Centennial College

Diabetes Prediction System

COMP 214 – Section 004: Advanced Database Concepts

Professor: Zakiyabanu Malek

Group 1:

Pham Tuong Nguyen

Aazain Ullah Khan

Reet Kaur

Gabriel Castellon

Karadag Ebru Nida

Date: 31st March 2024

# TABLE OF CONTENTS

[Data Requirements 3](#_Toc162794326)

[I. Introduction 3](#_Toc162794327)

[II. Database Structure 3](#_Toc162794328)

[III. The use of sequence 3](#_Toc162794329)

[IV. The use of index 3](#_Toc162794330)

[V. The use of procedures 3](#_Toc162794331)

[VI. The use of functions 4](#_Toc162794332)

[VII. The use of triggers 4](#_Toc162794333)

[ERD 5](#_Toc162794334)

[Database Tables 6](#_Toc162794335)

[SQL Commands 8](#_Toc162794336)

[I. Create tables 8](#_Toc162794337)

[II. Create sequences 12](#_Toc162794338)

[III. Create indexes 13](#_Toc162794339)

[IV. Create triggers 13](#_Toc162794340)

[V. Insert records 14](#_Toc162794341)

[PL/SQL Block codes 19](#_Toc162794342)

[Screenshot of the output 32](#_Toc162794343)

# Data Requirements

## Introduction

The objective of this project is to develop a database storing patient information to facilitate the diagnosis of diabetes through an AI model.

## Database Structure

Our database contains 7 tables:

* DP\_PATIENT: The table stores a patient's personal information.
* DP\_MEDICAL\_RECORDS: The table contains the patient’s weight, height, and lab results.
* DP\_FAMILY\_HISTORY: The table captures familial medical backgrounds related to diabetes.
* DP\_RISK\_FACTORS: The table contains lifestyle factors, environmental exposures, and other risk determinants.
* DP\_PREDICTION\_MODEL: The table stores the information of a prediction model. This table will fetch the data from the patient table and its associated tables make a prediction.
* DP\_PREDICTION\_RESULT: The table stores the diabetes diagnosis result of a patient after the model has predicted.
* DP\_MEDICATIONS: The table stores medications the model suggests if a patient has diabetes.

## The use of sequence

The sequence assigns unique identifiers to entities like DP\_PATIENT, DP\_MEDICAL\_RECORDS, and DP\_PREDICTION\_RESULT ensuring data integrity and facilitating retrieval.

## The use of index

The index is used to optimize query performance within our database by expediting data retrieval operations such as search, filtering, and sorting. This is applied across tables such as DP\_PATIENT, DP\_MEDICAL\_RECORDS, DP\_FAMILY\_HISTORY, and DP\_PREDICTION\_RESULT.

## The use of procedures

Add\_Patient\_Medication Procedure

This procedure is designed to insert a new medication record into the DP\_MEDICATIONS table for a specific patient. It's used when a patient is prescribed a new medication.

When a healthcare provider prescribes a new medication to a patient, this procedure is executed with the prescription details. It ensures that the patient's medication record is updated in the database.

Update\_Patient\_Medication Procedure

This procedure allows for the updating of an existing medication record in the DP\_MEDICATIONS table. It's particularly useful for adjusting medication dosages, changing administration instructions, or updating the medication schedule.

This procedure is called when there's a need to modify the details of an existing medication prescription for a patient. Changes may be due to various reasons such as response to treatment, side effects, or updated medical guidelines.

Calculate\_Medical\_Statistics

This procedure is used for calculating and displaying a patient’s statistical data from medical records from the DP\_MEDICAL\_RECORDS table. It summarizes data such as glucose levels, insulin intake, blood pressure, skin thickness, etc. Furthermore, it provides insights on the patient’s diabetes status. This procedure can be used to create reports on patient statistics or to analyze patient data in healthcare settings

## The use of functions

FUNC\_BMI\_CAL

The BMI calculation function computes the Body Mass Index (BMI) using the weight and height provided as input parameters. This function is useful for assessing the relationship between body mass index and diabetes. It can help healthcare professionals or researchers analyze how BMI correlates with diabetes prevalence.

