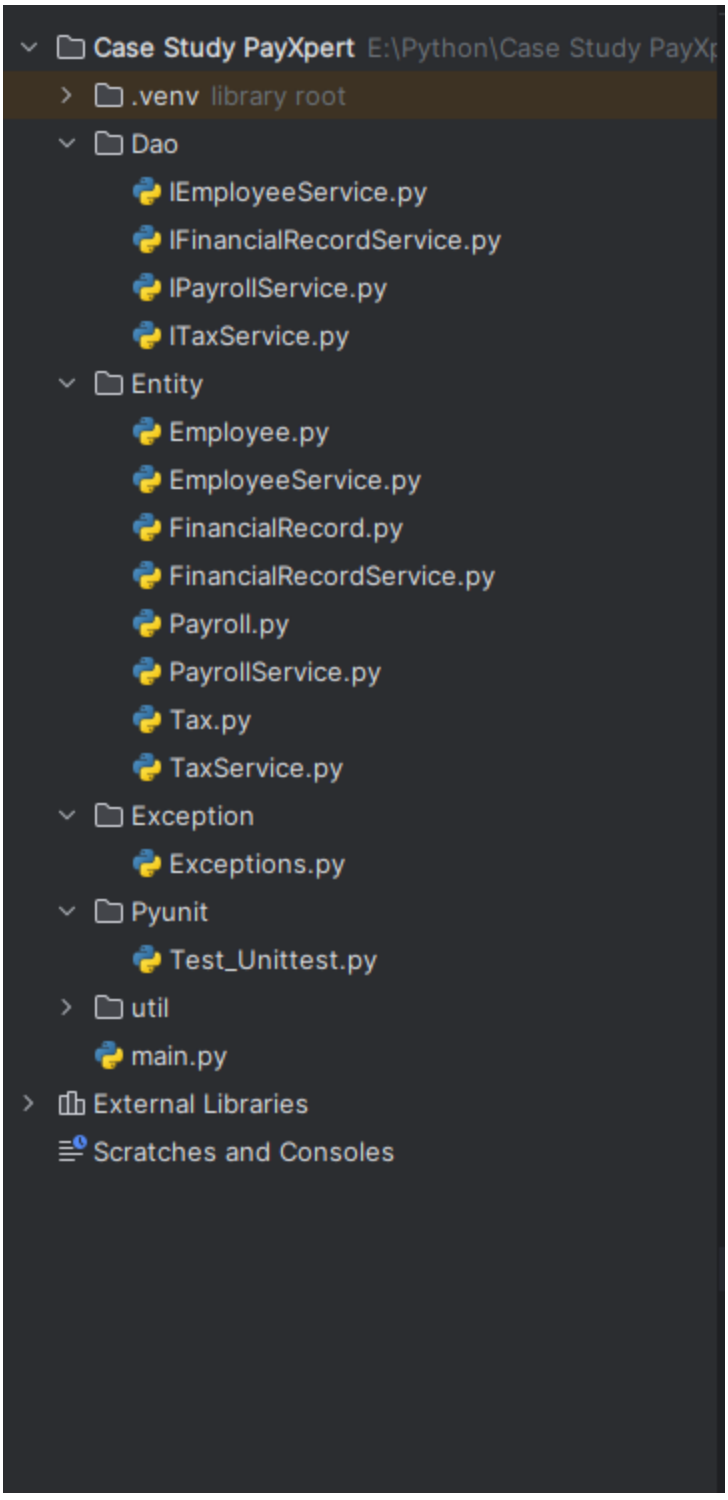

Case Study PayXpert E:\Python\Case Study PayXp
.venv library root
Dao
IEmployeeService.py
IFinancialRecordService.py
IPayrollService.py
ITaxService.py
Entity
Employee.py
EmployeeService.py
FinancialRecord.py
FinancialRecordService.py
Payroll.py
PayrollService.py
Tax.py
TaxService.py
Exception
Exceptions.py
Pyunit
Test_Unittest.py
util
main.py
External Libraries
Scratches and Consoles

SQL Tables:

1. Employee Table:

```
mysql> desc employee;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| EmployeeID      | int  | NO   | PRI | NULL    |       |
| FirstName       | text | YES  |     | NULL    |       |
| LastName        | text | YES  |     | NULL    |       |
| DateOfBirth     | date | YES  |     | NULL    |       |
| Gender          | text | YES  |     | NULL    |       |
| Email           | text | YES  |     | NULL    |       |
| PhoneNumber     | text | YES  |     | NULL    |       |
| Address         | text | YES  |     | NULL    |       |
| Position        | text | YES  |     | NULL    |       |
| JoiningDate     | date | YES  |     | NULL    |       |
| TerminationDate | date | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
11 rows in set (0.01 sec)
```

2. Payroll Table:

```
mysql> desc payroll;
+----------------------+------+------+-----+---------+-------+
| Field                | Type | Null | Key | Default | Extra |
+----------------------+------+------+-----+---------+-------+
| PayrollID            | int  | NO   | PRI | NULL    |       |
| EmployeeID           | int  | YES  | MUL | NULL    |       |
| PayPeriodStartingdate| date | YES  |     | NULL    |       |
| PayPeriodEndingdate  | date | YES  |     | NULL    |       |
| BasicSalary          | int  | YES  |     | NULL    |       |
| OvertimePay          | int  | YES  |     | NULL    |       |
| Deductions           | int  | YES  |     | NULL    |       |
| NetSalary            | int  | YES  |     | NULL    |       |
+----------------------+------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

3. Tax Table:

```
mysql> desc tax;
+---------------+------+------+-----+---------+-------+
| Field         | Type | Null | Key | Default | Extra |
+---------------+------+------+-----+---------+-------+
| TaxID         | int  | NO   | PRI | NULL    |       |
| EmployeeID    | int  | YES  | MUL | NULL    |       |
| TaxYear       | year | YES  |     | NULL    |       |
| TaxableIncome | int  | YES  |     | NULL    |       |
| TaxAmount     | int  | YES  |     | NULL    |       |
+---------------+------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

4. FinancialRecord Table:

```
mysql> desc FinancialRecord ;
+-------------+-----------------------------------------+------+-----+---------+-------+
| Field       | Type                                    | Null | Key | Default | Extra |
+-------------+-----------------------------------------+------+-----+---------+-------+
| RecordID    | int                                     | NO   | PRI | NULL    |       |
| EmployeeID  | int                                     | YES  | MUL | NULL    |       |
| RecordDate  | date                                    | YES  |     | NULL    |       |
| Description | text                                    | YES  |     | NULL    |       |
| Amount      | int                                     | YES  |     | NULL    |       |
| RecordType  | enum('Income','Expense','Tax Payment') | YES  |     | NULL    |       |
+-------------+-----------------------------------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Classes :

• Employee:

• Properties: EmployeeID, FirstName, LastName, DateOfBirth, Gender, Email,

PhoneNumber, Address, Position, JoiningDate, TerminationDate

```python

2 usages
class Employee:
    def __init__(self, employeeID=None, firstname=None, lastname=None, dateofbirth=None, gender=None, email=None,
                 phonenumber=None, address=None, position=None, joiningdate=None, terminationdate=None, db_connector=None):
        self._employeeID = employeeID
        self._Firstname = firstname
        self._Lastname = lastname
        self._DateofBirth = dateofbirth
        self._Gender = gender
        self._email = email
        self._Phonenumber = phonenumber
        self._Address = address
        self._Position = position
        self._joiningDate = joiningdate
        self._terminationDate = terminationdate
        self._db_connector = db_connector

    1 usage
    @property
    def employeeId(self):
        return self._employeeID

    @employeeId.setter
    def employeeId(self, new_employeeId):
        self._employeeID = new_employeeId

    1 usage
    @property
    def Firstname(self):
        return self._Firstname

    @Firstname.setter
    def Firstname(self, new_Firstname):
        self._Firstname = new_Firstname

    1 usage
    @property
    def Lastname(self):
        return self._Lastname

```

```
        return self._Lastname

    @Lastname.setter
    def Lastname(self, new_Lastname):
        self._Lastname = new_Lastname

    1 usage
    @property
    def DateofBirth(self):
        return self._DateofBirth

    @DateofBirth.setter
    def DateofBirth(self, new_DateofBirth):
        self._DateofBirth = new_DateofBirth

    1 usage
    @property
    def Gender(self):
        return self._Gender

    @Gender.setter
    def Gender(self, new_Gender):
        self._Gender = new_Gender

    1 usage
    @property
    def email(self):
        return self._email

    @email.setter
    def email(self, new_email):
        self._email = new_email

    1 usage
    @property
    def Phonenumber(self):
        return self._Phonenumber

