

# דו"ח פרויקט בבסיס נתונים

מגישים: מאיר קלמפנר ונתנאל בשן

תאריך: 08/06/2022

גרסה: 1.0



## תוכן עניינים

2	תוכן עניינים
3	אפיון
5	אכלוס טבלאות
6	שאלות
6	שאלות טריוויאליות
6	שאלות לא טריוויאליות
8	אינדקסים
10	אינטגרציה
10	שאלות האינטגרציה
11	Views
12	גרפים
13	תכנות PL/SQL
13	פרוצדורות
14	פונקציות
16	נספח
16	הקוד ליצירת הטבלאות
16	הקוד לאכלוס הטבלאות וקבצי CSV
16	הקוד של השאלות
18	הקוד של האינדקסים
19	הקוד של שאלות האינטגרציה
20	הקוד של Views

## אפיון

הנושא שנבחר בכיתה הינו ניהול בסיס נתונים של מוזיאון. והישויות שבחרנו הן: אגף, מחלקה ועובד.

החלטנו שיש היררכיה בין הישות מחלקה לבין אגף, כך שלכל אגף יש כמה מחלקות, בעצם קשר יחיד לרבים.

בנוסף החלטנו שלכל עובד יש כמה מחלקות בהן הוא עובד ולכן יש קשר רבים לרבים בין עובד לבין מחלקה.

להלן התכונות של הישויות:

אגף:

- מספר אגף (מספר רץ)
- שם האגף

מחלקה:

- מספר מחלקה (מספר רץ)
- שם המחלקה
- מספר האגף (מפתח של המחלקה אגף)

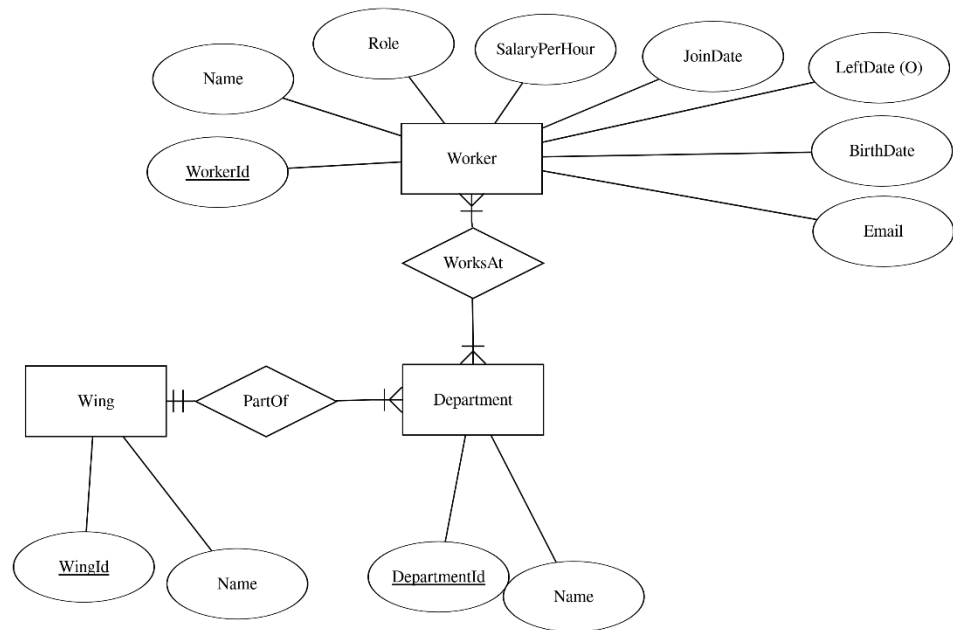
עובד:

- מספר עובד (מספר רץ)
- שם העובד
- תפקיד
- שכר לשעה
- תאריך הצטרפות
- תאריך עזיבה (יכול להיות NULL במקרה ועדיין עובד)
- תאריך לידה
- אימייל

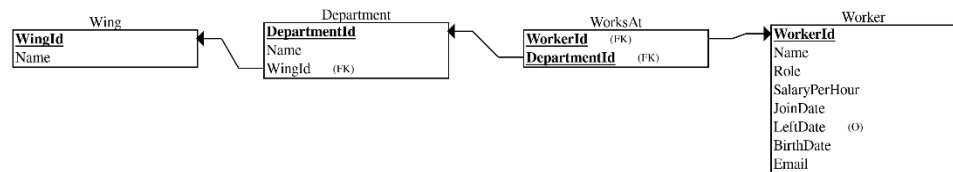
עובד במחלקה:

- מספר עובד (מפתח של המחלקה עובד)
- מספר מחלקה (מפתח של המחלקה אגף)

להלן הגרף ERD שהכנו:



להלן הגרף הרציונלי שנוצר מהגרף הקודם, כאן רואים את הטבלאות והמפתחות:



את הקוד שנוצר בשביל ליצור את הטבלאות אפשר לראות [בנספח](#).

