Group 15

Final Report Daycare

CIS 3400 EMWA

Arfin Jahan (arfin.jahan@baruchmail.cuny.edu)
Chi Ying Annice Chan (chiyingannice.chan@baruchmail.cuny.edu)
Christian Sierra Perez (christian.sierraperez1@baruchmail.cuny.edu)
Colleen Zeng (colleen.zeng@baruchmail.cuny.edu)
Yondonjamts Otgontamir (yondonjamts.otgontamir@baruchmail.cuny.edu)

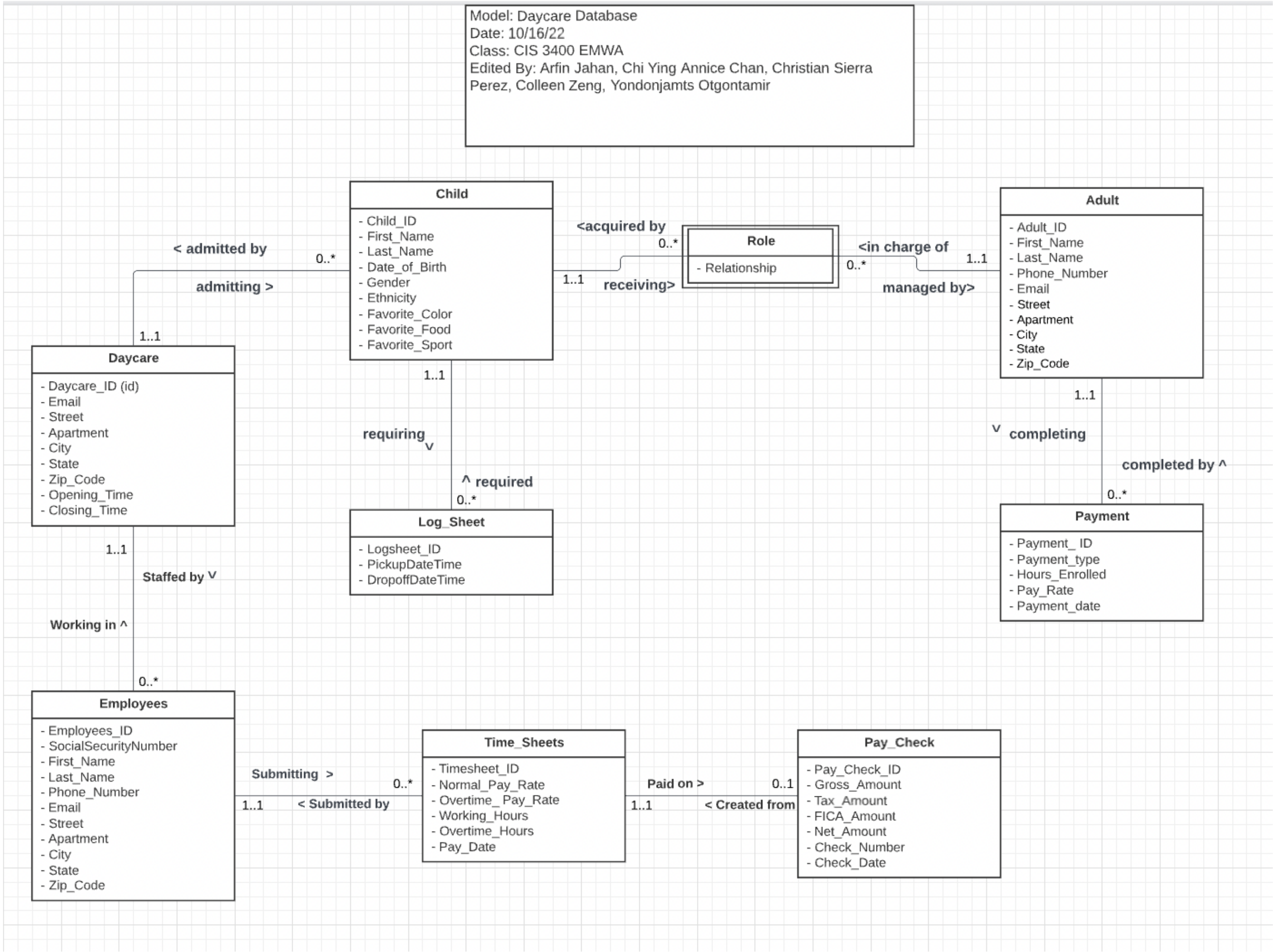
Introduction:

Nicole’s Family Daycare is a childcare business that has been running for 8 years. By running a daycare, Nicole has to keep track of employees, paychecks, parent's information, pay rates, children, etc. For the last 8 years, Nicole hasn’t modernized her data. She has been using file cabinets filled with papers and folders to keep track of all the information. However, Nicole has realized this is quite inefficient because she spends a lot of time going through papers and it’s difficult to organize everything.

In Nicole’s family business, we need to keep track of the child’s meals, allergies, equipment, parent’s contact information, payments expectation, and the duration the child stays at the Daycare. We create data to keep up with records of what it was needed to organize the Daycare to make sure we have the correct data to keep up on what was needed.Therefore, we also have created forms and reports to help Nicole’s Family Daycare to keep in record whenever she needs them.

Entity Relationship Model Diagram:

E-R Diagram on Lucid



Normalization:

Payment **(Payment_ID (Key)**, Payment_Type, Hours_Enrolled, Pay_Rate, Payment_Date, **Adult_ID (FK))**
Key: Payment_ID

Adult_Info **(Adult_ID (Key)**, First_Name, Last_Name, Phone_Number, Email, Street, Apartment, Zip_Code)
Key: Adult_ID

Adult_Location_Info **(Zip_Code (Key)**, City, State)
Key: Zip_Code

Role (Relationship, **Child_ID (FK)(Key)**, **Adult_ID (FK)(Key))**
Key: Child_ID, Adult_ID

Log_Sheets **(Logsheet_ID (Key)**, PickupDateTime, DropoffDateTime, **Child_ID (FK))**
Key: Logsheet_ID

Child **(Child_ID (Key)**, First_Name, Last_Name, Date_of_Birth, Gender, Ethnicity, Favorite_Color, Favorite_Food, Favorite_Sport, **Daycare_ID (FK))**

Key: Child_ID

Time_Sheets **(Timesheet_ID (Key),** Normal_Pay_Rate, Overtime_Pay_Rate, Working_Hours, Overtime_Hours, Pay_Date, **Employees_ID (FK))**
Key: Timesheet_ID

Pay_Check **(Pay_Check_ID (Key)**, Gross_Amount, Tax_Amount, FICA_Amount, Net_Amount, Check_Number, Check_Date, **Timesheet_ID (FK))**
Key: Pay_Check_ID

Employee_SS **(SocialSecurityNumber (Key)**, First_Name, Last_Name)
Key: SocialSecurityNumber

Employee_Info **(Empoyees_ID (Key)**, SocialSecurityNumber, Phone_Number, Email, Street, Apartment, Zip_Code, **Daycare_ID(FK))**
Key: Empoyees_ID

Employee_Location **(Zip_Code (Key)**, City, State)
Key: Zip_Code

Daycare_Zip **(Zip_Code (Key)**, City, State)
Key: Zip_Code

Daycare_Info **(Daycare_ID (Key),** Email, Street, Apartment, Zip_Code, Opening_Time, Closing_Time)
Key: Daycare_ID

Functional dependencies:

Daycare Relation (Daycare_ID (Key), Email, Street, Apartment, City, State, Zip_Code, Opening_Time, Closing_Time)
Key: Daycare_ID
FD 1: Daycare_ID → Email, Street, Apartment, City, State, Zip_Code, Opening_Time, Closing_Time
FD 2: Zip_Code → City, State
1NF: Yes, Given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: No, because there is a transitive dependency

Normalization of Daycare Relation:

Daycare_Info Relation (Daycare_ID, Email, Street, Apartment, Zip_Code, Opening_Time, Closing_Time)
Key: Daycare_ID
FD1: Daycare_ID → Email, Street, Apartment, Zip_Code, Opening_Time, Closing_Time
1NF: Yes, because it was split from a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Daycare_Zip Relation (Zip_Code, City, State)

Key: Zip_Code
FD1: Zip_Code → City, State
1NF: Yes, because it was split from a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Employees Relation (Employees_ID (Key), SocialSecurityNumber, First_Name, Last_Name, Phone_Number, Email, Street, Apartment, City, State, Zip_Code, Daycare_ID (FK))
Key: Employees_ID
FD 1: Employees_ID → SocialSecurityNumber, First_Name, Last_Name, Phone_Number, Email, Street, Apartment, City, State, Zip_Code, Daycare_ID
FD 2: Zip_Code → City, State
FD 3: SocialSecurityNumber → First_Name, Last_Name
1NF: Yes, because it's given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: No, because there are transitive dependencies

Normalization of Employee Relation:

Employee_Info Relation (Empoyees_ID, SocialSecurityNumber, Phone_Number, Email, Street, Apartment, Zip_Code, Daycare_ID(FK))
Key: Employees_ID
FD1: Employees_ID → SocialSecurityNumber, Phone_Number, Email, Street, Apartment, Zip_Code, Daycare_ID)
1NF: Yes, because it is split from a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependency

Employee_Info Relation (Empoyees_ID, SocialSecurityNumber, Phone_Number, Email, Street, Apartment, City, State, Zip_Code, Daycare_ID(FK))
Key: Employees_ID
FD1: Employees_ID → SocialSecurityNumber, Phone_Number, Email, Street, Apartment, Zip_Code, City, State, Daycare_ID)
1NF: Yes, because it is split from a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependency

Employee_Location Relation (Zip_Code, City, State)
Key: Zip_Code
FD1: Zip_Code → City, State
1NF: Yes, because it's split from a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there is no transitive dependency

Employee_SS Relation (SocialSecurityNumber, First_Name, Last_Name)