    @Phonenumber.setter
    def Phonenumber(self, new_Phonenumber):
        self._Phonenumber = new_Phonenumber
```

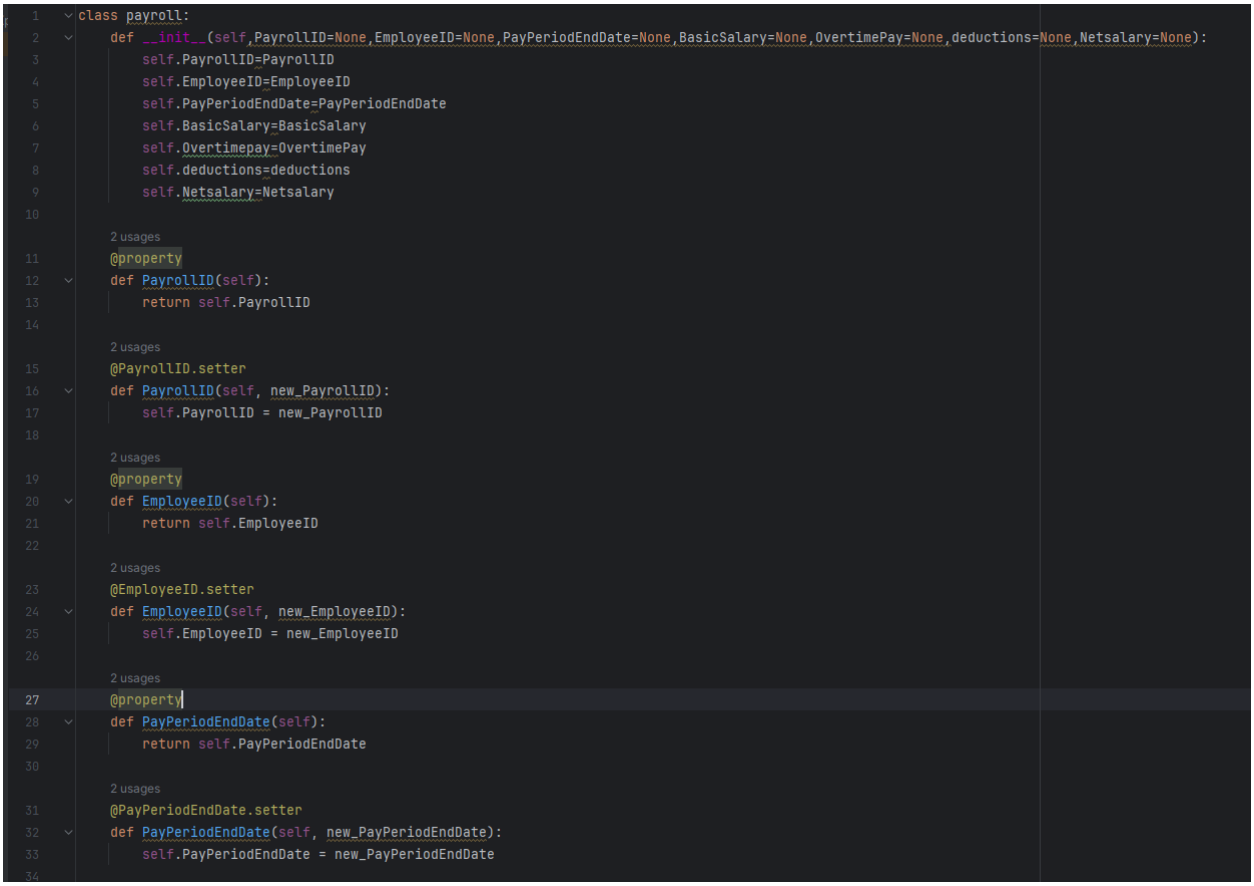
```
    1 usage
    @property
    def Address(self):
        return self._Address

    @Address.setter
    def Address(self, new_Address):
        self._Address = new_Address

    1 usage
    @property
    def Position(self):
        return self._Position

    @Position.setter
    def Position(self, new_Position):
        self._Position = new_Position

    1 usage
    @property
    def joiningDate(self):
        return self._joiningDate

    @joiningDate.setter
    def joiningDate(self, new_joiningDate):
        self._joiningDate = new_joiningDate

    1 usage
    @property
    def terminationDate(self):
        return self._terminationDate

    @terminationDate.setter
    def terminationDate(self, new_terminationDate):
        self._terminationDate = new_terminationDate
```

- **Methods: CalculateAge()**

```
1 usage
def CalculateAge(self):
    try:
        employeeID=input("Enter your EmployeeID for which you want to know the age : ")
        self._db_connector.open_connection()
        cur = self._db_connector.connection.cursor()
        query = "Select timestampdiff(YEAR,DateOfBirth,CurDate()) from Employee where EmployeeID=%s"
        value = (employeeID,)
        cur.execute(query, value)
        record = cur.fetchone()
        if record:
            print(f"The age of Employee whose employee ID is {employeeID} = {record[0]}")
        else:
            print("NO Employee Found")
    except Exception as e:
        print("Error in Fetching age")
    finally:
        self._db_connector.connection.close()
```

• Payroll:

• Properties: PayrollID, EmployeeID, PayPeriodStartDate, PayPeriodEndDate, BasicSalary, OvertimePay, Deductions, NetSalary



```python
class payroll:
    def __init__(self,PayrollID=None,EmployeeID=None,PayPeriodEndDate=None,BasicSalary=None,OvertimePay=None,deductions=None,Netsalary=None):
        self.PayrollID=PayrollID
        self.EmployeeID=EmployeeID
        self.PayPeriodEndDate=PayPeriodEndDate
        self.BasicSalary=BasicSalary
        self.Overtimepay=OvertimePay
        self.deductions=deductions
        self.Netsalary=Netsalary

    2 usages
    @property
    def PayrollID(self):
        return self.PayrollID

    2 usages
    @PayrollID.setter
    def PayrollID(self, new_PayrollID):
        self.PayrollID = new_PayrollID

    2 usages
    @property
    def EmployeeID(self):
        return self.EmployeeID

    2 usages
    @EmployeeID.setter
    def EmployeeID(self, new_EmployeeID):
        self.EmployeeID = new_EmployeeID

    2 usages
    @property
    def PayPeriodEndDate(self):
        return self.PayPeriodEndDate

    2 usages
    @PayPeriodEndDate.setter
    def PayPeriodEndDate(self, new_PayPeriodEndDate):
        self.PayPeriodEndDate = new_PayPeriodEndDate
```

```python
    2 usages
    @property
    def BasicSalary(self):
        return self.BasicSalary

    2 usages
    @BasicSalary.setter
    def BasicSalary(self, new_BasicSalary):
        self.BasicSalary = new_BasicSalary

    2 usages
    @property
    def Overtimepay(self):
        return self.Overtimepay

    2 usages
    @Overtimepay.setter
    def Overtimepay(self, new_Overtimepay):
        self.Overtimepay = new_Overtimepay

    2 usages
    @property
    def deductions(self):
        return self.deductions

    2 usages
    @deductions.setter
    def deductions(self, new_deductions):
        self.deductions = new_deductions

    2 usages
    @property
    def Netsalary(self):
        return self.Netsalary

    2 usages
    @Netsalary.setter
    def Netsalary(self, new_Netsalary):
        self.Netsalary = new_Netsalary
```

• Tax:

• Properties: TaxID, EmployeeID, TaxYear, TaxableIncome, TaxAmount

```python
class tax:
    def __init__(self,TaxID=None, EmployeeID=None, TaxYear=None, TaxableIncome=None, TaxAmount=None):
        self.TaxID=TaxID
        self.employeeId=EmployeeID
        self.taxYear=TaxYear
        self.TaxableIncome=TaxableIncome
        self.TaxAmount=TaxAmount

    2 usages
    @property
    def TaxID(self):
        return self.TaxID

    2 usages
    @TaxID.setter
    def TaxID(self, new_TaxID):
        self.TaxID = new_TaxID

    2 usages
    @property
    def EmployeeID(self):
        return self.EmployeeID

    1 usage
    @EmployeeID.setter
    def EmployeeID(self, new_EmployeeID):
        self.EmployeeID = new_EmployeeID

    2 usages
    @property
    def TaxYear(self):
        return self.TaxYear

    1 usage
    @TaxYear.setter
    def TaxYear(self, new_TaxYear):
        self.TaxYear = new_TaxYear

    2 usages
    @property
    def TaxableIncome(self):
```

```
        return self.TaxableIncome

    2 usages
    @TaxableIncome.setter
    def TaxableIncome(self, new_TaxableIncome):
        self.TaxableIncome = new_TaxableIncome