## אכלוס טבלאות

השתמשנו בpython בשביל להכין את scripts וגם השתמשנו בספריות faker ו random. הסקריפטים שלנו יצרו קובץ csv כיוון שרצינו להכניס הרבה ישויות והדרך המהירה והפשוטה ביותר שמצאנו זה באמצעות csv.

מספר הנתונים שנוצרו:

- עובדים: 20,911
- מחלקות: 1,002
- אגפים: 400
- עובדים במחלקות: 31,291

בעיות:

- היינו צריכים לגרום לכך שהעובדים גדולים מגיל 18.
- היינו צריכים לבדוק שהעובד התחיל לעבוד אחרי גיל 21 וסיים לעבוד לפני גיל 70.
- היינו צריכים לגרום שהעובד עזב אחרי שהצטרף לעבודה.
- היינו צריכים להזין את הנתונים של המחלקות לאחר שהכנסנו את הנתונים של האגפים, כיוון שיש בטבלה של המחלקות מפתח של האגפים.
- היינו צריכים להזין את הנתונים של העובדים במחלקות לאחר שהכנסנו את הנתונים של העובדים ושל המחלקות, כיוון שיש בטבלה של העובדים במחלקות מפתחות של העובדים ושל המחלקות.

את הקוד שנוצר בשביל ליצור את הטבלאות אפשר לראות ב[נספח](#).

## שאלות

### שאלות טריוויאליות

1. כל העובדים שעובדים הרבה זמן (30 שנה ומעלה) ומקבלים שכר נמוך (פחות מ100 לשעה).

**שימוש:** משאבי אנוש החליטו שהעובדים שעובדים כבר מעל 30 שנה הם נכס חשוב למוזיאון, לכן הם התחילו תהליך בעזרת הנהלת החשבונות להעלות את השכר של כל העובדים האלה שיהיה לפחות 100 לשעה.

**התכונות החוזרות:** מספר העובד, שם העובד, שכר לשעה, תאריך הצטרפות ואימייל.  
**מספר השורות:** 113.

**זמן הריצה:** 0.036 שניות.

2. ממוצע ההוצאות החודשיות על שכר העובדים.

**שימוש:** הנהלת חשבונות רוצים לדעת את ההוצאות החודשיות של שכר העובדים בחודש ממוצע (20 ימים בחודש עם 8 שעות בכל יום). על מנת לעשות את זה, הם צריכים את השאלתה הזאת.

**התכונות החוזרות:** מספר העובד, שם העובד, שכר לשעה, תאריך הצטרפות ואימייל.  
**מספר השורות:** 1.

**זמן הריצה:** 0.019 שניות.

3. תפקידים של עובדים ומספר העובדים שעובדים בהם (גם ברגע הנתון וגם לכל אורך השנים)

**שימוש:** משאבי אנוש יוכלו כך לראות את התפלגות התפקידים של העובדים כרגע ולכל אורך השנים, וכך לדעת באיזה תפקיד יש יותר מדי עובדים ובאיזה תפקיד אין מספיק.  
**התכונות החוזרות:** תפקיד, מספר העובדים לכל אורך השנים, מספר העובדים כרגע.  
**מספר השורות:** 9.

**זמן ריצה:** 0.171 שניות.

4. מספר העובדים שמקבלים שכר (מעוגל ל10 הקרוב)

**שימוש:** הנהלת החשבונות יוכלו להשתמש בשאלתה הזאת על מנת לבדוק מה ההתפלגות של שכר העובדים במוזיאון.  
**התכונות החוזרות:** שכר, מספר העובדים.