Key: SocialSecurityNumber
FD1: SocialSecurityNumber → First_Name, Last_Name
1NF: Yes, because it is split from a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Time_Sheets Relation (Timesheet_ID (Key), Normal_Pay_Rate, Overtime_Pay_Rate, Working_Hours, Overtime_Hours, Pay_Date, Employees_ID (FK))
Key: Timesheet_ID
FD 1: Timesheet_ID → Normal_Pay_Rate, Overtime_Pay_Rate, Working_Hours, Overtime_Hours, Pay_Date, Employees_ID
1NF: Yes, because given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Pay_Check Relation (Pay_Check_ID (Key), Gross_Amount, Tax_Amount, FICA_Amount, Net_Amount, Check_Number, Check_Date, Timesheet_ID (FK))
Key: Pay_Check_ID
FD 1: Pay_Check_ID → Gross_Amount, Tax_Amount, FICA_Amount, Net_Amount, Check_Number, Check_Date, Timesheet_ID
1NF: Yes, because given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Child Relation (Child_ID (Key), First_Name, Last_Name, Date_of_Birth, Gender, Ethnicity, Favorite_Color, Favorite_Food, Favorite_Sport, Daycare_ID (FK))
Key: Child_ID
FD 1: Child_ID → First_Name, Last_Name, Date_of_Birth, Gender, Ethnicity, Favorite_Color, Favorite_Food, Favorite_Sport, Daycare_ID
1NF: Yes, because given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Log_Sheets Relation (Logsheet_ID (Key), PickupDateTime, DropoffDateTime, Child_ID (FK))
Key: Logsheet_ID
FD 1: Logsheet_ID → PickupDateTime, DropoffDateTime, Child_ID (**FK**)
1NF: Yes, because given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Role (Relationship, Child_ID (FK)(Key), Adult_ID (FK)(Key))
Key: Child_ID, Adult_ID
FD1: Child_ID, Adult_ID → Relationship
1NF: Yes, given as a relation
2NF: Yes, because there are no partial key dependencies.

3NF: Yes, because there are no transitive dependencies.

Adult Relation (Adult_ID (Key), First_Name, Last_Name, Phone_Number, Email, Street, Apartment, City, State, Zip_Code)
Key: Adult_ID
FD 1: Adult_ID → First_Name, Last_Name, Phone_Number, Email, Street, Apartment, City, State, Zip_Code
FD 2: Zip_Code → City, State
1NF: Yes, because given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: No, because there is a transitive dependency

Normalization of Adult Relation:

Adult_Info Relation (Adult_ID (Key), First_Name, Last_Name, Phone_Number, Email, Street, Apartment, Zip_Code)
Key: Adult_ID
FD 1: Adult_ID → First_Name, Last_Name, Phone_Number, Email, Street, Apartment, Zip_Code
1NF: Yes, because given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

Adult_Location_Info Relation (Zip_Code, City, State)
Key: Zip_Code
FD1: Zip_Code → City, State
1NF: Yes, it is split from a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, there are no transitive dependencies

Payment Relation (Payment_ID (Key), Payment_Type, Hours_Enrolled, Pay_Rate, Payment_Date, Adult_ID (FK))
Key: Payment_ID
FD 1: Payment_ID → Payment_Type, Hours_Enrolled, Pay_Rate, Payment_Date, Adult_ID (**FK**)
1NF: Yes, because given as a relation
2NF: Yes, because there are no partial key dependencies
3NF: Yes, because there are no transitive dependencies

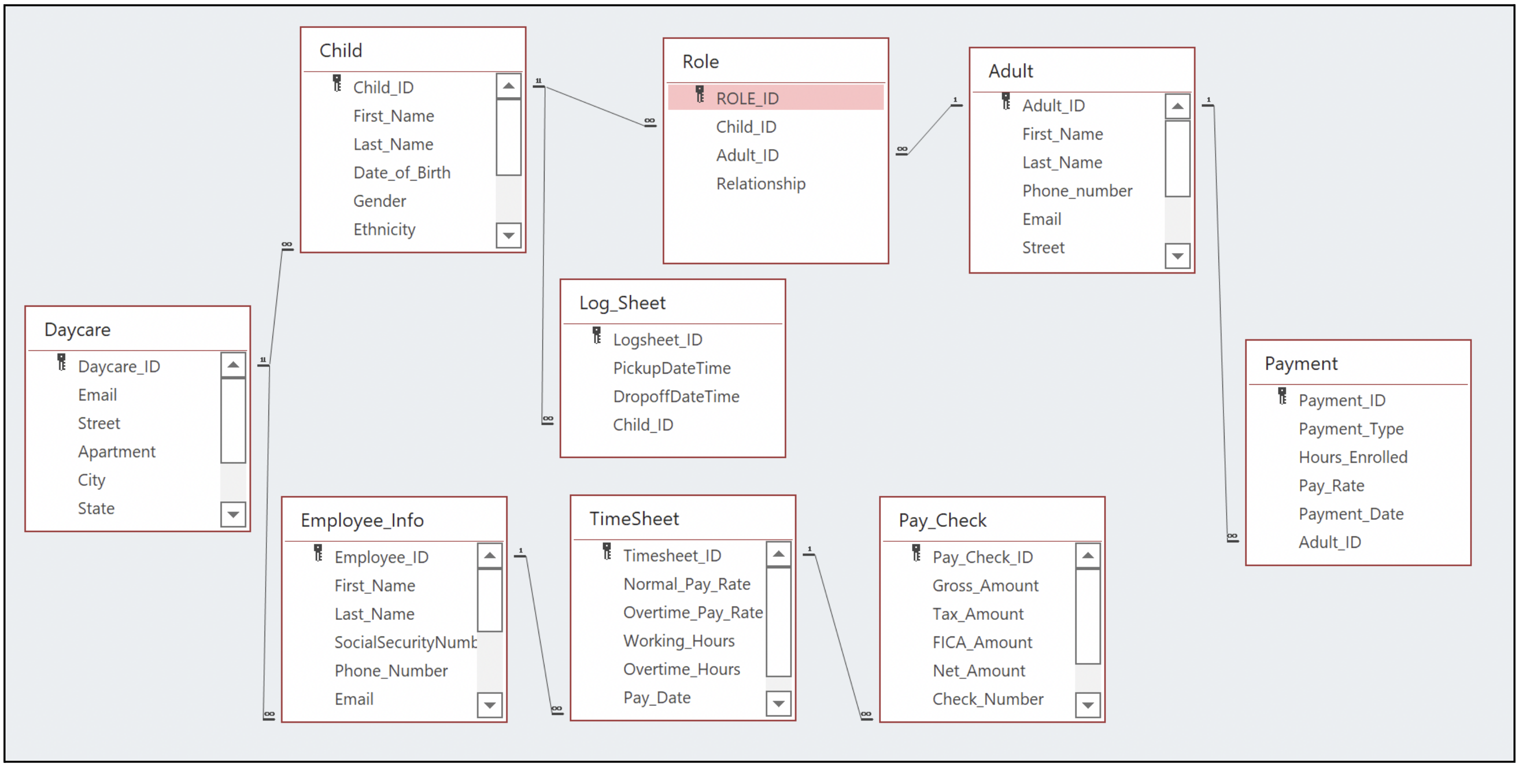
CREATING TABLE:

```
CREATE TABLE Daycare (
     Daycare_ID     NUMBER    NOT NULL  PRIMARY KEY,
     Email          VARCHAR(30),
     Street         VARCHAR(30),
     Apartment      VARCHAR(30),
     City           VARCHAR(30),
     State          VARCHAR(2),
     Zip_Code       VARCHAR(10),
     Opening_Time   VARCHAR(20),
     Closing_Time   VARCHAR(20));

CREATE TABLE Employee_Info (
     Employee_ID         NUMBER         NOT NULL  PRIMARY KEY,
     First_Name          VARCHAR(20)    NOT NULL,
     Last_Name           VARCHAR(20)    NOT NULL,
     SocialSecurityNumberVARCHAR(15),
     Phone_Number        VARCHAR(15),
     Email               VARCHAR(30),
     Street              VARCHAR(15),
     Apartment           VARCHAR(15),
     City                VARCHAR (20),
     State               VARCHAR(2),
     Zip_Code            VARCHAR (15),
     Daycare_ID          NUMBER         NOT NULL);

CREATE TABLE TimeSheet (
     Timesheet_ID        NUMBER    NOT NULL  PRIMARY KEY,
     Normal_Pay_Rate     NUMBER,
     Overtime_Pay_Rate   NUMBER,
     Working_Hours       NUMBER,
     Overtime_Hours      NUMBER,
     Pay_Date            DATETIME  NOT NULL,
     Employee_ID         NUMBER    NOT NULL);

CREATE TABLE Pay_Check(
     Pay_Check_ID        NUMBER    NOT NULL  PRIMARY KEY,
     Gross_Amount        NUMBER,
     Tax_Amount          NUMBER,
     FICA_Amount         NUMBER,
     Net_Amount          NUMBER,
     Check_Number        INTEGER,
     Check_Date          DATETIME,
     Timesheet_ID        NUMBER    NOT NULL);

CREATE TABLE Adult  (
     Adult_ID       NUMBER         NOT NULL  PRIMARY KEY,
     First_Name     VARCHAR(30)    NOT NULL,
     Last_Name      VARCHAR(30)    NOT NULL,
     Phone_number   VARCHAR(20),
     Email          VARCHAR(30),
     Street         VARCHAR(30),
     Apartment      VARCHAR(30),
     City           VARCHAR(30),
     State          VARCHAR(2),
     Zip_Code       VARCHAR(10));

CREATE TABLE Payment  (
     Payment_ID     NUMBER    NOT NULL  PRIMARY KEY,
     Payment_Type   VARCHAR(20),
     Hours_Enrolled NUMBER,
     Pay_Rate       NUMBER,
     Payment_Date   DATETIME,
     Adult_ID       NUMBER    NOT NULL);

CREATE TABLE Child (
     Child_ID       NUMBER         NOT NULL  PRIMARY KEY,
     First_Name     VARCHAR(30)    NOT NULL,
     Last_Name      VARCHAR(30)    NOT NULL,
     Date_of_Birth  DATETIME       NOT NULL,
     Gender         VARCHAR(20),
     Ethnicity      VARCHAR(20),
     Favorite_Color VARCHAR(20),
     Favorite_Food  VARCHAR(20),
     Favorite_Sport VARCHAR(20),
     Daycare_ID     NUMBER         NOT NULL);

CREATE TABLE Log_Sheet (
     Logsheet_ID         NUMBER         NOT NULL  PRIMARY KEY,
     Pickup_Date_Time    DATETIME       NOT NULL,
     Dropoff_Date_Time   DATETIME       NOT NULL,
     Child_ID            NUMBER         NOT NULL);

CREATE TABLE Role (
     ROLE_ID        NUMBER    NOT NULL  PRIMARY KEY
     Child_ID       NUMBER    NOT NULL,
     Adult_ID       NUMBER    NOT NULL,
     Relationship   VARCHAR(20));
```

