

מיני פרויקט בבסיסי נתונים

פרויקט: בסיס נתונים עבור בנק

היחידה הנבחרת: מידע ומשכורות של עובדים

מגישים:

אלי ארזי 336301304

אלעד רדומסקי 318645850

תוכן עניינים:

3.....	תאור המערכת
4.....	ERD
4.....	DSD
5.....	יצירת טבלאות
6-7.....	Data generator
8-9.....	DESC
10.....	גיבוי ושחזור
11-24.....	שלב 2 - שאילתות ואילוצים
25-35.....	שלב 3 – פונקציות ופרוצדורות

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

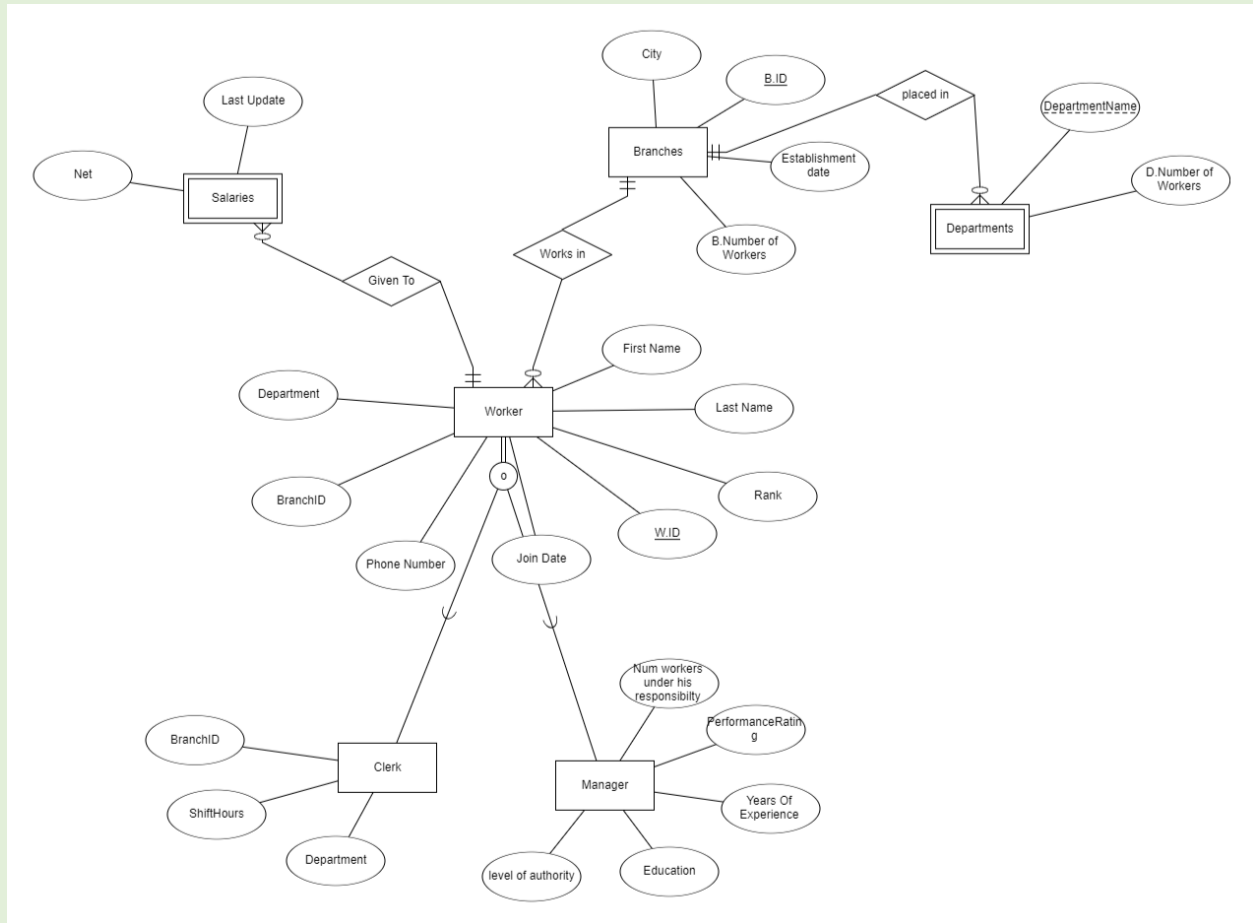
אהרון עובד בסניף: תל אביב, כמנהל מחלקת הלוואות

