Whitford Country Club

A Database Project by Mark Miller

Table of Contents:

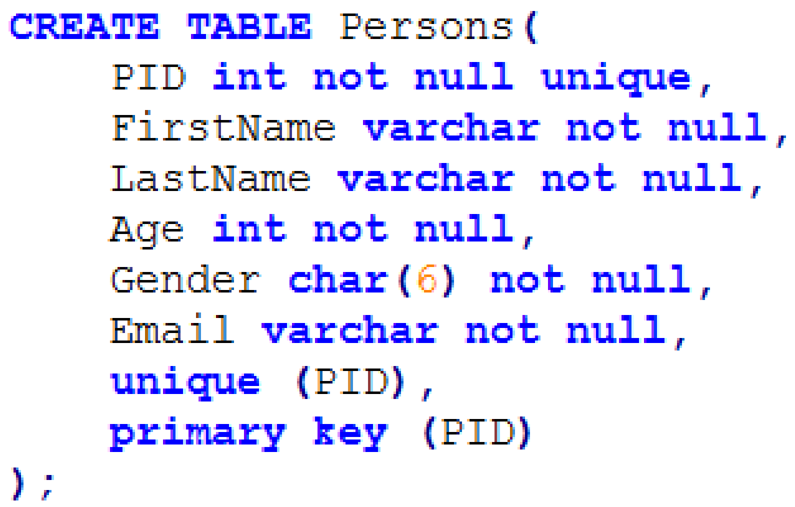
Executive Summary:

ER-Diagram

Create Table Statements:

Persons Table

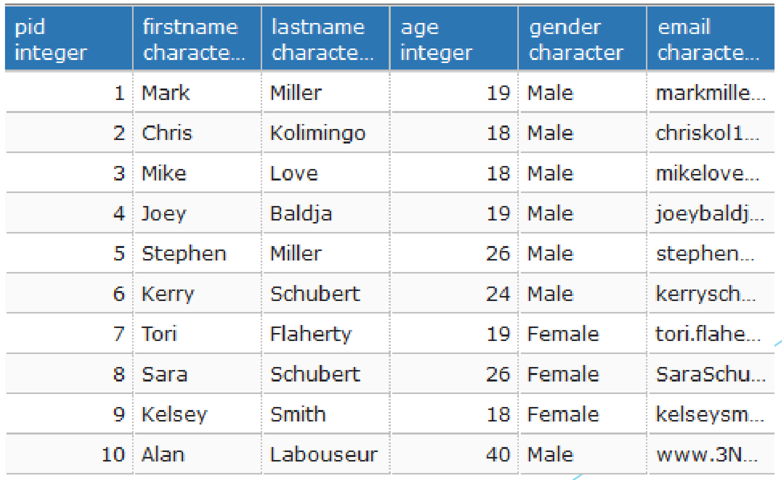
- This is the most important table as it is the beginning in which we distinguish every person with a PID, name and important information about themselves. The PID is the primary key as it acts as the beginning to distinguishing a member from an employee.



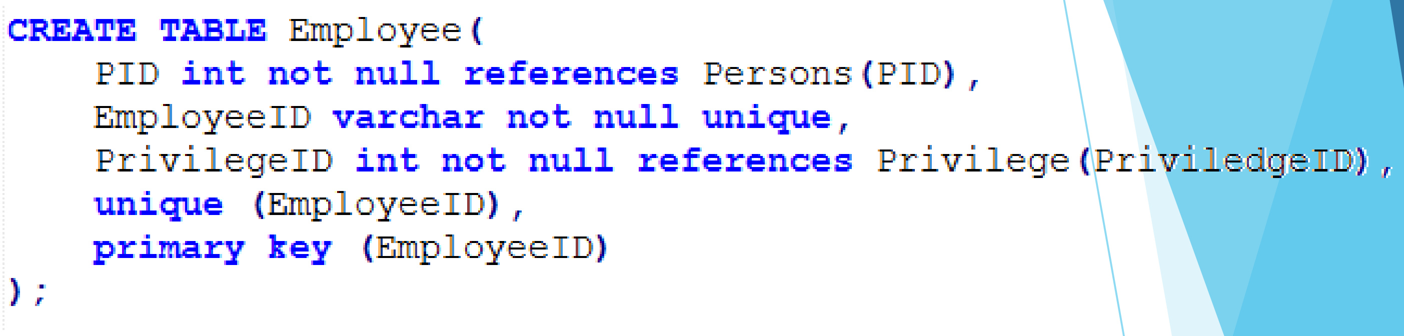
```
CREATE TABLE Persons(
    PID int not null unique,
    FirstName varchar not null,
    LastName varchar not null,
    Age int not null,
    Gender char(6) not null,
    Email varchar not null,
    unique (PID),
    primary key (PID)
);
```