ALTER TABLES:

```
ALTER TABLE TimeSheet
    ADD FOREIGN KEY (Employee_ID)
              REFERENCES  Employee_Info(Employee_ID);

ALTER TABLE Pay_Check
   ADD FOREIGN KEY (Timesheet_ID)
        REFERENCES TimeSheets (Timesheet_ID);

ALTER TABLE Payment
    ADD FOREIGN KEY (Adult_ID)
             REFERENCES Adult(Adult_ID);

ALTER TABLE Child
    ADD FOREIGN KEY (Daycare_ID)
            REFERENCES Daycare_Relation (Daycare_ID);

ALTER TABLE Log_Sheet
    ADD FOREIGN KEY (Child_ID)
               REFERENCES Child (Child_ID);

ALTER TABLE Role
     ADD FOREIGN KEY (Child_ID)
          REFERENCES Child(Child_ID);
```

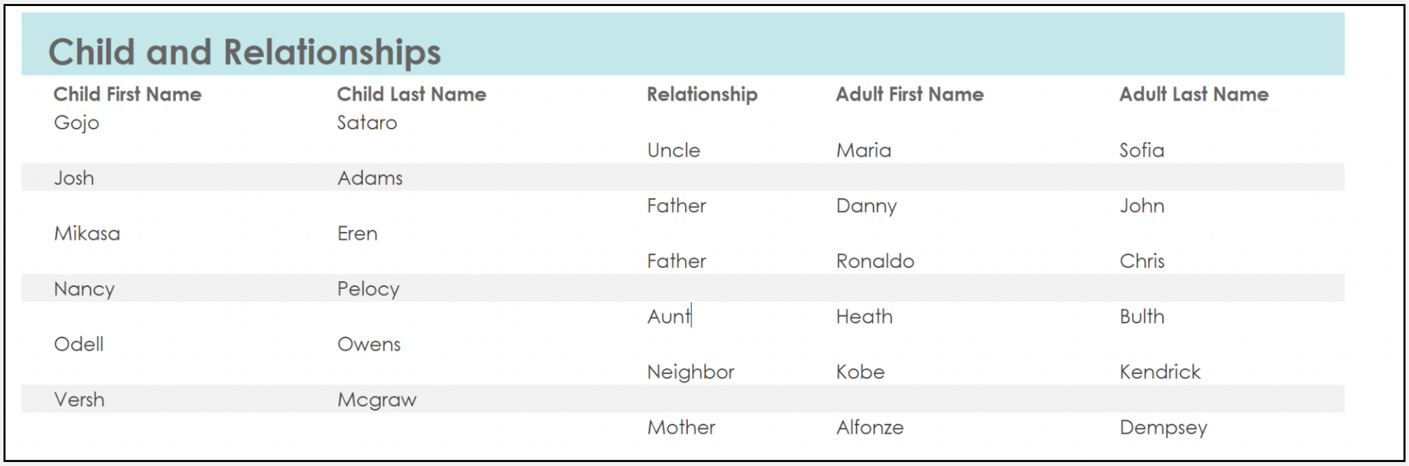
```
ALTER TABLE Role
     ADD FOREIGN KEY(Adult_ID)
          REFERENCES Adult(Adult_ID);

ALTER TABLE Employee_Info
     ADD FOREIGN KEY (Daycare_ID)
          REFERENCES Daycare(Daycare_ID);
```



Reporting Questions with SQL Code:

1. List all children with the names and relationships of related adults

   SELECT Child.First_Name, Child.Last_Name, Role.Relationship, Adult.First_Name, Adult.Last_Name
   FROM ((Child **INNER JOIN** Role **ON** Child.Child_ID = Role.Child_ID)
   INNER JOIN Adult ON Role.Adult_ID = Adult.Adult_ID)
   ORDER BY Child.First_Name ASC;

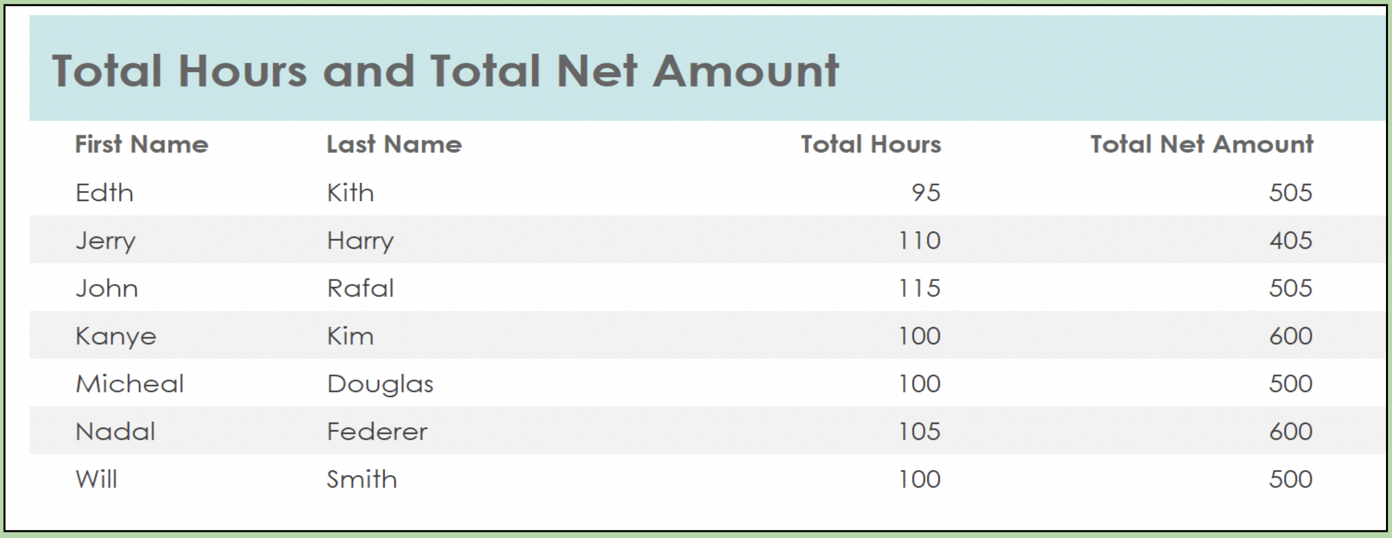


Child and Relationships

Child First Name	Child Last Name	Relationship	Adult First Name	Adult Last Name
Gojo	Sataro			
		Uncle	Maria	Sofia
Josh	Adams			
		Father	Danny	John
Mikasa	Eren			
		Father	Ronaldo	Chris
Nancy	Pelocy			
		Aunt	Heath	Bulth
Odell	Owens			
		Neighbor	Kobe	Kendrick
Versh	Mcgraw			
		Mother	Alfonze	Dempsey

2. List all employees with a sum of hours worked (overtime, etc.) per timesheet and total of net amount

   SELECT First_Name, Last_Name, (Working_Hours + Overtime_Hours) AS Total_Hours, SUM(Pay_Check.Net_Amount) AS Total_Net_Amount
   FROM (Employee_Info **INNER JOIN** TimeSheet ON Employee_Info.Employee_ID = TimeSheet.Employee_ID) **INNER JOIN** Pay_Check ON TimeSheet.Timesheet_ID = Pay_Check.Timesheet_ID
   GROUP BY First_Name, Last_Name, (Working_Hours + Overtime_Hours);