    2 usages
    @property
    def TaxAmount(self):
        return self.TaxAmount

    2 usages
    @TaxAmount.setter
    def TaxAmount(self, new_TaxAmount):
        self.TaxAmount = new_TaxAmount
```

• FinancialRecord:

• Properties: RecordID, EmployeeID, RecordDate, Description, Amount, RecordType

```
class financialrecord:
    def __init__(self, RecordID=None, EmployeeID=None, RecordDate=None, Description=None, Amount=None, RecordType=None):
        self.recordID=RecordID
        self.employeeID=EmployeeID
        self.RecordDate=RecordDate
        self.description=Description
        self.Amount=Amount
        self.RecordType=RecordType

    2 usages
    @property
    def recordID(self):
        return self.recordID

    2 usages
    @recordID.setter
    def recordID(self, new_recordID):
        self.recordID = new_recordID

    2 usages
    @property
    def employeeID(self):
        return self.employeeID

    2 usages
    @employeeID.setter
    def employeeID(self, new_employeeID):
        self.employeeID = new_employeeID

    2 usages
    @property
    def RecordDate(self):
        return self.RecordDate

    2 usages
    @RecordDate.setter
    def RecordDate(self, new_RecordDate):
        self.RecordDate = new_RecordDate

    2 usages
```

```
        return self.RecordDate

    2 usages
    @RecordDate.setter
    def RecordDate(self, new_RecordDate):
        self.RecordDate = new_RecordDate

    2 usages
    @property
    def description(self):
        return self.description

    2 usages
    @description.setter
    def description(self, new_description):
        self.description = new_description

    2 usages
    @property
    def Amount(self):
        return self.Amount

    2 usages
    @Amount.setter
    def Amount(self, new_Amount):
        self.Amount = new_Amount

    2 usages
    @property
    def RecordType(self):
        return self.RecordType

    2 usages
    @RecordType.setter
    def RecordType(self, new_RecordType):
        self.RecordType = new_RecordType
```

• EmployeeService (implements IEmployeeService):

• Methods:

GetEmployeeById,

```python
from Dao.IEmployeeService import IEmployeeService
from Exception.Exceptions import EmployeeNotFoundException
2 usages
class employeeservice(IEmployeeService):
    def __init__(self,db_connector):
        self.db_connector=db_connector

    1 usage
    def GetEmployeeById(self):
        self.db_connector.open_connection()
        cur=self.db_connector.connection.cursor()
        employeeId=input("Please enter your employeeID : ")
        cur.execute("Select * from Employee Where employeeID =%s",(employeeId,))
        record=cur.fetchone()
        if record:
            print("Here are the details of th employee : ")
            print(f"Employee ID = {record[0]}")
            print(f"First name = {record[1]}")
            print(f"Last Name = {record[2]}")
            print(f"Date Of Birth = {record[3]}")
            print(f"Gender = {record[4]}")
            print(f"Email = {record[5]}")
            print(f"Phone Number = {record[6]}")
            print(f"Address = {record[7]}")
            print(f"Position = {record[8]}")
            print(f"Joining Date = {record[9]}")
            print(f"Termination Date = {record[10]}")

        else:
            raise EmployeeNotFoundException(employeeId)
        self.db_connector.close_connection()
```

GetAllEmployees,

```
1 usage
def GetAllEmployees(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    cur.execute("Select FirstName from Employee")
    record=cur.fetchall()
    if record:
        for i in record:
            print(i[0])
    else:
        print("No employee work here")
    self.db_connector.close_connection()
```

AddEmployee,

```
1 usage
def AddEmployee(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    print("Enter the details of employee you want to add")
    employeeID=self.GetuniqueEmployeeID()
    FirstName=input("Enter Frist Name : ")
    LastName=input("Enter Last Name : ")
    DateofBirth=input("Enter Date of birth in the format 'YYYY-MM-DD' : ")
    Gender=input("Enter Gender (Male,Female) : ")
    Email=input("Enter the email address : ")
    PhoneNumber=input("Enter Phone Number : ")
    Address=input("Enter Address : ")
    Positon=input("Enter Position : ")
    Joiningdate=input("Enter Joining Date in the format 'YYYY-MM-DD' :")
    query = "Insert into Employee (EmployeeId, Firstname, Lastname, Dateofbirth, Gender, Email, Phonenumber, Address, Position, Joiningdate) values(%s,%s,%s,%s,%s,%s,%s,%s,
    values=(employeeID,FirstName,LastName,DateofBirth,Gender,Email,PhoneNumber,Address,Positon,Joiningdate,)
    cur.execute(query,values)
    print("Employee Added Successfully")
    self.db_connector.connection.commit()
    self.db_connector.close_connection()
```

UpdateEmployee,

```
1 usage
def UpdateEmployee(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    employeeId = input("Enter the employeeID you want to update : ")
    print("Choose what you want to update : ")
    print("1.First name ")
    print("2.Last Name ")
    print("3.Date Of Birth ")
    print("4.Gender ")
    print("5.Email ")
    print("6.Phone Number ")
    print("7.Address ")
    print("8.Position ")
    print("9.Joining Date ")
    print("10.Termination Date ")
    choice = int(input("Enter your choice : "))
    cur.execute("Select * from Employee Where employeeID =%s", (employeeId,))
    record = cur.fetchone()
    if record:
        FirstName = record[1]
        LastName = record[2]
        DateOfBirth = record[3]
        Gender = record[4]
        Email = record[5]
        PhoneNumber = record[6]
        Address = record[7]
        Position = record[8]
        JoiningDate = record[9]
        TerminationDate = record[10]

        if choice == 1:
            FirstName = input("Enter Frist Name : ")
        elif choice == 2:
            LastName = input("Enter Last Name : ")
        elif choice == 3:
            DateOfBirth = input("Enter Date of birth in the format 'YYYY-MM-DD' : ")
```

```
    if choice == 1:
        FirstName = input("Enter Frist Name : ")
    elif choice == 2:
        LastName = input("Enter Last Name : ")
    elif choice == 3:
        DateOfBirth = input("Enter Date of birth in the format 'YYYY-MM-DD' : ")
    elif choice == 4:
        Gender = input("Enter Gender (Male,Female) : ")
    elif choice == 5:
        Email = input("Enter the email address : ")
    elif choice == 6:
        PhoneNumber = input("Enter Phone Number : ")
    elif choice == 7:
        Address = input("Enter Address : ")
    elif choice == 8:
        Position = input("Enter Position : ")
    elif choice == 9:
        JoiningDate = input("Enter Joining Date in the format 'YYYY-MM-DD' :")
    elif choice == 10:
        TerminationDate = input("Enter Termination Date in the format 'YYYY-MM-DD' : ")
    else:
        print("Enter correct choice !!! ")
    query = "Update employee set Firstname=%s,Lastname=%s,Dateofbirth=%s, Gender=%s, Email=%s, Phonenumber=%s, Address=%s, position=%s, joiningdate=%s, TerminationDat
    values = (FirstName, LastName, DateOfBirth, Gender, Email, PhoneNumber, Address, Position, JoiningDate,
              TerminationDate, employeeId,)
    cur.execute(query, values)
    print("Updated Successfully")
    self.db_connector.connection.commit()
else:
    raise EmployeeNotFoundException(employeeId)
self.db_connector.close_connection()
```

RemoveEmployee

```
1 usage
def RemoveEmployee(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    emploeeID=input("Enter the employee id of the employee you want to remove : ")
    cur.execute("Delete from employee Where employeeId=%s",(emploeeID,))
    print("Employee successfully removed")
    self.db_connector.connection.commit()
    self.db_connector.close_connection()



1 usage
def GetuniqueEmployeeID(self):
    return len(self.get_allemployee())+1

1 usage
def get_allemployee(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    cur.execute("select FirstName from employee")
    record=cur.fetchall()
    return record
```

• PayrollService (implements IPayrollService):

• Methods:

GeneratePayroll,

```
from Dao.IPayrollService import IPayrollService
from Exception.Exceptions import PayrollGenerationException
4 usages
class payrollservice(IPayrollService):
    def __init__(self,db_connector):
        self.db_connector= db_connector


    1 usage
    def GeneratePayroll(self):
        self.db_connector.open_connection()
        cur=self.db_connector.connection.cursor()
        employeeId=input("Enter the employeeID for which you want to generate payroll : ")
        payrollid=self.getuniquepayrollid()
        basesalary=int(input("Enter the base salary of the employee"))
        Payperiodstartingdate=input("Enter the pay period starting date in the format 'YYYY-MM-DD' : ")
        Payperiodendingdate = input("Enter the pay period ending date in the format 'YYYY-MM-DD' : ")
        OvertimePay=int(input("Enter the overtime pay of the employee"))
        Deductions=int(input("Enter the deduction for the employee : "))
        NetSalary=basesalary+OvertimePay-Deductions
        query="Insert into payroll values(%s,%s,%s,%s,%s,%s,%s,%s)"
        values=(payrollid,employeeId,Payperiodstartingdate,Payperiodendingdate,basesalary,OvertimePay,Deductions,NetSalary,)
        cur.execute(query,values)
        self.db_connector.connection.commit()
        self.db_connector.close_connection()
```

GetPayrollById,

