**PROLOG:**

% -----------------------------

% Dynamic predicates

:- dynamic tax\_slab/3.

:- dynamic income/1.

:- dynamic age/1.

:- dynamic regime/1.

:- dynamic deduction/2.

% -----------------------------

% Tax slabs FY 2024–25

% Old Regime

tax\_slab(old, 250000, 0).

tax\_slab(old, 500000, 0.05).

tax\_slab(old, 1000000, 0.2).

tax\_slab(old, 5000000, 0.3).

tax\_slab(old, 99999999, 0.3). % surcharge applies separately

% New Regime

tax\_slab(new, 300000, 0).

tax\_slab(new, 700000, 0.05).

tax\_slab(new, 1000000, 0.1).

tax\_slab(new, 1200000, 0.15).

tax\_slab(new, 1500000, 0.2).

tax\_slab(new, 5000000, 0.3).

tax\_slab(new, 99999999, 0.3). % surcharge applies separately

% -----------------------------

% Surcharge slabs

% Format: surcharge(Regime, IncomeThreshold, Rate)

surcharge(\_, 50000000, 0.1). % ₹50L–₹1Cr

surcharge(\_, 100000000, 0.15). % ₹1Cr–₹2Cr

surcharge(old, 200000000, 0.25). % ₹2Cr–₹5Cr (only old)

surcharge(old, 999999999, 0.37). % ₹5Cr+ (only old)

surcharge(new, 999999999, 0.15). % capped for new

% -----------------------------

% Max deduction limits (all section names quoted and upper-case to match Python)

max\_deduction('80C', 150000).

max\_deduction('80D', Max) :- age(A), (A >= 60 -> Max = 50000 ; Max = 25000).

max\_deduction('80CCD(1B)', 50000).

max\_deduction('EPF', 150000).

max\_deduction('Life Insurance', 150000).

max\_deduction('Standard', 50000).

max\_deduction('NPS', 50000). % optional alias

% -----------------------------

% Utility Rules

print\_max\_deduction\_limits :-

writeln("🔢 Maximum Deduction Limits (FY 2024–25):"),

age\_based\_80d\_limit(L),

format(" - Medical Insurance (80D): ₹~w~n", [L]),

format(" - Life Insurance (80C): ₹150000~n"),

format(" - EPF (80C): ₹150000~n"),

format(" - NPS (80CCD(1B)): ₹50000~n"),

format(" - Standard Deduction: ₹50000 (auto applied)~n~n").

age\_based\_80d\_limit(Limit) :-

age(A),

( A >= 60 -> Limit = 50000 ; Limit = 25000 ).

total\_deductions(Sum) :-

findall(A, deduction(\_, A), L),

sum\_list(L, Partial),

max\_deduction('Standard', Std),

Sum is Partial + Std.

age\_based\_exemption(E) :-

age(A),

(A >= 60 -> E is 300000 ; E is 250000).

get\_sorted\_slabs(Regime, Sorted) :-

findall((Limit, Rate), tax\_slab(Regime, Limit, Rate), L),

sort(L, Sorted).

% -----------------------------

% Taxable Income

taxable\_income(old, TI) :-

income(I),

total\_deductions(D),

age\_based\_exemption(E),

Raw is I - D - E,

TI is max(0, Raw).

taxable\_income(new, TI) :-

income(I),

TI is I - 50000. % Standard deduction only

% -----------------------------

% Progressive Tax Computation

progressive\_tax(\_, \_, [], \_, 0).

progressive\_tax(Income, PrevLimit, [(Limit, Rate)|Rest], Acc, Tax) :-

Income =< Limit,

Portion is Income - PrevLimit,

Temp is Portion \* Rate,

Tax is Acc + Temp.

progressive\_tax(Income, PrevLimit, [(Limit, Rate)|Rest], Acc, Tax) :-

Income > Limit,

Portion is Limit - PrevLimit,

Temp is Portion \* Rate,

NewAcc is Acc + Temp,

progressive\_tax(Income, Limit, Rest, NewAcc, Tax).