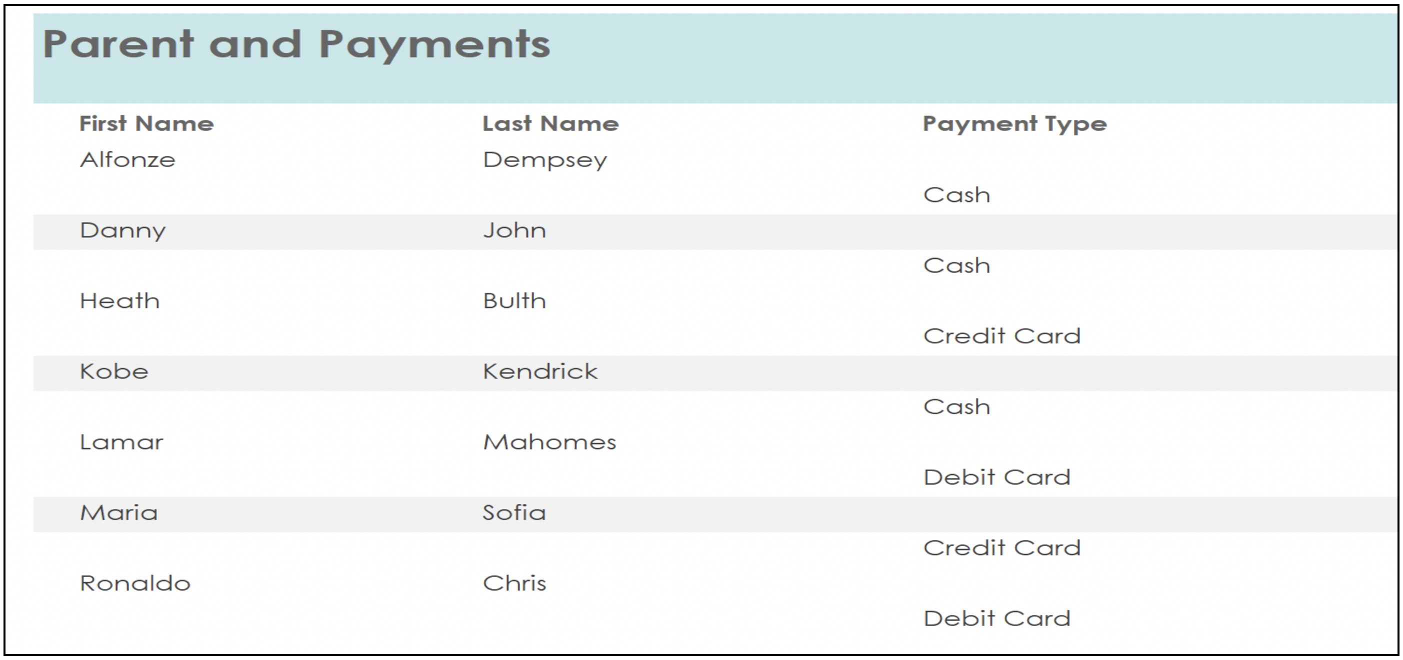


Total Hours and Total Net Amount

First Name	Last Name	Total Hours	Total Net Amount
Edth	Kith	95	505
Jerry	Harry	110	405
John	Rafal	115	505
Kanye	Kim	100	600
Micheal	Douglas	100	500
Nadal	Federer	105	600
Will	Smith	100	500

3. List all parents and their preferred payment types

   SELECT Adult.First_Name, Adult.Last_Name, Payment.Payment_Type
   FROM Adult **INNER JOIN** Payment **ON** Adult.Adult_ID = Payment.Adult_ID;



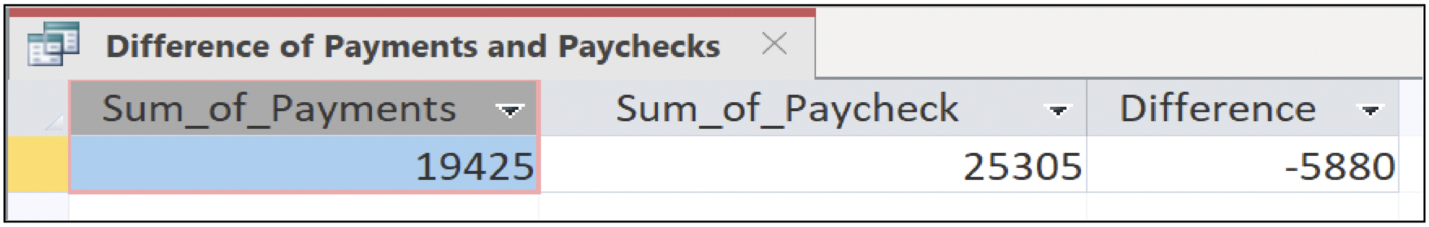
Parent and Payments

First Name	Last Name	Payment Type
Alfonze	Dempsey	Cash
Danny	John	Cash
Heath	Bulth	Credit Card
Kobe	Kendrick	Cash
Lamar	Mahomes	Debit Card
Maria	Sofia	Credit Card
Ronaldo	Chris	Debit Card

Specific questions with SQL Code:

1. Sum of payments, a sum of payroll then get the difference (by month?) Can the business cover payroll from the payments?

   SELECT SUM(Hours_Enrolled * Pay_Rate) AS Sum_of_Payments, SUM(Net_Amount) AS Sum_of_Paycheck, (Sum_of_Payments - Sum_of_Paycheck) AS Difference
   FROM Payment, Pay_Check;



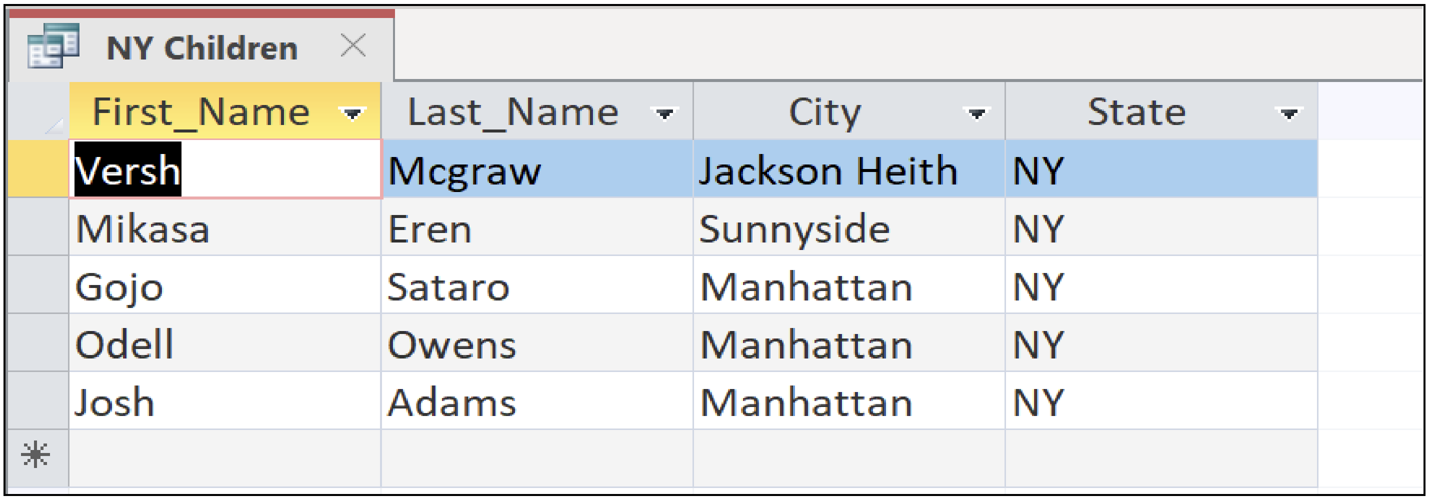
Difference of Payments and Paychecks

Sum_of_Payments	Sum_of_Paycheck	Difference
19425	25305	-5880

   No, the business cannot cover payroll from the payments.

2. For all the children that are registered in our daycare, show me all the children located in NY.

   SELECT Child.First_Name, Child.Last_Name, Adult.City, Adult.State
   FROM (Child **INNER JOIN** Role **ON** Child.Child_ID = Role.Child_ID)
   INNER JOIN Adult **ON** Role.Adult_ID = Adult.Adult_ID
   WHERE Adult.State = "NY";

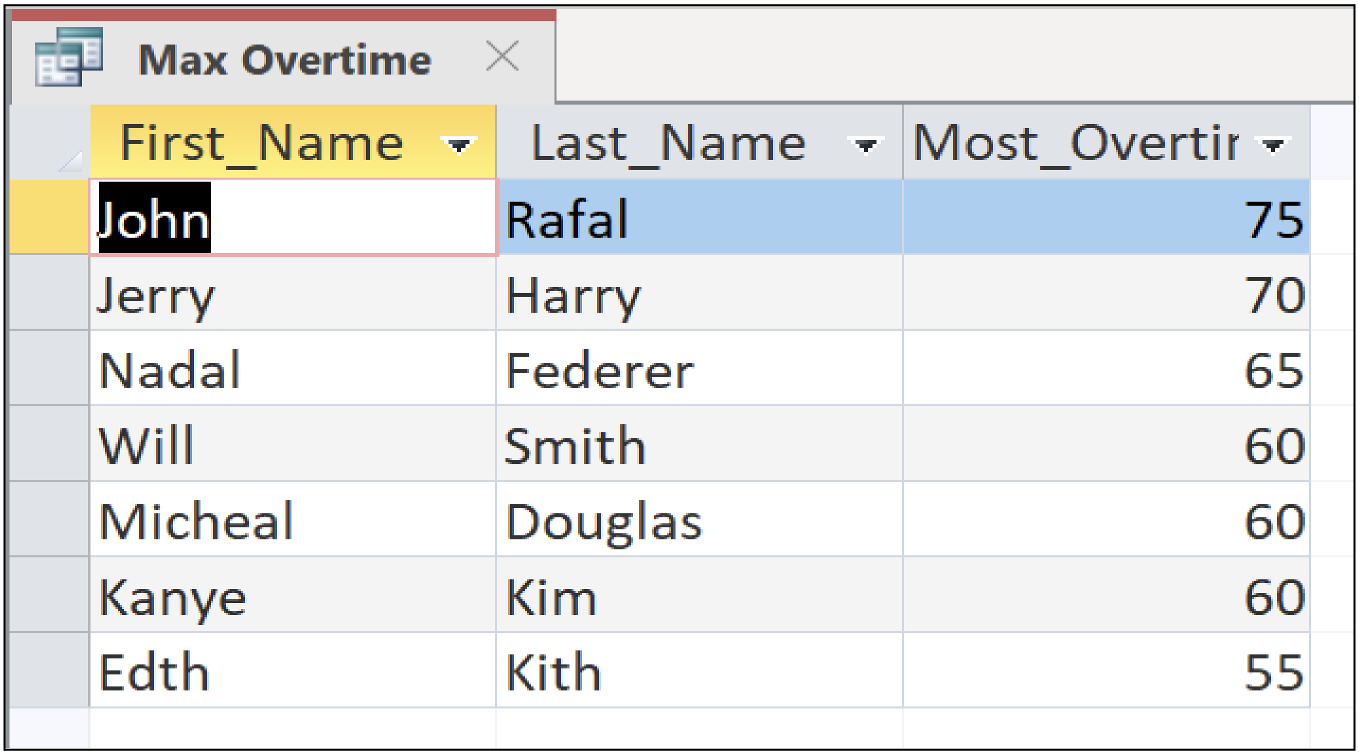


NY Children

First_Name	Last_Name	City	State
Versh	Mcgraw	Jackson Heith	NY
Mikasa	Eren	Sunnyside	NY
Gojo	Sataro	Manhattan	NY
Odell	Owens	Manhattan	NY
Josh	Adams	Manhattan	NY