```
1 usage
def GetPayrollById(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    payrollID=input("Enter the payrollId for details : ")
    cur.execute("Select * from payroll where PayrollId=%s",(payrollID,))
    record=cur.fetchone()
    if record:
        print("Here are the details : ")
        print(f"Payroll If : {record[0]}")
        print(f"Employee Id : {record[1]}")
        print(f"Pay period starting date : {record[2]}")
        print(f"Pay period ending date : {record[3]}")
        print(f"Base Salary : {record[4]}")
        print(f"Overt time pay : {record[5]}")
        print(f"deductions : {record[6]}")
        print(f"Net salary : {record[7]}")
    else:
        raise PayrollGenerationException()
    self.db_connector.close_connection()
```

GetPayrollsForEmployee,

```
1 usage
def GetPayrollsForEmployee(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    employeeID=input("Enter the employee Id for which you want Payrolls : ")
    cur.execute("Select * from payroll where EmployeeId=%s", (employeeID,))
    record = cur.fetchall()
    if record:
        print("Here are the details : ")
        for i in record:
            print(f"Payroll Id : {i[0]}")
            print(f"Employee Id : {i[1]}")
            print(f"Pay period starting date : {i[2]}")
            print(f"Pay period ending date : {i[3]}")
            print(f"Base Salary : {i[4]}")
            print(f"Overt time pay : {i[5]}")
            print(f"deductions : {i[6]}")
            print(f"Net salary : {i[7]}")
            print(" ")
    else:
        raise PayrollGenerationException()
    self.db_connector.close_connection()
```

GetPayrollsForPeriod

```python
def GetPayrollsForPeriod(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    startingdate=input("Enter the Starting date in the format 'YYYY-MM-DD' : ")
    endingdate=input("Enter the ending date in the format 'YYYY-MM-DD' : ")
    query="select * from payroll where Payperiodstartingdate >= %s and Payperiodendingdate <= %s"
    values=(startingdate,endingdate,)
    cur.execute(query,values)
    record=cur.fetchall()
    if record:
        print("Here are the payroll for the given period : ")
        for i in record:
            print(f"Payroll Id : {i[0]}")
            print(f"Employee Id : {i[1]}")
            print(f"Pay period starting date : {i[2]}")
            print(f"Pay period ending date : {i[3]}")
            print(f"Base Salary : {i[4]}")
            print(f"Overt time pay : {i[5]}")
            print(f"deductions : {i[6]}")
            print(f"Net salary : {i[7]}")
            print(" ")
    else:
        raise PayrollGenerationException()
    self.db_connector.close_connection()


1 usage
def getuniquepayrollid(self):
    return len(self.geallpayroll())+1

1 usage
def geallpayroll(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    cur.execute("select * from payroll")
    record=cur.fetchall()
    self.db_connector.close_connection()
    return record
```

• TaxService (implements ITaxService):

• Methods:

CalculateTax,

```
from Dao.ITaxService import ITaxService
from Exception.Exceptions import TaxCalculationException
from Exception.Exceptions import InvalidInputException
2 usages
class taxservice(ITaxService):
    def __init__(self,db_connector):
        self.db_connector=db_connector

    1 usage
    def CalculateTax(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        cur.execute("Select TaxableIncome from tax")
        record=cur.fetchall()
        if record:
            for i in record:
                if i[0] <= 50000:
                    tax=i[0]*0.05
                    cur.execute("Update tax set TaxAmount=%s Where TaxableIncome=%s",(tax,i[0]))

                elif i[0]>50000:
                    tax = i[0] * 0.1
                    cur.execute("Update tax set TaxAmount=%s Where TaxableIncome=%s", (tax, i[0]))
        else:
            raise TaxCalculationException()

        self.db_connector.connection.commit()
        self.db_connector.close_connection()
        self.GetTaxesForEmployee()
```

GetTaxById,

```
1 usage
def GettaxbyID(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    taxID = input("Enter the tax ID for details : ")
    cur.execute("Select * from tax where taxid=%s", (taxID,))
    record = cur.fetchone()
    if record:
        print("Here are the details : ")
        print(f"Tax ID : {record[0]}")
        print(f"Employee Id : {record[1]}")
        print(f"Tax Year : {record[2]}")
        print(f"Taxable Income : {record[3]}")
        print(f"Tax Amount : {record[4]}")
    else:
        raise InvalidInputException
    self.db_connector.close_connection()
```

GetTaxesForEmployee,

```
2 usages
def GetTaxesForEmployee(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    employeeID = input("Enter the employee Id for which you want tax : ")
    cur.execute("Select * from tax where EmployeeId=%s", (employeeID,))
    record = cur.fetchall()
    if record:
        print("Here are the details : ")
        for i in record:
            print(f"Tax ID : {i[0]}")
            print(f"Employee Id : {i[1]}")
            print(f"Tax Year : {i[2]}")
            print(f"Taxable Income : {i[3]}")
            print(f"Tax Amount : {i[4]}")
            print(" ")
    else:
        raise InvalidInputException
    self.db_connector.close_connection()
```

GetTaxesForYear

```
1 usage
def GetTaxesForYear(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    Year = input("Enter the Year for which you want taxes : ")
    cur.execute("Select * from tax where TaxYear =%s", (Year,))
    record = cur.fetchall()
    if record:
        print("Here are the details : ")
        for i in record:
            print("Here are the details : ")
            print(f"Tax ID : {i[0]}")
            print(f"Employee Id : {i[1]}")
            print(f"Tax Year : {i[2]}")
            print(f"Taxable Income : {i[3]}")
            print(f"Tax Amount : {i[4]}")
            print(" ")
    else:
        print("No taxes Found")
    self.db_connector.close_connection()
```

• FinancialRecordService (implements IFinancialRecordService):

• Methods:

AddFinancialRecord,

```python
from Dao.IFinancialRecordService import IFinancialRecordService
from Exception.Exceptions import FinancialRecordException
2 usages
class FinancialRecordService(IFinancialRecordService):
    def __init__(self,db_connector):
        self.db_connector=db_connector


    1 usage
    def AddFinancialRecord(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        recordID=self.GetuniquerecordID()
        employeeId=input("Enter the Employee Id : ")
        RecordDate=input("Enter the record date in the format 'YYYY-MM-DD' : ")
        description=input("Enter the description : ")
        amount=input("Enter the amount : ")
        recordType=input("Enter the Record type from three options ('Income','Expense','Tax Payment') : ")
        query="Insert into financialrecord values(%s,%s,%s,%s,%s,%s)"
        values=(recordID,employeeId,RecordDate,description,amount,recordType,)
        cur.execute(query,values)
        print("Successfully Added")
        self.db_connector.connection.commit()
        self.db_connector.close_connection()
```

GetFinancialRecordById,

```python
1 usage
def GetFinancialRecordById(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    recordId=input("Enter the record Id : ")
    cur.execute("Select * from financialrecord where recordid=%s",(recordId,))
    record=cur.fetchone()
    if record:
        print("Here are the details : ")
        print(f"Record ID : {record[0]}")
        print(f"employee ID : {record[1]}")
        print(f"Record Date : {record[2]}")
        print(f"description : {record[3]}")
        print(f"Amount : {record[4]}")
        print(f"Record Type : {record[5]}")
    else:
        raise FinancialRecordException
    self.db_connector.close_connection()
```

GetFinancialRecordsForEmployee,

```
1 usage
def GetFinancialRecordsForEmployee(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    employeeID = input("Enter the employee Id for which you want record : ")
    cur.execute("Select * from financialrecord where EmployeeId=%s", (employeeID,))
    record = cur.fetchall()
    if record:
        print("Here are the details : ")
        for i in record:
            print(f"Record ID : {i[0]}")
            print(f"Employee Id : {i[1]}")
            print(f"Record Date : {i[2]}")
            print(f"Description : {i[3]}")
            print(f"Amount : {i[4]}")
            print(f"Record Type : {i[5]}")
            print(" ")
    else:
        raise FinancialRecordException
    self.db_connector.close_connection()
```

GetFinancialRecordsForDate

```
1 usage
def GetFinancialRecordsForDate(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    recorddate = input("Enter the record date in the format 'YYYY-MM-DD' : ")
    cur.execute("Select * from financialrecord where recorddate=%s", (recorddate,))
    record = cur.fetchall()
    if record:
        print("Here are the details : ")
        for i in record:
            print(f"Record ID : {i[0]}")
            print(f"Employee Id : {i[1]}")
            print(f"Record Date : {i[2]}")
            print(f"Description : {i[3]}")
            print(f"Amount : {i[4]}")
            print(f"Record Type : {i[5]}")
            print(" ")
    else:
        raise FinancialRecordException
    self.db_connector.close_connection()