**מספר השורות:** 10.

**זמן ריצה:** 0.023

### שאלות לא טריוויאליות

5. מספר העובדים שעובדים כרגע בכל מחלקה

**שימוש:** נתון למשאבי אנוש שיש מחלקות בהן יש יותר מדי עובדים ומחלקות שבהן חסר

עובדים. לכן על מנת שמשאבי אנוש יוכלו כך לבדוק באיזה מחלקות יש יותר מדי עובדים ובאיזה מחלקות חסר עובדים, הם יוכלו להשתמש בשאילתה הזאת.

**התכונות החוזרות:** מספר מחלקה, שם מחלקה ומספר העובדים במחלקה.

**מספר השורות:** 953.

**זמן ריצה:** 0.36 שניות.

6. מספר המחלקות שכל עובד עובד בהן

**שימוש:** על מנת שמשאבי אנוש יוכלו לחלק את העובדים בין המחלקות, משאבי אנוש צריכים לדעת מי העובדים שעובדים קשה ומי העובדים שכמעט ואין להם עבודה. בעזרת השאילתה הזאת הם יכולים לדעת את הנתונים האלו.

**התכונות החוזרות:** מספר עובד, שם עובד, שכר לשעה, תאריך הצטרפות, אימייל

ומספר המחלקות שהעובד עובד בהן.

**מספר השורות:** 1893.

**זמן ריצה:** 0.884 שניות.

7. עובדים שיש להם את אותו יום ההולדת ממיון לפי יום ההולדת.

**שימוש:** ההנהלה החליטה במקום לחגוג לכל עובד יום הולדת בפני עצמו, לחגוג לכל העובדים באותו שחוגגים באותו היום מסיבת יום הולדת אחת וכך לחסוך כספים.

**התכונות החוזרות:** מספר עובד, שם עובד, אימייל, תאריך לידה, תאריך הצטרפות

ותאריך עזיבה.

**מספר השורות:** 20911.

**זמן ריצה:** 9.448 שניות.

8. כל המחלקות שאין להן מנהל.

**שימוש:** משאבי אנוש רוצים שלכל מחלקה יהיה לפחות מנהל אחד, ועל מנת לעשות זאת הם צריכים שאילתה שתציג להם את המחלקות שלהן אין מנהל.

**התכונות החוזרות:** מספר עובד, שם עובד, אימייל, תאריך לידה, תאריך הצטרפות

ותאריך עזיבה.

**מספר השורות:** 34.

**זמן ריצה:** 0.073 שניות.

הקודים של כל השאילתות נמצאים ב**[נספח](#)**.

## אינדקסים

1. ימי ההולדת של העובדים.

**שאלתה:** האינדקס משפר את השאילתה מספר 7.

**התפלגות:** מספר התאריכים המיוחדים הם 12361 מתוך 20911.

**שיפור:** חל שיפור כבשניה וחצי.

**אחוז השינוי:** ירד בכ14 אחוזים מהזמן המקורי.

**הסבר למה התקבלו התוצאות:** השאילתה עושה הרבה מאוד שימושים בתאריך הלידה וביוון שהנדקסנו את התכונה הזאת אז זמן החיפוש קטן.

**ריצה לפני האינדקס:**

20911 rows selected in 9.448 seconds

**ריצה אחרי האינדקס:**

20911 rows selected in 8.101 seconds

2. זמן סיום העבודה של העובד.

**שאלתה:** האינדקס משפר את השאילתה מספר 6.

**התפלגות:** יש 1894 תאריכים שהם Null (עובדים בפועל) לעומת 20,911 שורות בטבלה של העובדים.

	NULLEMENTS	ALLELEMENTS
▶ 1	1894	20911

**שיפור:** חל שיפור ב0.072 שניות.

**אחוז השינוי:** ירד בכ8 אחוזים.

**הסבר למה התקבלו התוצאות:** השאילתה בודקת האם העובד עדיין עובד לפי האם תאריך העזיבה הוא NULL, לכן האינדקס היה צריך לייעל את זמן השאילתה.

**ריצה לפני האינדקס:**

1893 rows selected in 0.884 seconds

**ריצה אחרי האינדקס:**

➦ 1893 rows selected in 0.812 seconds

3. מספר עובד בטבלה של עובד במחלקה.