3. Employees with the most overtime hours

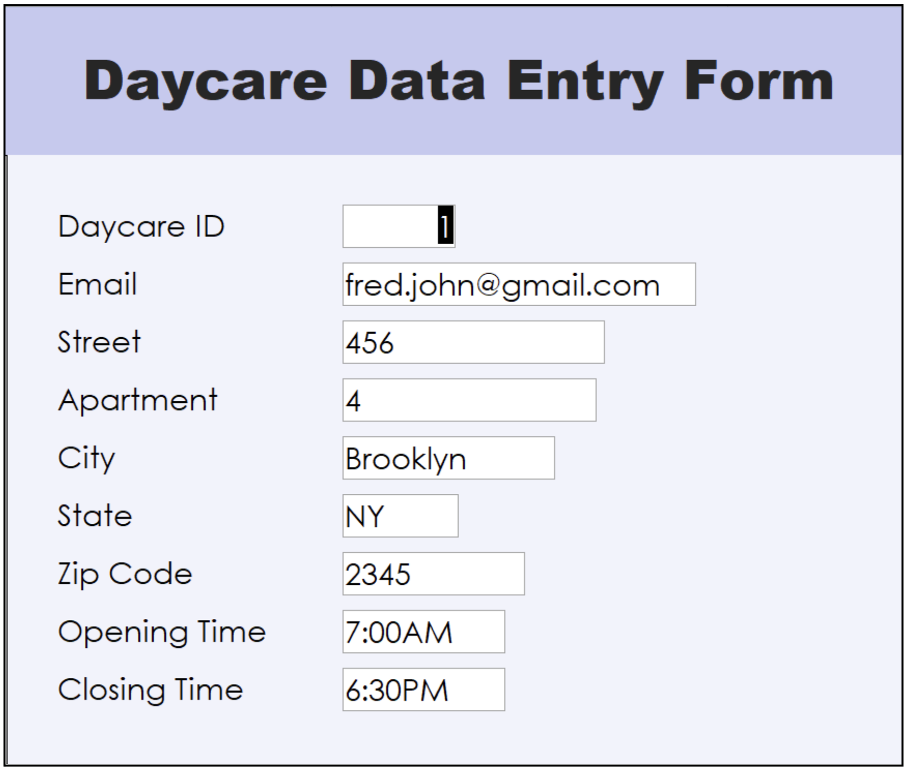
SELECT First_Name, Last_Name, MAX(Overtime_Hours) AS Most_Overtime
FROM Employee_Info **INNER JOIN** TimeSheet **ON** Employee_Info.Employee_ID =
TimeSheet.Employee_ID
GROUP BY First_Name, Last_Name
ORDER BY MAX(Overtime_Hours) DESC;



Max Overtime

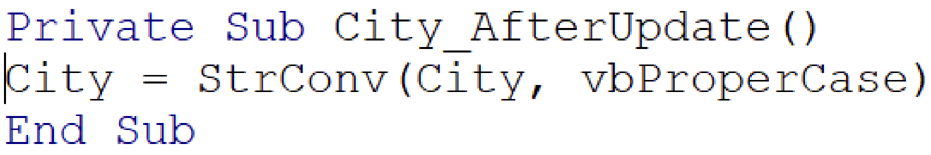
First_Name	Last_Name	Most_Overtir
John	Rafal	75
Jerry	Harry	70
Nadal	Federer	65
Will	Smith	60
Micheal	Douglas	60
Kanye	Kim	60
Edth	Kith	55

Forms, Reports, and Queries:



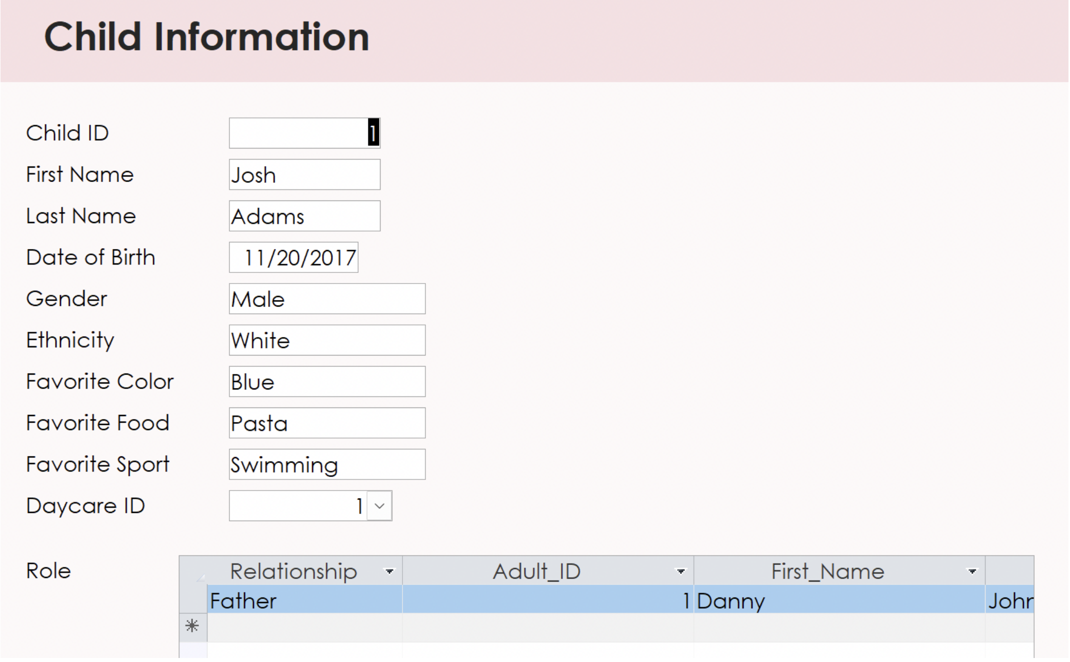
Daycare Data Entry Form

Daycare ID	1
Email	fred.john@gmail.com
Street	456
Apartment	4
City	Brooklyn
State	NY
Zip Code	2345
Opening Time	7:00AM
Closing Time	6:30PM



```
Private Sub City_AfterUpdate()
City = StrConv(City, vbProperCase)
End Sub
```

The **Daycare Data Entry Form** is used to input and update the daycare basic information. After the previous person enters their information and presses the enter on the keyboard to save, the next person can enter their information. The VBA code capitalizes the first letter of the City field.

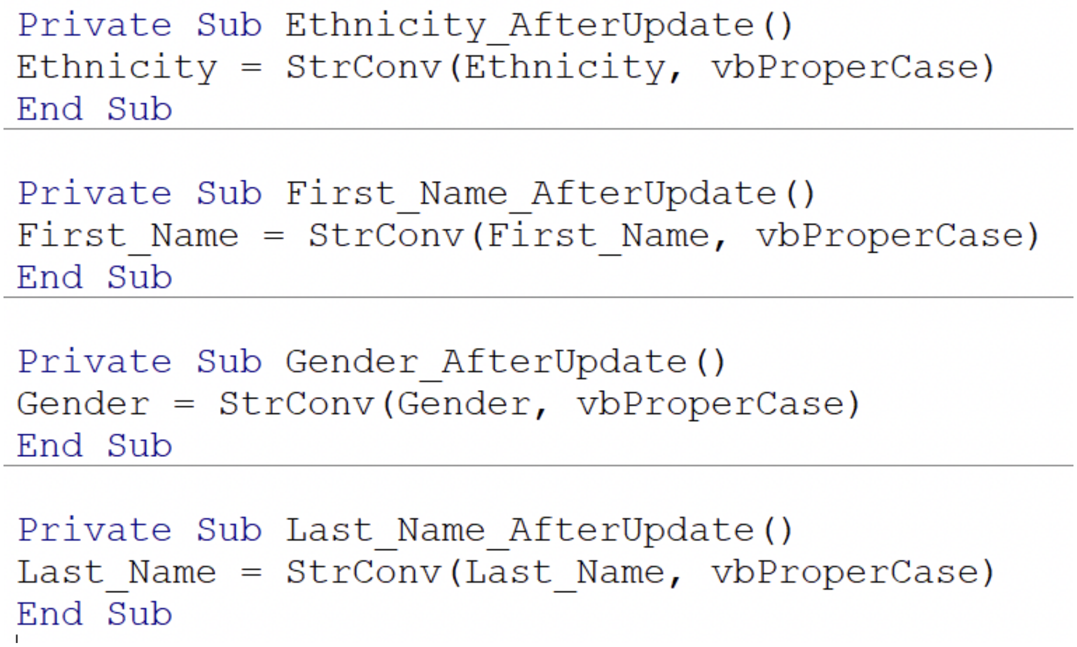


Child Information

Child ID	1
First Name	Josh
Last Name	Adams
Date of Birth	11/20/2017
Gender	Male
Ethnicity	White
Favorite Color	Blue
Favorite Food	Pasta
Favorite Sport	Swimming
Daycare ID	1

