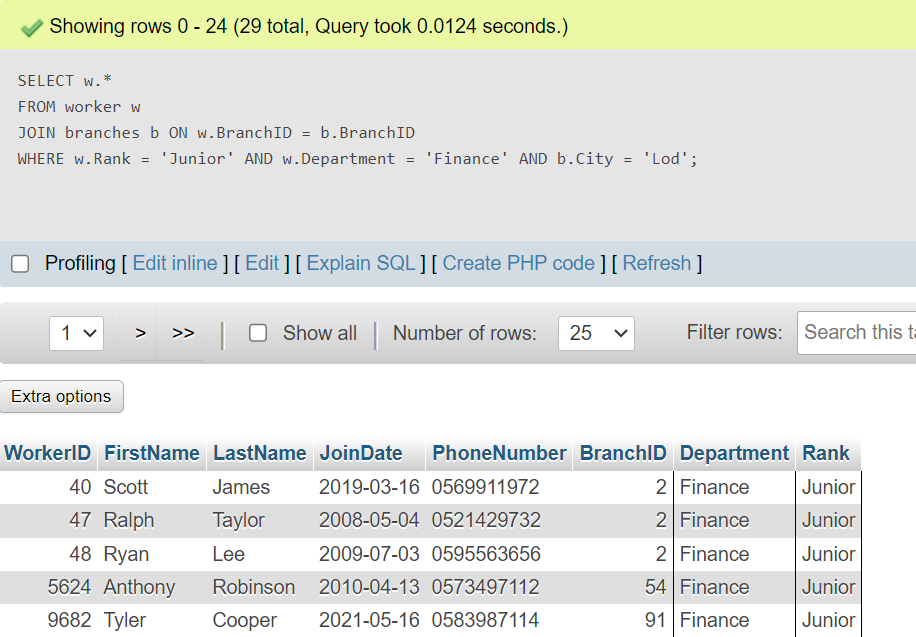
**הרצה על בסיס הנתונים:**



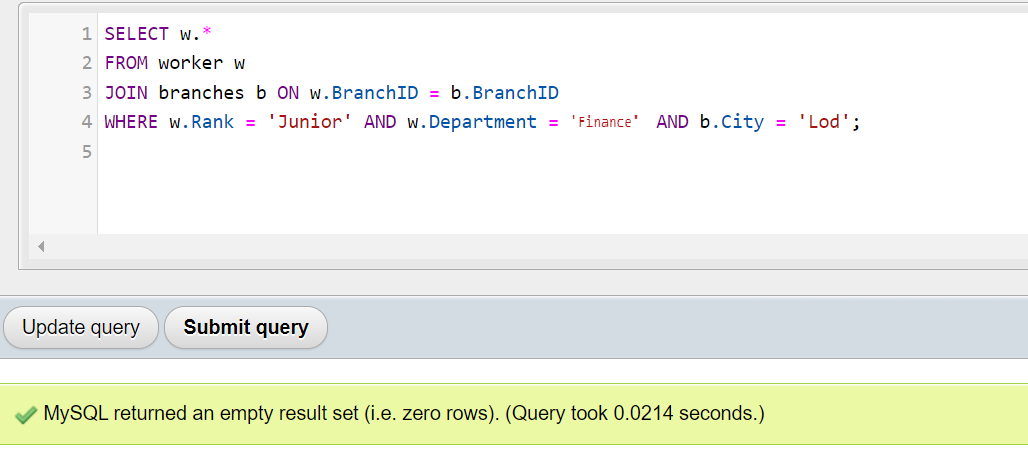
**שאילתת delete 1:**

שאילתה שמוחקת את כל העובדים שדרגתם Junior במחלקת פיננסים בעיר לוד.





לפני ביצוע הפקודה:



לאחר ביצוע

הפקודה:

ניתן לראות

כי לא קיימים

יותר עובדים

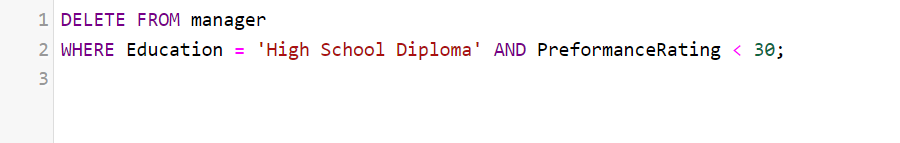
בדרגת Junior

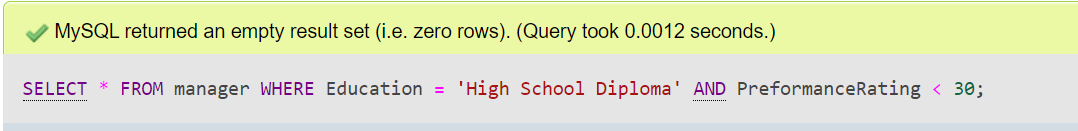
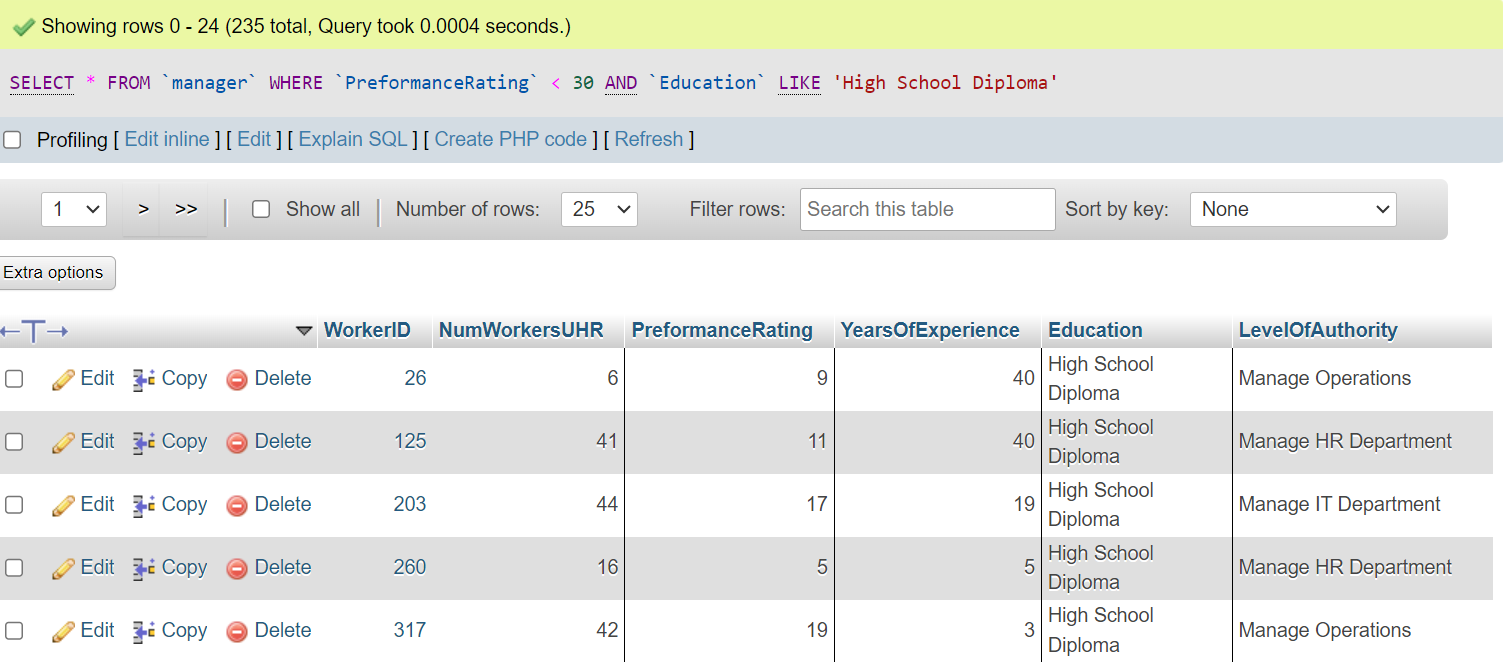
במחלקת פיננסים בסניף

לוד.

**שאילתת delete 2:**

שאילתה שמוחקת את כל המנהלים עם השכלה תיכונית ועם דירוג נמוך מ30.