1 usage
def GetuniquerecordID(self):
    return len(self.get_allrecord()+1)

1 usage
def get_allrecord(self):
    self.db_connector.open_connection()
    cur = self.db_connector.connection.cursor()
    cur.execute("select recordId from financialrecord")
    record=cur.fetchall()
    return record
```

Abstract Class :

IEmployeeService:

```python
from abc import ABC, abstractmethod

2 usages
class IEmployeeService(ABC):
    @abstractmethod
    def GetEmployeeById(self):
        pass

    @abstractmethod
    def GetAllEmployees(self):
        pass

    @abstractmethod
    def AddEmployee(self):
        pass

    @abstractmethod
    def UpdateEmployee(self):
        pass

    @abstractmethod
    def RemoveEmployee(self):
        pass
```

IPayrollService:

```python
from abc import ABC, abstractmethod

2 usages
class IPayrollService(ABC):
    @abstractmethod
    def GeneratePayroll(self):
        pass

    @abstractmethod
    def GetPayrollById(self):
        pass

    @abstractmethod
    def GetPayrollsForEmployee(self):
        pass

    @abstractmethod
    def GetPayrollsForPeriod(self):
        pass

    @abstractmethod
    def getuniquepayrollid(self):
        pass
```

ITaxService:

```python
from abc import ABC, abstractmethod

2 usages
class ITaxService(ABC):
    @abstractmethod
    def CalculateTax(self):
        pass

    @abstractmethod
    def GettaxbyID(self):
        pass

    @abstractmethod
    def GetTaxesForEmployee(self):
        pass

    @abstractmethod
    def GetTaxesForYear(self):
        pass
```

IFinancialRecordService:

```python
from abc import ABC, abstractmethod

2 usages
class IFinancialRecordService(ABC):
    @abstractmethod
    def AddFinancialRecord(self):
        pass

    @abstractmethod
    def GetFinancialRecordById(self):
        pass

    @abstractmethod
    def GetFinancialRecordsForEmployee(self):
        pass

    @abstractmethod
    def GetFinancialRecordsForDate(self):
        pass
```

Exceptions :

```
3 usages
class EmployeeNotFoundException(Exception):

    def __init__(self, employee_id):
        self.employee_id = employee_id
        self.message = f"Employee with ID {employee_id} not found"
        super().__init__(self.message)

4 usages
class PayrollGenerationException(Exception):

    def __init__(self, message="Error in payroll generation"):
        self.message = message
        super().__init__(self.message)

2 usages
class TaxCalculationException(Exception):

    def __init__(self, message="Error in tax calculation"):
        self.message = message
        super().__init__(self.message)

4 usages
class FinancialRecordException(Exception):

    def __init__(self, message="Financial record management error"):
        self.message = message
        super().__init__(self.message)

3 usages
class InvalidInputException(Exception):

    def __init__(self, message="Invalid input data"):
        self.message = message
        super().__init__(self.message)
```

```
class DatabaseConnectionException(Exception):

    def __init__(self, message="Database connection error"):
        self.message = message
        super().__init__(self.message)
```

Database Connectivity :

```python
import mysql.connector
2 usages
class DBConnector:
    def __init__(self, host, user, password, port, database):
        self.host = host
        self.user = user
        self.password = password
        self.port = port
        self.database = database
        self.connection = None

    21 usages (21 dynamic)
    def open_connection(self):
        if self.connection is None or not self.connection.is_connected():
            self.connection = mysql.connector.connect(
                host=self.host,
                user=self.user,
                password=self.password,
                port=self.port,
                database=self.database
            )

    18 usages (18 dynamic)
    def close_connection(self):
        if self.connection is not None and self.connection.is_connected():
            self.connection.close()
```

Main() :

```python
from Entity.Employee import Employee
from util.DB_connection import DBConnector
from Entity.EmployeeService import employeeservice
from Entity.PayrollService import payrollservice
from Entity.TaxService import taxservice
from Entity.FinancialRecordService import FinancialRecordService

1 usage
class Payxpert:
    5 usages
    def main(self):
        db_connector = DBConnector( host: "localhost",  user: "root",  password: "root",  port: 3306,  database: "payxpert")
        Emp = Employee(db_connector=db_connector)
        ES = employeeservice(db_connector)
        PS = payrollservice(db_connector)
        TS = taxservice(db_connector)
        FS = FinancialRecordService(db_connector)

        print("Hi folks !! This is Pay Xpert")
        print("Choose from the following Options : ")
        print("1. If you need help with the employee database ")
        print("2. If you need help with the Payroll database ")
        print("3. If you need help with the Tax database ")
        print("4. If you need help with the Financial Record database ")
        choice=input("Now Enter Your Choice : ")
        if choice=='1':
            print("Employee Database Loading...")
            print("Now, Choose from the following options ")
            print("1. If you want to know the age of employee")
            print("2. If you want employee Details")
            print("3. All the Employees working here")
            print("4. If you want to Add a new Employee")
            print("5. If you want to update database")
            print("6. If you want to delete the employee details from the database")
            print("7. Exit")
            emp=input("Enter the choice : ")
            if emp=='1':
                Emp.CalculateAge()
            elif emp=='2':
                ES.GetEmployeeById()
            elif emp=='3':
```

```
        elif emp=='3':
            ES.GetAllEmployees()
        elif emp=='4':
            ES.AddEmployee()
        elif emp=='5':
            ES.UpdateEmployee()
        elif emp=='6':
            ES.RemoveEmployee()
        elif emp=='7':
            self.main()
        else:
            print("Invalid Choice")
    elif choice=='2':
        print("Payroll Databasse Loading ...")
        print("Now, Choose from the following options : ")
        print("1. If you want to generate payroll")
        print("2. IF you want to Payroll Details")
        print("3. If you want to know all Payrolls for an employee")
        print("4. If you want all payrolls for a specific period")
        print("5. Exit")
        pay=input("Enter your choice : ")
        if pay=='1':
            PS.GeneratePayroll()
        elif pay=='2':
            PS.GetPayrollById()
        elif pay=='3':
            PS.GetPayrollsForEmployee()
        elif pay=='4':
            PS.GetPayrollsForPeriod()
        elif pay=='5':
            self.main()
        else:
            print("Invalid Choice")
    elif choice=='3':
        print("Tax Database Loading...")
        print("Now, Choose from the following options ")
        print("1. If you want to Calculate Tax")
        print("2. If you want to Know tax details")
        print("3. If you want to know all Taxes for an employee")
        print("4. If you want to know all taxes in a year ")
        print("5. Exit")
```

```
            if tax== '1':
                TS.CalculateTax()
            elif tax=='2':
                TS.GettaxbyID()
            elif tax=='3':
                TS.GetTaxesForEmployee()
            elif tax=='4':
                TS.GetTaxesForYear()
            elif tax=='5':
                self.main()
            else:
                print("Invalid Choice")
        elif choice=='4':
            print("Financial Record Loading...")
            print("Now, Choose from the following options ")
            print("1. IF you want to add a financial record")
            print("2. If you want to Financial Record details")
            print("3. If you want to know all finacial records for an employee")
            print("4. If you want to know all Financial Records for a specific date")
            print("5. Exit")
            fin=input("Enter Your Choice : ")
            if fin=='1':
                FS.AddFinancialRecord()
            elif fin=='2':
                FS.GetFinancialRecordById()
            elif fin=='3':
                FS.GetFinancialRecordsForEmployee()
            elif fin=='4':
                FS.GetFinancialRecordsForDate()
            elif fin=='5':
                self.main()
            else:
                print("Invalid Choice")
        else:
            print("Invalid Choice")