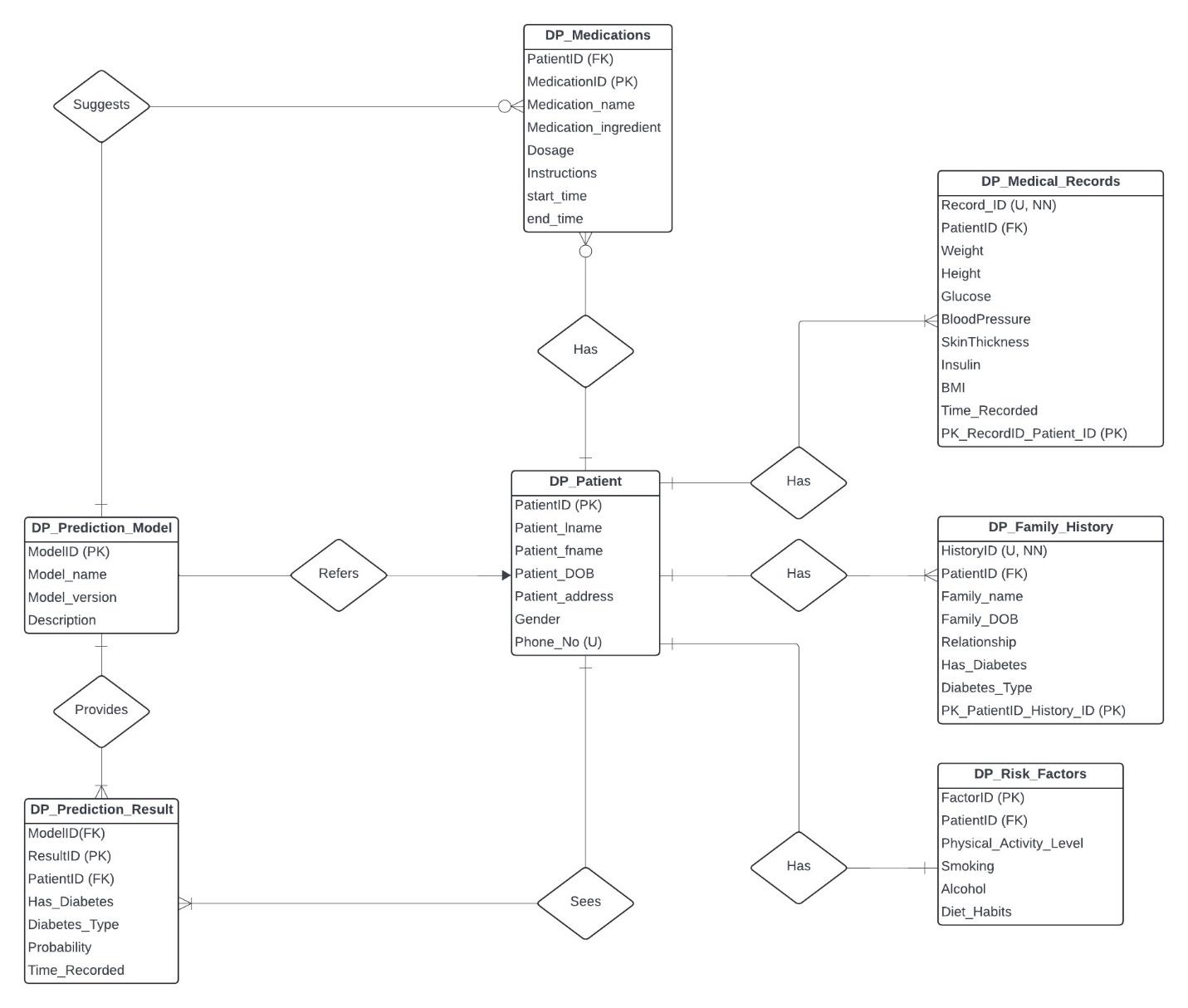
CALC\_AGE

We cannot store age as a column in our database as age keeps changing and it is not effective to manually update the column. As a result, we have implemented a function that calculates the age of patients based on the date of birth column in DB\_PATIENT table. It calculates the age in years using the months\_between function and returns the result.

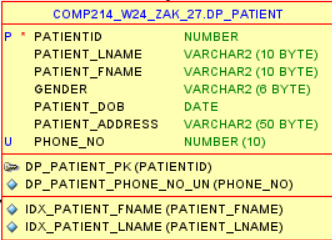
## The use of triggers

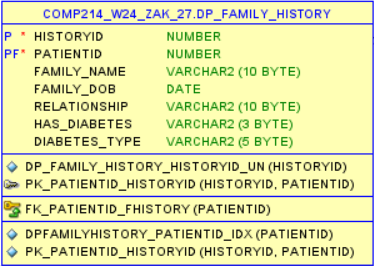
Triggers automatically execute actions in response to database events, ensuring data insertion and updating operations are carried out correctly.