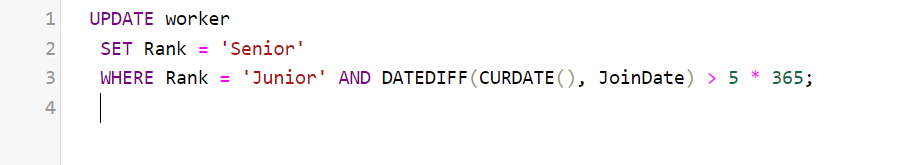


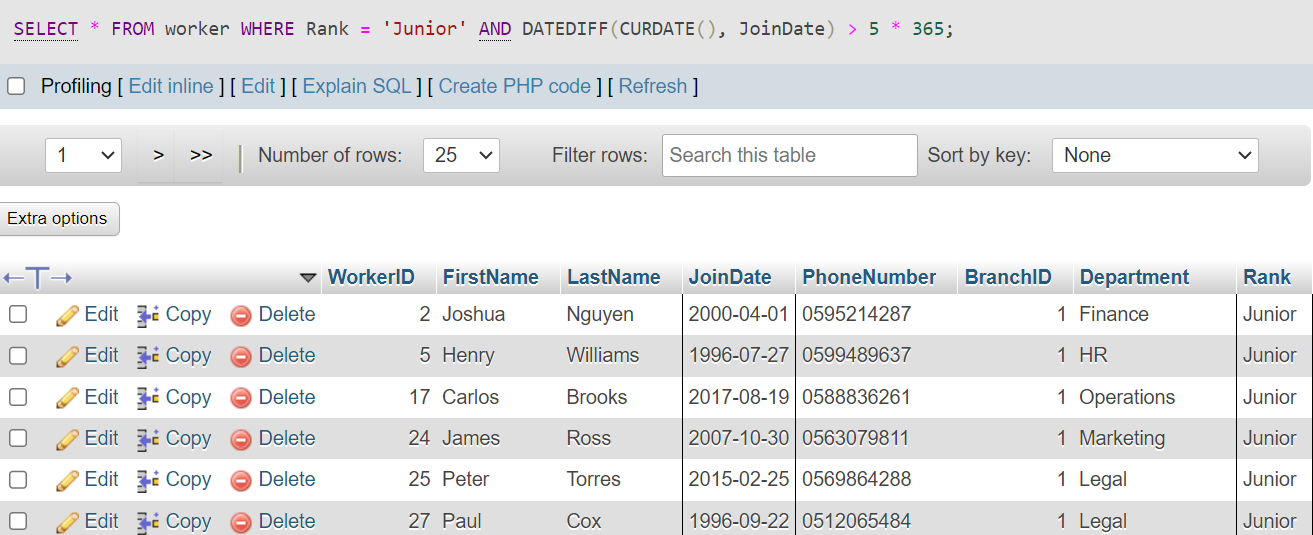
לפני ביצוע הפקודה:

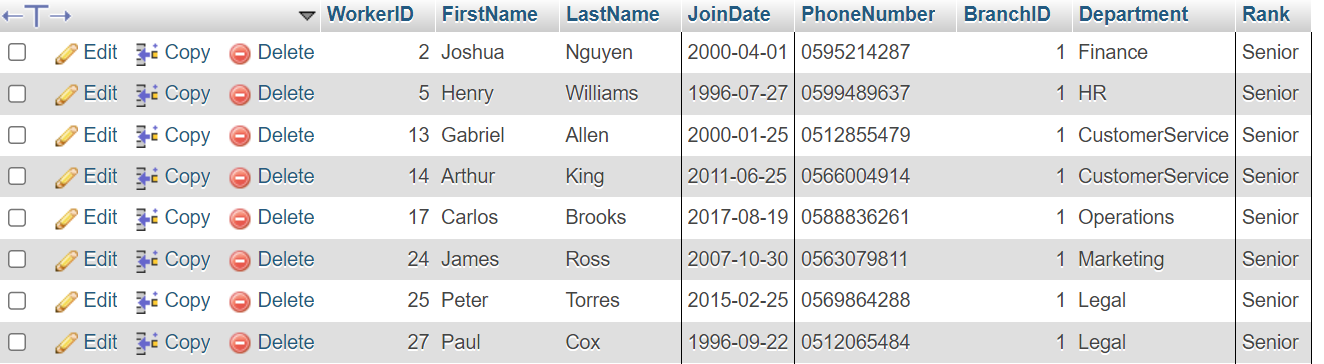
לאחר ביצוע הפקודה:

**שאילתת update 1:**

שאילתה שמקדמת את כל העובדים בדרגת junior לדרגת senior בתנאי שהם עובדים בבנק יותר מ5 שנים.