if __name__=='__main__':
    obj=Payxpert()
    obj.main()
```

Output :

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice :
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 1
Employee Database Loading...
Now, Choose from the following options
1. If you want to know the age of employee
2. If you want employee Details
3. All the Employees working here
4. If you want to Add a new Employee
5. If you want to update database
6. If you want to delete the employee details from the database
7. Exit
Enter the choice : 1
Enter your EmployeeID for which you want to know the age : 5
The age of Employee whose employee ID is 5 = 43

Process finished with exit code 0
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 1
Employee Database Loading...
Now, Choose from the following options
1. If you want to know the age of employee
2. If you want employee Details
3. All the Employees working here
4. If you want to Add a new Employee
5. If you want to update database
6. If you want to delete the employee details from the database
7. Exit
Enter the choice : 2
Please enter your employeeID : 5
Here are the details of th employee :
Employee ID = 5
First name = Daniel
Last Name = Miller
Date Of Birth = 1980-07-05
Gender = Male
Email = daniel.miller@example.com
Phone Number = 777-888-9999
Address = 202 Oak St, City, Country
Position = Project Manager
Joining Date = 2022-05-15
Termination Date = 2023-10-30

Process finished with exit code 0
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 1
Employee Database Loading...
Now, Choose from the following options
1. If you want to know the age of employee
2. If you want employee Details
3. All the Employees working here
4. If you want to Add a new Employee
5. If you want to update database
6. If you want to delete the employee details from the database
7. Exit
Enter the choice : 3
John
Jane
Michael
Emily
Daniel
Eva
Andrew
Olivia
Matthew
Sophia
William
Emma
Christopher
Isabella
Alexander

Process finished with exit code 0
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 1
Employee Database Loading...
Now, Choose from the following options
1. If you want to know the age of employee
2. If you want employee Details
3. All the Employees working here
4. If you want to Add a new Employee
5. If you want to update database
6. If you want to delete the employee details from the database
7. Exit
Enter the choice : 5
Enter the employeeID you want to update : 2
Choose what you want to update :
1.First name
2.Last Name
3.Date Of Birth
4.Gender
5.Email
6.Phone Number
7.Address
8.Position
9.Joining Date
10.Termination Date
Enter your choice : 2
Enter Last Name : Foster
Updated Successfully

Process finished with exit code 0
```

```
mysql> select*from employee;
+------------+-------------+----------+-------------+--------+-------------------------------+-------------+-------------------------------+--------------------+-------------+-----------------+
| EmployeeID | FirstName   | LastName | DateOfBirth | Gender | Email                         | PhoneNumber | Address                       | Position           | JoiningDate | TerminationDate |
+------------+-------------+----------+-------------+--------+-------------------------------+-------------+-------------------------------+--------------------+-------------+-----------------+
|          1 | John        | Doe      | 1990-01-01  | Male   | john.doe@example.com          | 123-456-7890 | 123 Main St, City, Country   | Software Developer | 2022-01-01  | NULL            |
|          2 | Jane        | Smith    | 1985-05-15  | Female | jane.smith@example.com        | 987-654-3210 | 456 Oak St, City, Country    | Software Developer | 2022-02-15  | 2023-05-20      |
|          3 | Michael     | Jakson   | 1988-09-20  | Male   | michael.johnson@example.com   | 555-123-4567 | 789 Pine St, City, Country   | Manager            | 2021-12-10  | NULL            |
|          4 | Emily       | Davis    | 1993-03-12  | Female | emily.davis@example.com       | 111-222-3333 | 101 Elm St, City, Country    | HR Coordinator     | 2023-03-01  | NULL            |
|          5 | Daniel      | Miller   | 1980-07-05  | Male   | daniel.miller@example.com     | 777-888-9999 | 202 Oak St, City, Country    | Project Manager    | 2022-05-15  | 2023-10-30      |
|          6 | Eva         | Anderson | 1992-04-18  | Female | eva.anderson@example.com      | 444-555-6666 | 303 Cedar St, City, Country  | Software Developer | 2022-07-01  | NULL            |
|          7 | Andrew      | Taylor   | 1987-11-25  | Male   | andrew.taylor@example.com     | 999-888-7777 | 505 Maple St, City, Country  | Software Developer | 2023-02-28  | NULL            |
|          8 | Olivia      | Clark    | 1995-08-08  | Female | olivia.clark@example.com      | 777-666-5555 | 707 Walnut St, City, Country | Manager            | 2022-04-15  | 2023-09-30      |
|          9 | Matthew     | Baker    | 1983-12-03  | Male   | matthew.baker@example.com     | 111-999-4444 | 909 Pine St, City, Country   | Manager            | 2022-09-01  | NULL            |
|         10 | Sophia      | Wright   | 1998-06-22  | Female | sophia.wright@example.com     | 333-222-1111 | 101 Walnut St, City, Country | Software Developer | 2023-01-10  | 2023-07-15      |
|         11 | William     | Harris   | 1991-02-15  | Male   | william.harris@example.com    | 666-777-8888 | 202 Cedar St, City, Country  | HR Coordinator     | 2022-05-20  | NULL            |
|         12 | Emma        | Martin   | 1989-09-30  | Female | emma.martin@example.com       | 444-333-2222 | 404 Oak St, City, Country    | HR Coordinator     | 2023-03-15  | NULL            |
|         13 | Christopher | Robinson | 1981-07-12  | Male   | christopher.robinson@example.com | 222-333-4444 | 606 Pine St, City, Country | Software Developer | 2022-08-10  | NULL            |
|         14 | Isabella    | Garcia   | 1994-05-29  | Female | isabella.garcia@example.com   | 888-999-1111 | 303 Elm St, City, Country    | Analyst            | 2023-04-01  | NULL            |
|         15 | Alexander   | Young    | 1986-03-08  | Male   | alexander.young@example.com   | 555-444-3333 | 808 Maple St, City, Country  | HR Coordinator     | 2022-10-15  | NULL            |
+------------+-------------+----------+-------------+--------+-------------------------------+-------------+-------------------------------+--------------------+-------------+-----------------+
15 rows in set (0.00 sec)

mysql> select*from employee;
+------------+-------------+----------+-------------+--------+-------------------------------+-------------+-------------------------------+--------------------+-------------+-----------------+
| EmployeeID | FirstName   | LastName | DateOfBirth | Gender | Email                         | PhoneNumber | Address                       | Position           | JoiningDate | TerminationDate |
+------------+-------------+----------+-------------+--------+-------------------------------+-------------+-------------------------------+--------------------+-------------+-----------------+
|          1 | John        | Doe      | 1990-01-01  | Male   | john.doe@example.com          | 123-456-7890 | 123 Main St, City, Country   | Software Developer | 2022-01-01  | NULL            |
|          2 | Jane        | Foster   | 1985-05-15  | Female | jane.smith@example.com        | 987-654-3210 | 456 Oak St, City, Country    | Software Developer | 2022-02-15  | 2023-05-20      |
|          3 | Michael     | Jakson   | 1988-09-20  | Male   | michael.johnson@example.com   | 555-123-4567 | 789 Pine St, City, Country   | Manager            | 2021-12-10  | NULL            |
|          4 | Emily       | Davis    | 1993-03-12  | Female | emily.davis@example.com       | 111-222-3333 | 101 Elm St, City, Country    | HR Coordinator     | 2023-03-01  | NULL            |
|          5 | Daniel      | Miller   | 1980-07-05  | Male   | daniel.miller@example.com     | 777-888-9999 | 202 Oak St, City, Country    | Project Manager    | 2022-05-15  | 2023-10-30      |
|          6 | Eva         | Anderson | 1992-04-18  | Female | eva.anderson@example.com      | 444-555-6666 | 303 Cedar St, City, Country  | Software Developer | 2022-07-01  | NULL            |
|          7 | Andrew      | Taylor   | 1987-11-25  | Male   | andrew.taylor@example.com     | 999-888-7777 | 505 Maple St, City, Country  | Software Developer | 2023-02-28  | NULL            |
|          8 | Olivia      | Clark    | 1995-08-08  | Female | olivia.clark@example.com      | 777-666-5555 | 707 Walnut St, City, Country | Manager            | 2022-04-15  | 2023-09-30      |
|          9 | Matthew     | Baker    | 1983-12-03  | Male   | matthew.baker@example.com     | 111-999-4444 | 909 Pine St, City, Country   | Manager            | 2022-09-01  | NULL            |
|         10 | Sophia      | Wright   | 1998-06-22  | Female | sophia.wright@example.com     | 333-222-1111 | 101 Walnut St, City, Country | Software Developer | 2023-01-10  | 2023-07-15      |
|         11 | William     | Harris   | 1991-02-15  | Male   | william.harris@example.com    | 666-777-8888 | 202 Cedar St, City, Country  | HR Coordinator     | 2022-05-20  | NULL            |
|         12 | Emma        | Martin   | 1989-09-30  | Female | emma.martin@example.com       | 444-333-2222 | 404 Oak St, City, Country    | HR Coordinator     | 2023-03-15  | NULL            |
|         13 | Christopher | Robinson | 1981-07-12  | Male   | christopher.robinson@example.com | 222-333-4444 | 606 Pine St, City, Country | Software Developer | 2022-08-10  | NULL            |
|         14 | Isabella    | Garcia   | 1994-05-29  | Female | isabella.garcia@example.com   | 888-999-1111 | 303 Elm St, City, Country    | Analyst            | 2023-04-01  | NULL            |
|         15 | Alexander   | Young    | 1986-03-08  | Male   | alexander.young@example.com   | 555-444-3333 | 808 Maple St, City, Country  | HR Coordinator     | 2022-10-15  | NULL            |
+------------+-------------+----------+-------------+--------+-------------------------------+-------------+-------------------------------+--------------------+-------------+-----------------+
15 rows in set (0.00 sec)
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 1
Employee Database Loading...
Now, Choose from the following options
1. If you want to know the age of employee
2. If you want employee Details
3. All the Employees working here
4. If you want to Add a new Employee
5. If you want to update database
6. If you want to delete the employee details from the database
7. Exit
Enter the choice : 4
Enter the details of employee you want to add
Enter Frist Name : Manas
Enter Last Name : Rustagi
Enter Date of birth in the format 'YYYY-MM-DD' : 2002-06-16
Enter Gender (Male,Female) : Male
Enter the email address : manasrustagi16@gmail.com
Enter Phone Number : 7982681438
Enter Address : karol bagh
Enter Position : PGET
Enter Joining Date in the format 'YYYY-MM-DD' :2024-02-01
Employee Added Successfully

Process finished with exit code 0
```

```
mysql> select*from employee;
+------------+-----------+----------+-------------+--------+-----------------------------------+--------------+-----------------------------+--------------------+-------------+-----------------+
| EmployeeID | FirstName | LastName | DateOfBirth | Gender | Email                             | PhoneNumber  | Address                     | Position           | JoiningDate | TerminationDate |
+------------+-----------+----------+-------------+--------+-----------------------------------+--------------+-----------------------------+--------------------+-------------+-----------------+
|          1 | John      | Doe      | 1990-01-01  | Male   | john.doe@example.com              | 123-456-7890 | 123 Main St, City, Country  | Software Developer | 2022-01-01  | NULL            |