**שאלתה:** האינדקס צריך לשפר את השאילתה מספר 5.

**התפלגות:** יש 20910 מספרי עובדים שונים בתוך הטבלה עם 31291 שורות.

**שיפור:** חל שיפור ב0.016 שניות.

**אחוז השינוי:** ירד בכ5 אחוזים.

**הסבר למה התקבלו התוצאות:** השאילתה עושה natural join בין הטבלאות של עובד ועובד במחלקה.



### ריצה לפני האינדקס:

953 rows selected in 0.327 seconds
------------------------------------

### ריצה אחרי האינדקס:

953 rows selected in 0.311 seconds
------------------------------------

את הקוד של האינדקסים אפשר לראות ב**[נספח](#)**.

## אינטגרציה

נתנו הרשאות select reference לטבלה Worker לשתי קבוצות.  
את הקוד שנותן את הגישה, אפשר לראות בנספח.

קיבלנו הרשאה מהקבוצה של חננאל זגורי ואביחי סוריה לגשת אל הטבלאות שלהם.

## שאלות האינטגרציה

1. הצגת מחיר לכל מחלקה של כל מוצגי התערוכות שפועלות כרגע.  
**שימוש:** על מנת שמשאבי אנוש יוכלו לחלק בצורה הוגנת את כל מאבטחי המוזיאון בין התערוכות, הם צריכים לדעת כמה יקר כל תערוכה, שאת זה השאלתה נותנת.  
**התכונות החוזרות:** מספר מחלקה, שם מחלקה ומחיר.  
**מספר השורות:** 505.  
**זמן ריצה:** 0.226 שניות.
  2. כל המחלקות שאין אצלן תערוכה כרגע.  
**שימוש:** על מנת שהמעצבים יוכלו לפתוח תערוכות חדשות, הם צריכים לדעת באיזה מחלקות אין כרגע תערוכה בשביל לפתוח שם, שאת זה השאלתה נותנת.  
**התכונות החוזרות:** מספר מחלקה ושם מחלקה.  
**מספר השורות:** 991.  
**זמן ריצה:** 0.281 שניות.
  3. כל המחלקות שמתקיים אצלן תערוכה כרגע אבל אין שם שומרים.  
**שימוש:** חשוב מאוד שבכל מחלקה שמתקיימת אצלה תערוכה כרגע שיהיה שם לפחות שומר אחד, לכן צריך שאלתה שנותנת את כל המחלקות שבהן אין שומר שעובד כרגע.  
**התכונות החוזרות:** מספר מחלקה ושם מחלקה.  
**מספר השורות:** 365.  
**זמן ריצה:** 0.158 שניות.
- הקודים של כל השאלות נמצאים ב**נספח**.

יצרנו 4 תצוגות שונות, שניים בשביל הנהלת חשבונות ושניים בשביל משאבי אנוש.

### 1. הנהלת חשבונות

- העובדים מסודרים בסדר יורד לפי השכר לשעה  
הנהלת החשבונות ישתמשו בזה על מנת לבדוק אצל אילו עובדים יורד הכי הרבה כסף בשכר החודשי.
- בממוצע סך כל ההוצאות על שכר העובדים בחודש  
הנהלת החשבונות ישתמשו בזה על מנת לבדוק את ההוצאה החודשית בממוצע של המוזיאון על העובדים.