% -----------------------------

% Surcharge Application

apply\_surcharge(Regime, BaseTax, FinalTax) :-

income(I),

findall((Limit, Rate), surcharge(Regime, Limit, Rate), Slabs),

sort(Slabs, Sorted),

find\_surcharge\_rate(I, Sorted, 0, SurchargeRate),

FinalTax is BaseTax \* (1 + SurchargeRate).

find\_surcharge\_rate(\_, [], Rate, Rate).

find\_surcharge\_rate(I, [(Limit, Rate)|\_], \_, Rate) :- I =< Limit, !.

find\_surcharge\_rate(I, [\_|Rest], \_, Final) :- find\_surcharge\_rate(I, Rest, \_, Final).

% -----------------------------

% Compute Final Tax

compute\_tax(Regime, FinalTax) :-

taxable\_income(Regime, TI),

get\_sorted\_slabs(Regime, Slabs),

progressive\_tax(TI, 0, Slabs, 0, BaseTax),

apply\_surcharge(Regime, BaseTax, FinalTax).

% -----------------------------

% Regime Suggestion

suggest\_regime(BestRegime, OldTax, NewTax) :-

compute\_tax(old, OldTax),

compute\_tax(new, NewTax),

(OldTax < NewTax -> BestRegime = old ; BestRegime = new).

% -----------------------------

% Deduction Gap Logic

% 80C combined cap

deduction\_gap('80C', Gap) :-

findall(Amount,

(member(Section, ['80C', 'EPF', 'LifeInsurance', 'NPS']),

deduction(Section, Amount)),

Amounts),

sum\_list(Amounts, Total),

Gap is 150000 - Total,

Gap > 0.

% 80D logic based on age

deduction\_gap('80D', Gap) :-

(deduction('80D', Used) -> true ; Used = 0),

max\_deduction('80D', Max),

Gap is Max - Used,

Gap > 0.

% Other deductions (non-80C, non-80D)

deduction\_gap(S, Gap) :-

\+ member(S, ['80C', 'EPF', 'LifeInsurance', 'NPS', '80D']),

(deduction(S, Used) -> true ; Used = 0),

max\_deduction(S, Max),

Gap is Max - Used,

Gap > 0.

% -----------------------------

% Deduction Tips (including 80C)

print\_deduction\_tips :-

deduction\_gap(S, G),

format("💡 Tip: Invest ₹~w more in ~w to save tax.~n", [G, S]),

fail.

print\_deduction\_tips.

% -----------------------------

% Regime Explanation

explain\_choice(OldTax, NewTax, Deductions) :-

Diff is OldTax - NewTax,

Diff > 0,

format("📌 The New Regime is suggested as it saves ₹~2f more tax than the Old Regime. You claimed deductions of ₹~w, which may not be fully beneficial in the Old Regime.~n", [Diff, Deductions]).

explain\_choice(OldTax, NewTax, Deductions) :-

Diff is NewTax - OldTax,

Diff >= 0,

format("📌 The Old Regime is suggested as it saves ₹~2f more tax due to your total deductions of ₹~w.~n", [Diff, Deductions]).

% -----------------------------

% Final Tax Summary

tax\_summary :-

income(Income),

age(Age),

total\_deductions(TotalDeduction),

compute\_tax(old, OldTax),

compute\_tax(new, NewTax),

suggest\_regime(BestRegime, OldTax, NewTax),

taxable\_income(BestRegime, TaxableIncome),

format("📊 Tax Analysis Suggestion:~n~n"),

format("🪙 Income: ₹~w~n", [Income]),

format("👤 Age: ~w~n", [Age]),

format("📉 Deductions (incl. ₹50,000 standard): ₹~w~n", [TotalDeduction]),

format("💰 Taxable Income: ₹~w~n", [TaxableIncome]),

format("🧮 Old Regime Tax (w/ surcharge): ₹~2f~n", [OldTax]),

format("🧮 New Regime Tax (w/ surcharge): ₹~2f~n", [NewTax]),

format("🎯 Suggested Regime: ~w~n", [BestRegime]),

explain\_choice(OldTax, NewTax, TotalDeduction),

print\_deduction\_tips.