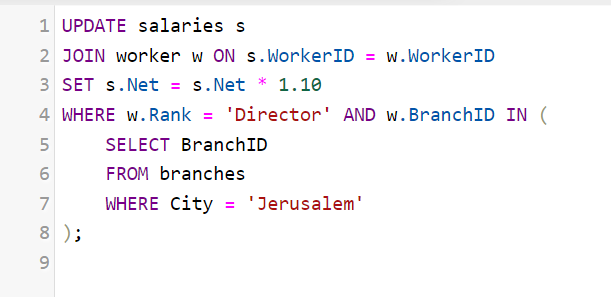


טבלה לפני ההרצה:

 לאחר ההרצה:

**שאילתת update 2:**

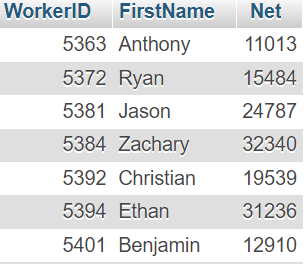
שאילתה שמעלה שכר לכל העובדים שהם directors בעשר אחוז, בתנאי שהסניפים שלהם בירושלים.



השאילתה לוקחת את הדרגה מטבלת עובד, ואת מספר הסניף, השאילתה בודקת מה מספרי הסניפים שנמצאים בירושלים, השאילתה מוצאת את השכר של כל עובד מטבלת השכר בעזרת צירוף של טבלת עובד ושכר על בסיס תז, ומעלה את השכר ב 10%.

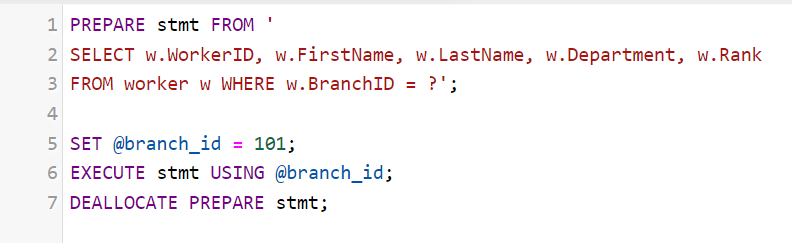
אחרי: לפני:



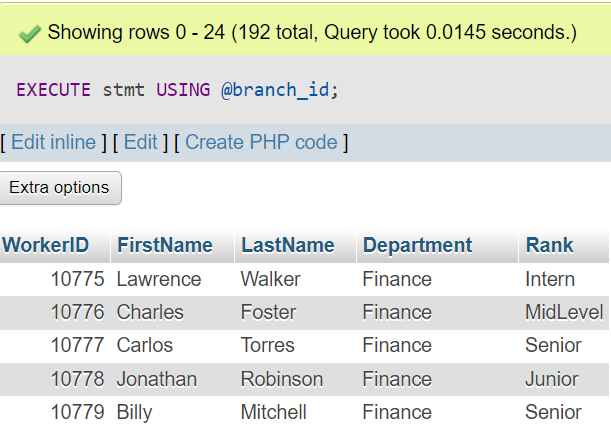


**שאילתה 1 עם פרמטר:**

השאילתה משתמשת בפרמטר אחד BranchID, היא בוחרת נתונים של עובדים מסניף מסוים, בשאילתה הזו 101.

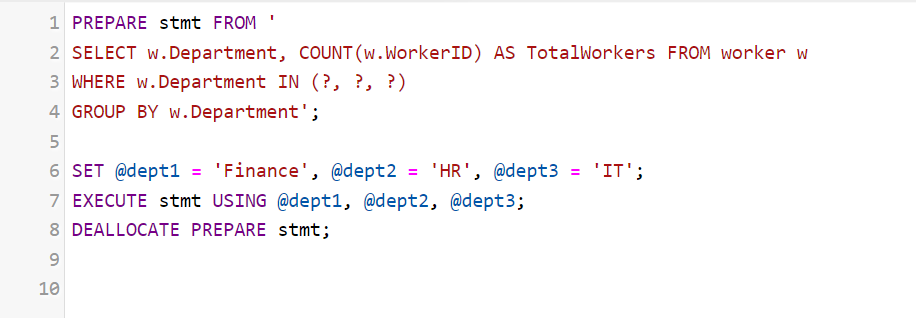


ראשית השאילתה מכינה את ה״שאילתה״ ונותנת את זה לhandler בשם ,stmt במקום המתאים עבור branchID ישנו סימן שאלה וזה בעצם מחזיק מקום (placeholder).

לאחר מכן אנu נותנים לערך branchesID את הערך שאנחנו רוצים במקרה הזה 101, ואז אומרים לשאילתה להפעיל את עצמה עם הערך של branchID שהגדרנו, השאילתה תופעל כאשר branchID = 101, לבסוף משחררים את stmt.