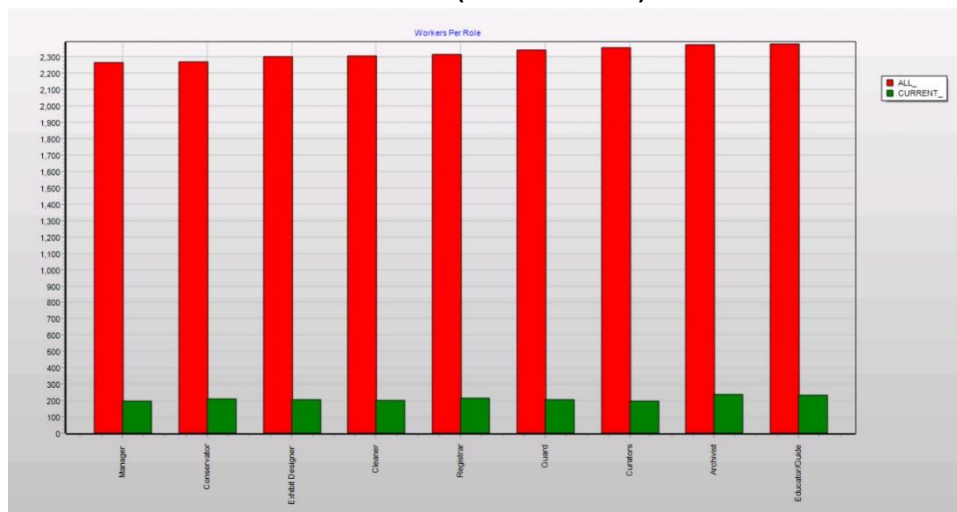
### 2. משאבי אנוש

- המנהלים שעובדים כרגע  
משאבי אנוש יוכלו להשתמש בזה על מנת לקבל מידע אודות המנהלים שעובדים במוזיאון.
- העובדים שעובדים לצד המחלקות בהם עובדים  
משאבי אנוש יוכלו להשתמש בזה על מנת לבדוק כל עובד באיזו מחלקה הוא עובד.

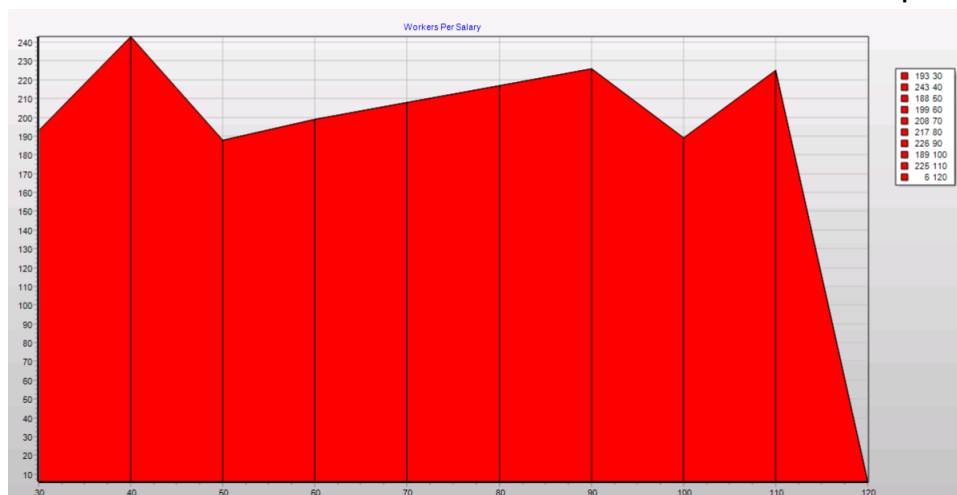
את הקוד של התצוגות אפשר לראות ב[נספח](#).

## גרפים

בגרף הבא רואים בציר X את כל התפקידים שיש לעובדים ובציר Y מספר העובדים בכל תפקיד. בנוסף אפשר לראות מספר העובדים שכרגע עובדים (בצבע הירוק) ביחס למספר העובדים הכללי שעבדו עד עכשיו (בצבע האדום).



בגרף הבא רואים בציר X את המשכורות של העובדים (מעוגל) ובציר Y את מספר העובדים שמקבלים משכורת בתחום הזה.



## תכנות PL/SQL

### פרוצדורות

1. שינוי שכר לעובדים שעבדו הרבה זמן ומקבלים שכר לשעה נמוך  
**מטרה:** כאמור לפני כן, משאבי אנוש רצו להעלות את השכר לשעה של העובדים שעבדו מעל 30 שנה ומקבלים פחות מ-100 לשעה. על מנת לעדכן את השכר, יצרנו את הפרוצדורה הבאה:

```
1 CREATE OR REPLACE PROCEDURE SetSalaryToOldWorkers(yearsWorking INT, salaryPH FLOAT, salaryPHLessThan FLOAT, updatedRows OUT INT)
2 AS
3 BEGIN
4     UPDATE Worker SET SalaryPerHour=salaryPH
5     WHERE LeftDate IS NULL
6     AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM JoinDate) >= yearsWorking)
7     AND (SalaryPerHour < salaryPHLessThan);
8     updatedRows := SQL%ROWCOUNT;
9 END;
```

כאן בדקנו שהפרוצדורה באמת עובדת:

The screenshot shows the SQL Developer interface. The 'Test script' tab is active, displaying the following PL/SQL code:

```
1 declare
2     updatedRows integer;
3 begin
4     SetSalaryToOldWorkers(30, 100, 100, :updatedRows);
5 end;
```

Below the code, the 'DBMS Output' tab shows the execution results in a table:

Variable	Type	Value
updatedRows	String	113

2. פיטור עובד.