Functional Dependencies:
PID → FirstName, LastName, Age, Gender, Email

Sample data:



pid integer	firstname characte...	lastname characte...	age integer	gender character	email characte...
1	Mark	Miller	19	Male	markmille...
2	Chris	Kolimingo	18	Male	chriskol1...
3	Mike	Love	18	Male	mikelove...
4	Joey	Baldja	19	Male	joeybaldj...
5	Stephen	Miller	26	Male	stephen...
6	Kerry	Schubert	24	Male	kerrysch...
7	Tori	Flaherty	19	Female	tori.flahe...
8	Sara	Schubert	26	Female	SaraSchu...
9	Kelsey	Smith	18	Female	kelseysm...
10	Alan	Labouseur	40	Male	www.3N...

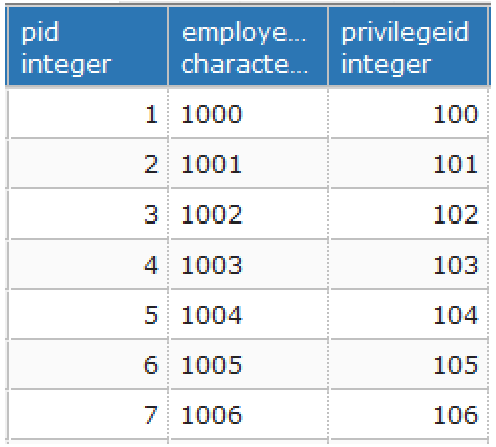


```
CREATE TABLE Employee(
    PID int not null references Persons(PID),
    EmployeeID varchar not null unique,
    PrivilegeID int not null references Privilege(PriviledgeID),
    unique (EmployeeID),
    primary key (EmployeeID)
);
```

Create Table Statements:
Employee Table

- The Employee Table places a crucial role as it links many of the other tables that are associated with an employee. We also define and store the EmployeeID into this table. This table references the Persons table through the PID and uses the EmployeeID as its primary key. There is a decision node placed on the employee side because of the choice between a employee or member.

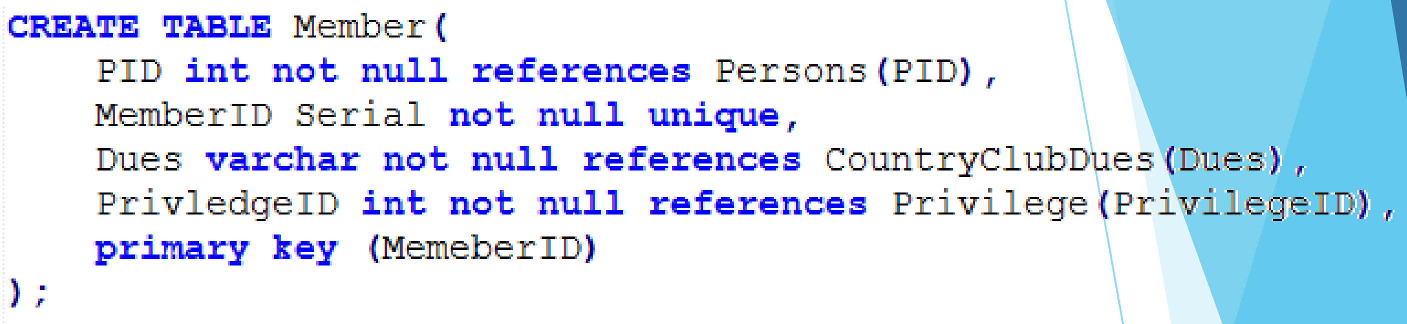
Functional Dependencies:
EmployeeID → PID, PrivilegeID



pid integer	employe... characte...	privilegeid integer
1	1000	100
2	1001	101
3	1002	102
4	1003	103
5	1004	104
6	1005	105
7	1006	106

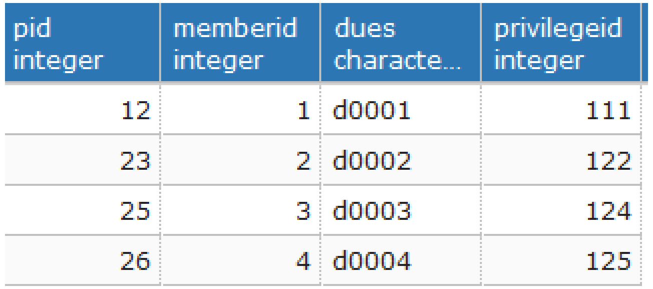
Create Table Statements:
Member Table

- Another crucial table as it links all member related table to it by a MemberID. The MemberID servers as the primary key and references the Persons table by the PID. Again we place a decision node on the member side because you can either be a member or an employee.



```
CREATE TABLE Member(
    PID int not null references Persons(PID),
    MemberID Serial not null unique,
    Dues varchar not null references CountryClubDues(Dues),
    PrivledgeID int not null references Privilege(PrivilegeID),
    primary key (MemeberID)
);
```