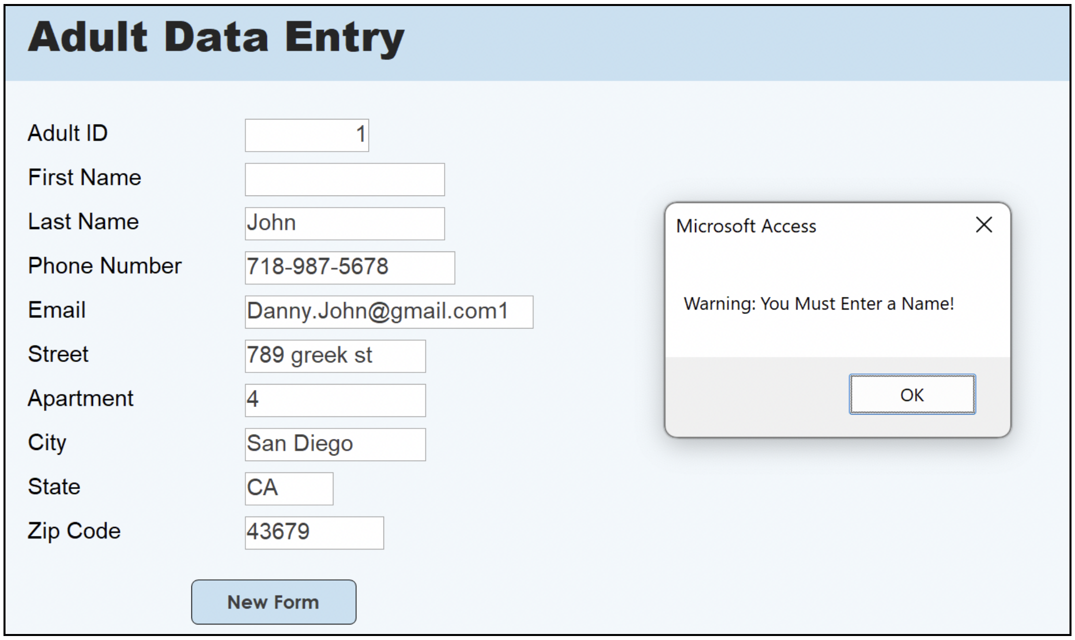
Role

Relationship	Adult_ID	First_Name	
Father	1	Danny	Johr
*			



```
Private Sub Ethnicity_AfterUpdate()
Ethnicity = StrConv(Ethnicity, vbProperCase)
End Sub

Private Sub First_Name_AfterUpdate()
First_Name = StrConv(First_Name, vbProperCase)
End Sub

Private Sub Gender_AfterUpdate()
Gender = StrConv(Gender, vbProperCase)
End Sub

Private Sub Last_Name_AfterUpdate()
Last_Name = StrConv(Last_Name, vbProperCase)
End Sub
```

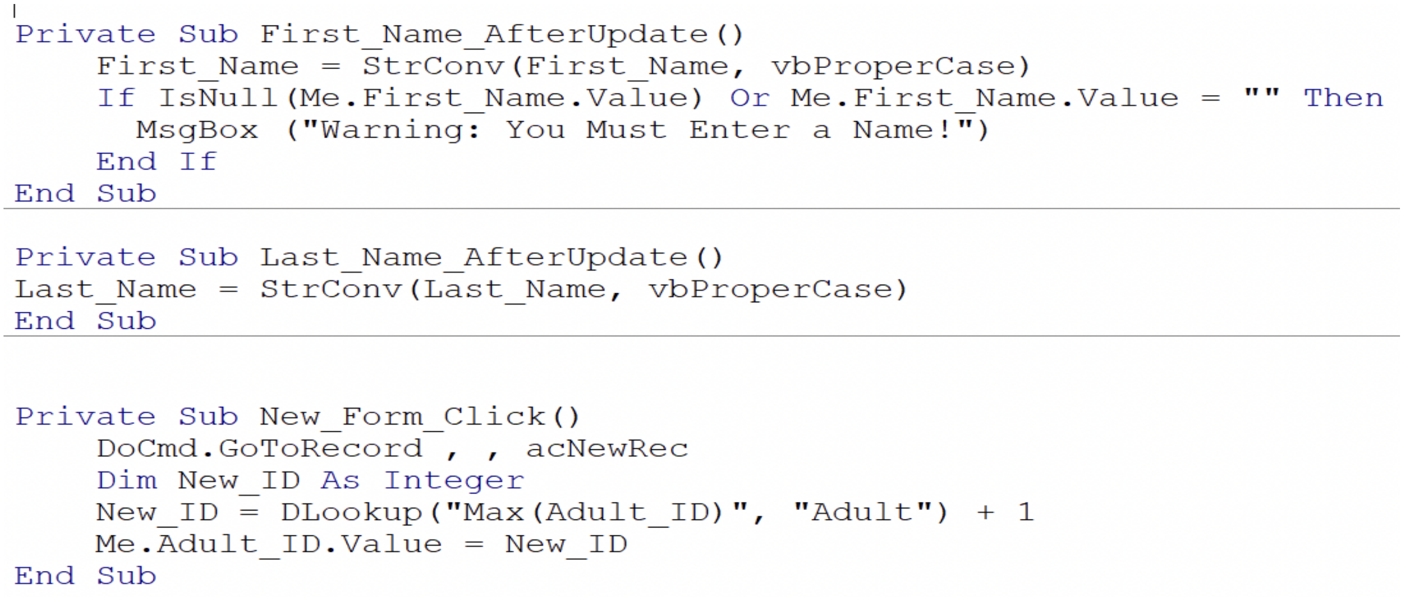
The **Child Form** is used to input interesting and basic information about the child. The daycare that the child is enrolled in can be selected from the Daycare_ID as shown. Under Role Subform, users can input the relationship of the child and the adult, making it easier to track in the system. Additionally, VBA code has been implemented to auto-capitalize the first letter of the Ethnicity, City, First_Name, and Last_Name fields.



Adult Data Entry

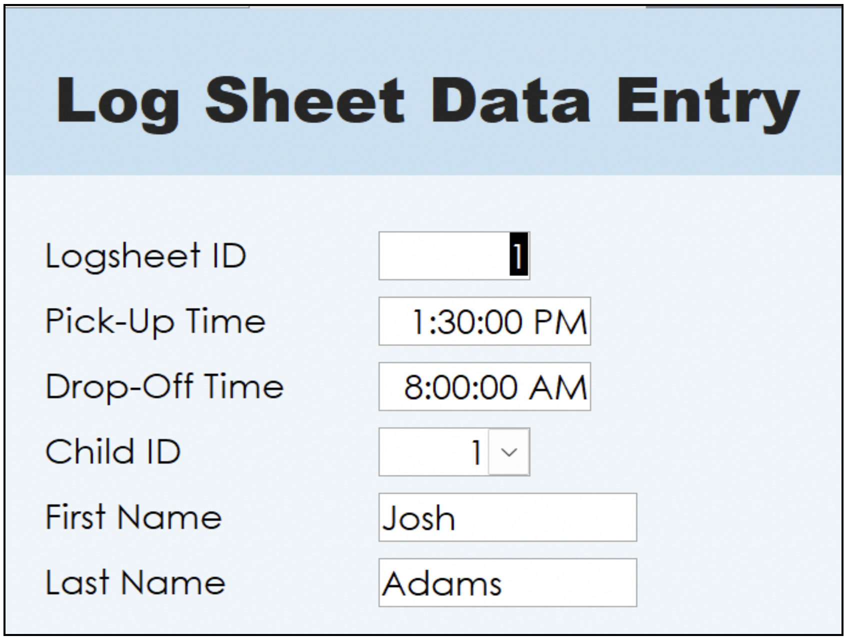
Adult ID 1
First Name
Last Name John
Phone Number 718-987-5678
Email Danny.John@gmail.com1
Street 789 greek st
Apartment 4
City San Diego
State CA
Zip Code 43679

Warning: You Must Enter a Name!



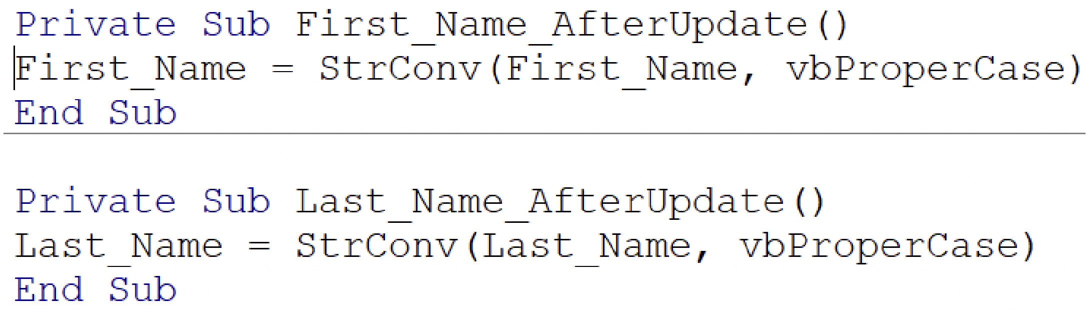
```
Private Sub First_Name_AfterUpdate()
    First_Name = StrConv(First_Name, vbProperCase)
    If IsNull(Me.First_Name.Value) Or Me.First_Name.Value = "" Then
      MsgBox ("Warning: You Must Enter a Name!")
    End If
End Sub

Private Sub Last_Name_AfterUpdate()
Last_Name = StrConv(Last_Name, vbProperCase)
End Sub


Private Sub New_Form_Click()
    DoCmd.GoToRecord , , acNewRec
    Dim New_ID As Integer
    New_ID = DLookup("Max(Adult_ID)", "Adult") + 1
    Me.Adult_ID.Value = New_ID
End Sub
```

The **Adult Data Entry Form** is used to input information of the adult, who's related to the child(ren) entrusted in the daycare. This form features VBA code to capitalize the first letter of the First_Name, Last_Name fields. A warning message also pops up if the First_Name field is left blank. Lastly, when the button "New Form" is clicked, a new form populates with a unique Adult_ID.