מנהל סניף תל אביב: אברהם

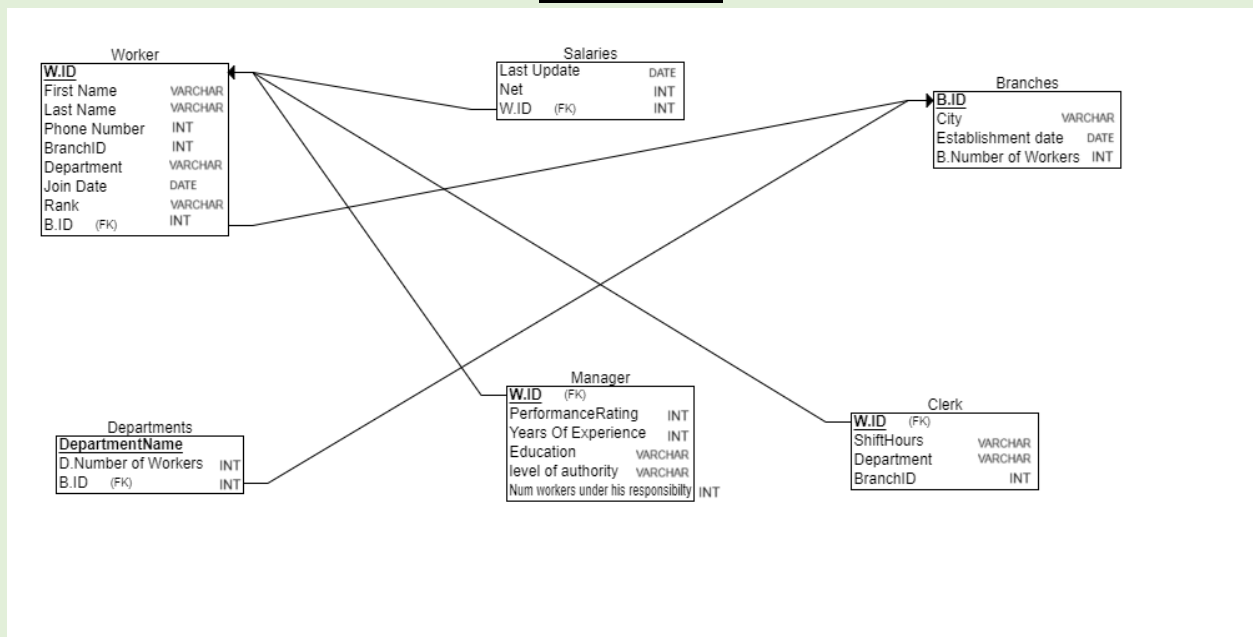
מנהל סניף ירוחם: משה

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:

```
import random
from datetime import datetime, timedelta
# Function to generate random dates
def random_date(start, end):
    return (start + timedelta(days=random.randint(0, (end - start).days))).strftime('%Y-%m-%d')
# Function to calculate Net salary from Gross salary
def calculate_net_salary(gross_salary):
    # Example formula: net salary is gross salary minus 20% taxes
    net_salary = int(gross_salary * 0.80)
    return net_salary
department_names = ['Finance', 'HR', 'IT', 'CustomerService', 'Operations', 'Marketing', 'Legal',
                    'Compliance', 'Audit', 'Risk']
# Example educations and authority levels
educations = ["Bachelor's Degree", "Master's Degree", "PhD", "Associate Degree", "High School Diploma"]
authority_levels = [
    'Manage IT Department', 'Manage HR Department', 'Manage Finance Department',
    'Manage Operations', 'Manage Customer Service', 'Manage Marketing'
]
# Date range for last update
start_date = datetime.strptime('01-01-1990', '%d-%m-%Y')
end_date = datetime.strptime('26-05-2024', '%d-%m-%Y')
# Salaries Table
salary_insert_commands = []
for worker_id in range(1, 43571):
    gross_salary = random.randint(30000, 120000) # Random gross salary between 30k and 120k
    net_salary = calculate_net_salary(gross_salary)
    last_update = random_date(start_date, end_date)
    salary_insert_commands.append(f"INSERT INTO Salaries (WorkerID, Gross, Net, LastUpdate) VALUES ({worker_id}, {gross_salary}, {net_salary}, '{last_update}');")
# Manager Table
manager_insert_commands = []
num_managers = int(0.1 * 43570) # Assuming 10% of workers are managers
manager_worker_ids = random.sample(range(1, 43571), num_managers) # Unique WorkerIDs for managers
for worker_id in manager_worker_ids:
    num_workers_uhr = random.randint(5, 50) # Number of workers under responsibility
    performance_rating = random.randint(1, 100)
    years_of_experience = random.randint(1, 40)
    education = random.choice(educations).replace("'", "")
    level_of_authority = random.choice(authority_levels).replace("'", "")
    manager_insert_commands.append(f"INSERT INTO Manager (WorkerID, NumWorkersUHR, PerformanceRating, YearsOfExperience, Education, LevelOfAuthority) VALUES ({worker_id}, {num_workers_uhr}, {performance_rating}, {years_of_experience}, '{education}', '{level_of_authority}');")
# Clerk Table
clerk_insert_commands = []
all_worker_ids = set(range(1, 43571))
clerk_worker_ids = all_worker_ids - set(manager_worker_ids) # WorkerIDs for clerks, excluding managers
for worker_id in clerk_worker_ids:
    branch_id = random.randint(1, 400)
    department = random.choice(department_names).replace("'", "")
    shift_hours = random.choice(['Morning', 'Afternoon', 'Night'])
    clerk_insert_commands.append(f"INSERT INTO Clerk (WorkerID, BranchID, Department, ShiftHours) VALUES ({worker_id}, {branch_id}, '{department}', '{shift_hours}');")
# Write the SQL commands to files
with open('insert_salaries.sql', 'w') as file:
    file.write('\n'.join(salary_insert_commands))
with open('insert_managers.sql', 'w') as file:
    file.write('\n'.join(manager_insert_commands))
with open('insert_clerks.sql', 'w') as file:
    file.write('\n'.join(clerk_insert_commands))
print("SQL insert commands for Salaries, Managers, and Clerks have been generated and written to files.")
```

Branches, Departments and Workers Tables:

```
import random
from datetime import datetime, timedelta

cities = ['Jerusalem', 'TelAviv', 'Ramle', 'Haifa', 'BeerSheva', 'Netanya', 'Petahtikva', 'RamatGan',
          'Yavne', 'Ashdod', 'Ashkelon', 'Hadera', 'KiryatShmona', 'Afula', 'Lod', 'Herzelia', 'KfarSaba',
          'Raanana', 'Teveriya', 'MaaleEdomim']
department_names = ['Finance', 'HR', 'IT', 'CustomerService', 'Operations', 'Marketing', 'Legal',
                    'Compliance', 'Audit', 'Risk']

first_names = ['James', 'John', 'Robert', 'Michael', 'William', 'David', 'Richard', 'Joseph',
               'Charles', 'Thomas', 'Christopher', 'Daniel', 'Matthew', 'Anthony', 'Mark', 'Donald', 'Steven', 'Paul',
               'Andrew', 'Joshua', 'Kenneth', 'Kevin', 'Brian', 'George', 'Edward', 'Ronald', 'Timothy', 'Jason',
               'Jeffrey', 'Ryan', 'Jacob', 'Gary', 'Nicholas', 'Eric', 'Jonathan', 'Stephen', 'Larry', 'Justin',
               'Scott', 'Brandon', 'Benjamin', 'Samuel', 'Gregory', 'Frank', 'Alexander', 'Raymond', 'Patrick', 'Jack',
               'Dennis', 'Jerry', 'Tyler', 'Aaron', 'Jose', 'Adam', 'Nathan', 'Henry', 'Douglas', 'Zachary', 'Peter',
               'Kyle', 'Walter', 'Ethan', 'Jeremy', 'Harold', 'Keith', 'Christian', 'Roger', 'Noah', 'Gerald',
               'Carl', 'Terry', 'Sean', 'Austin', 'Arthur', 'Lawrence', 'Jesse', 'Dylan', 'Bryan', 'Joe',
               'Jordan', 'Billy', 'Bruce', 'Albert', 'Willie', 'Gabriel', 'Logan', 'Alan', 'Juan', 'Wayne',
               'Roy', 'Ralph', 'Randy', 'Eugene', 'Carlos', 'Russell', 'Bobby', 'Victor', 'Martin', 'Ernest',
               'Phillip']

last_names = ['Smith', 'Johnson', 'Williams', 'Brown', 'Jones', 'Garcia', 'Miller', 'Davis',
              'Rodriguez', 'Martinez', 'Hernandez', 'Lopez', 'Gonzalez', 'Wilson', 'Anderson', 'Thomas', 'Taylor',
              'Moore', 'Jackson', 'Martin', 'Lee', 'Perez', 'Thompson', 'White', 'Harris', 'Sanchez', 'Clark',
              'Ramirez', 'Lewis', 'Robinson', 'Walker', 'Young', 'Allen', 'King', 'Wright', 'Scott', 'Torres',
              'Nguyen', 'Hill', 'Flores', 'Green', 'Adams', 'Nelson', 'Baker', 'Hall', 'Rivera', 'Campbell',
              'Mitchell', 'Carter', 'Roberts', 'Gomez', 'Phillips', 'Evans', 'Turner', 'Diaz', 'Parker', 'Cruz',
              'Edwards', 'Collins', 'Reyes', 'Stewart', 'Morris', 'Morales', 'Murphy', 'Cook', 'Rogers', 'Gutierrez',
              'Ortiz', 'Morgan', 'Cooper', 'Peterson', 'Bailey', 'Reed', 'Kelly', 'Howard', 'Ramos', 'Kim', 'Cox',
              'Ward', 'Richardson', 'Watson', 'Brooks', 'Chavez', 'Wood', 'James', 'Bennett', 'Gray', 'Mendoza',
              'Ruiz', 'Hughes', 'Price', 'Alvarez', 'Castillo', 'Sanders', 'Patel', 'Myers', 'Long', 'Ross', 'Foster',
              'Jimenez']

def random_date(start, end):
    return (start + timedelta(days=random.randint(0, (end - start).days))).strftime('%Y-%m-%d')

def random_phone_number():
    return f"05{random.randint(0, 9)}-{random.randint(1000000, 9999999)}"

# Branches Table
branch_insert_commands = []
branch_data = []
start_date = datetime.strptime('01-01-1990', '%d-%m-%Y')
end_date = datetime.strptime('26-05-2024', '%d-%m-%Y')

for branch_id in range(1, 401):
    city = random.choice(cities)
    establishment_date = random_date(start_date, end_date)
    number_of_workers = random.randint(20, 200)
    branch_insert_commands.append(f"INSERT INTO Branches (city, EstablishmentDate, NumberOfWorkers)
VALUES ({city}, '{establishment_date}', {number_of_workers});")
    branch_data.append((branch_id, number_of_workers))

# Departments Table
department_insert_commands = []
department_data = []
for branch_id, num_workers in branch_data:
    department_workers = [num_workers // 10 + (1 if x < num_workers % 10 else 0) for x in range(10)]
    for dept_index, dept_name in enumerate(department_names):
        department_insert_commands.append(f"INSERT INTO Departments (DepartmentName, BranchID,
NumberOfWorkers) VALUES ({dept_name}, {branch_id}, {department_workers[dept_index]});")
        department_data.append((dept_name, branch_id, department_workers[dept_index]))

# Workers Table
worker_insert_commands = []
worker_id = 1
for dept_name, branch_id, dept_workers in department_data:
    for _ in range(dept_workers):
        first_name = random.choice(first_names)
        last_name = random.choice(last_names)
        join_date = random_date(start_date, end_date)
        phone_number = random_phone_number()
        worker_insert_commands.append(f"INSERT INTO Worker (FirstName, LastName, JoinDate, PhoneNumber,
BranchID, Department) VALUES ({first_name}, '{last_name}', '{join_date}', '{phone_number}',
{branch_id}, '{dept_name}');")
        worker_id += 1

# Write the SQL commands to files
with open('insert_branches.sql', 'w') as file:
    file.write('\n'.join(branch_insert_commands))

with open('insert_departments.sql', 'w') as file:
    file.write('\n'.join(department_insert_commands))

with open('insert_workers.sql', 'w') as file:
    file.write('\n'.join(worker_insert_commands))
print("SQL insert commands have been generated and written to files.")
```