**מטרה:** משאבי אנוש צריכים את הפרוצדורה הזאת על מנת לפטר עובד (או שהמוזיאון מפטר אותו או שהוא עוזב), לכן קיימת הפרוצדורה הזאת.

```
1 CREATE OR REPLACE PROCEDURE FireWorker(workerId_ INT)
2 AS
3 BEGIN
4     UPDATE Worker SET LeftDate = CURRENT_DATE
5     WHERE workerId = workerId_ AND LeftDate IS NULL;
6 END;
```

כאן בדקנו שהפרוצדורה באמת עובדת:

```

Test script DBMS Output Statistics Profiler Trace
1
2 declare
3     id INT;
4     d Worker.LeftDate%TYPE;
5 begin
6     id := 19786;
7     FireWorker(id);
8     SELECT LeftDate INTO d FROM Worker WHERE WorkerId = id;
9 end;

```

## פונקציות

1. האם עובד עדיין עובד.

**מטרה:** במהלך כתיבת השאילתות והתצוגות השתמשנו הרבה בתנאי LeftDate IS NULL בשביל לבדוק שהעובד עדיין עובד. על מנת לשפר את הקריאות של השאילתות הוספנו את הפונקציה IsWorking(LeftDate) וכך יהיה מובן מה הכוונה.

```

1 CREATE OR REPLACE FUNCTION IsWorking(leftDate DATE)
2 RETURN INT
3 AS
4 BEGIN
5     IF leftDate IS NULL THEN
6         RETURN 1;
7     ELSE
8         RETURN 0;
9     END IF;
10 END;

```

להלן אפשר לראות שהפונקציה עובדת:

```
SELECT * FROM Worker WHERE IsWorking(leftDate) = 1;
```

WORKERID	NAME	ROLE	SALARYPERHOUR	JOINDATE	LEFTDATE	BIRTHDATE	EMAIL
1	19787 Courtney Arroyo	Cleaner	44.52	30/06/2018		28/04/1988	Courtney_Arroyo577@gmail.com
2	19788 Luis Raymond	Manager	118.26	23/08/2008		04/01/1976	LuisRaymond644@gmail.com
3	19789 Richard Bowman	Cleaner	120	21/07/1996		24/01/1966	RichardBowman991@gmail.com
4	19790 Christopher Moore PhD	Conservator	73.64	19/09/2011		10/08/1971	ChristopherMoorePhD634@walla.com
5	19791 Kelsey Pennington	Conservator	80.67	23/09/1997		15/12/1960	Kelsey-Pennington542@hotmail.com
6	19792 Rachael Beasley	Exhibit Designer	62.15	30/10/2019		18/11/1994	RachaelBeasley773@hotmail.com
7	19793 Sarah Johnson	Exhibit Designer	102.4	20/12/2007		20/04/1972	Sarah-Johnson316@hotmail.com
8	19794 Darren Flowers	Manager	87.45	17/11/2019		14/03/1982	Darren-Flowers231@outlook.com
9	19795 Scott Cox	Curators	121.79	13/06/2018		03/04/1963	Scott_Cox383@outlook.com
10	19796 Kenneth Frank	Guard	81.41	17/09/2004		08/01/1973	KennethFrank543@hotmail.com

2. מספר העובדים במחלקה.

**מטרה:** הפונקציה נותנת את מספר העובדים במחלקה עם מספר המחלקה בקלט.

```
1 CREATE OR REPLACE FUNCTION NumberWorkersOfDepartment (departmentId_ INT)
2 RETURN INT
3 AS
4     result INT;
5 BEGIN
6     SELECT COUNT(*) INTO result
7     FROM Department NATURAL JOIN WorksAt wa
8         INNER JOIN Worker w ON w.WorkerId = wa.WorkerId
9     WHERE IsWorking(leftDate) = 1 AND DepartmentId = departmentId_
10    GROUP BY DepartmentId;
11
12    RETURN result;
13 END;
```

להלן אפשר לראות שהפונקציה עובדת:

Test script DBMS Output Statistics Profiler Trace

1  
2 declare  
3 res integer;  
4 begin  
5 res := NumberWorkersOfDepartment (0);  
6 end;