**שאילתה 2 עם פרמטר:**

השאילתה מקבלת רשימה בתור פרמטר ומחזירה את מספר העובדים במחלקות הללו ומקבצת את הפלט לפי מחלקה.



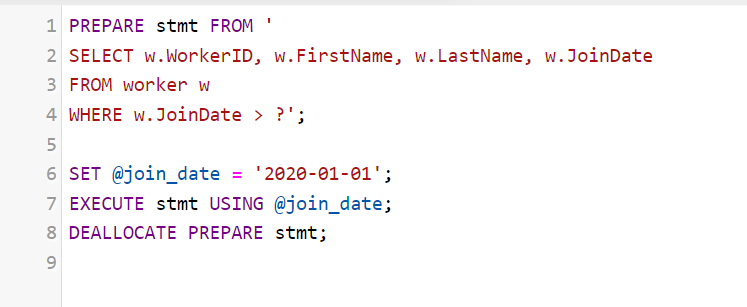
השאילתה ראשית מכינה את ה״שאילתה״ והכל מוחזק על ידי stmt, השאילתה הזו לוקחת רשימה של מחלקות וסופרת את סכום העובדים במחלקה, הפלט מקובץ לפי מחלקה, לאחר מכן מגדירים משתנים עם ערכים כל ערך יהיה מחלקה, ואז מפעילים את stmt עם הערכים שהגדרנו, בסוף מוחקים את המחזיק מקום (כדי לשחרר מקום).

הצבנו את הערכים "Finance" , "HR" , "IT" בפרמטרים והרצנו



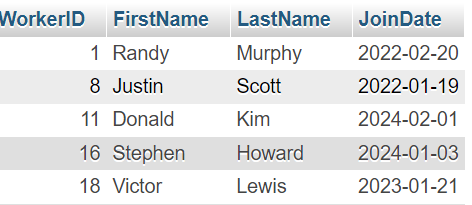
**שאילתה 3 עם פרמטר:**

השאילתה מקבלת כפרמטר ערך של תאריך, השאילתה מחזירה פרטים של עובד (תז שם ותאריך הצטרפות) עבור כל העובדים שהצטרפו לאחר התאריך שקיבלה.



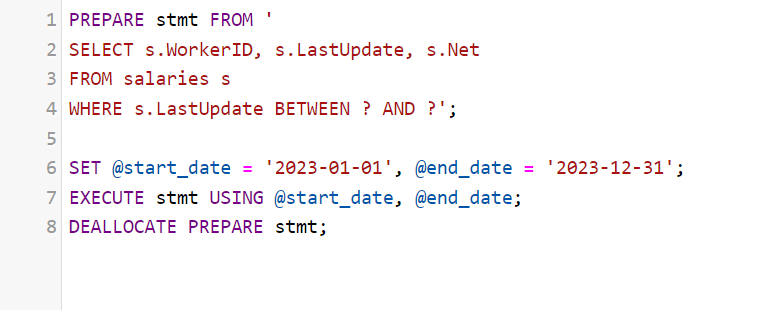
ראשית השאילתה מכינה את ה״שאילתה״ שלוקחת מידע של עובד ובמקום של joindate יש סימן של גדול מ? והמשתנה stmt מחזיק אותה, לאחר מכן מגדירים משתנה שמחזיק תאריך, ואז מפעילים את השאילתה עם המשתנה שהגדרנו, והיא תחזיר את כל העובדים שהצטרפו לארגון לאחר התאריך הנתון.

הצבנו את התאריך 1 לינואר 2020 בפרמטר והרצנו:



**שאילתה 4 עם פרמטר:**

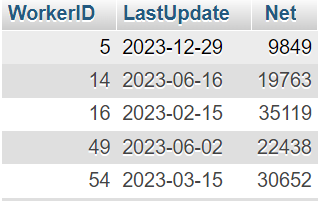
השאילתה הזו מקבלת כפרמטר טווח של תאריכים (ת.התחלה ת.סוף) ופולטת תז, תאריך עדכון אחרון, ונטו של משכורת שעודכנו בטווח הנ"ל, כאשר תאריך עדכון אחרון צריך להיות בטווח תאריכים שקיבלנו כפרמטר.



השאילתה מכינה את ה״שאילתה״ שלוקחת תז תאריך עדכון אחרון ומשכורת מטבלת משכורת, כאשר תאריך עדכון אחרון צריך להיות בין שתי סימני שאלה שהם פרמטרים את כל זה היא שמה במחזיק מקום בשם stmt.