DESC commands:

DESC branches;

Field	Type	Null	Key	Default	Extra
BranchID	int(11)	NO	PRI	NULL	auto_increment
City	varchar(20)	NO		NULL	
EstablishmentDate	date	NO		NULL	
NumberOfWorkers	int(11)	NO		NULL	

DESC departments;

Field	Type	Null	Key	Default	Extra
DepartmentName	varchar(20)	NO	PRI	NULL	
BranchID	int(11)	NO	PRI	NULL	
NumberOfWorkers	int(11)	NO		NULL	

DESC worker;

Field	Type	Null	Key	Default	Extra
WorkerID	int(11)	NO	PRI	NULL	auto_increment
FirstName	varchar(20)	NO		NULL	
LastName	varchar(20)	NO		NULL	
JoinDate	date	NO		NULL	
PhoneNumber	int(11)	NO		NULL	
BranchID	int(11)	NO		NULL	
Department	varchar(20)	NO		NULL	
Rank	varchar(20)	NO		Junior	

DESC salaries;

Field	Type	Null	Key	Default	Extra
WorkerID	int(11)	NO	PRI	NULL	
LastUpdate	date	NO		NULL	
Net	int(11)	NO		NULL	

DESC manager;

Field	Type	Null	Key	Default	Extra
WorkerID	int(11)	NO	PRI	NULL	
NumWorkersUHR	int(11)	NO		NULL	
PreformanceRating	int(11)	NO		NULL	
YearsOfExperience	int(11)	NO		NULL	
Education	varchar(50)	NO		NULL	
LevelOfAuthority	varchar(50)	NO		NULL	

DESC clerk;

Field	Type	Null	Key	Default	Extra
WorkerID	int(11)	NO	MUL	NULL	
Department	varchar(20)	NO	MUL	NULL	
BranchID	int(11)	NO	MUL	NULL	
ShiftHours	int(11)	NO		NULL	

גיבוי בסיס הנתונים:

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Tracking Designer Central columns

Exporting tables from "workersandsalaries" database

Export templates:

New template:

Template name: Create

Existing templates:

Template: -- Select a template -- Update Delete

Export method:

☒ Quick - display only the minimal options
☐ Custom - display all possible options

Format:

SQL ▼

Export

שחזור בסיס הנתונים:

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Tracking Designer Central columns

Importing into the database "workersandsalaries"

File to import:

File may be compressed (gzip, bzip2) or uncompressed.
A compressed file's name must end in `{format}.{compression}` Example: `.sql.zip`

Browse your computer: (Max: 40MiB)

בחירת קובץ

You may also drag and drop a file on any page.

Character set of the file:

▼

שלב 2

שאלת select 1:

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

```
1 SELECT m.WorkerID, w.FirstName, w.LastName, w.Rank, b.BranchID, b.City, b.EstablishmentDate
2 FROM manager m
3 JOIN worker w ON m.WorkerID = w.WorkerID
4 JOIN branches b ON w.BranchID = b.BranchID
5 WHERE w.Rank = 'Director'
6 ORDER BY m.YearsOfExperience DESC, m.PreformanceRating DESC
7 LIMIT 1;
8
```

השאלתה 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 הנסיון בסדר יורד, במקרה של שוויון השאלתה מסדרת לפי דירוג (performance rating). כלומר השאלתה הזאת מחפשת את פרטי המנהל עם מס הכי גבוה של שנות נסיון עם הדירוג הגבוה ביותר ועם הדרגה הכי גבוהה (director) השאלתה מחזירה את הפרטים של אותו מנהל כולל מידע על המחלקה בו הוא עובד.

הרצה על בסיס הנתונים:

✓ Showing rows 0 - 0 (1 total, Query took 0.0020 seconds.)

```
SELECT m.WorkerID, w.FirstName, w.LastName, w.Rank, b.BranchID, b.City, b.EstablishmentDate
FROM manager m
JOIN worker w ON m.WorkerID = w.WorkerID
JOIN branches b ON w.BranchID = b.BranchID
WHERE w.Rank = 'Director'
ORDER BY m.YearsOfExperience DESC, m.PreformanceRating DESC
LIMIT 1;
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

Extra options

WorkerID	FirstName	LastName	Rank	BranchID	City	EstablishmentDate
21878	Gary	Martinez	Director	206	jerusalem	2005-04-08

Query results operations

שאלת select 2:

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

```
1 SELECT d.DepartmentName, b.City, COUNT(w.WorkerID) AS TotalWorkers
2 FROM departments d
3 JOIN branches b ON d.BranchID = b.BranchID
4 JOIN worker w ON d.DepartmentName = w.Department AND d.BranchID = w.BranchID
5 WHERE YEAR(b.EstablishmentDate) > 2010
6 GROUP BY d.DepartmentName, b.City
7 ORDER BY TotalWorkers DESC;
```

השאלתה לוקחת שם מחלקה, עיר, וסופרת כמה עובדים יש בכל מחלקה וקוראת לסכום TotalWorkers, השאלתה 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:

✓ Showing rows 0 - 24 (200 total, Query took 0.0524 seconds.) [TotalWorkers: 173... - 119...]