|          2 | Jane      | Foster   | 1985-05-15  | Female | jane.smith@example.com            | 987-654-3210 | 456 Oak St, City, Country   | Software Developer | 2022-02-15  | 2023-05-20      |
|          3 | Michael   | Jakson   | 1988-09-20  | Male   | michael.johnson@example.com       | 555-123-4567 | 789 Pine St, City, Country  | Manager            | 2021-12-10  | NULL            |
|          4 | Emily     | Davis    | 1993-03-12  | Female | emily.davis@example.com           | 111-222-3333 | 101 Elm St, City, Country   | HR Coordinator     | 2023-03-01  | NULL            |
|          5 | Daniel    | Miller   | 1980-07-05  | Male   | daniel.miller@example.com         | 777-888-9999 | 202 Oak St, City, Country   | Project Manager    | 2022-05-15  | 2023-10-30      |
|          6 | Eva       | Anderson | 1992-04-18  | Female | eva.anderson@example.com          | 444-555-6666 | 303 Cedar St, City, Country | Software Developer | 2022-07-01  | NULL            |
|          7 | Andrew    | Taylor   | 1987-11-25  | Male   | andrew.taylor@example.com         | 999-888-7777 | 505 Maple St, City, Country | Software Developer | 2023-02-28  | NULL            |
|          8 | Olivia    | Clark    | 1995-08-08  | Female | olivia.clark@example.com          | 777-666-5555 | 707 Walnut St, City, Country | Manager           | 2022-04-15  | 2023-09-30      |
|          9 | Matthew   | Baker    | 1983-12-03  | Male   | matthew.baker@example.com         | 111-999-4444 | 909 Pine St, City, Country  | Manager            | 2022-09-01  | NULL            |
|         10 | Sophia    | Wright   | 1998-06-22  | Female | sophia.wright@example.com         | 333-222-1111 | 101 Walnut St, City, Country | Software Developer | 2023-01-10  | 2023-07-15      |
|         11 | William   | Harris   | 1991-02-15  | Male   | william.harris@example.com        | 666-777-8888 | 202 Cedar St, City, Country | HR Coordinatorr    | 2022-05-20  | NULL            |
|         12 | Emma      | Martin   | 1989-09-30  | Female | emma.martin@example.com           | 444-333-2222 | 404 Oak St, City, Country   | HR Coordinator     | 2023-03-15  | NULL            |
|         13 | Christopher | Robinson | 1981-07-12 | Male   | christopher.robinson@example.com  | 222-333-4444 | 606 Pine St, City, Country  | Software Developer | 2022-08-10  | NULL            |
|         14 | Isabella  | Garcia   | 1994-05-29  | Female | isabella.garcia@example.com       | 888-999-1111 | 303 Elm St, City, Country   | Analyst            | 2023-04-01  | NULL            |
|         15 | Alexander | Young    | 1986-03-08  | Male   | alexander.young@example.com       | 555-444-3333 | 808 Maple St, City, Country | HR Coordinator     | 2022-10-15  | NULL            |
+------------+-----------+----------+-------------+--------+-----------------------------------+--------------+-----------------------------+--------------------+-------------+-----------------+
15 rows in set (0.00 sec)

mysql> select*from employee;
+------------+-----------+----------+-------------+--------+-----------------------------------+--------------+-----------------------------+--------------------+-------------+-----------------+
| EmployeeID | FirstName | LastName | DateOfBirth | Gender | Email                             | PhoneNumber  | Address                     | Position           | JoiningDate | TerminationDate |
+------------+-----------+----------+-------------+--------+-----------------------------------+--------------+-----------------------------+--------------------+-------------+-----------------+
|          1 | John      | Doe      | 1990-01-01  | Male   | john.doe@example.com              | 123-456-7890 | 123 Main St, City, Country  | Software Developer | 2022-01-01  | NULL            |
|          2 | Jane      | Foster   | 1985-05-15  | Female | jane.smith@example.com            | 987-654-3210 | 456 Oak St, City, Country   | Software Developer | 2022-02-15  | 2023-05-20      |
|          3 | Michael   | Jakson   | 1988-09-20  | Male   | michael.johnson@example.com       | 555-123-4567 | 789 Pine St, City, Country  | Manager            | 2021-12-10  | NULL            |
|          4 | Emily     | Davis    | 1993-03-12  | Female | emily.davis@example.com           | 111-222-3333 | 101 Elm St, City, Country   | HR Coordinator     | 2023-03-01  | NULL            |
|          5 | Daniel    | Miller   | 1980-07-05  | Male   | daniel.miller@example.com         | 777-888-9999 | 202 Oak St, City, Country   | Project Manager    | 2022-05-15  | 2023-10-30      |
|          6 | Eva       | Anderson | 1992-04-18  | Female | eva.anderson@example.com          | 444-555-6666 | 303 Cedar St, City, Country | Software Developer | 2022-07-01  | NULL            |
|          7 | Andrew    | Taylor   | 1987-11-25  | Male   | andrew.taylor@example.com         | 999-888-7777 | 505 Maple St, City, Country | Software Developer | 2023-02-28  | NULL            |
|          8 | Olivia    | Clark    | 1995-08-08  | Female | olivia.clark@example.com          | 777-666-5555 | 707 Walnut St, City, Country | Manager           | 2022-04-15  | 2023-09-30      |
|          9 | Matthew   | Baker    | 1983-12-03  | Male   | matthew.baker@example.com         | 111-999-4444 | 909 Pine St, City, Country  | Manager            | 2022-09-01  | NULL            |
|         10 | Sophia    | Wright   | 1998-06-22  | Female | sophia.wright@example.com         | 333-222-1111 | 101 Walnut St, City, Country | Software Developer | 2023-01-10  | 2023-07-15      |
|         11 | William   | Harris   | 1991-02-15  | Male   | william.harris@example.com        | 666-777-8888 | 202 Cedar St, City, Country | HR Coordinatorr    | 2022-05-20  | NULL            |
|         12 | Emma      | Martin   | 1989-09-30  | Female | emma.martin@example.com           | 444-333-2222 | 404 Oak St, City, Country   | HR Coordinator     | 2023-03-15  | NULL            |
|         13 | Christopher | Robinson | 1981-07-12 | Male   | christopher.robinson@example.com  | 222-333-4444 | 606 Pine St, City, Country  | Software Developer | 2022-08-10  | NULL            |
|         14 | Isabella  | Garcia   | 1994-05-29  | Female | isabella.garcia@example.com       | 888-999-1111 | 303 Elm St, City, Country   | Analyst            | 2023-04-01  | NULL            |
|         15 | Alexander | Young    | 1986-03-08  | Male   | alexander.young@example.com       | 555-444-3333 | 808 Maple St, City, Country | HR Coordinator     | 2022-10-15  | NULL            |
|         16 | Manas     | Rustagi  | 2002-06-16  | Male   | manasrustagi16@gmail.com          | 7982681438   | karol bagh                  | PGET               | 2024-02-01  | NULL            |
+------------+-----------+----------+-------------+--------+-----------------------------------+--------------+-----------------------------+--------------------+-------------+-----------------+
16 rows in set (0.00 sec)
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 2
Payroll Databasse Loading ...
Now, Choose from the following options :
1. If you want to generate payroll
2. IF you want to Payroll Details
3. If you want to know all Payrolls for an employee
4. If you want all payrolls for a specific period
5. Exit
Enter your choice : 3
Enter the employee Id for which you want Payrolls : 5
Here are the details :
Payroll Id : 5
Employee Id : 5
Pay period starting date : 2022-06-01
Pay period ending date : 2022-06-15
Base Salary : 70000
Overt time pay : 300
deductions : 500
Net salary : 69800


Process finished with exit code 0
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 2
Payroll Databasse Loading ...
Now, Choose from the following options :
1. If you want to generate payroll
2. IF you want to Payroll Details
3. If you want to know all Payrolls for an employee
4. If you want all payrolls for a specific period
5. Exit
Enter your choice : 4
Enter the Starting date in the format 'YYYY-MM-DD' : 2022-01-01
Enter the ending date in the format 'YYYY-MM-DD' : 2022-05-01
Here are the payroll for the given period :
Payroll Id : 1
Employee Id : 1
Pay period starting date : 2022-01-01
Pay period ending date : 2022-01-15
Base Salary : 50000
Overt time pay : 200
deductions : 300
Net salary : 49900

Payroll Id : 2
Employee Id : 2
Pay period starting date : 2022-02-01
Pay period ending date : 2022-02-15
Base Salary : 55000
Overt time pay : 150
deductions : 250
Net salary : 54900
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 2
Payroll Databasse Loading ...
Now, Choose from the following options :
1. If you want to generate payroll
2. IF you want to Payroll Details
3. If you want to know all Payrolls for an employee
4. If you want all payrolls for a specific period
5. Exit
Enter your choice : 5
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 3
Tax Database Loading...
Now, Choose from the following options
1. If you want to Calculate Tax
2. If you want to Know tax details
3. If you want to know all Taxes for an employee
4. If you want to know all taxes in a year
5. Exit
Enter your Choice : 1
Enter the employee Id for which you want tax : 5
Here are the details :
Tax ID : 5
Employee Id : 5
Tax Year : 2022
Taxable Income : 69800
Tax Amount : 6980
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 3
Tax Database Loading...
Now, Choose from the following options
1. If you want to Calculate Tax
2. If you want to Know tax details
3. If you want to know all Taxes for an employee
4. If you want to know all taxes in a year
5. Exit
Enter your Choice : 4
Enter the Year for which you want taxes : 2021
No taxes Found

Process finished with exit code 0
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 4
Financial Record Loading...
Now, Choose from the following options
1. IF you want to add a financial record
2. If you want to Financial Record details
3. If you want to know all finacial records for an employee
4. If you want to know all Financial Records for a specific date
5. Exit
Enter Your Choice : 1
Enter the Employee Id : 5
Enter the record date in the format 'YYYY-MM-DD' : 2024-02-01
Enter the description : phone
Enter the amount : 5000
Enter the Record type from three options ('Income','Expense','Tax Payment') : Expense
Successfully Added

Process finished with exit code 0
```

```
mysql> select*from financialrecord;
+----------+------------+------------+-------------+--------+------------+
| RecordID | EmployeeID | RecordDate | Description | Amount | RecordType |
+----------+------------+------------+-------------+--------+------------+
|        1 |          1 | 2022-01-05 | Income      |   2000 | Income     |
|        2 |          2 | 2022-02-10 | Expense     |    500 | Expense    |
|        3 |          3 | 2022-02-20 | Income      |   2500 | Income     |
|        4 |          4 | 2023-03-10 | Expense     |    300 | Expense    |
|        5 |          5 | 2022-06-05 | Income      |   3500 | Income     |
|        6 |          6 | 2022-08-05 | Income      |   2500 | Income     |
|        7 |          7 | 2023-03-20 | Expense     |    700 | Expense    |
|        8 |          8 | 2022-05-10 | Income      |   3000 | Income     |
|        9 |          9 | 2022-10-20 | Expense     |    800 | Expense    |
|       10 |         10 | 2023-02-05 | Income      |   2000 | Income     |
|       11 |         11 | 2022-06-20 | Expense     |    600 | Expense    |
|       12 |         12 | 2023-04-20 | Income      |   3500 | Income     |
|       13 |         13 | 2022-09-10 | Expense     |    900 | Expense    |
|       14 |         14 | 2023-05-05 | Income      |   4000 | Income     |
|       15 |         15 | 2022-11-05 | Expense     |    700 | Expense    |
|       16 |          5 | 2024-02-01 | phone       |   5000 | Expense    |
+----------+------------+------------+-------------+--------+------------+
16 rows in set (0.00 sec)
```