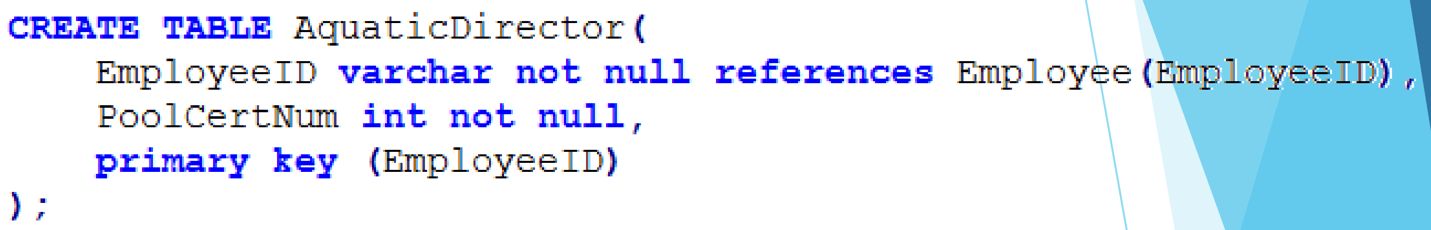
Functional Dependencies:
MemberID → PID, Dues, PrivilegeID



pid integer	memberid integer	dues characte…	privilegeid integer
12	1	d0001	111
23	2	d0002	122
25	3	d0003	124
26	4	d0004	125

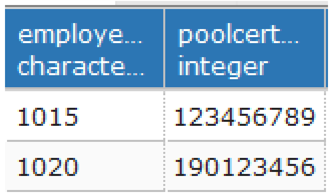
Create Table Statements:
AquaticDirector Table

- This is one of many jobs that are associated with being an Employee. EmployeeID is again the primary key and references the Employee Table. The AqauticDirector also has a Pool Certificate Number associated with him. A decision node is placed on this side as it is a job decision.



```
CREATE TABLE AquaticDirector(
    EmployeeID varchar not null references Employee(EmployeeID),
    PoolCertNum int not null,
    primary key (EmployeeID)
);
```

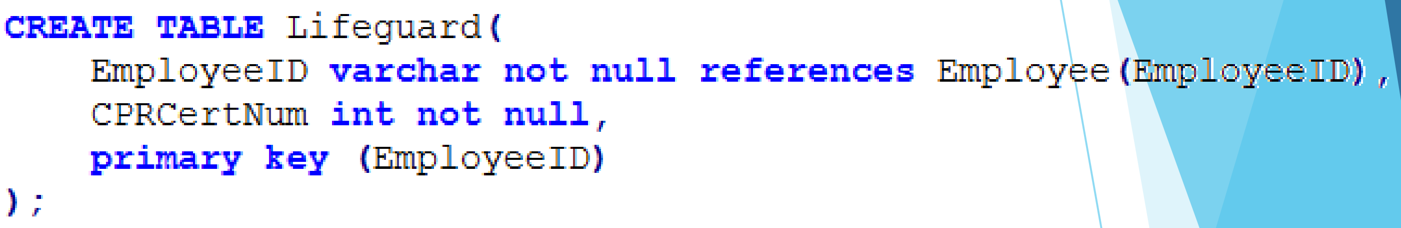
Functional Dependencies:
EmployeeID → PoolCertNum



employe... characte...	poolcert... integer
1015	123456789
1020	190123456

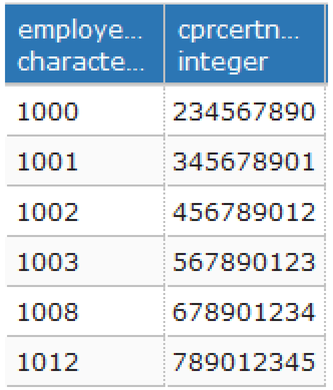
Create Table Statements:
Lifeguard Table

- Another job associated with being an Employee and follows the same principles as the AquaticDirector table. It references EmployeeID and has an CPR Certificate Number associated with it too. We again place a decision node on this side of the table to show the choice of being a lifeguard.



```
CREATE TABLE Lifeguard(
    EmployeeID varchar not null references Employee(EmployeeID),
    CPRCertNum int not null,
    primary key (EmployeeID)
);
```