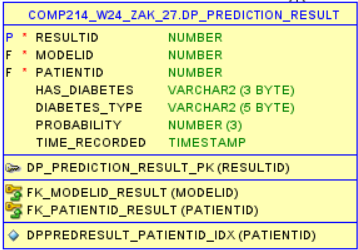
# ERD

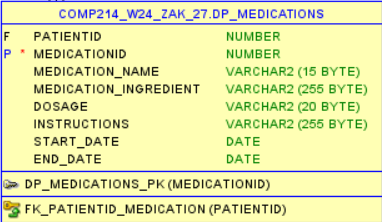


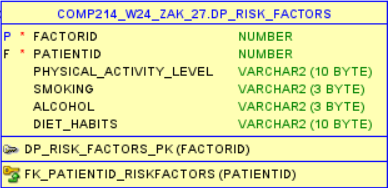
# Database Tables

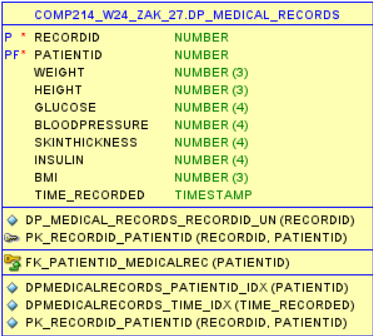
****

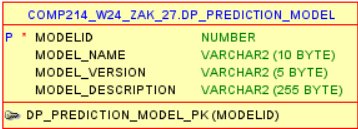
****

****

****

****

****

****

# SQL Commands

## Create tables

CREATE TABLE DP\_PATIENT(

PatientID NUMBER PRIMARY KEY,

Patient\_lname VARCHAR2(10),

Patient\_fname VARCHAR2(10),

Gender VARCHAR2(6),

Patient\_DOB DATE,

Patient\_address VARCHAR2(50),

Phone\_No NUMBER(10) UNIQUE

);

CREATE TABLE DP\_MEDICAL\_RECORDS(

RecordID NUMBER UNIQUE NOT NULL,

PatientID NUMBER NOT NULL,

Weight NUMBER(3),

Height NUMBER(3),

Glucose NUMBER(4),

BloodPressure NUMBER(4),

SkinThickness NUMBER(4),

Insulin NUMBER(4),

BMI NUMBER(3),

Time\_Recorded TIMESTAMP,

CONSTRAINT FK\_PatientID\_MedicalRec FOREIGN KEY (PatientID) REFERENCES DP\_PATIENT(PatientID),

CONSTRAINT PK\_RecordID\_PatientID PRIMARY KEY (RecordID, PatientID)

);

CREATE TABLE DP\_FAMILY\_HISTORY(

HistoryID NUMBER UNIQUE NOT NULL,

PatientID NUMBER NOT NULL,

Family\_name VARCHAR2(10),

Family\_DOB DATE,

Relationship VARCHAR2(10),

Has\_Diabetes VARCHAR2(3),

Diabetes\_Type VARCHAR2(5),

CONSTRAINT FK\_PatientID\_FHistory FOREIGN KEY (PatientID) REFERENCES DP\_PATIENT(PatientID),

CONSTRAINT PK\_PatientID\_HistoryID PRIMARY KEY (HistoryID, PatientID)

);

CREATE TABLE DP\_RISK\_FACTORS(

FactorID NUMBER PRIMARY KEY,

PatientID NUMBER NOT NULL,

Physical\_Activity\_Level VARCHAR2(10),

Smoking VARCHAR2(3),

Alcohol VARCHAR2(3),

Diet\_Habits VARCHAR2(10),

CONSTRAINT FK\_PatientID\_RiskFactors FOREIGN KEY (PatientID) REFERENCES DP\_PATIENT(PatientID)

);

CREATE TABLE DP\_PREDICTION\_MODEL(

ModelID NUMBER PRIMARY KEY,

Model\_name VARCHAR2(10),

Model\_version VARCHAR2(5),

Model\_description VARCHAR2(255)

);

CREATE TABLE DP\_PREDICTION\_RESULT(

ResultID NUMBER PRIMARY KEY,

ModelID NUMBER NOT NULL,

PatientID NUMBER NOT NULL,

Has\_Diabetes VARCHAR2(3),

Diabetes\_Type VARCHAR2(5),

Probability NUMBER(3),

Time\_Recorded TIMESTAMP,

CONSTRAINT FK\_PatientID\_Result FOREIGN KEY (PatientID) REFERENCES DP\_PATIENT(PatientID),

CONSTRAINT FK\_ModelID\_Result FOREIGN KEY (ModelID) REFERENCES DP\_PREDICTION\_MODEL(ModelID)

);

CREATE TABLE DP\_MEDICATIONS(

PatientID NUMBER,

MedicationID NUMBER PRIMARY KEY,

Medication\_name VARCHAR2(15),

Medication\_ingredient VARCHAR2(255),

Dosage VARCHAR2(20),

Instructions VARCHAR2(255),

Start\_Date DATE,

End\_Date DATE,

CONSTRAINT FK\_PatientID\_Medication FOREIGN KEY (PatientID) REFERENCES DP\_PATIENT(PatientID)

);

## Create sequences

CREATE SEQUENCE DP\_PATIENT\_PATIENTID\_SEQ

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE DP\_MEDICAL\_RECORDS\_SEQ

START WITH 100

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE DP\_PREDICTION\_RESULT\_SEQ

START WITH 900

INCREMENT BY 1

NOCACHE

NOCYCLE;

-- Apply to tables

ALTER TABLE DP\_PATIENT

MODIFY PatientID DEFAULT DP\_PATIENT\_PATIENTID\_SEQ.NEXTVAL;

ALTER TABLE DP\_MEDICAL\_RECORDS

MODIFY RecordID DEFAULT DP\_MEDICAL\_RECORDS\_SEQ.NEXTVAL;

ALTER TABLE DP\_PREDICTION\_RESULT

MODIFY RecordID DEFAULT DP\_PREDICTION\_RESULT\_SEQ.NEXTVAL;

## Create indexes

--DP\_PATIENT table

CREATE INDEX idx\_Patient\_lname ON DP\_PATIENT(Patient\_lname);

CREATE INDEX idx\_Patient\_fname ON DP\_PATIENT(Patient\_fname);

--DP\_MEDICAL\_RECORDS table

CREATE INDEX dpmedicalrecords\_patientid\_idx ON DP\_MEDICAL\_RECORDS(PatientID);

CREATE INDEX dpmedicalrecords\_time\_idx ON DP\_MEDICAL\_RECORDS(Time\_Recorded);

--DP\_FAMILY\_HISTORY table

CREATE INDEX dpfamilyhistory\_patientid\_idx ON DP\_FAMILY\_HISTORY(PatientID);

--DP\_PREDICTION\_RESULT table

CREATE INDEX dppredresult\_patientid\_idx ON DP\_PREDICTION\_RESULT(PatientID);

## Create triggers

CREATE OR REPLACE TRIGGER TRIGGER\_ON\_BMI\_TIME\_RECORDED

BEFORE INSERT OR UPDATE ON DP\_MEDICAL\_RECORDS

FOR EACH ROW

BEGIN

:NEW.BMI := FUNC\_BMI\_CAL(:NEW.WEIGHT, :NEW.HEIGHT);

IF :NEW.BMI = -1 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Invalid Input. Please check the weight and height');

END IF;

:NEW.Time\_Recorded := SYSTIMESTAMP;

END;

/

CREATE OR REPLACE TRIGGER UPDATE\_TIME\_PREDICTION

BEFORE INSERT OR UPDATE ON DP\_PREDICTION\_RESULT

FOR EACH ROW

BEGIN

:NEW.Time\_Recorded := SYSTIMESTAMP;

END;

/

CREATE OR REPLACE TRIGGER CHECK\_DIABETES\_CONDITION

BEFORE INSERT OR UPDATE ON DP\_FAMILY\_HISTORY

FOR EACH ROW

BEGIN

IF LOWER(:NEW.Has\_Diabetes) = 'no' AND :NEW.Diabetes\_Type IS NOT NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'If Has\_Diabetes is No, Diabetes\_Type must be empty.');

END IF;

END;

/

## Insert records

-- DP\_PATIENT

INSERT INTO DP\_PATIENT(Patient\_lname, Patient\_fname, Patient\_DOB, Patient\_address, Gender, Phone\_No) VALUES

('Smith', 'John', TO\_DATE('1990-05-15', 'YYYY-MM-DD'), '123 Main St, Anytown, Canada', 'Male', 1234567890);

INSERT INTO DP\_PATIENT(Patient\_lname, Patient\_fname, Patient\_DOB, Patient\_address, Gender, Phone\_No) VALUES

('Johnson', 'Emily', TO\_DATE('1985-09-22', 'YYYY-MM-DD'), '456 Elm St, Anycity, Canada', 'Female', 9876543210);

INSERT INTO DP\_PATIENT(Patient\_lname, Patient\_fname, Patient\_DOB, Patient\_address, Gender, Phone\_No) VALUES

('Williams', 'David', TO\_DATE('1978-03-10', 'YYYY-MM-DD'), '789 Oak St, Anystate, Canada', 'Male', 1112223333);

INSERT INTO DP\_PATIENT(Patient\_lname, Patient\_fname, Patient\_DOB, Patient\_address, Gender, Phone\_No) VALUES

('Brown', 'Sarah', TO\_DATE('1995-11-28', 'YYYY-MM-DD'), '321 Pine St, Anymetro, Canada', 'Female', 4445556666);

INSERT INTO DP\_PATIENT(Patient\_lname, Patient\_fname, Patient\_DOB, Patient\_address, Gender, Phone\_No) VALUES

('Jones', 'Michael', TO\_DATE('1980-07-07', 'YYYY-MM-DD'), '555 Cedar St, Anysuburb, Canada', 'Male', 7778889999);

INSERT INTO DP\_PATIENT(Patient\_lname, Patient\_fname, Patient\_DOB, Patient\_address, Gender, Phone\_No) VALUES

('Garcia', 'Maria', TO\_DATE('1972-12-03', 'YYYY-MM-DD'), '888 Maple St, Anyvillage, Canada', 'Female', 2223334444);

INSERT INTO DP\_PATIENT(Patient\_lname, Patient\_fname, Patient\_DOB, Patient\_address, Gender, Phone\_No) VALUES

('Martinez', 'Christ', TO\_DATE('1992-02-18', 'YYYY-MM-DD'), '1010 Birch St, Anyhamlet, Canada', 'Male', 5556667777);

-- DP\_PREDICTION\_MODEL

INSERT INTO DP\_PREDICTION\_MODEL(ModelID, Model\_name, Model\_version, Model\_description) VALUES

(1, 'KNN', '1.0', 'KNN is a machine learning algorithm used for classification and regression tasks.');

INSERT INTO DP\_PREDICTION\_MODEL(ModelID, Model\_name, Model\_version, Model\_description) VALUES

(2, 'DT', '2.3', 'Decision Tree is a predictive modeling algorithm that maps observations about an item to conclusions about the items target value.');

INSERT INTO DP\_PREDICTION\_MODEL(ModelID, Model\_name, Model\_version, Model\_description) VALUES

(3, 'RF', '1.5', 'Random Forest is an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees.');

-- DP\_FAMILY\_HISTORY

INSERT INTO DP\_FAMILY\_HISTORY (HistoryID, PatientID, Family\_name, Family\_DOB, Relationship, Has\_Diabetes, Diabetes\_Type)

VALUES (1001, 1, 'Smith', TO\_DATE('1970-05-15', 'YYYY-MM-DD'), 'Mother', 'Yes', 'Type2');

INSERT INTO DP\_FAMILY\_HISTORY (HistoryID, PatientID, Family\_name, Family\_DOB, Relationship, Has\_Diabetes, Diabetes\_Type)

VALUES (1002, 1, 'Smith', TO\_DATE('1975-08-20', 'YYYY-MM-DD'), 'Father', 'No', NULL);

INSERT INTO DP\_FAMILY\_HISTORY (HistoryID, PatientID, Family\_name, Family\_DOB, Relationship, Has\_Diabetes, Diabetes\_Type)

VALUES (1003, 2, 'Johnson', TO\_DATE('1965-03-10', 'YYYY-MM-DD'), 'Mother', 'Yes', 'Type1');

INSERT INTO DP\_FAMILY\_HISTORY (HistoryID, PatientID, Family\_name, Family\_DOB, Relationship, Has\_Diabetes, Diabetes\_Type)

VALUES (1004, 2, 'Johnson', TO\_DATE('1968-11-25', 'YYYY-MM-DD'), 'Father', 'Yes', 'Type2');

INSERT INTO DP\_FAMILY\_HISTORY (HistoryID, PatientID, Family\_name, Family\_DOB, Relationship, Has\_Diabetes, Diabetes\_Type)

VALUES (1005, 3, 'Williams', TO\_DATE('1982-07-03', 'YYYY-MM-DD'), 'Mother', 'No', NULL);

INSERT INTO DP\_FAMILY\_HISTORY (HistoryID, PatientID, Family\_name, Family\_DOB, Relationship, Has\_Diabetes, Diabetes\_Type)

VALUES (1006, 6, 'Garcia', TO\_DATE('1978-09-12', 'YYYY-MM-DD'), 'Father', 'Yes', 'Type1');

INSERT INTO DP\_FAMILY\_HISTORY (HistoryID, PatientID, Family\_name, Family\_DOB, Relationship, Has\_Diabetes, Diabetes\_Type)

VALUES (1007, 4, 'Brown', TO\_DATE('1955-12-28', 'YYYY-MM-DD'), 'Mother', 'No', NULL);

-- DP\_RISK\_FACTORS

INSERT INTO DP\_RISK\_FACTORS (FactorID, PatientID, Physical\_Activity\_Level, Smoking, Alcohol, Diet\_Habits)

VALUES (2001, 1, 'Active', 'No', 'No', 'Balanced');

INSERT INTO DP\_RISK\_FACTORS (FactorID, PatientID, Physical\_Activity\_Level, Smoking, Alcohol, Diet\_Habits)

VALUES (2002, 2, 'Sedentary', 'Yes', 'No', 'High Sugar');

INSERT INTO DP\_RISK\_FACTORS (FactorID, PatientID, Physical\_Activity\_Level, Smoking, Alcohol, Diet\_Habits)

VALUES (2003, 3, 'Moderate', 'No', 'Yes', 'Low Fiber');

INSERT INTO DP\_RISK\_FACTORS (FactorID, PatientID, Physical\_Activity\_Level, Smoking, Alcohol, Diet\_Habits)

VALUES (2004, 4, 'Active', 'No', 'No', 'Balanced');

INSERT INTO DP\_RISK\_FACTORS (FactorID, PatientID, Physical\_Activity\_Level, Smoking, Alcohol, Diet\_Habits)

VALUES (2005, 5, 'Active', 'Yes', 'No', 'Low Carb');

INSERT INTO DP\_RISK\_FACTORS (FactorID, PatientID, Physical\_Activity\_Level, Smoking, Alcohol, Diet\_Habits)

VALUES (2006, 6, 'Sedentary', 'Yes', 'Yes', 'High Fat');

INSERT INTO DP\_RISK\_FACTORS (FactorID, PatientID, Physical\_Activity\_Level, Smoking, Alcohol, Diet\_Habits)

VALUES (2007, 7, 'Moderate', 'No', 'Yes', 'Balanced');

--DP\_MEDICAL\_RECORDS

INSERT INTO DP\_MEDICAL\_RECORDS (PatientID, Weight, Height, Glucose, BloodPressure, skinthickness, Insulin)

VALUES (1, 70, 1.80, 85, 66, 29, 0);

INSERT INTO DP\_MEDICAL\_RECORDS (PatientID, Weight, Height, Glucose, BloodPressure, skinthickness, Insulin)

VALUES (2, 100, 1.85, 149, 66, 0, 168);

INSERT INTO DP\_MEDICAL\_RECORDS (PatientID, Weight, Height, Glucose, BloodPressure, skinthickness, Insulin)

VALUES (3, 66, 1.70, 183, 72, 35, 0);

INSERT INTO DP\_MEDICAL\_RECORDS (PatientID, Weight, Height, Glucose, BloodPressure, skinthickness, Insulin)

VALUES (4, 50, 1.66, 137, 40, 32, 0);

INSERT INTO DP\_MEDICAL\_RECORDS (PatientID, Weight, Height, Glucose, BloodPressure, skinthickness, Insulin)

VALUES (5, 75, 1.73, 116, 70, 45, 88);

INSERT INTO DP\_MEDICAL\_RECORDS (PatientID, Weight, Height, Glucose, BloodPressure, skinthickness, Insulin)

VALUES (6, 85, 1.90, 197, 60, 0, 0);

INSERT INTO DP\_MEDICAL\_RECORDS (PatientID, Weight, Height, Glucose, BloodPressure, skinthickness, Insulin)

VALUES (7, 66, 1.65, 78, 50, 0, 543);

# PL/SQL Block codes

CREATE OR REPLACE PACKAGE DP\_PACKAGE--header

AS

FUNCTION FUNC\_BMI\_CAL(

in\_weight IN DP\_MEDICAL\_RECORDS.WEIGHT%TYPE,

in\_height IN DP\_MEDICAL\_RECORDS.HEIGHT%TYPE)

RETURN NUMBER;

FUNCTION calc\_age

(p\_id in dp\_patient.patientid%type)

return number;

PROCEDURE PROC\_PREDICTION\_RESULT (

p\_ModelID IN DP\_PREDICTION\_MODEL.ModelID%TYPE,

p\_Mec\_RecID IN DP\_MEDICAL\_RECORDS.RecordID%TYPE);

PROCEDURE Add\_Patient\_Medication (

pro\_PatientID IN DP\_MEDICATIONS.PatientID%TYPE,

pro\_MedicationName IN DP\_MEDICATIONS.Medication\_name%TYPE,

pro\_MedicationIngredient IN DP\_MEDICATIONS.Medication\_ingredient%TYPE,

pro\_Dosage IN DP\_MEDICATIONS.Dosage%TYPE,

pro\_Instructions IN DP\_MEDICATIONS.Instructions%TYPE,

pro\_StartDate IN DP\_MEDICATIONS.Start\_Date%TYPE,

pro\_EndDate IN DP\_MEDICATIONS.End\_Date%TYPE);

PROCEDURE Calculate\_Medical\_Statistics (

pro\_PatientID IN DP\_PATIENT.PatientID%TYPE);

END;

/

CREATE OR REPLACE PACKAGE BODY DP\_PACKAGE--BODY

AS

FUNCTION FUNC\_BMI\_CAL(

in\_weight IN DP\_MEDICAL\_RECORDS.WEIGHT%TYPE,

in\_height IN DP\_MEDICAL\_RECORDS.HEIGHT%TYPE)

RETURN NUMBER

IS

out\_bmi DP\_MEDICAL\_RECORDS.BMI%TYPE;

BEGIN

IF in\_weight IS NULL OR in\_weight = 0 AND in\_height IS NULL OR in\_height = 0 THEN

out\_bmi := -1;

raise\_application\_error(-20015,'Invalid input values');

END IF;

IF in\_height > 3 THEN

out\_bmi := -1;

raise\_application\_error(-20016,'Height must be in meter and smaller than 3');

ELSE

out\_bmi := in\_weight / POWER(in\_height,2);

END IF;

RETURN out\_bmi;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END FUNC\_BMI\_CAL;

function calc\_age

(p\_id in dp\_patient.patientid%type)

return number is

dob dp\_patient.patient\_dob%type;

age number;

begin

select patient\_dob into dob from dp\_patient where patientid = p\_id;

age := floor(months\_between(sysdate, dob) / 12);

return age;

exception

when no\_data\_found then

return null;

when others then

return null;

end calc\_age;

PROCEDURE PROC\_PREDICTION\_RESULT (

p\_ModelID IN DP\_PREDICTION\_MODEL.ModelID%TYPE,

p\_Mec\_RecID IN DP\_MEDICAL\_RECORDS.RecordID%TYPE)

AS

v\_PatientID DP\_MEDICAL\_RECORDS.PatientID%TYPE;

v\_BMI DP\_MEDICAL\_RECORDS.BMI%TYPE;

v\_Insulin DP\_MEDICAL\_RECORDS.Insulin%TYPE;

v\_Has\_Diabetes DP\_PREDICTION\_RESULT.Has\_Diabetes%TYPE;

v\_Diabetes\_Type DP\_PREDICTION\_RESULT.Diabetes\_Type%TYPE;

v\_Probability DP\_PREDICTION\_RESULT.Probability%TYPE;

CURSOR c\_medical\_records IS

SELECT PatientID, BMI, Insulin

FROM DP\_MEDICAL\_RECORDS WHERE RecordID = p\_Mec\_RecID;

BEGIN

FOR medical\_rec IN c\_medical\_records LOOP

v\_PatientID := medical\_rec.PatientID;

v\_BMI := medical\_rec.BMI;

v\_Insulin := medical\_rec.Insulin;

-- Just example

-- Replace this with the actual prediction logic in Python

IF v\_BMI > 20 AND v\_Insulin > 20 THEN

v\_Has\_Diabetes := 'Yes';

v\_Diabetes\_Type := 'Type2';

v\_Probability := 0.85;

ELSE

v\_Has\_Diabetes := 'No';

v\_Diabetes\_Type := NULL;

v\_Probability := 0.15;

END IF;

INSERT INTO DP\_PREDICTION\_RESULT (ModelID, PatientID, Has\_Diabetes, Diabetes\_Type, Probability)

VALUES (p\_ModelID, v\_PatientID, v\_Has\_Diabetes, v\_Diabetes\_Type, v\_Probability);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Prediction results inserted successfully.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END PROC\_PREDICTION\_RESULT;

PROCEDURE Add\_Patient\_Medication (

pro\_PatientID IN DP\_MEDICATIONS.PatientID%TYPE,

pro\_MedicationName IN DP\_MEDICATIONS.Medication\_name%TYPE,

pro\_MedicationIngredient IN DP\_MEDICATIONS.Medication\_ingredient%TYPE,

pro\_Dosage IN DP\_MEDICATIONS.Dosage%TYPE,

pro\_Instructions IN DP\_MEDICATIONS.Instructions%TYPE,

pro\_StartDate IN DP\_MEDICATIONS.Start\_Date%TYPE,

pro\_EndDate IN DP\_MEDICATIONS.End\_Date%TYPE)

AS

BEGIN

INSERT INTO DP\_MEDICATIONS (

PatientID,

MedicationID,

Medication\_name,

Medication\_ingredient,

Dosage,

Instructions,

Start\_Date,

End\_Date

) VALUES (

pro\_PatientID,

DP\_MEDICAL\_RECORDS\_SEQ.NEXTVAL,

pro\_MedicationName,

pro\_MedicationIngredient,

pro\_Dosage,

pro\_Instructions,

pro\_StartDate,

pro\_EndDate

);

EXCEPTION

WHEN OTHERS THEN

-- Handle exceptions (e.g., invalid patient ID, constraint violations)

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END Add\_Patient\_Medication;

PROCEDURE Calculate\_Medical\_Statistics (

pro\_PatientID IN DP\_PATIENT.PatientID%TYPE)

AS

records NUMBER := 0;

avg\_glucose NUMBER := 0;

max\_bp NUMBER := 0;

min\_skinthickness NUMBER := NULL;

totalinsulin NUMBER := 0;

p\_lname DP\_PATIENT.Patient\_lname%TYPE;

p\_fname DP\_PATIENT.Patient\_fname%TYPE;

has\_db VARCHAR2(3);

dbtype VARCHAR2(5);

CURSOR medical\_stats IS

SELECT

COUNT(\*) AS total\_records,

AVG(Glucose) AS average\_glucose,

MAX(BloodPressure) AS max\_blood\_pressure,

MIN(SkinThickness) AS min\_skin\_thickness,

SUM(Insulin) AS total\_insulin,

p.Patient\_lname,

p.Patient\_fname,

fh.Has\_Diabetes,

fh.Diabetes\_Type

FROM

DP\_MEDICAL\_RECORDS mr

JOIN

DP\_PATIENT p ON mr.PatientID = p.PatientID

LEFT JOIN

DP\_FAMILY\_HISTORY fh ON p.PatientID = fh.PatientID

WHERE

mr.PatientID = pro\_PatientID

GROUP BY

p.PatientID, p.Patient\_lname, p.Patient\_fname, fh.Has\_Diabetes, fh.Diabetes\_Type;

BEGIN

-- Open cursor and fetch values into variables

OPEN medical\_stats;

FETCH medical\_stats INTO

records,

avg\_glucose,

max\_bp,

min\_skinthickness,

totalinsulin,

p\_lname,

p\_fname,

has\_db,

dbtype;

CLOSE medical\_stats;

-- Display statistics

DBMS\_OUTPUT.PUT\_LINE('Displaying statistics for patient: ' || p\_fname || ' ' || p\_lname);

DBMS\_OUTPUT.PUT\_LINE('Total Medical Records: ' || records);

DBMS\_OUTPUT.PUT\_LINE('Average Glucose Level: ' || avg\_glucose);

DBMS\_OUTPUT.PUT\_LINE('Maximum Blood Pressure: ' || max\_bp);

DBMS\_OUTPUT.PUT\_LINE('Minimum Skin Thickness: ' || min\_skinthickness);

DBMS\_OUTPUT.PUT\_LINE('Total Insulin Administered: ' || totalinsulin);

IF has\_db = 'Yes' THEN

DBMS\_OUTPUT.PUT\_LINE('Patient has diabetes');

DBMS\_OUTPUT.PUT\_LINE('Diabetes Type: ' || dbtype);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Patient does not have diabetes');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No medical records found for the specified patient.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END Calculate\_Medical\_Statistics;

END;

/

# Screenshot of the output