Variable	Value	Call stack
res	6	[Line 6] end;
*		

## נספח

### הקוד ליצירת הטבלאות

```
1 CREATE TABLE Worker
2 (
3     WorkerId INT NOT NULL,
4     Name VARCHAR(50) NOT NULL,
5     Role VARCHAR(50) NOT NULL,
6     SalaryPerHour FLOAT NOT NULL,
7     JoinDate DATE NOT NULL,
8     LeftDate DATE,
9     BirthDate DATE NOT NULL,
10    Email VARCHAR(50) NOT NULL,
11    PRIMARY KEY (WorkerId)
12 );
13
14 CREATE TABLE Wing
15 (
16     WingId INT NOT NULL,
17     Name VARCHAR(50) NOT NULL,
18     PRIMARY KEY (WingId)
19 );
20
21 CREATE TABLE Department
22 (
23     DepartmentId INT NOT NULL,
24     Name VARCHAR(50) NOT NULL,
25     WingId INT NOT NULL,
26     PRIMARY KEY (DepartmentId),
27     FOREIGN KEY (WingId) REFERENCES Wing(WingId)
28 );
29
30 CREATE TABLE WorksAt
31 (
32     WorkerId INT NOT NULL,
33     DepartmentId INT NOT NULL,
34     PRIMARY KEY (WorkerId, DepartmentId),
35     FOREIGN KEY (WorkerId) REFERENCES Worker(WorkerId),
36     FOREIGN KEY (DepartmentId) REFERENCES Department(DepartmentId)
37 );
```

### הקוד לאכילוס הטבלאות וקבצי CSV

<https://github.com/nbashan/MiniProjectInDataBases/tree/master/scripts>

### הקוד של השאילתות

.1

```
1 -- Find all working workers that works for a long time (30 years and above) and earns a low salary (below 100 per hour)
2 SELECT WorkerId, Name, SalaryPerHour, JoinDate, Email, BirthDate
3 FROM Worker WHERE LeftDate IS NULL
4     AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM JoinDate) >= 30)
5     AND (SalaryPerHour < 100.0);
```

.2

```
1 -- Calculate the salary to pay for workers per an average month (20 days with 8 hours per day)
2 SELECT SUM(SalaryPerHour) * 8 * 20 AS Salary
3 FROM Worker WHERE LeftDate IS NULL;
```



.3

```
-- Show all roles with their workers and working workers
SELECT Role, Count(*) AS All_, Sum(IsWorking) AS Current_
FROM (SELECT Role, IsWorking(LeftDate) AS IsWorking FROM Worker)
GROUP BY Role
ORDER BY All_;
```

.4

```
1 -- Shows the distribution of salary between the workers (rounds to the closest ten)
2 SELECT (ROUND(SalaryPerHour) - MOD(ROUND(SalaryPerHour),10)) AS Salary, COUNT(*) AS Workers
3 FROM Worker
4 WHERE LeftDate IS NULL
5 GROUP BY (ROUND(SalaryPerHour) - MOD(ROUND(SalaryPerHour),10))
6 ORDER BY salary
```

.5

```
1 -- Number of working workers per department
2 SELECT DepartmentId, Name, COUNT(*) AS Workers
3 FROM WorksAt NATURAL JOIN Department
4 WHERE WorkerId IN (SELECT WorkerId FROM Worker WHERE LeftDate IS NULL)
5 GROUP BY DepartmentId, Name;
```

.6

```
1 -- Number of departments each working worker works on
2 SELECT WorkerId, Name, SalaryPerHour, JoinDate, Email, COUNT(*) AS Departments
3 FROM WorksAt NATURAL JOIN Worker
4 WHERE LeftDate IS NULL
5 GROUP BY WorkerId, SalaryPerHour, Name, JoinDate, Email;
```

.7

```
1 -- Workers with the same birth date
2 SELECT w1.WorkerId, w1.Name, w1.Email, w1.BirthDate, w1.JoinDate, w1.LeftDate
3 FROM Worker w1 JOIN Worker w2
4 ON TO_CHAR(w1.BirthDate, 'MM-dd') = TO_CHAR(w2.BirthDate, 'MM-dd')
5 GROUP BY w1.WorkerId, w1.Name, w1.Email, w1.BirthDate, w1.JoinDate, w1.LeftDate
6 ORDER BY TO_CHAR(w1.BirthDate, 'MM-dd');
```