Functional Dependencies:
EmployeeID → CPRCertNum

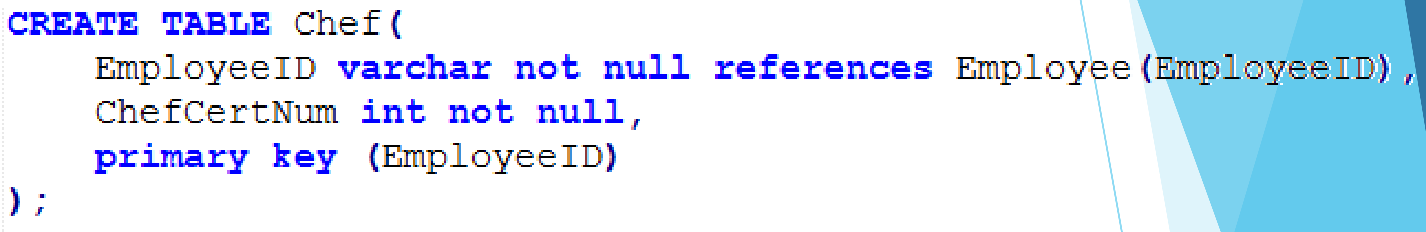


employe... characte...	cprcertn... integer
1000	234567890
1001	345678901
1002	456789012
1003	567890123
1008	678901234
1012	789012345

Create Table Statements:

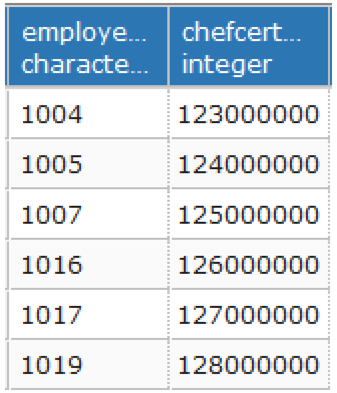
Chef Table

- Following the other job tables, this table sets the primary key as the EmployeeID and references the Employee table with it. The Chef table also has a Chef Certificate Number within it. The decision node is used on this side of the table to show the choice of being a chef.



```
CREATE TABLE Chef(
    EmployeeID varchar not null references Employee(EmployeeID),
    ChefCertNum int not null,
    primary key (EmployeeID)
);
```

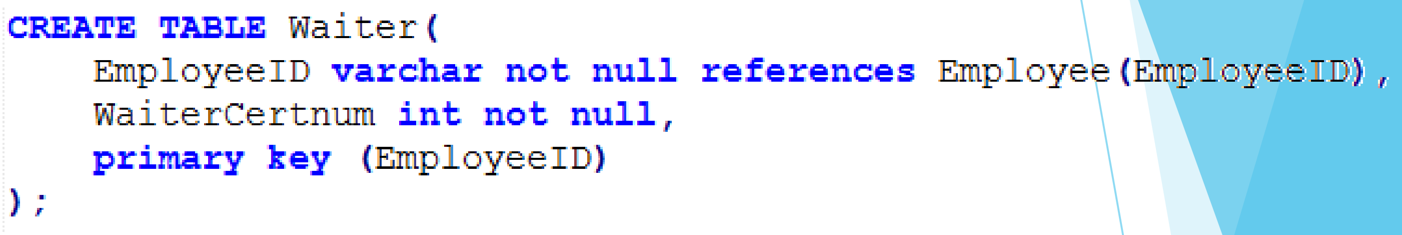
Functional Dependencies:
EmployeeID → ChefCertNum



employe... characte...	chefcert... integer
1004	123000000
1005	124000000
1007	125000000
1016	126000000
1017	127000000
1019	128000000

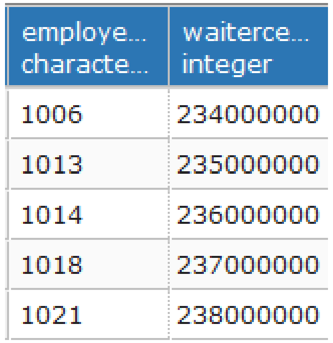
Create Table Statements:
Waiter Table

- The waiter table follows all other job tables in that we set the EmployeeID as the primary key and reference the Employee table to get it. The Waiter table also as a Waiter Certificate number corresponding with it. A decision node is used to show a decision is made on choosing this job.



```
CREATE TABLE Waiter(
    EmployeeID varchar not null references Employee(EmployeeID),
    WaiterCertnum int not null,
    primary key (EmployeeID)
);
```