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 4
Financial Record Loading...
Now, Choose from the following options
1. IF you want to add a financial record
2. If you want to Financial Record details
3. If you want to know all finacial records for an employee
4. If you want to know all Financial Records for a specific date
5. Exit
Enter Your Choice : 3
Enter the employee Id for which you want record : 5
Here are the details :
Record ID : 5
Employee Id : 5
Record Date : 2022-06-05
Description : Income
Amount : 3500
Record Type : Income

Record ID : 16
Employee Id : 5
Record Date : 2024-02-01
Description : phone
Amount : 5000
Record Type : Expense


Process finished with exit code 0
```

Exception

```
"E:\Python\Case Study PayXpert\.venv\Scripts\python.exe" "E:\Python\Case Study PayXpert\main.py"
Hi folks !! This is Pay Xpert
Choose from the following Options :
1. If you need help with the employee database
2. If you need help with the Payroll database
3. If you need help with the Tax database
4. If you need help with the Financial Record database
Now Enter Your Choice : 4
Financial Record Loading...
Now, Choose from the following options
1. IF you want to add a financial record
2. If you want to Financial Record details
3. If you want to know all finacial records for an employee
4. If you want to know all Financial Records for a specific date
5. Exit
Enter Your Choice : 4
Enter the record date in the format 'YYYY-MM-DD' : 2024-05-01
Traceback (most recent call last):
  File "E:\Python\Case Study PayXpert\main.py", line 121, in <module>
    obj.main()
  File "E:\Python\Case Study PayXpert\main.py", line 109, in main
    FS.GetFinancialRecordsForDate()
  File "E:\Python\Case Study PayXpert\Entity\FinancialRecordService.py", line 79, in GetFinancialRecordsForDate
    raise FinancialRecordException
Exception.Exceptions.FinancialRecordException: Financial record management error

Process finished with exit code 1
```

main.py | Exceptions.py | Test_Unittest.py | FinancialRecordService.py | PayrollService.py | EmployeeService.py | DB_connection.py

```python
import unittest

from util.DB_connection import DBConnector
from Entity.EmployeeService import employeeservice
from Entity.PayrollService import payrollservice
db_connector = DBConnector( host: "localhost", user: "root", password: "root", port: 3306, database: "payxpert")

esl = employeeservice(db_connector)
psl = payrollservice(db_connector)



class PayrollSystemTestCase(unittest.TestCase):
    db_connector = DBConnector( host: "localhost", user: "root", password: "root", port: 3306, database: "payxpert")
    def test_calculate_net_salary_after_deductions(self):
        self.db_connector.open_connection()
        cur = self.db_connector.connection.cursor()
        query = "SELECT NetSalary, Overtime, Deductions FROM payroll"
        netCheck = []
        cur.execute(query)
        record=cur.fetchall()
        for NetSalary, Overtime, ded in record:
            netCheck.append(int(NetSalary + Overtime - ded))
        expected_net_salary = netCheck
        actual_net_salary = psl.CalculateSalary(record[0],record[1],record[2])
        self.assertEqual(actual_net_salary, expected_net_salary)
```

```python
        cur.execute(query)
        record=cur.fetchall()
        for NetSalary, Overtime, ded in record:
            netCheck.append(int(NetSalary + Overtime - ded))
        expected_net_salary = netCheck
        actual_net_salary = psl.CalculateSalary(record[0],record[1],record[2])
        self.assertEqual(actual_net_salary, expected_net_salary)

    def test_verify_error_handling_for_invalid_employee_data(self):
        employee = {
            'empID': 1,
            'firstName': 'John',
            'lastName': 'Doe',
            'dob': '1980-05-15',
            'gender': 'Male',
            'email': 'fwadw@gmail.com',
            'phone': '12',
            'address': '123 Main St, London',
            'position': 'Manager',
            'joiningDate': '2020-01-01'
        }
        validation_result = esl.checkEmployeeDataValidation(employee)
        self.assertTrue(validation_result, msg: "Invalid Data Found.")
```