**PYTHON:**

from pyswip import Prolog

import sys

import os

# Fix encoding to display ₹ on Windows

if os.name == 'nt':

    try:

        sys.stdout.reconfigure(encoding='utf-8')

    except AttributeError:

        import ctypes

        ctypes.windll.kernel32.SetConsoleOutputCP(65001)

# --------------------------

# User Input

# --------------------------

try:

    age = int(input("Enter your age: "))

    income = int(input("Enter your annual income (₹): "))

except ValueError:

    print("❌ Invalid input. Please enter numbers.")

    exit()

# --------------------------

# Display Max Deduction Limits

# --------------------------

print("\n🔒 Maximum Deduction Limits for FY 2024-25:")

print(" - 80C (Investments, LIC, Tuition, etc.): ₹150,000")

print(f" - 80D (Medical Insurance): ₹{'50,000' if age >= 60 else '25,000'}")

print(" - 80CCD(1B) (NPS): ₹50,000")

print(" - EPF (Part of 80C): Included in ₹150,000")

print(" - NPS (Part of 80C): Included in ₹150,000")

print(" - LifeInsurance (Part of 80C): Included in ₹150,000")

# --------------------------

# Deduction Input

# --------------------------

deduction\_input = input("\nEnter deductions (e.g., 80C=120000,80D=25000,80CCD(1B)=20000): ")

deduction\_pairs = [d.strip() for d in deduction\_input.split(",") if "=" in d]

deductions = {}

for pair in deduction\_pairs:

    try:

        sec, amt = pair.split("=")

        deductions[sec.strip().upper()] = int(amt.strip())

    except ValueError:

        print(f"⚠️  Skipping invalid entry: {pair}")

# --------------------------

# Prolog Setup

# --------------------------

prolog = Prolog()

try:

    prolog.consult("tax\_advisor2.pl")

except Exception as e:

    print(f"❌ Failed to load Prolog file: {e}")

    exit()

# Clear old facts

for q in ["retractall(income(\_))", "retractall(age(\_))", "retractall(deduction(\_, \_))"]:

    list(prolog.query(q))

# Add new facts

prolog.assertz(f"income({income})")

prolog.assertz(f"age({age})")

for section, amount in deductions.items():

    # Enclose section in single quotes to support strings with special characters

    prolog.assertz(f"deduction('{section}', {amount})")

# --------------------------

# Output

# --------------------------

try:

    list(prolog.query("tax\_summary."))  # Prolog will print everything from tax\_summary/0

except Exception as e:

    print(f"❌ Error running tax\_summary: {e}")

#chcp 65001

**SAMPLE INPUT AND OUTPUT:**

D:\AI lab\tax\_bracket\_advisor>python tax\_advisor.py

Enter your age: 68

Enter your annual income (₹): 5000000

🔒 Maximum Deduction Limits for FY 2024-25:

- 80C (Investments, LIC, Tuition, etc.): ₹150,000

- 80D (Medical Insurance): ₹50,000

- 80CCD(1B) (NPS): ₹50,000

- EPF (Part of 80C): Included in ₹150,000

- NPS (Part of 80C): Included in ₹150,000

- LifeInsurance (Part of 80C): Included in ₹150,000

Enter deductions (e.g., 80C=120000,80D=25000,80CCD(1B)=20000): 0

Warning: d:/ai lab/tax\_bracket\_advisor/tax\_advisor2.pl:96:

Warning: Singleton variables: [Rest]

📊 Tax Analysis Suggestion:

🪙 Income: ₹5000000

👤 Age: 68

📉 Deductions (incl. ₹50,000 standard): ₹50000

💰 Taxable Income: ₹4950000

🧮 Old Regime Tax (w/ surcharge): ₹1328250.00

🧮 New Regime Tax (w/ surcharge): ₹1292500.00

🎯 Suggested Regime: new

📌 The New Regime is suggested as it saves ₹35750.00 more tax than the Old Regime. You claimed deductions of ₹50000, which may not be fully beneficial in the Old Regime.

💡 Tip: Invest ₹150000 more in 80C to save tax.

💡 Tip: Invest ₹50000 more in 80D to save tax.

D:\AI lab\tax\_bracket\_advisor>