```
SELECT d.DepartmentName, b.City, COUNT(w.WorkerID) AS TotalWorkers
FROM departments d
JOIN branches b ON d.BranchID = b.BranchID
JOIN worker w ON d.DepartmentName = w.Department AND d.BranchID = w.BranchID
WHERE YEAR(b.EstablishmentDate) > 2010
GROUP BY d.DepartmentName, b.City
ORDER BY TotalWorkers DESC;
```

[\[Edit inline \]](#) [\[Edit \]](#) [\[Create PHP code \]](#)

1 ▾

> >>



Show all

Number of rows:

25 ▾

Filter rows:

Search this table

Extra options

DepartmentName	City	TotalWorkers ▾ 1
Finance	RamatGan	173
HR	RamatGan	172
CustomerService	RamatGan	170
IT	RamatGan	170
Operations	RamatGan	168

שאלת select 3:

שאלתה שמוצאת את השכר הממוצע לכל מחלקה, עבור מחלקות עם יותר מ-5 עובדים.

```
1 SELECT w.Department, AVG(s.Net) AS AverageSalary
2 FROM worker w
3 JOIN salaries s ON w.WorkerID = s.WorkerID
4 GROUP BY w.Department
5 HAVING COUNT(w.WorkerID) > 5
6 ORDER BY AverageSalary DESC;
```

השאלתה 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 מ-5 עובדים לבסוף מסדרים הכל בסדר יורד על בסיס המשכורת הממוצעת.

הרצה על בסיס הנתונים:

✓ Showing rows 0 - 9 (10 total, Query took 0.0545 seconds.) [AverageSalary: 24818.1234... - 24331.5631...]

```
SELECT w.Department, AVG(s.Net) AS AverageSalary
FROM worker w
JOIN salaries s ON w.WorkerID = s.WorkerID
GROUP BY w.Department
HAVING COUNT(w.WorkerID) > 5
ORDER BY AverageSalary DESC;
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

☐ Show all | Number of rows: 25 ▼ Filter rows:

Extra options

Department	AverageSalary ▼ 1
Marketing	24818.1234
Risk	24713.4061
Compliance	24659.1078
CustomerService	24651.7253
Legal	24627.1158

שאלת select 4:

שאלתה שמחפשת ומחזירה את הפרטים של כל הפקידים בדרגת Senior שעובדים באשקלון ויש להם משמרות של יותר מ-30 שעות בשבוע.

```
1 SELECT c.WorkerID, w.FirstName, w.LastName, w.Rank, c.ShiftHoursPW, b.BranchID, b.City
2 FROM clerk c
3 JOIN worker w ON c.WorkerID = w.WorkerID
4 JOIN branches b ON c.BranchID = b.BranchID
5 WHERE b.city = 'Ashkelon' AND w.Rank = 'Senior' AND c.ShiftHoursPW > 30;
6
7
```

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 BranchID, לאחר מכן 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 מ-30 שעות בשבוע.

הרצה על בסיס הנתונים:

✓ Showing rows 0 - 24 (124 total, Query took 0.0052 seconds.)

```
SELECT c.WorkerID, w.FirstName, w.LastName, w.Rank, c.ShiftHoursPW, b.BranchID, b.City
FROM clerk c
JOIN worker w ON c.WorkerID = w.WorkerID
JOIN branches b ON c.BranchID = b.BranchID
WHERE b.city = 'Ashkelon' AND w.Rank = 'Senior' AND c.ShiftHoursPW > 30;
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

1 ▾

> >>



Show all

Number of rows:

25 ▾

Filter rows:

Search this table

Extra options

WorkerID	FirstName	LastName	Rank	ShiftHoursPW	BranchID	City
14601	Donald	Ross	Senior	36	15	Ashkelon
19302	Tyler	Garcia	Senior	35	15	Ashkelon
28837	Carlos	Johnson	Senior	40	15	Ashkelon
14601	Donald	Ross	Senior	36	15	Ashkelon
19302	Tyler	Garcia	Senior	35	15	Ashkelon

שאלת delete :1

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

```
1 DELETE w
2   FROM worker w
3   JOIN branches b ON w.BranchID = b.BranchID
4   WHERE w.Rank = 'Junior' AND w.Department = 'Finance' AND b.City = 'Lod';
5
```

✓ Showing rows 0 - 24 (29 total, Query took 0.0124 seconds.)

```
SELECT w.*
FROM worker w
JOIN branches b ON w.BranchID = b.BranchID
WHERE w.Rank = 'Junior' AND w.Department = 'Finance' AND b.City = 'Lod';
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 ▾

>

>>

☐

Show all

Number of rows:

25 ▾

Filter rows:

Search this t

Extra options

WorkerID	FirstName	LastName	JoinDate	PhoneNumber	BranchID	Department	Rank
40	Scott	James	2019-03-16	0569911972	2	Finance	Junior
47	Ralph	Taylor	2008-05-04	0521429732	2	Finance	Junior
48	Ryan	Lee	2009-07-03	0595563656	2	Finance	Junior
5624	Anthony	Robinson	2010-04-13	0573497112	54	Finance	Junior
9682	Tyler	Cooper	2021-05-16	0583987114	91	Finance	Junior



```
1 SELECT w.*
2 FROM worker w
3 JOIN branches b ON w.BranchID = b.BranchID
4 WHERE w.Rank = 'Junior' AND w.Department = 'Finance' AND b.City = 'Lod';
5
```

Update query

Submit query

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0214 seconds.)

לפני ביצוע הפקודה:

לאחר ביצוע
הפקודה:
ניתן לראות
כי לא קיימים
יותר עובדים
בדרגת Junior
במחלקת
פיננסים בסניף
לוד.

שאלתת 2 delete:

שאלתת שמוחקת את כל המנהלים עם השכלה תיכונית ועם דירוג נמוך מ-30.

```
1 DELETE FROM manager
2 WHERE Education = 'High School Diploma' AND PerformanceRating < 30;
3
```

לפני ביצוע הפקודה:

✓ Showing rows 0 - 24 (235 total, Query took 0.0004 seconds.)

```
SELECT * FROM `manager` WHERE `PerformanceRating` < 30 AND `Education` LIKE 'High School Diploma'
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1

>

>>

☐

Show all

Number of rows:

25

Filter rows:

Search this table

Sort by key:

None

Extra options

		WorkerID	NumWorkersUHR	PerformanceRating	YearsOfExperience	Education	LevelOfAuthority
<input type="checkbox"/>	Edit Copy Delete	26	6	9	40	High School Diploma	Manage Operations
<input type="checkbox"/>	Edit Copy Delete	125	41	11	40	High School Diploma	Manage HR Department
<input type="checkbox"/>	Edit Copy Delete	203	44	17	19	High School Diploma	Manage IT Department
<input type="checkbox"/>	Edit Copy Delete	260	16	5	5	High School Diploma	Manage HR Department
<input type="checkbox"/>	Edit Copy Delete	317	42	19	3	High School Diploma	Manage Operations



לאחר ביצוע הפקודה:

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0012 seconds.)