.8

```
1 -- Find all departments that don't have a manager
2 SELECT DepartmentId, Name FROM Department
3 WHERE DepartmentId IN (SELECT DepartmentId FROM Department
4 MINUS
5 SELECT d.DepartmentId
6 FROM Worker NATURAL JOIN WorksAt wa INNER JOIN Department d ON d.DepartmentId = wa.DepartmentId
7 WHERE Role = 'Manager');
```

הקוד של האינדקסים

.1

```
1 CREATE INDEX i_WorkerBirthdate
2 ON Worker(BirthDate)
```

.2

```
1 CREATE INDEX i_WorkerLeftdate
2 ON Worker(LeftDate)
```

.3

```
1 CREATE INDEX i_WorksAtWorkerId
2 ON WorksAt(WorkerId);
```

הקוד של שאילתות האינטגרציה

הרשאות גישה:

```
1  GRANT SELECT, REFERENCES ON Department
2  TO aheller;
3
4  GRANT SELECT, REFERENCES ON Wing
5  TO aheller;
6
7  GRANT SELECT, REFERENCES ON Department
8  TO silon;
9
10 GRANT SELECT, REFERENCES ON Wing
11 TO silon;
12
13 GRANT SELECT, REFERENCES ON Worker
14 TO yyohanan;
```

.1

```
1  -- Select all departments and their price of current exhibitions
2  SELECT d.DepartmentId, d.Name, SUM(Price)
3  FROM Department d
4       INNER JOIN Soria.ExhibitionAtDepartment ead ON d.DepartmentId = ead.Department_Id
5       NATURAL JOIN Soria.Exhibition
6       NATURAL JOIN Soria.ExhibitAtExhibition ee
7       INNER JOIN Soria.Exhibit e ON e.ExhibitId = ee.ExhibitId
8  WHERE StartTime <= CURRENT_DATE AND CURRENT_DATE <= EndTime
9  GROUP BY d.DepartmentId, d.Name
```

.2

```
1  -- Find all departments that currently don't contains any exhibitions.
2  SELECT DepartmentId, Name FROM Department
3  WHERE DepartmentId NOT IN (
4      SELECT Department_Id
5      FROM Soria.ExhibitionAtDepartment NATURAL JOIN Soria.Exhibition
6      WHERE StartTime <= CURRENT_DATE AND CURRENT_DATE <= EndTime
7  );
```

.3

```
1  -- Finds departments that currently contains an exhibition but without any guards.
2  SELECT DepartmentId, Name
3  FROM Department
4      INNER JOIN Soria.ExhibitionAtDepartment ON DepartmentId = Department_Id
5      NATURAL JOIN Soria.Exhibition
6  WHERE StartTime <= CURRENT_DATE AND CURRENT_DATE <= EndTime AND DepartmentId NOT IN (
7      SELECT DepartmentId
8      FROM Department NATURAL JOIN WorksAt wa INNER JOIN Worker w ON wa.WorkerId = w.WorkerId
9      WHERE LeftDate IS NULL AND Role = 'Guard'
10 )
11 GROUP BY DepartmentId, Name
12 ORDER BY DepartmentId;
```

## הקוד של הViews

.1

```
1  CREATE VIEW workersOrderedBySalary
2  AS
3  SELECT WorkerId, Name, Role, SalaryPerHour, Email, JoinDate, BirthDate
4  FROM Worker
5  WHERE IsWorking(LeftDate)
6  ORDER BY SalaryPerHour desc
```

.2

```
1  CREATE VIEW totalSalaryPerMonth
2  AS
3  SELECT sum(SalaryPerHour)*160
4  FROM Worker
```

.3

```
1 CREATE VIEW v_WorkingManagers
2 AS
3 SELECT WorkerId, Name, Role, SalaryPerHour, Email, JoinDate, BirthDate
4 FROM Worker WHERE LeftDate IS NULL AND Role == 'Manager';
```

.4

```
1 CREATE VIEW v_WorkingWorkersWithDepartment
2 AS
3 SELECT DepartmentId, DepartmentName, WorkerId, Name AS WorkerName, Role, SalaryPerHour, Email, JoinDate, BirthDate
4 FROM (SELECT DepartmentId, Name AS DepartmentName, WorkerId
5       FROM WorksAt NATURAL JOIN Department) NATURAL JOIN
6 (SELECT * FROM Worker WHERE LeftDate IS NULL)
7 ORDER BY DepartmentId;
```