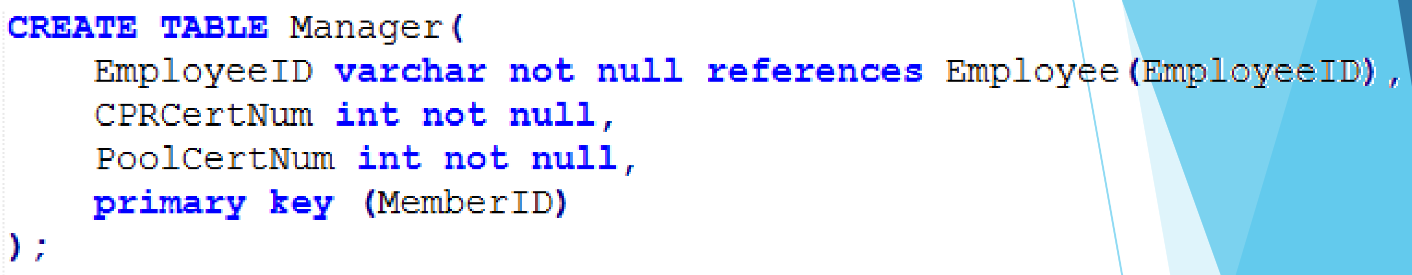
Functional Dependencies:
EmployeeID → WaiterCertNum



employe... characte...	waiterce... integer
1006	234000000
1013	235000000
1014	236000000
1018	237000000
1021	238000000

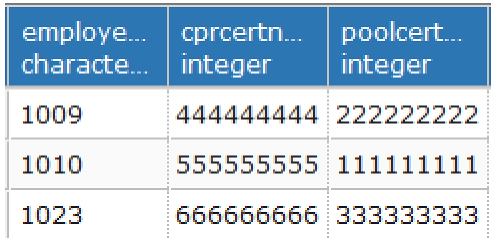
Create Table Statements:
Manager Table

- The final job table is the Manager table. This again sets the EmployeeID to be the primary key and references the Employee table to get it. It then also has both a CPR and Pool Certificate Number in its table. Like the rest a decision node is used to represent the choice between a manager or any other job.



```
CREATE TABLE Manager(
    EmployeeID varchar not null references Employee(EmployeeID),
    CPRCertNum int not null,
    PoolCertNum int not null,
    primary key (MemberID)
);
```

Functional Dependencies:
EmployeeID → PoolCertNum, CPRCertNum

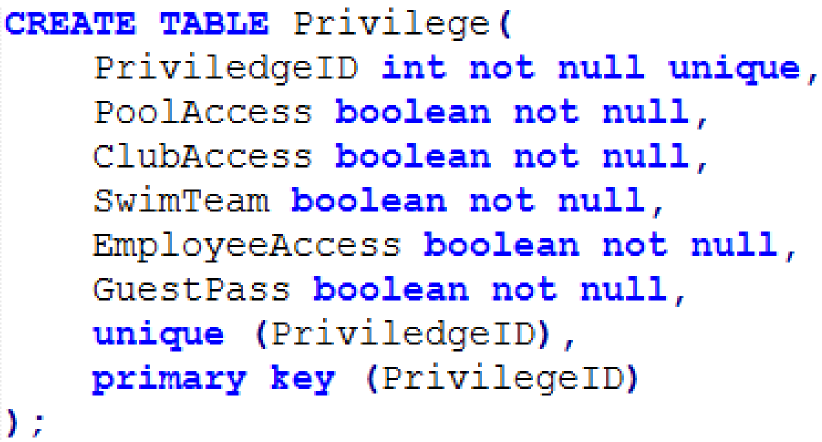


employe...	cprcertn...	poolcert...
characte...	integer	integer
1009	444444444	222222222
1010	555555555	111111111
1023	666666666	333333333

Create Table Statements:

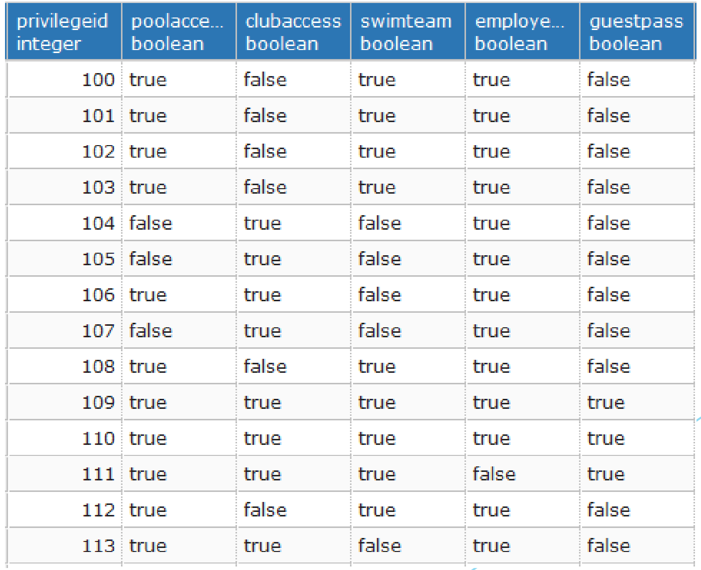
Privilege Table

- The privilege table represents the privileges that each employee or member has access to. The PrivilegeID is stored within this table and is referenced in both the employee and member table. The PrivilegeID is used as the primary key in this table.



```
CREATE TABLE Privilege(
    PriviledgeID int not null unique,
    PoolAccess boolean not null,
    ClubAccess boolean not null,
    SwimTeam boolean not null,
    EmployeeAccess boolean not null,
    GuestPass boolean not null,
    unique (PriviledgeID),
    primary key (PrivilegeID)
);
```

Functional Dependencies:
PrivilegeID → PoolAccess, ClubAccess, SwimTeam, EmployeeAccess, GuestPass

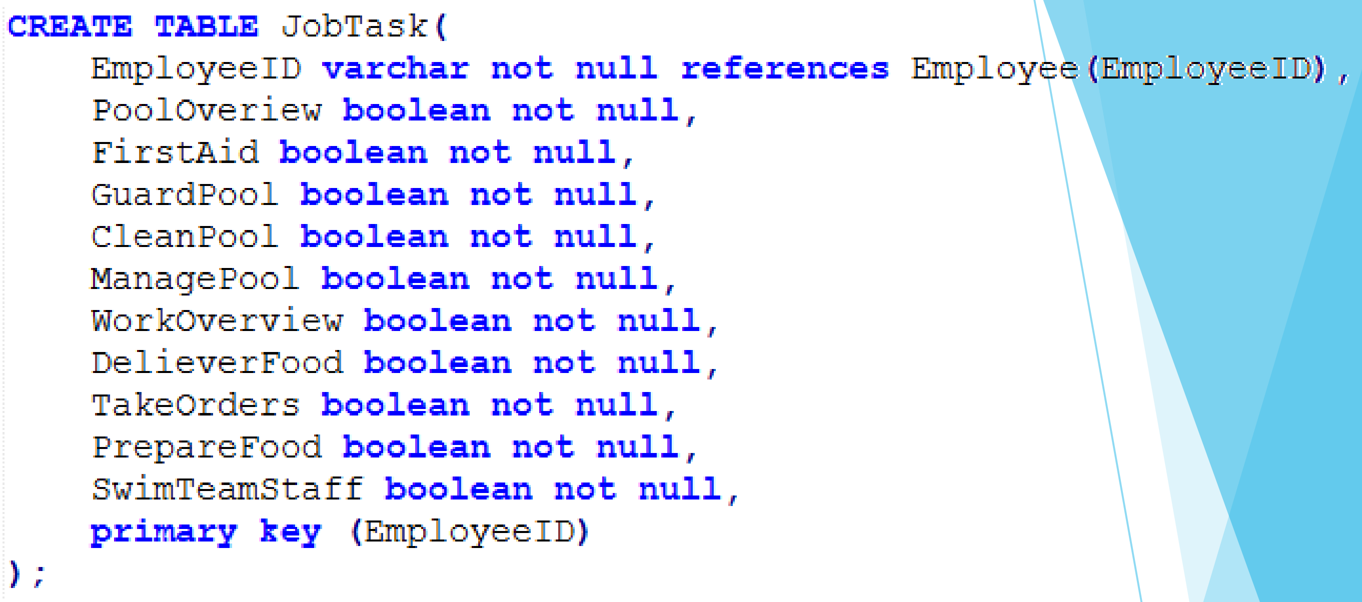


privilegeid integer	poolacce... boolean	clubaccess boolean	swimteam boolean	employe... boolean	guestpass boolean
100	true	false	true	true	false
101	true	false	true	true	false
102	true	false	true	true	false
103	true	false	true	true	false
104	false	true	false	true	false
105	false	true	false	true	false
106	true	true	false	true	false
107	false	true	false	true	false
108	true	false	true	true	false
109	true	true	true	true	true
110	true	true	true	true	true
111	true	true	true	false	true
112	true	false	true	true	false
113	true	true	false	true	false

Create Table Statements:

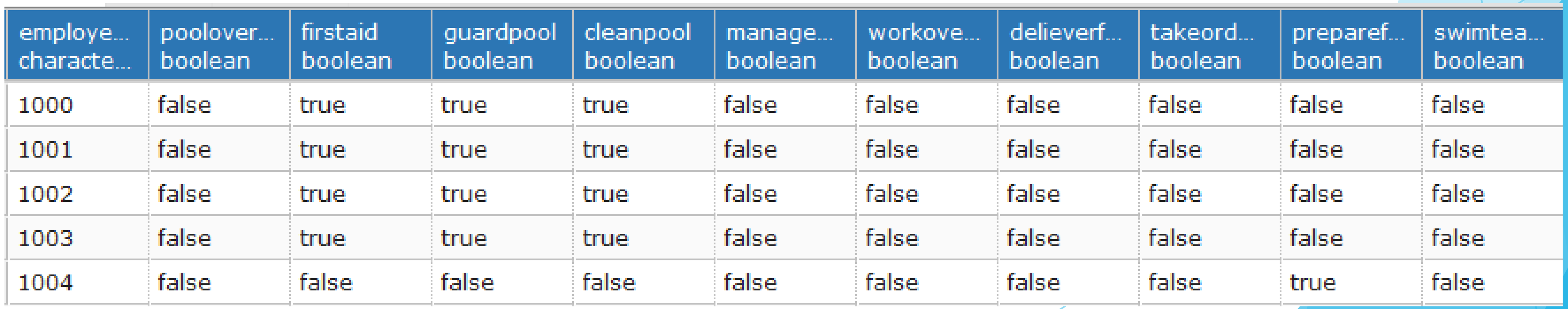
JobTask Table

- This table is very similar to the Privilege table in that it defines all of the job task assigned to a specific person. We use the EmployeeID as the primary key and references the Employee table once again.



```
CREATE TABLE JobTask(
    EmployeeID varchar not null references Employee(EmployeeID),
    PoolOveriew boolean not null,
    FirstAid boolean not null,
    GuardPool boolean not null,
    CleanPool boolean not null,
    ManagePool boolean not null,
    WorkOverview boolean not null,
    DelieverFood boolean not null,
    TakeOrders boolean not null,
    PrepareFood boolean not null,
    SwimTeamStaff boolean not null,
    primary key (EmployeeID)
);
```

Functional Dependencies:
EmployeeID → PoolOverview, FirstAid, GuardPool, CleanPool, ManagePool, WorkOverview, DelieverFood, TakeOrders, SwimTeamStaff

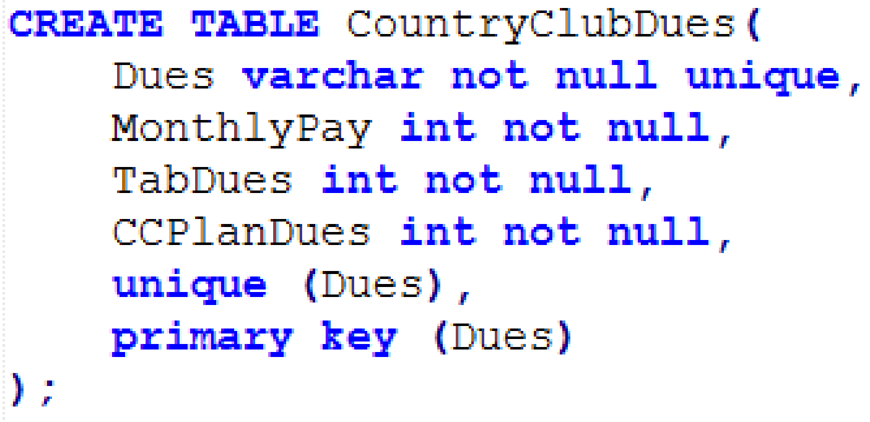


employe... characte...	poolover... boolean	firstaid boolean	guardpool boolean	cleanpool boolean	manage... boolean	workove... boolean	delieverf... boolean	takeord... boolean	preparef... boolean	swimtea... boolean
1000	false	true	true	true	false	false	false	false	false	false
1001	false	true	true	true	false	false	false	false	false	false
1002	false	true	true	true	false	false	false	false	false	false
1003	false	true	true	true	false	false	false	false	false	false
1004	false	false	false	false	false	false	false	false	true	false

Create Table Statements:

CountryClubDues Table

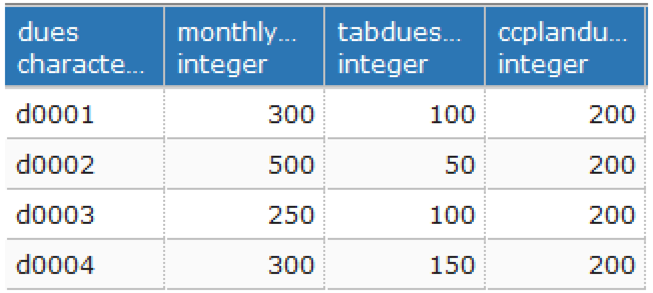
- This table signifies all of the payments that a member must make to the Country Club. We use Dues as the primary key and store it within this table. Dues is referenced by the Member table in order to gain access.



```
CREATE TABLE CountryClubDues(
    Dues varchar not null unique,
    MonthlyPay int not null,
    TabDues int not null,
    CCPlanDues int not null,
    unique (Dues),
    primary key (Dues)
);
```

Functional Dependencies:

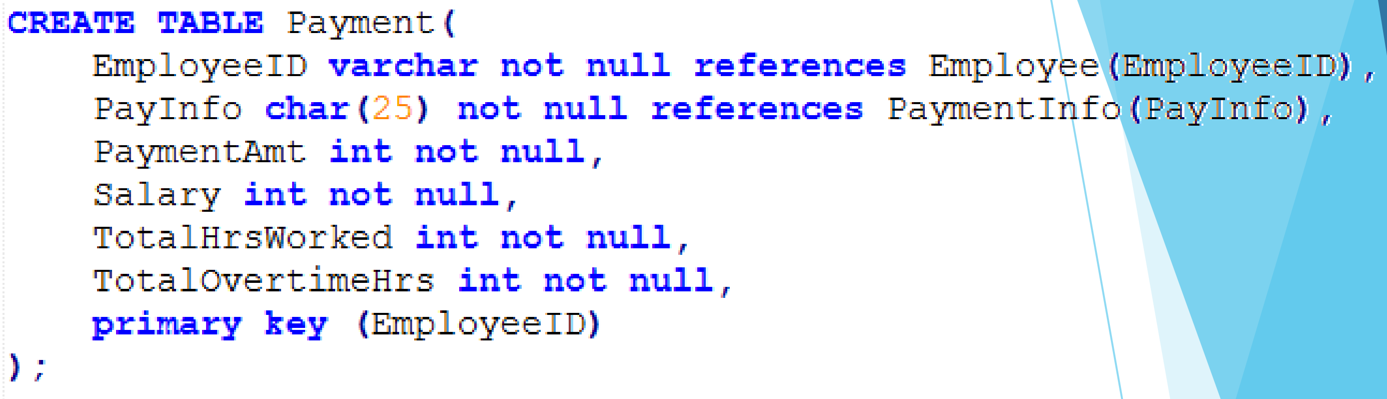
Dues → MonthlyPay, TabDues, CCPlanDues



dues characte...	monthly... integer	tabdues... integer	ccplandu... integer
d0001	300	100	200
d0002	500	50	200
d0003	250	100	200
d0004	300	150	200

Create Table Statements:
Payment Table

- This table uses a EmployeeID once again as its primary id and describes all of the factors that go into your payments and the total payment that you get.

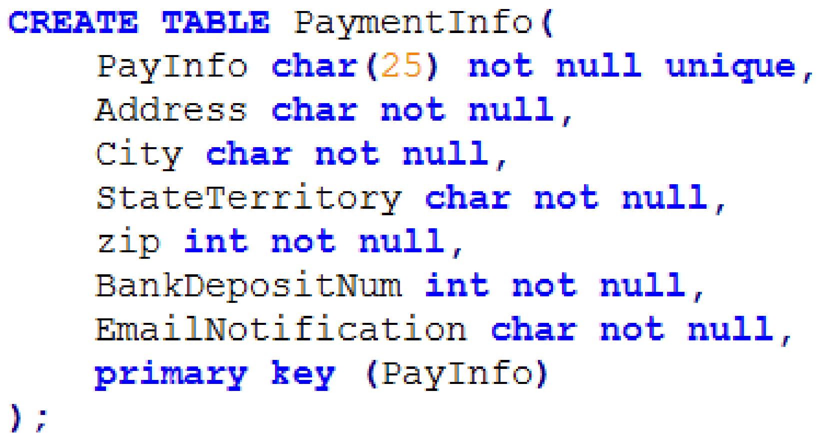


```
CREATE TABLE Payment(
    EmployeeID varchar not null references Employee(EmployeeID),
    PayInfo char(25) not null references PaymentInfo(PayInfo),
    PaymentAmt int not null,
    Salary int not null,
    TotalHrsWorked int not null,
    TotalOvertimeHrs int not null,
    primary key (EmployeeID)
);
```

Functional Dependencies:
EmployeeID → PAyInfo, PaymentAmt, Salary, TotalHrsWorked, TotalOvertimeHrs

Create Table Statements:
PaymentInfo Table

- The final table in our database is the PaymentInfo table which specifies all of the needed information when a employee is being paid. The primary key is the PayInfo which is unqiue for every person.



```
CREATE TABLE PaymentInfo(
    PayInfo char(25) not null unique,
    Address char not null,
    City char not null,
    StateTerritory char not null,
    zip int not null,
    BankDepositNum int not null,
    EmailNotification char not null,
    primary key (PayInfo)
);
```

Functional Dependencies:
PayInfo → Address, City, StateTerritory, zip, BankDepositNum, EmailNotification

SQL Code Link:

https://github.com/markyMARK0702/LinkedIn-Projects/blob/master/Mark_Miller_Final_Project_WCC.sql