```
SELECT * FROM manager WHERE Education = 'High School Diploma' AND PerformanceRating < 30;
```


שאלתת update 1:

שאלתה שמקדמת את כל העובדים בדרגת junior לדרגת senior בתנאי שהם עובדים בבנק יותר מ 5 שנים.

```
1 UPDATE worker
2   SET Rank = 'Senior'
3   WHERE Rank = 'Junior' AND DATEDIFF(CURDATE(), JoinDate) > 5 * 365;
4
```

טבלה לפני ההרצה:

```
SELECT * FROM worker WHERE Rank = 'Junior' AND DATEDIFF(CURDATE(), JoinDate) > 5 * 365;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

1 > >> | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

		WorkerID	FirstName	LastName	JoinDate	PhoneNumber	BranchID	Department	Rank
<input type="checkbox"/>	Edit Copy Delete	2	Joshua	Nguyen	2000-04-01	0595214287	1	Finance	Junior
<input type="checkbox"/>	Edit Copy Delete	5	Henry	Williams	1996-07-27	0599489637	1	HR	Junior
<input type="checkbox"/>	Edit Copy Delete	17	Carlos	Brooks	2017-08-19	0588836261	1	Operations	Junior
<input type="checkbox"/>	Edit Copy Delete	24	James	Ross	2007-10-30	0563079811	1	Marketing	Junior
<input type="checkbox"/>	Edit Copy Delete	25	Peter	Torres	2015-02-25	0569864288	1	Legal	Junior
<input type="checkbox"/>	Edit Copy Delete	27	Paul	Cox	1996-09-22	0512065484	1	Legal	Junior



לאחר ההרצה:

		WorkerID	FirstName	LastName	JoinDate	PhoneNumber	BranchID	Department	Rank
<input type="checkbox"/>	Edit Copy Delete	2	Joshua	Nguyen	2000-04-01	0595214287	1	Finance	Senior
<input type="checkbox"/>	Edit Copy Delete	5	Henry	Williams	1996-07-27	0599489637	1	HR	Senior
<input type="checkbox"/>	Edit Copy Delete	13	Gabriel	Allen	2000-01-25	0512855479	1	CustomerService	Senior
<input type="checkbox"/>	Edit Copy Delete	14	Arthur	King	2011-06-25	0566004914	1	CustomerService	Senior
<input type="checkbox"/>	Edit Copy Delete	17	Carlos	Brooks	2017-08-19	0588836261	1	Operations	Senior
<input type="checkbox"/>	Edit Copy Delete	24	James	Ross	2007-10-30	0563079811	1	Marketing	Senior
<input type="checkbox"/>	Edit Copy Delete	25	Peter	Torres	2015-02-25	0569864288	1	Legal	Senior
<input type="checkbox"/>	Edit Copy Delete	27	Paul	Cox	1996-09-22	0512065484	1	Legal	Senior

שאלתת 2 update:

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

```
1 UPDATE salaries s
2 JOIN worker w ON s.WorkerID = w.WorkerID
3 SET s.Net = s.Net * 1.10
4 WHERE w.Rank = 'Director' AND w.BranchID IN (
5     SELECT BranchID
6     FROM branches
7     WHERE City = 'Jerusalem'
8 );
9
```

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

לפני:

אחרי:

WorkerID	FirstName	Net
5363	Anthony	11013
5372	Ryan	15484
5381	Jason	24787
5384	Zachary	32340
5392	Christian	19539
5394	Ethan	31236
5401	Benjamin	12910



WorkerID	FirstName	Net
5363	Anthony	12114
5372	Ryan	17032
5381	Jason	27266
5384	Zachary	35574
5392	Christian	21493
5394	Ethan	34360
5401	Benjamin	14201

שאלתה 1 עם פרמטר:

השאלתה משתמשת בפרמטר אחד BranchID, היא בוחרת נתונים של עובדים מסניף מסוים, בשאלתה הזו 101.

```
1 PREPARE stmt FROM '  
2 SELECT w.WorkerID, w.FirstName, w.LastName, w.Department, w.Rank  
3 FROM worker w WHERE w.BranchID = ?';  
4  
5 SET @branch_id = 101;  
6 EXECUTE stmt USING @branch_id;  
7 DEALLOCATE PREPARE stmt;
```

ראשית השאלתה מכינה את ה"שאלתה" ונותנת את זה ל handler בשם stmt, במקום המתאים עבור branchID ישנו סימן שאלה וזה בעצם מחזיק מקום (placeholder).

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

✓ Showing rows 0 - 24 (192 total, Query took 0.0145 seconds.)

```
EXECUTE stmt USING @branch_id;
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

Extra options

WorkerID	FirstName	LastName	Department	Rank
10775	Lawrence	Walker	Finance	Intern
10776	Charles	Foster	Finance	MidLevel
10777	Carlos	Torres	Finance	Senior
10778	Jonathan	Robinson	Finance	Junior
10779	Billy	Mitchell	Finance	Senior

שאלתה 2 עם פרמטר:

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

```
1 PREPARE stmt FROM '  
2 SELECT w.Department, COUNT(w.WorkerID) AS TotalWorkers FROM worker w  
3 WHERE w.Department IN (?, ?, ?)  
4 GROUP BY w.Department';  
5  
6 SET @dept1 = 'Finance', @dept2 = 'HR', @dept3 = 'IT';  
7 EXECUTE stmt USING @dept1, @dept2, @dept3;  
8 DEALLOCATE PREPARE stmt;  
9  
10
```

השאלתה ראשית מכינה את ה"שאלתה" והכל מוחזק על ידי stmt, השאלתה 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

Department	TotalWorkers
Finance	4543
HR	4505
IT	4454

שאלתה 3 עם פרמטר:

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

```
1 PREPARE stmt FROM '  
2 SELECT w.WorkerID, w.FirstName, w.LastName, w.JoinDate  
3 FROM worker w  
4 WHERE w.JoinDate > ?';  
5  
6 SET @join_date = '2020-01-01';  
7 EXECUTE stmt USING @join_date;  
8 DEALLOCATE PREPARE stmt;  
9
```

ראשית השאלתה מכינה את ה"שאלתה" 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 השאלתה עם המשתנה שהגדרנו, והיא תחזיר את כל העובדים שהצטרפו לארגון לאחר התאריך הנתון.

הצבנו את התאריך 1 לינואר 2020 בפרמטר והרצנו:

WorkerID	FirstName	LastName	JoinDate
1	Randy	Murphy	2022-02-20
8	Justin	Scott	2022-01-19
11	Donald	Kim	2024-02-01
16	Stephen	Howard	2024-01-03
18	Victor	Lewis	2023-01-21

שאלתה 4 עם פרמטר:

השאלתה 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.

```
1 PREPARE stmt FROM '  
2 SELECT s.WorkerID, s.LastUpdate, s.Net  
3 FROM salaries s  
4 WHERE s.LastUpdate BETWEEN ? AND ?';  
5  
6 SET @start_date = '2023-01-01', @end_date = '2023-12-31';  
7 EXECUTE stmt USING @start_date, @end_date;  
8 DEALLOCATE PREPARE stmt;
```

השאלתה מכינה את ה"שאלתה" שלוקחת תז תאריך עדכון 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 השאלתה 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:

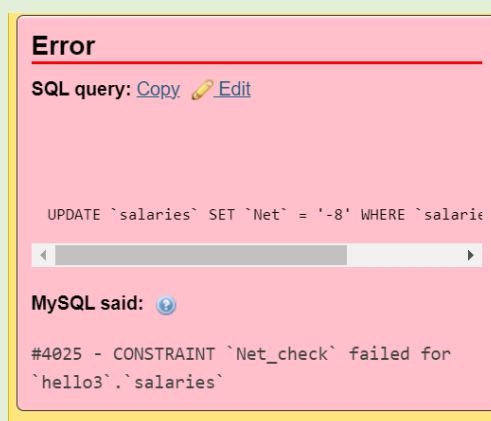
WorkerID	LastUpdate	Net
5	2023-12-29	9849
14	2023-06-16	19763
16	2023-02-15	35119
49	2023-06-02	22438
54	2023-03-15	30652

אילוח 1:

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

```
1 ALTER TABLE salaries
2   ADD CONSTRAINT Net_check CHECK (Net >= 0);
```

נסיון של להכניס ערך -8 לשכר נפגש בשגיאה

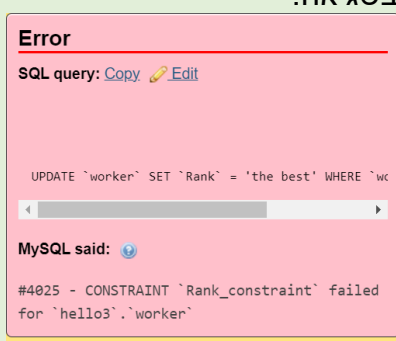


אילוח 2:

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

```
1 ALTER TABLE Worker
2   ADD CONSTRAINT rank_constraint CHECK (Rank IN ('Junior', 'Intern', 'MidLevel', 'Manager', 'Senior', 'Expert',
3   'Director'));
```

נסיון של הכנסת ערך דרגה the best נתקל בשגיאה:

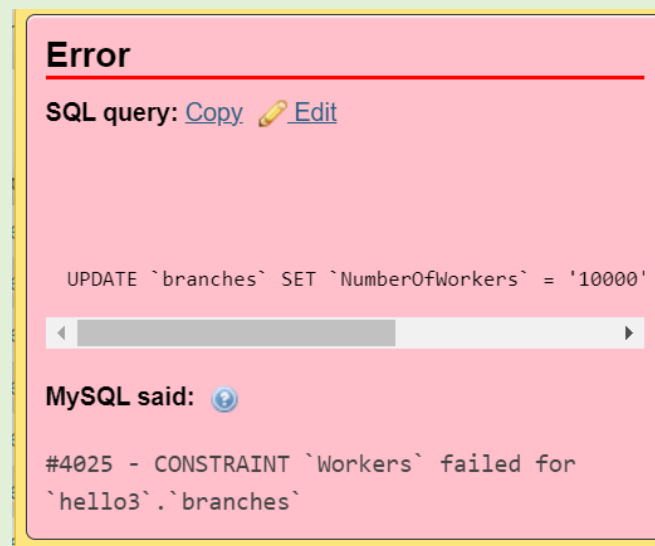


אילוח 3:

מאלץ שלכל סניף יהיה מקסימום של 999 עובדים.

```
1 ALTER TABLE branches
2 ADD CONSTRAINT Workers CHECK (NumberOfWorkers <=1000);
3
```

נסיון להכניס ערך 10000 במספר העובדים בסניף נתקל בשגיאה:



שלב 3

פונקציה מס' 1 – AvgManagerPerformance

הפונקציה מחשבת את ממוצע ציוני הביצוע (PerformanceRating) של כל המנהלים בסניף מסויים הניתן כפרמטר:

```
DELIMITER $$
--Function that calculates the average PerformanceRating of managers in specific branch
CREATE FUNCTION AvgManagerPerformance(branch_id INT)
RETURNS DECIMAL(5,2)
BEGIN
    DECLARE avg_rating DECIMAL(5,2);

    SELECT AVG(PerformanceRating) INTO avg_rating
    FROM manager m
    JOIN worker w ON m.WorkerID = w.WorkerID
    WHERE w.BranchID = branch_id;

    RETURN IFNULL(avg_rating, 0);
END$$

DELIMITER;
```

הפונקציה מקבלת כפרמטר את מספר הסניף, מבצעת שאילתת select על צירוף הטבלאות עובד ומנהל כשמספר הסניף הוא הניתן בפרמטר מחשבת ממוצע (AVG) ציוני ביצוע ושומרת את זה במשתנה avg_rating, ומחזירה אותו אם הוא לא ריק.

הרצה:

שמרנו את הפונקציה בבסיס הנתונים שלנו, וכעת הרצנו אותה עם פרמטר מספר סניף : 91

```
SET @p0='91'; SELECT `AvgManagerPerformance`(@p0) AS `AvgManagerPerformance`;
```

Execution results of routine `AvgManagerPerformance`

AvgManagerPerformance
93.00

יצא שממוצע ציוני הביצועים של המנהלים בסניף זה הוא 93!

פרוצדורה מס' 1 – GiveRaise

הפרוצדורה מקבלת מספר סניף, מספר מחלקה ואחוז – ומבצעת העלאה בשכר במספר אחוזים הניתן כפרמטר, לכל העובדים במחלקה ובסניף הנתונים בתנאי שיש למחלקה מנהל עם ציון ביצועים מעל 80.

```
DELIMITER $$
-- Procedure that gives raise in salary to all workers who works in the department
-- only if the manager have more then 80 PreformanceRating in the specifice department
CREATE PROCEDURE GiveRaise(branch_id INT, department_name VARCHAR(20), raise_percentage
DECIMAL(5,2))
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE worker_id INT;
    DECLARE new_salary INT;
    DECLARE cur CURSOR FOR
        SELECT w.WorkerID, s.Net
        FROM worker w
        JOIN salaries s ON w.WorkerID = s.WorkerID
        WHERE w.BranchID = branch_id AND w.Department = department_name
        AND w.Department IN (SELECT Department FROM manager m JOIN worker w ON m.WorkerID =
w.WorkerID WHERE PerformanceRating > 80);

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Exception handling
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        -- Rollback the actions when Exception is caught
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'An error occurred while giving raises.';
    END;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO worker_id, new_salary;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Calculate new salary
        SET new_salary = new_salary + (new_salary * raise_percentage / 100);

        -- Update the salary
        UPDATE salaries
        SET Net = new_salary
        WHERE WorkerID = worker_id;
    END LOOP;

    CLOSE cur;

    COMMIT;
END$$

DELIMITER ;
```

הפרוצדורה מגדירה Cursor בשם cur שתפקידו לרוץ בלולאה על שאילתת select שלוקחת את כל המשכורות של כל העובדים במחלקה וסניף הנתונים שיש לו מנהל עם ציון מעל 80 ושומרת במשתנה את מספר העובד ומשכורתו הנוכחית. לאחר מכן מחשבת במשתנה חדש new_salary את המשכורת החדשה הכוללת את ההעלאה בשכר. ומעדכנת ע"י שאילתת UPDATE את המשכורות לעובדים הרלוונטיים, בסוף ניתן ליראות טיפול בחריגות שיכולות להיות – טיפול הכולל ביצוע rollback במטרה לשמור על איכות הנתונים שבסיס הנתונים במידה ונהרסו או נפגמו.

הרצה:

שמרנו את הפרוצדורה במסד הנתונים, ובדקנו את השכר של כל העובדים במחלקת CostumerService בסניף 91:

```
SELECT w.WorkerID,w.Department, s.Net AS Salary FROM worker w JOIN salaries s ON w.WorkerID = s.WorkerID WHERE w.BranchID = 91 AND w.Department = 'CustomerService';
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

WorkerID	Department	Salary
9689	CustomerService	28263
9690	CustomerService	13101
9691	CustomerService	28162

וכעת הרצנו את הפרוצדורה עם הפרמטרים: סניף 91 מחלקה CostumerService ו20 אחוז

```
SET @p0='91'; SET @p1='CustomerService'; SET @p2='20'; CALL `GiveRaise2`(@p0, @p1, @p2);
```

Execution results of routine 'GiveRaise2'

כעת נבדוק שוב ונראה שאכן עודכנו המשכורות וכל העובדים במחלקה קיבלו העלה של 20 אחוז:

Showing rows 0 - 2 (3 total, Query took 0.1004 seconds.)

```
SELECT w.WorkerID,w.Department, s.Net AS Salary FROM worker w JOIN salaries s ON w.WorkerID = s.WorkerID WHERE w.BranchID = 91 AND w.Department = 'CustomerService';
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

WorkerID	Department	Salary
9689	CustomerService	33916
9690	CustomerService	15721
9691	CustomerService	33794

```
DELIMITER $$
-- main program that iterate through each branch, check if the average performance rating of
managers in the branch is above a
-- a threshold given in the parameter "threshold_rating" and if it is - call the GiveRaise
procedure to give raises of "raise_percentage"
-- parameter to all the good workers in all departments of that branch.
CREATE PROCEDURE MainProgram(threshold_rating DECIMAL(5,2), raise_percentage DECIMAL(5,2))
BEGIN
    DECLARE branch_id INT;
    DECLARE department_name VARCHAR(20);
    DECLARE avg_performance DECIMAL(5,2);
    DECLARE done_branches INT DEFAULT FALSE;
    DECLARE done_departments INT DEFAULT FALSE;

    -- Cursor to iterate through all branches
    DECLARE cur_branches CURSOR FOR
        SELECT BranchID FROM branches;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done_branches = TRUE;

    -- Open the branches cursor
    OPEN cur_branches;

    -- Loop through each branch
    branch_loop: LOOP
        FETCH cur_branches INTO branch_id;

        IF done_branches THEN
            LEAVE branch_loop;
        END IF;

        -- Calculate the average performance rating for the current branch
        SET avg_performance = AvgManagerPerformance(branch_id);

        -- Check if the average performance rating is above the threshold
        IF avg_performance > threshold_rating THEN

            -- Cursor to iterate through all departments in the current branch
            DECLARE cur_departments CURSOR FOR
                SELECT DepartmentName FROM departments WHERE BranchID = branch_id;

            DECLARE CONTINUE HANDLER FOR NOT FOUND SET done_departments = TRUE;

            -- Open the departments cursor
            OPEN cur_departments;

            -- Loop through each department in the branch
            department_loop: LOOP
                FETCH cur_departments INTO department_name;

                IF done_departments THEN
                    LEAVE department_loop;
                END IF;

                -- Call the procedure to give raises to eligible workers in the current
                department
                CALL GiveRaise(branch_id, department_name, raise_percentage);

            END LOOP department_loop;

            -- Close the departments cursor
            CLOSE cur_departments;
        END IF;

    END LOOP branch_loop;

    -- Close the branches cursor
    CLOSE cur_branches;

    -- Exception handling
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        -- Rollback the transaction in case of error
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'An error occurred in the main program.';
    END;

    COMMIT;
END$$
DELIMITER ;
```

תיאור התכנית: התכנית מקבלת 2 פרמטרים: רף הצטיינות באחוזים ואחוז העלאה בשכר. התכנית רצה על כל סניף, בודקת ע"י שימוש בפונקציה את ממוצע ציוני הביצועים של כל המנהלים בסניף -אם הוא גדול מהרף המתקבל בפרמטר – קוראת לפרוצדורה GiveRaise ונותנת העלאה של האחוז שניתן כפרמטר לכל העובדים בכל המחלקות המצטיינות בסניף (שבהם המנהל עם ציון מעל 80).

הרצת התכנית הראשית:

ראשית, נמצא את כל הסניפים בהם ממוצע ציוני הביצועים של המנהלים הוא גדול מ80:

Showing rows 0 - 4 (5 total, Query took 5.4491 seconds.)

```
SELECT w.BranchID, AVG(m.PerformanceRating) AS AveragePerformanceRating FROM manager m JOIN worker w ON m.WorkerID = w.WorkerID GROUP BY w.BranchID HAVING AvgManagerPerformance(w.BranchID) > 80;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

BranchID	AveragePerformanceRating
17	80.2500
91	93.0000
160	93.5000
291	85.7500
356	92.0000

קיבלנו 5 סניפים כאלו. כעת נציג את כל משכורות העובדים בסניפים אלו שנמצאים במחלקה שבה יש מנהל עם ציון ביצועים מעל 80 (מחלקה מצטיינת):

Showing rows 0 - 197 (198 total, Query took 0.1866 seconds.)

```
SELECT w.WorkerID, w.Department, w.BranchID, s.Net AS Salary FROM worker w JOIN salaries s ON w.WorkerID = s.WorkerID WHERE w.Department IN ( SELECT DISTINCT w2.Department FROM worker w2 JOIN manager m ON w2.WorkerID = m.WorkerID WHERE m.PerformanceRating > 80 ) AND (w.BranchID = '17' OR w.BranchID = '91' OR w.BranchID = '160' OR w.BranchID = '291' OR w.BranchID = '356');
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 250 | Filter rows: Search this table

Extra options

WorkerID	Department	BranchID	Salary
1912	Finance	17	21838
1913	Finance	17	14969
1914	Finance	17	24275
1915	Finance	17	17241
1916	Finance	17	38269
1917	Finance	17	14542
1918	HR	17	14887
1919	HR	17	17109
1920	HR	17	35704
1921	HR	17	31843
1971	Risk	17	20812
9680	Finance	91	15483
9681	Finance	91	19965
9682	Finance	91	17979
9683	HR	91	35119
9684	HR	91	10324
9706	Risk	91	16241
17348	Finance	160	14867
17349	Finance	160	30495
17350	Finance	160	15468
17351	HR	160	31917
17352	HR	160	14915

(נציג מדגמים מכל מיני סניפים כי מספר השורות בטבלה זו יצא לנו 200-250):

כעת נפעיל את התכנית הראשית עם הפרמטרים: רף הצטיינות: 80, העלה בשכר: 20

```
SET @p0='80'; SET @p1='20'; CALL `MainProgram`(@p0, @p1);
```

עכשיו נפעיל **שוב** את השאילתה המראה לנו את כל משכורות העובדים בסניפים שממוצע ציוני הביצועים שלהם הוא מעל 80 ושנמצאים במחלקה שבה יש מנהל עם ציון ביצועים מעל 80 (מחלקה מצטיינת)
ונראה כיצד המשכורת של כל העובדים הללו עלתה ב-20%:

```
SELECT w.WorkerID, w.Department, w.BranchID, s.Net AS Salary FROM worker w JOIN salaries s ON w.WorkerID = s.WorkerID WHERE w.Department IN ( SELECT DISTINCT w2.Department FROM worker w2 JOIN manager m ON w2.WorkerID = m.WorkerID WHERE m.PerformanceRating > 80 ) AND (w.BranchID = '17' OR w.BranchID = '91' OR w.BranchID = '160' OR w.BranchID = '291' OR w.BranchID = '356');
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

1 > >> ☐ Show all Number of rows: 25 Filter rows: Search this table

Extra options

WorkerID	Department	BranchID	Salary
1912	Finance	17	26206
1913	Finance	17	17963
1914	Finance	17	29130
1915	Finance	17	20689
1916	Finance	17	45923
1917	Finance	17	17450
1918	HR	17	17864
1919	HR	17	20531
1920	HR	17	42845
1921	HR	17	38212

1971	Risk	17	24974
9680	Finance	91	18580
9681	Finance	91	23958
9682	Finance	91	21575
9683	HR	91	42143
9684	HR	91	12389

9706	Risk	91	19489
17348	Finance	160	17840
17349	Finance	160	36594
17350	Finance	160	18562
17351	HR	160	38300
17352	HR	160	17898

(נציג מדגמים מכל מיני סניפים כי מספר השורות בטבלה זו יצא לנו 200-250):

פונקציה מס' 1 – avg_branch_salary

הפונקציה מחשבת את ממוצע השכר של סניף שנתון בקלט. הפלט יהיה הממוצע.

```
1 DELIMITER $$
2 CREATE DEFINER=`root`@`localhost` FUNCTION `avg_branch_salary`(branch_id INT) RETURNS decimal(10,2)
3 DETERMINISTIC
4 BEGIN
5     DECLARE avg_salary DECIMAL(10,2);
6
7     SELECT AVG(Net)
8     INTO avg_salary
9     FROM salaries s
10    JOIN worker w ON s.WorkerID = w.WorkerID
11   WHERE w.BranchID = branch_id;
12
13     RETURN avg_salary;
14 END$$
15 DELIMITER ;
```

הקלט של הפונקציה הוא מספר מזהה הסניף עליו נחשב את הממוצע. הפרוצדורה מצרפת את טבלאות המשכורת והעובדים על בסיס מספר זהות. ומחזירה את השכר הממוצע של כל העובדים בסניף וזה יהיה הפלט.

הרצה:

נחשב את השכר הממוצע של סניף מספר 100:

```
SET @p0='100'; SELECT `avg_branch_salary`(@p0) AS `avg_branch_salary`;
```

Execution results of routine `avg_branch_salary`

avg_branch_salary
24419.50

פרוצדורה מס' 2 – raise_salary

הפרוצדורה מעלה את השכר בסניף ב-5% אם השכר הממוצע שלו קטן מ-26000.

```
1 DELIMITER $$
2 CREATE DEFINER='root'@'localhost' PROCEDURE `raise_salary`(IN `branch_id` INT)
3 BEGIN
4     DECLARE avg_salary DECIMAL(10,2);
5     DECLARE done INT DEFAULT 0;
6     DECLARE worker_id INT;
7     DECLARE cur CURSOR FOR
8         SELECT w.WorkerID
9         FROM worker w
10        JOIN salaries s ON w.WorkerID = s.WorkerID
11        WHERE w.BranchID = branch_id;
12    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
13
14    -- average salary
15    SET avg_salary = avg_branch_salary(branch_id);
16
17    -- If the average salary is under 26000, give a 5% raise to all workers in branch
18    IF avg_salary < 26000 THEN
19        OPEN cur;
20
21        read_loop: LOOP
22            FETCH cur INTO worker_id;
23            IF done THEN
24                LEAVE read_loop;
25            END IF;
26
27            BEGIN
28
29                DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
30                BEGIN
31
32                END;
33                UPDATE salaries SET Net = Net * 1.05 WHERE WorkerID = worker_id;
34            END;
35        END LOOP;
36
37        CLOSE cur;
38    END IF;
39 END$$
40 DELIMITER ;
```

קלט הפרוצדורה זה מספר מזהה הסניף הפרוצדורה משתמשת בפונקציה avg_branch_salary שמחזירה את ממוצע המשכורות של הסניף, לאחר מכן יש בדיקה אם השכר הממוצע נמוך מ-26000 אז יש לולאה שמעדכנת את כל המשכורות של העובדים ומעלה אותם ב-5%. ישנו טיפול בחריגים במהלך תהליך העדכון.

הרצה:

נריץ את הפרוצדורה על סניף מספר 100 שראינו שהשכר הממוצע שלו הוא 24419.5:

```
SET @p0='100'; CALL `raise_salary`(@p0);
```

וכעת נראה שהשכר הממוצע בסניף זה עלה ב-5 אחוז! :

<pre>SELECT `avg_branch_salary`(@p0) AS `avg_branch_salary`;</pre>	
<input type="checkbox"/> Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]	
<input type="checkbox"/> Show all	Number of rows: 25 <input type="button" value="v"/>
Filter rows: <input type="text" value="Search this"/>	
<input type="button" value="Extra options"/>	
avg_branch_salary	
	25640.52

פונקציה ראשית 2 -

הפונקציה הראשית הזו עוברת על כל הסניפים ומפעילה את הפרוצדורה שמעלה ב5 אחוז רק אם ממוצע המשכורות הוא מתחת ל26000 ע"י קריאה לפונקציית חישוב הממוצע בכל סניף.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`process_all_branches`()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE branch_id INT;
    DECLARE cur CURSOR FOR SELECT BranchID FROM branches;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO branch_id;
        IF done THEN
            LEAVE read_loop;
        END IF;

        BEGIN
            DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
            BEGIN

            END;

            CALL raise_salary(branch_id);
        END;
    END LOOP;

    CLOSE cur;
END$$
DELIMITER ;
```

הפונקציה הראשית הזו פותחת לולאה ועוברת על כל הסניפים, עבור כל סניף מפעילה את הפרוצדורה של העלאת שכר, והיא בתורה מפעילה את הפונקציה לחישוב שכר ממוצע, כאמור השכר של כל עובד יעלה אם השכר הממוצע קטן מ26000, גם כאן ישנו טיפול בחריגים (לא עושה כלום).

ריצה:

להלן הממוצע של המשכורות של כמה סניפים לדוגמא:

```
1 SET @p0='3'; SELECT `avg_branch_salary`(@p0) AS `avg_branch_salary1`;  
2 SET @p0='122'; SELECT `avg_branch_salary`(@p0) AS `avg_branch_salary2`;  
3 SET @p0='256'; SELECT `avg_branch_salary`(@p0) AS `avg_branch_salary3`;  
4 SET @p0='311'; SELECT `avg_branch_salary`(@p0) AS `avg_branch_salary4`;
```

avg_branch_salary1
23533.54

avg_branch_salary2
23986.18

avg_branch_salary3
24445.69

avg_branch_salary4
25754.57

וכעת נפעיל את התכנית הראשית:

```
CALL `process_all_branches`();
```

ונראה איך ממוצע המשכורות באותם סניפים שראינו שהיה נמוך מ26000 עלה ב5 אחוז:

avg_branch_salary1
24710.26

avg_branch_salary2
25185.48

avg_branch_salary3
25667.96

avg_branch_salary4
27042.36