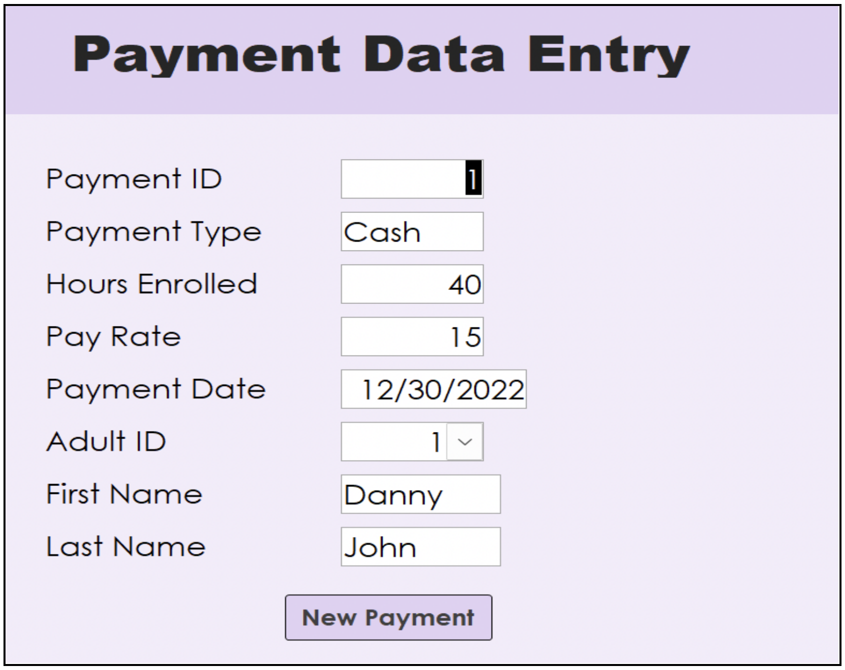
Log Sheet Data Entry

Logsheet ID	1
Pick-Up Time	1:30:00 PM
Drop-Off Time	8:00:00 AM
Child ID	1
First Name	Josh
Last Name	Adams



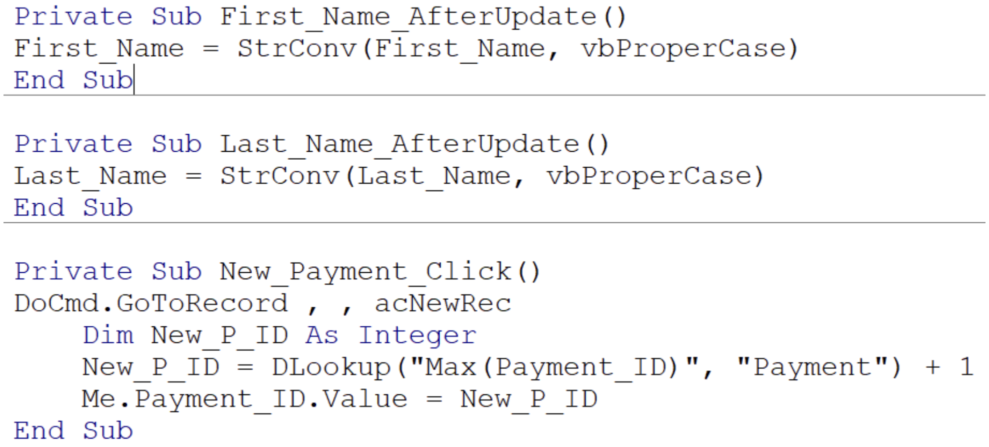
```
Private Sub First_Name_AfterUpdate()
First_Name = StrConv(First_Name, vbProperCase)
End Sub

Private Sub Last_Name_AfterUpdate()
Last_Name = StrConv(Last_Name, vbProperCase)
End Sub
```

The **Log_Sheet Data Entry Form** is used to log when and how long the child is in the daycare, which will be related to payment after. Users can select the child from the Child_ID as shown. This VBA code automatically capitalizes the first letter of the first and last names.



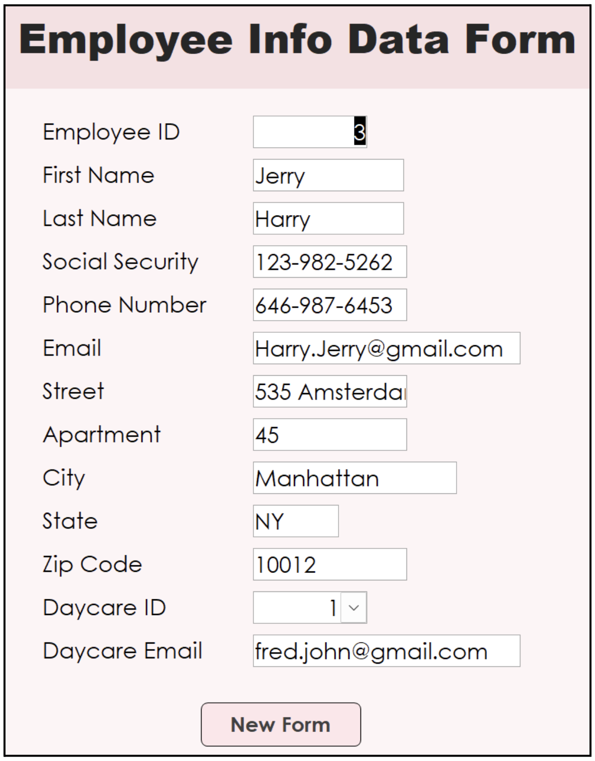
Payment Data Entry

Payment ID	1
Payment Type	Cash
Hours Enrolled	40
Pay Rate	15
Payment Date	12/30/2022
Adult ID	1
First Name	Danny
Last Name	John



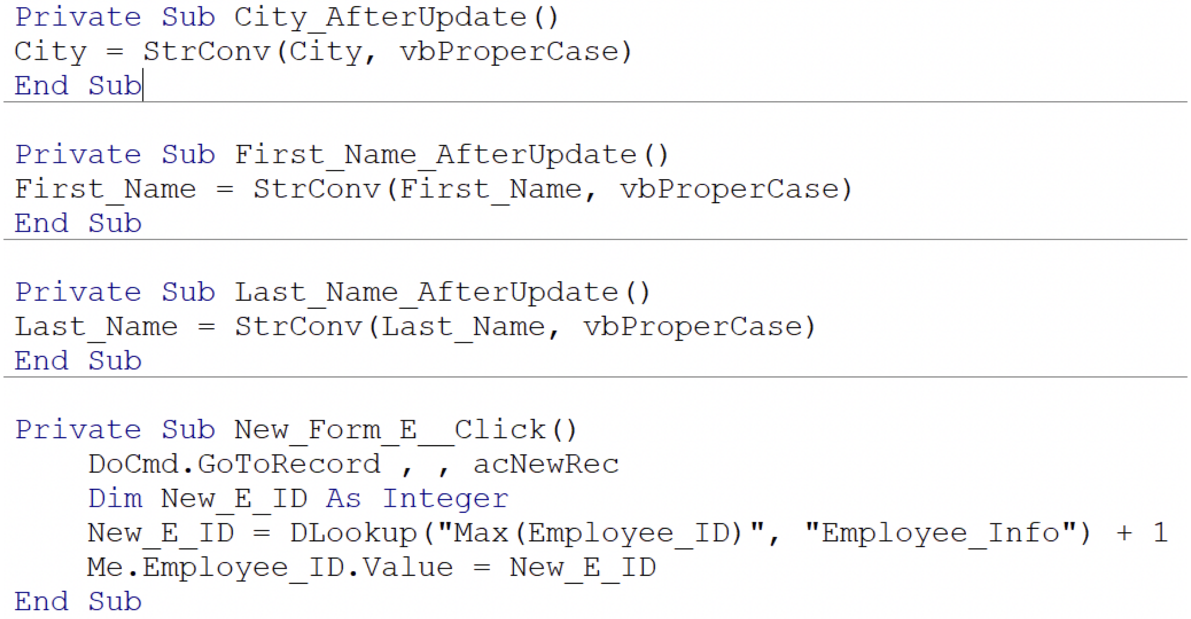
```
Private Sub First_Name_AfterUpdate()
First_Name = StrConv(First_Name, vbProperCase)
End Sub

Private Sub Last_Name_AfterUpdate()
Last_Name = StrConv(Last_Name, vbProperCase)
End Sub

Private Sub New_Payment_Click()
DoCmd.GoToRecord , , acNewRec
    Dim New_P_ID As Integer
    New_P_ID = DLookup("Max(Payment_ID)", "Payment") + 1
    Me.Payment_ID.Value = New_P_ID
End Sub
```

The **Payment Data Entry Form** is used to input and update adult payment to the daycare. Users can select the adult from the Adult_ID as shown. Additionally, VBA codes were inputted to auto-capitalize the first letters of the first and last names. There’s also an option to create a new form which automatically creates a new unique Payment ID.



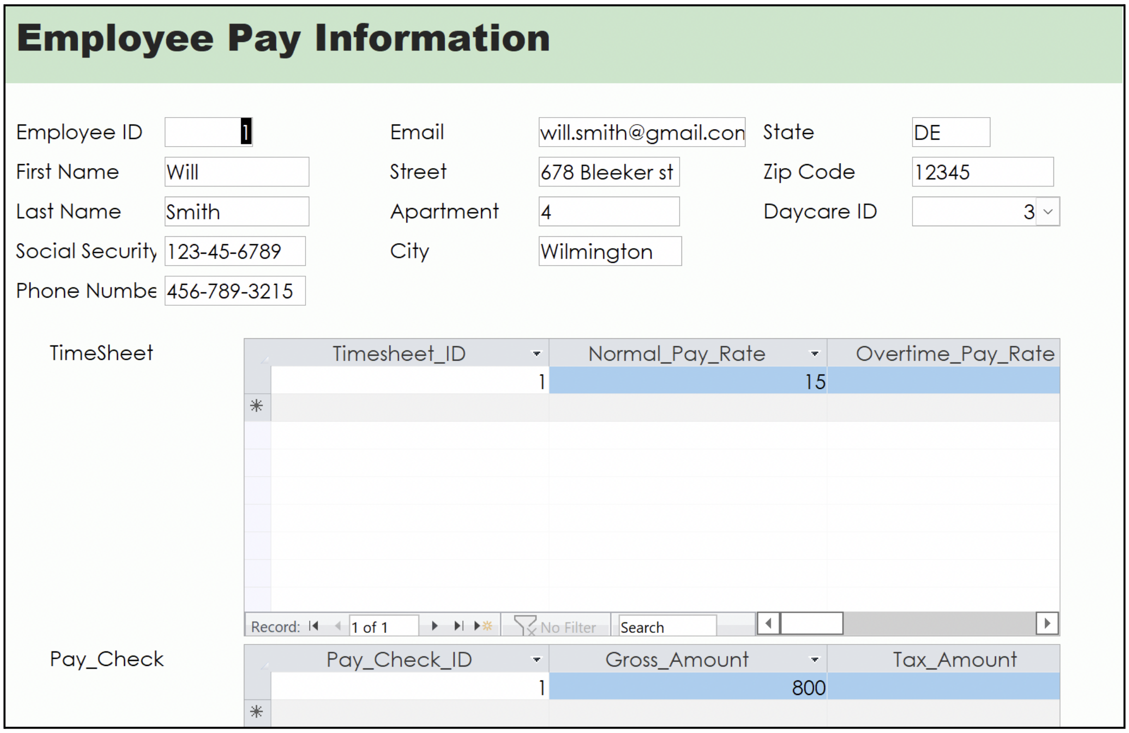
Employee Info Data Form

Employee ID	3
First Name	Jerry
Last Name	Harry
Social Security	123-982-5262
Phone Number	646-987-6453
Email	Harry.Jerry@gmail.com
Street	535 Amsterdai
Apartment	45
City	Manhattan
State	NY
Zip Code	10012
Daycare ID	1
Daycare Email	fred.john@gmail.com



```
Private Sub City_AfterUpdate()
City = StrConv(City, vbProperCase)
End Sub

Private Sub First_Name_AfterUpdate()
First_Name = StrConv(First_Name, vbProperCase)
End Sub

Private Sub Last_Name_AfterUpdate()
Last_Name = StrConv(Last_Name, vbProperCase)
End Sub

Private Sub New_Form_E__Click()
    DoCmd.GoToRecord , , acNewRec
    Dim New_E_ID As Integer
    New_E_ID = DLookup("Max(Employee_ID)", "Employee_Info") + 1
    Me.Employee_ID.Value = New_E_ID
End Sub
```

The **Employee_Info Data Form** is used to input and update the employee basic information. Users can select the daycare based on where they work from the Daycare_ID as shown. VBA code was implemented to auto-capitalize the first letters of the First_Name, Last_Name, and City fields. Additionally, users can click on the button to create a new form that will automatically generate a unique Employee ID as well.



Employee Pay Information

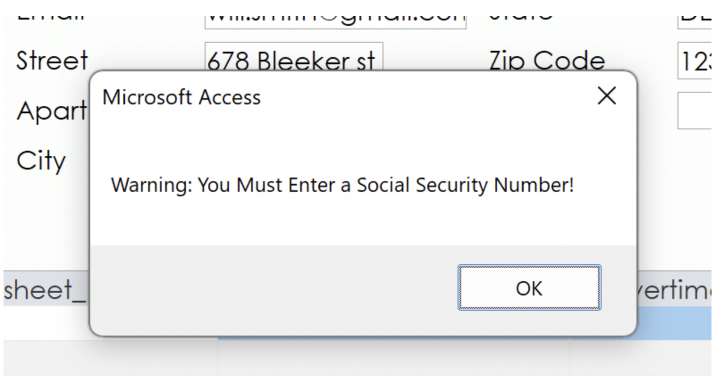
Employee ID	1	Email	will.smith@gmail.con	State	DE
First Name	Will	Street	678 Bleeker st	Zip Code	12345
Last Name	Smith	Apartment	4	Daycare ID	3
Social Security	123-45-6789	City	Wilmington		
Phone Numbe	456-789-3215				

TimeSheet

Timesheet_ID	Normal_Pay_Rate	Overtime_Pay_Rate
1	15	

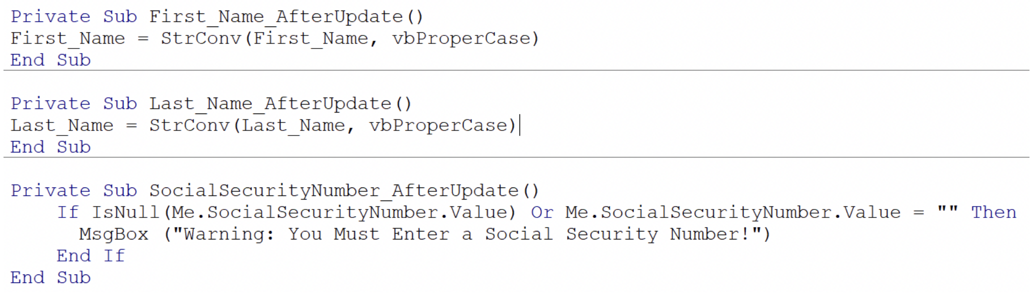
Pay_Check

Pay_Check_ID	Gross_Amount	Tax_Amount
1	800	



Microsoft Access

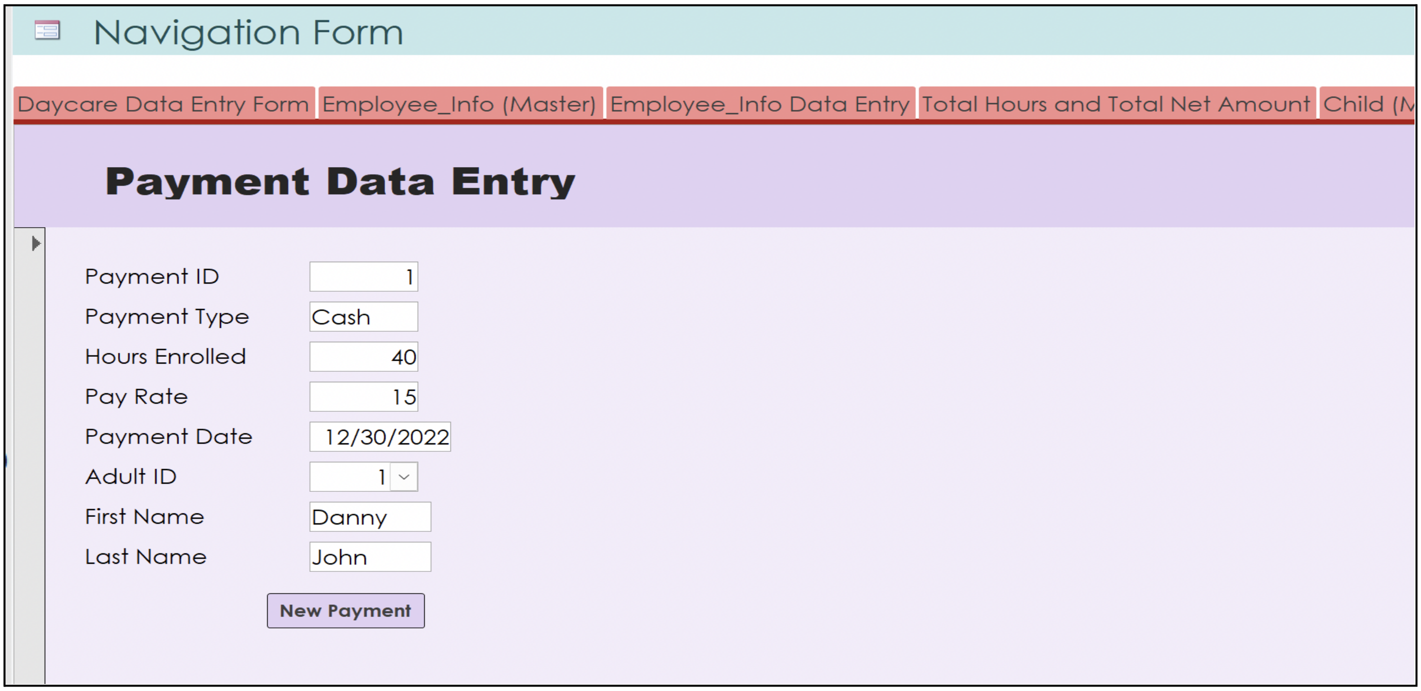
Warning: You Must Enter a Social Security Number!



```
Private Sub First_Name_AfterUpdate()
First_Name = StrConv(First_Name, vbProperCase)
End Sub

Private Sub Last_Name_AfterUpdate()
Last_Name = StrConv(Last_Name, vbProperCase)
End Sub

Private Sub SocialSecurityNumber_AfterUpdate()
    If IsNull(Me.SocialSecurityNumber.Value) Or Me.SocialSecurityNumber.Value = "" Then
      MsgBox ("Warning: You Must Enter a Social Security Number!")
    End If
End Sub
```

The Employee Pay Information Form can be used to input and update Employee information and payment information in the TimeSheet and Pay_Check tables. VBA code was implemented to capitalize First_Name, Last_Name. Additionally, a warning message appears if the user leaves the Social Security field blank.

Navigation Form:



Navigation Form

Payment Data Entry

Payment ID	1
Payment Type	Cash
Hours Enrolled	40
Pay Rate	15
Payment Date	12/30/2022
Adult ID	1
First Name	Danny
Last Name	John

The Navigation Form features all of the forms and reports grouped into sections:

- Daycare Business Related Forms
  - Daycare Data Entry Form
  - Employee_Info (Master)
  - Employee_Info Data Entry
  - Total Hours and Total Net Amount Report
- Child Related Forms
  - Child (Master Detail-Lookup)
  - Log_Sheet Data Entry
  - Child and Relationships Report
- Adult Related Forms
  - Adult Data Entry Form
  - Parent and Payments Report
  - Payment Data Entry

Conclusion:

By developing a database application, we help Nicole’s Family Daycare business to systemize its software and services. Where she can easily input and track her daycare, customers, employees, payments and other important information. Our Daycare Model highlights the key elements of a database development lifecycle. From crafting a proposal based on her needs, creating a conceptual data model to provide a visualized understanding, and finalizing the database with logical, physical data models and application developments, making simple access for users.

The E-R Model, normalization and relation model were the simpler steps. Mostly, we just need to make sure to convert the functional dependencies properly, and many were already in third normal form. While database and application implementation require the most attention and effort, especially when the codes and forms might not work or slightly off. Overall, the project brought us a great experience, taught us how to use Access and it is a great skillset to put on our resume to find more internships and carrer. We learned much about using the wizard queries and, unexpectedly, the relation model as well. Next time, we can improve on cutting back a few entities and focusing more on each entity in detail instead to create a better, solid database system.

Project Link