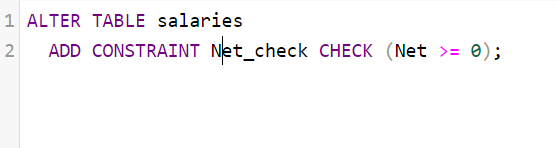
לאחר מכן השאילתה מגדירה שתי תאריכים ונותנת להם ערכי תאריך, ואז stmt מופעל עם שתי הפרמטרים שיוצבו במקום סימני השאלה, והפלט יהיה פרטי עובד שהמשכורת שלו עודכנה בין שתי תאריכים נתונים.

הצבנו את התאריך 1 לינואר 2023 כתאריך התחלה ו 31 לדצמבר 2023 כתאריך סוף בפרמטר והרצנו:

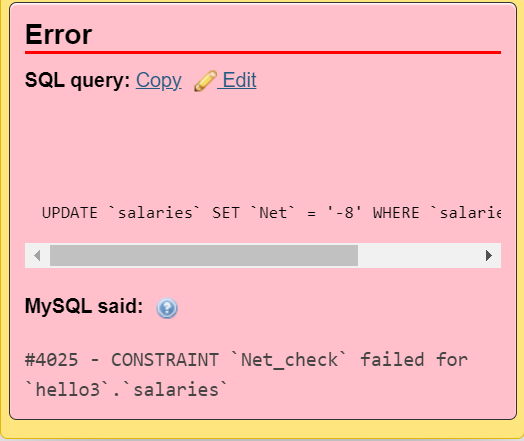


**אילוץ 1:**

אילוץ שכל ערכי השכר יהיו גדולים או שווים מ-0.

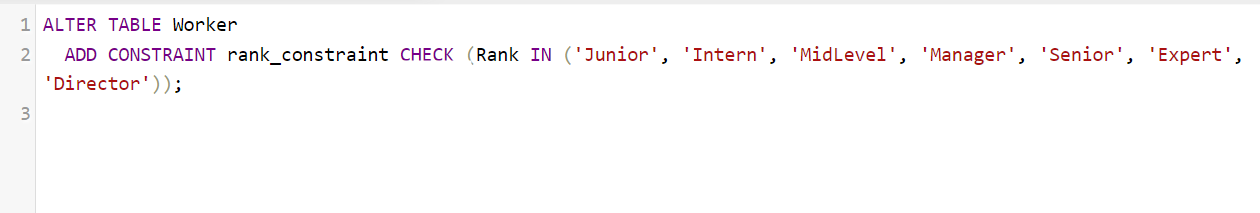


נסיון של להכניס ערך -8 לשכר נפגש בשגיאה

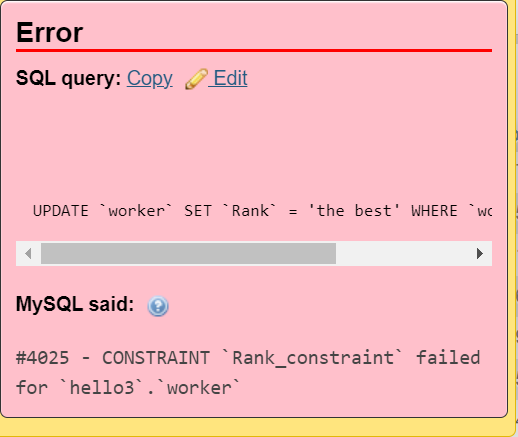


**אילוץ 2:**

אילוץ שהערך של Rank יהיה ערכים מסויימים בלבד, כלומר אם נכניס עובד בדרגה שהיא לא מבין הדרגות הספציפיות נקבל שגיאה.

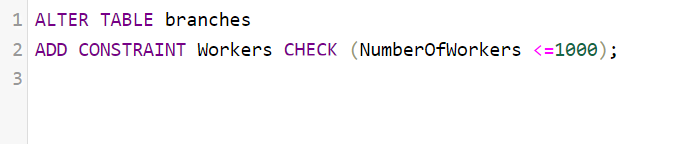


נסיון של הכנסת ערך דרגה the best נתקל בשגיאה:



**אילוץ 3:**

מאלץ שלכל סניף יהיה מקסימום של 999 עובדים.

****

נסיון להכניס ערך 10000 במספר העובדים בסניף נתקל בשגיאה:

