מיני פרויקט בבסיסי נתונים

פרויקט: בסיס נתונים עבור בנק

היחידה הנבחרת: מידע ומשכורות של עובדים

מגישים:
אלי ארזי 336301304
אלעד רדומסקי 318645850

תוכן עניינים:

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

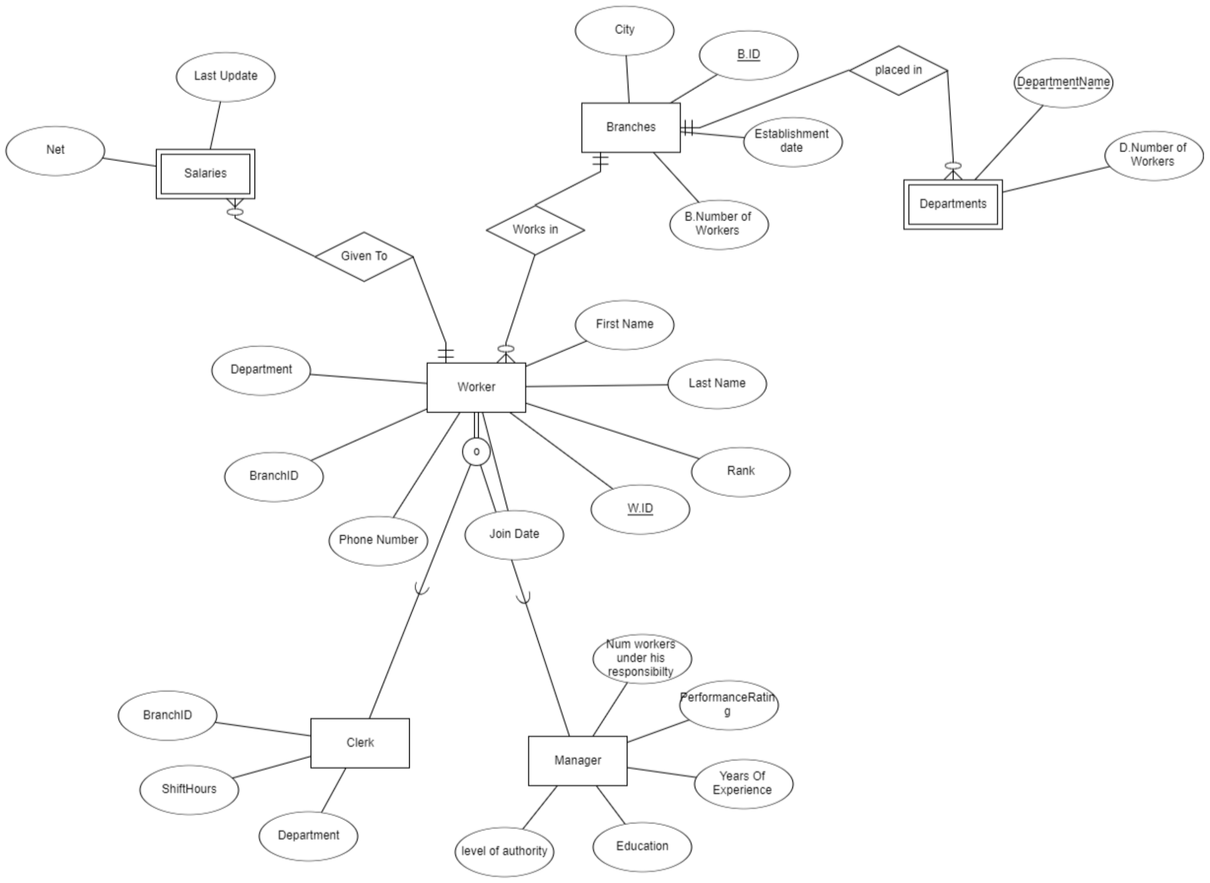
שכר של יעקב 8k
שכר של יצחק 8k
שכר של אברהם 15k
שכר של משה: 15k
שכר של אהרון: 13k

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

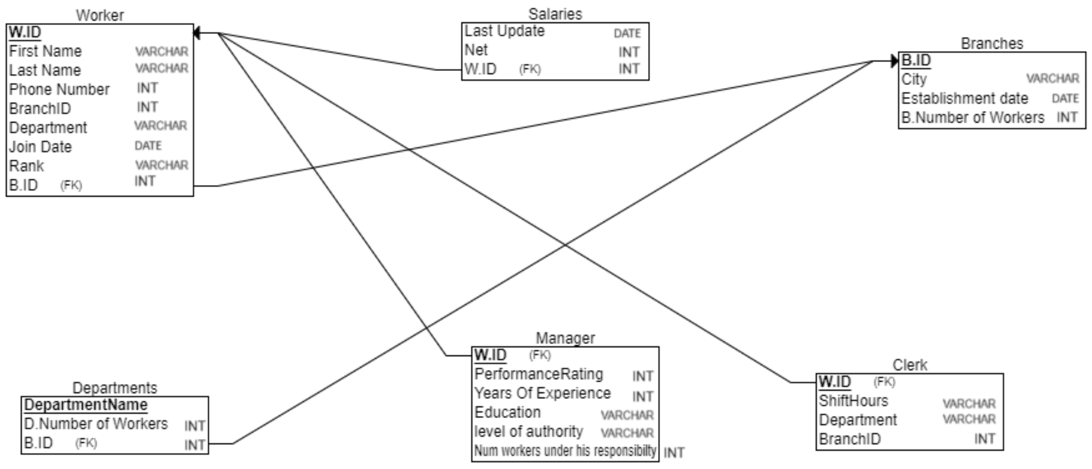
מנהל סניף תל אביב: אברהם
מנהל סניף ירוחם: משה

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



Worker

W.ID		
First Name		VARCHAR
Last Name		VARCHAR
Phone Number		INT
BranchID		INT
Department		VARCHAR
Join Date		DATE
Rank		VARCHAR
B.ID	(FK)	INT

Salaries

Last Update		DATE
Net		INT
W.ID	(FK)	INT

Branches

B.ID	
City	VARCHAR
Establishment date	DATE
B.Number of Workers	INT

Departments

DepartmentName		
D.Number of Workers		INT
B.ID	(FK)	INT

Manager

W.ID	(FK)	
PerformanceRating		INT
Years Of Experience		INT
Education		VARCHAR
level of authority		VARCHAR
Num workers under his responsibilty		INT

Clerk

W.ID	(FK)	
ShiftHours		VARCHAR
Department		VARCHAR
BranchID		INT

Create Tables

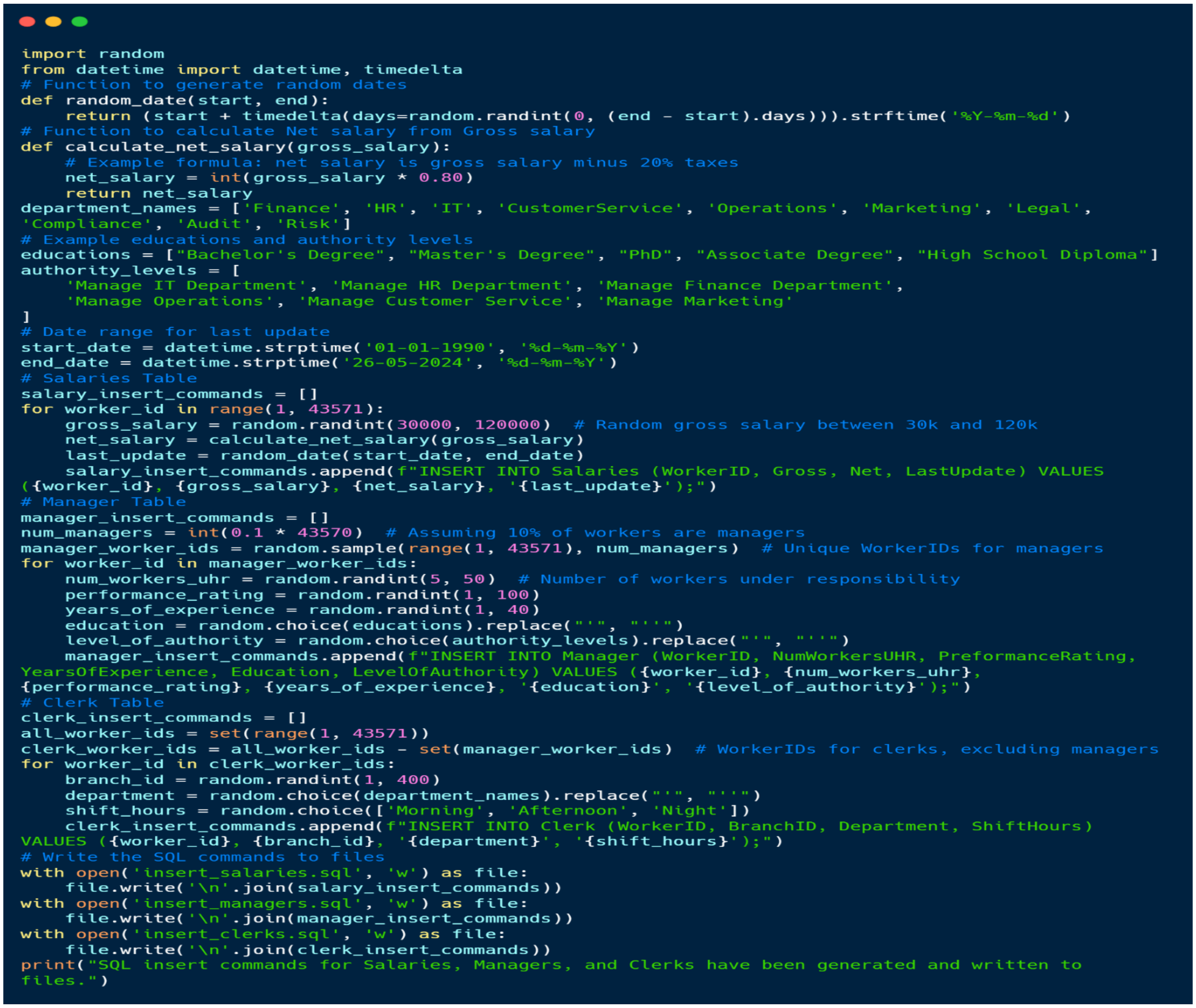
```
CREATE TABLE `branches` (
 `BranchID` int(11) NOT NULL AUTO_INCREMENT,
 `City` varchar(20) NOT NULL,
 `EstablishmentDate` date NOT NULL,
 `NumberOfWorkers` int(11) NOT NULL,
 PRIMARY KEY (`BranchID`)
)


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`)
)

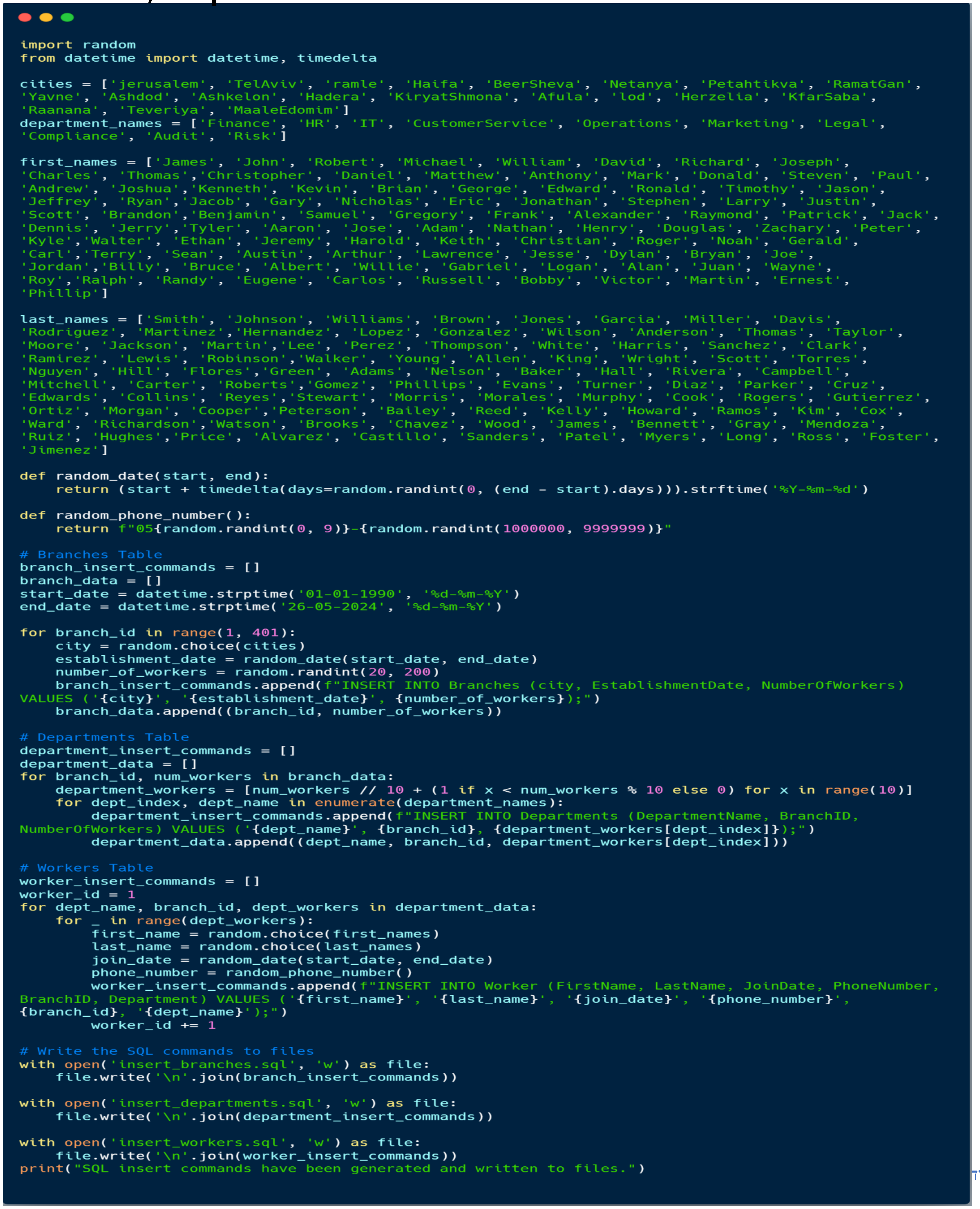
Python script-data generator

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, PreformanceRating,
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.")
```

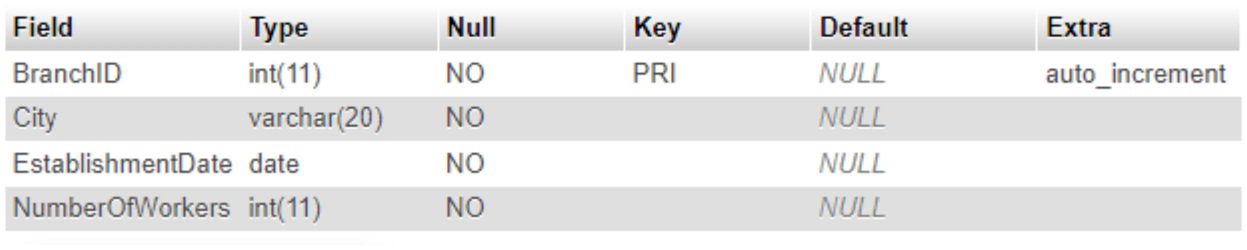
Branches, Departments and Workers Tables:



```python
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.")
```

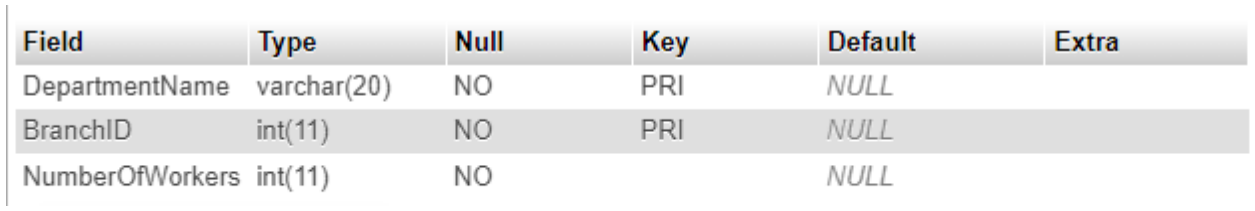
DESC commands:

```
DESC branches;
```



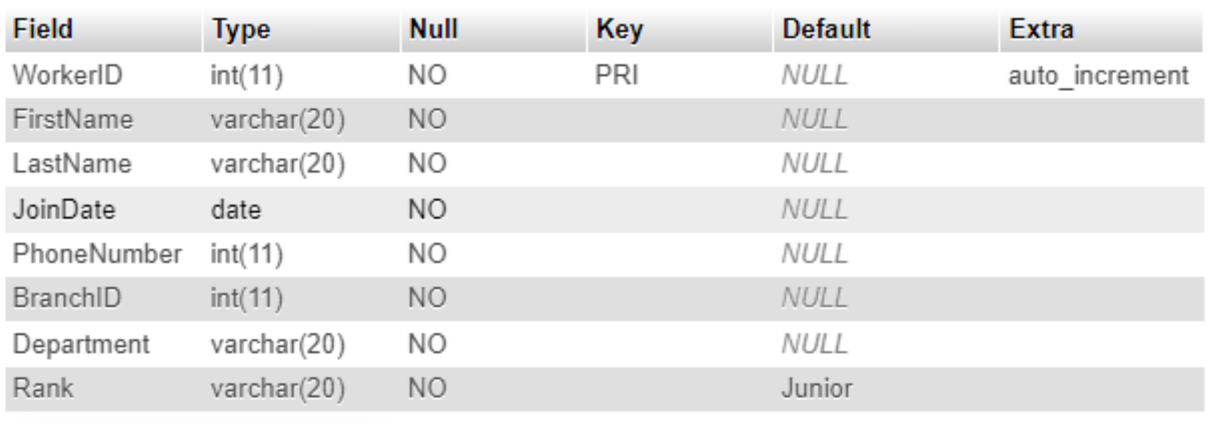
Field	Type	Null	Key	Default	Extra
BranchID	int(11)	NO	PRI	*NULL*	auto_increment
City	varchar(20)	NO		*NULL*	
EstablishmentDate	date	NO		*NULL*	
NumberOfWorkers	int(11)	NO		*NULL*	

```
DESC departments;
```



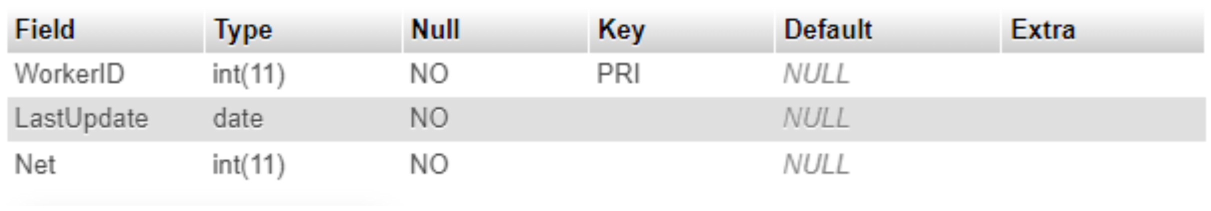
Field	Type	Null	Key	Default	Extra
DepartmentName	varchar(20)	NO	PRI	*NULL*	
BranchID	int(11)	NO	PRI	*NULL*	
NumberOfWorkers	int(11)	NO		*NULL*	

```
DESC worker;
```



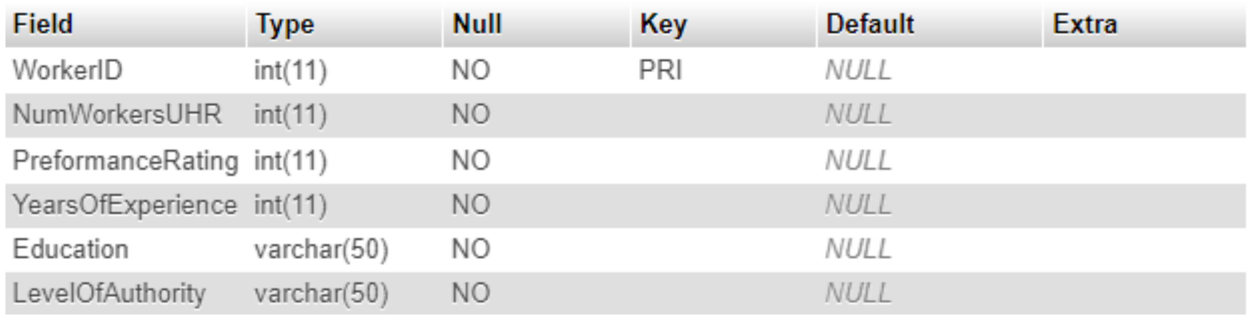
Field	Type	Null	Key	Default	Extra
WorkerID	int(11)	NO	PRI	*NULL*	auto_increment
FirstName	varchar(20)	NO		*NULL*	
LastName	varchar(20)	NO		*NULL*	
JoinDate	date	NO		*NULL*	
PhoneNumber	int(11)	NO		*NULL*	
BranchID	int(11)	NO		*NULL*	
Department	varchar(20)	NO		*NULL*	
Rank	varchar(20)	NO		Junior	

```
DESC salaries;
```



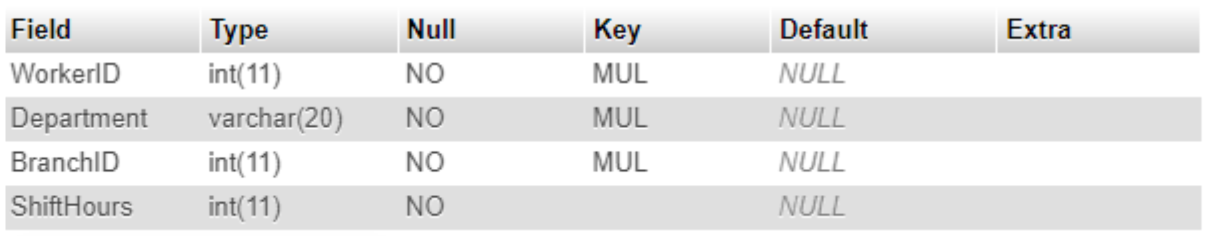
Field	Type	Null	Key	Default	Extra
WorkerID	int(11)	NO	PRI	*NULL*	
LastUpdate	date	NO		*NULL*	
Net	int(11)	NO		*NULL*	

```
DESC manager;
```



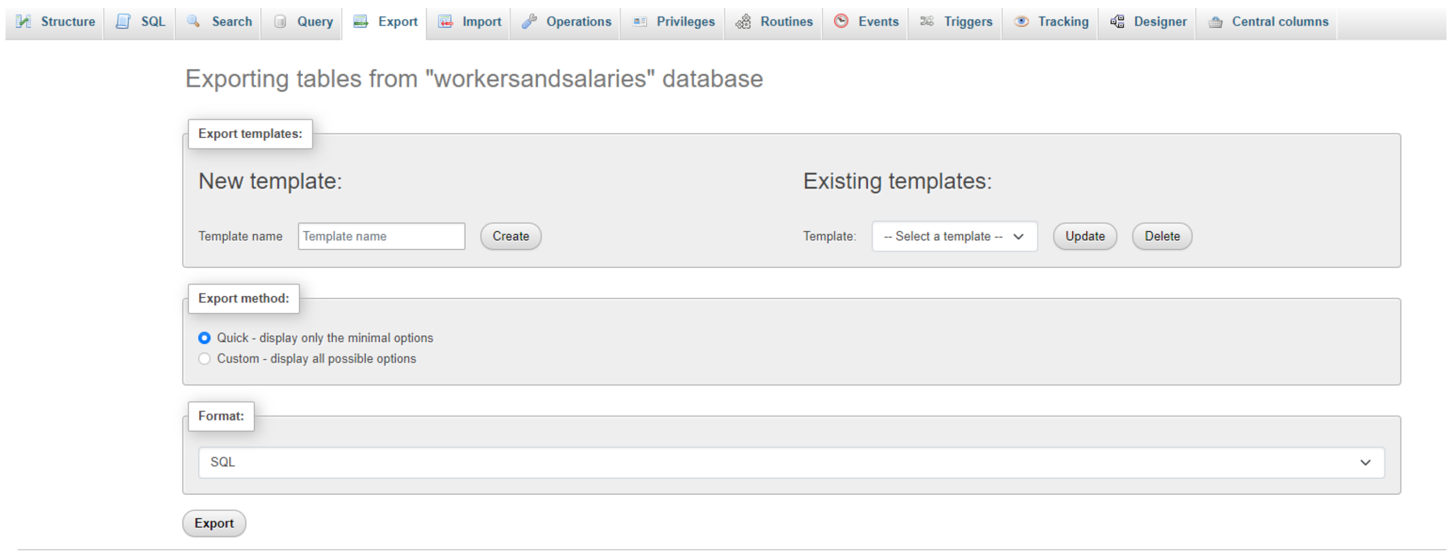
Field	Type	Null	Key	Default	Extra
WorkerID	int(11)	NO	PRI	*NULL*	
NumWorkersUHR	int(11)	NO		*NULL*	
PreformanceRating	int(11)	NO		*NULL*	
YearsOfExperience	int(11)	NO		*NULL*	
Education	varchar(50)	NO		*NULL*	
LevelOfAuthority	varchar(50)	NO		*NULL*	

```
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:



Exporting tables from "workersandsalaries" database

Export templates:

New template:

Template name: Template name [Create]

Existing templates:

Template: -- Select a template -- [Update] [Delete]

Export method:

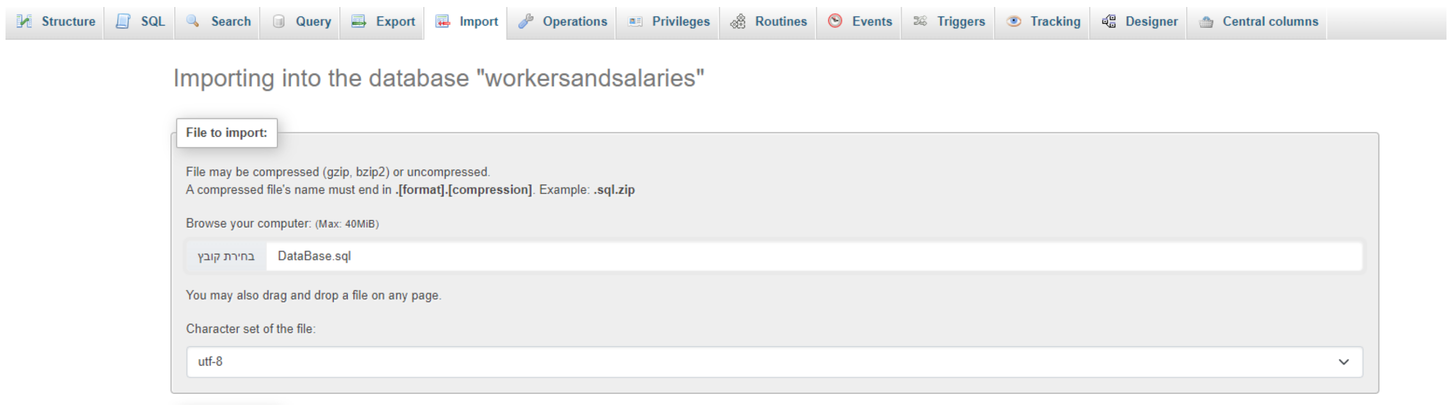
[✓ Quick - display only the minimal options]
Custom - display all possible options

Format:

SQL

[Export]

שחזור בסיס הנתונים:



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 DataBase.sql

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

Character set of the file:

utf-8