CREATE TABLE Patients (

PatientID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

DateOfBirth DATE,

Gender VARCHAR2(10),

ContactNumber VARCHAR2(15),

Address VARCHAR2(250),

InsuranceLimit NUMBER(10, 2)

);

---------------------------------------------------------

CREATE TABLE Users (

UserID NUMBER PRIMARY KEY,

Username VARCHAR2(50),

Password VARCHAR2(50),

Role VARCHAR2(20)

);

------------------------------------------------------------

CREATE TABLE Diagnosis (

DiagnosisID NUMBER PRIMARY KEY,

PatientID NUMBER,

DiagnosisDate DATE,

DiagnosisDetails CLOB,

FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)

);

--------------------------------------------------------------

CREATE TABLE Bills (

BillID NUMBER PRIMARY KEY,

PatientID NUMBER,

BillAmount NUMBER(10, 2),

BillingDate DATE,

FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)

);

-------------------------------------------------------------

CREATE VIEW PatientDetails AS

SELECT Patients.\*, Diagnosis.DiagnosisDetails

FROM Patients

JOIN Diagnosis ON Patients.PatientID = Diagnosis.PatientID;

-----------------------------------------------------------------

CREATE VIEW UserRoles AS

SELECT DISTINCT Role FROM Users;

---------------------------------------------------------------

CREATE INDEX idx\_PatientID ON Patients(PatientID);

----------------------------------------------------------

CREATE SEQUENCE Patients\_Seq START WITH 1 INCREMENT BY 1;

-------------------------------------------------------------------

CREATE OR REPLACE TRIGGER Patients\_BeforeInsert

BEFORE INSERT ON Patients

FOR EACH ROW

BEGIN

SELECT Patients\_Seq.NEXTVAL INTO :new.PatientID FROM DUAL;

END;

/

----------------------------------------------------------------

CREATE OR REPLACE PROCEDURE RegisterUserRole(p\_Role VARCHAR2)

AS

BEGIN

INSERT INTO Users (Role) VALUES (p\_Role);

COMMIT;

END;

/

----------------------------------------------------------------------

CREATE OR REPLACE PROCEDURE AddDiagnosis(p\_PatientID NUMBER, p\_DiagnosisDate DATE, p\_DiagnosisDetails CLOB)

AS

BEGIN

INSERT INTO Diagnosis (PatientID, DiagnosisDate, DiagnosisDetails) VALUES (p\_PatientID, p\_DiagnosisDate, p\_DiagnosisDetails);

COMMIT;

END;

/

-------------------------------------------------------------------------------

CREATE OR REPLACE PROCEDURE FetchPatientDetails(p\_PatientID NUMBER)

AS

v\_PatientDetails PatientDetails%ROWTYPE;

BEGIN

SELECT \* INTO v\_PatientDetails

FROM PatientDetails

WHERE PatientID = p\_PatientID;

DBMS\_OUTPUT.PUT\_LINE('PatientID: ' || v\_PatientDetails.PatientID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_PatientDetails.Name);

END;

/

CREATE OR REPLACE PROCEDURE GenerateBill(p\_PatientID NUMBER)

AS

v\_TotalBill NUMBER(10, 2);

BEGIN

SELECT SUM(BillAmount) INTO v\_TotalBill

FROM Bills

WHERE PatientID = p\_PatientID;

INSERT INTO Bills (PatientID, BillAmount, BillingDate)

VALUES (p\_PatientID, v\_TotalBill, SYSDATE);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bill generated successfully. Total amount: ' || v\_TotalBill);

EXCEPTION

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

-------------------------------------------------------------------------

CREATE OR REPLACE TRIGGER CheckInsuranceLimit

BEFORE UPDATE ON Patients

FOR EACH ROW

BEGIN

IF :new.InsuranceLimit <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Patient insurance limit expired');

END IF;

END;

/

--------------------------------------------------------------------------------------

CREATE SEQUENCE UserID\_Seq

START WITH 1

INCREMENT BY 1

NOMAXVALUE

NOCACHE

NOCYCLE;

--------------------------------------------------------------------------------------------------

CREATE SEQUENCE DIAGNOSISID\_SEQ

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

------------------------------------------------------------------------------------------------------------------

---------SAMPLE DATA------------

INSERT INTO Patients (Name, DateOfBirth, Gender, ContactNumber, Address, InsuranceLimit)

VALUES ('pratik patil', TO\_DATE('1995-05-15', 'YYYY-MM-DD'), 'Male', '7276322887', 'nakshatra plot no.23 sangli', 5000.00);

INSERT INTO Patients (Name, DateOfBirth, Gender, ContactNumber, Address, InsuranceLimit)

VALUES ('vitthal patil', TO\_DATE('2002-05-15', 'YYYY-MM-DD'), 'Male', '7276322887', 'nakshatra plot no.23 sangli', 7000.00);

INSERT INTO Users (UserID, Username, Password, Role)

VALUES (UserID\_Seq.NEXTVAL, 'pratik patil', 'password123', 'Doctor');

INSERT INTO Diagnosis (DIAGNOSISID, PatientID, DiagnosisDate, DiagnosisDetails)

VALUES (DIAGNOSISID\_SEQ.NEXTVAL, 1, TO\_DATE('2024-01-25', 'YYYY-MM-DD'), 'Fever');

INSERT INTO Bills (BILLID, PatientID, BillAmount, BillingDate)

VALUES (1, 1, 100.00, SYSDATE);

-------------------------------------------------------------------------------------------------------------

1) Write necessary queries to register new user roles and personas

CREATE OR REPLACE PROCEDURE RegisterUserRole(p\_Role VARCHAR2)

AS

BEGIN

INSERT INTO Users (UserID, Role) VALUES (UserID\_Seq.NEXTVAL, p\_Role);

COMMIT;

END;

/

2)Write necessary queries to add to the list of diagnosis of the patient tagged by date

INSERT INTO DIAGNOSIS (DIAGNOSISID, PatientID, DiagnosisDate, DiagnosisDetails)

VALUES (DIAGNOSISID\_SEQ.NEXTVAL, 1, TO\_DATE('2024-02-10', 'YYYY-MM-DD'), 'Headache');

3)Write necessary queries to fetch required details of a particular patient.

CREATE OR REPLACE PROCEDURE FetchPatientDetails(p\_PatientID NUMBER)

AS

v\_PatientDetails PatientDetails%ROWTYPE;

BEGIN

FOR c IN (SELECT \* FROM PatientDetails WHERE PatientID = p\_PatientID) LOOP

v\_PatientDetails := c;

DBMS\_OUTPUT.PUT\_LINE('PatientID: ' || v\_PatientDetails.PatientID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_PatientDetails.Name);

END LOOP;

END;

/

4)Write necessary queries to prepare bill for the patient at the end of checkout.

CREATE OR REPLACE PROCEDURE GenerateBill(p\_PatientID NUMBER)

AS

v\_TotalBill NUMBER(10, 2);

BEGIN

v\_TotalBill := 200.00;

INSERT INTO Bills (PatientID, BillAmount, BillingDate)

VALUES (p\_PatientID, v\_TotalBill, SYSDATE);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bill generated successfully. Total amount: ' || v\_TotalBill);

EXCEPTION

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

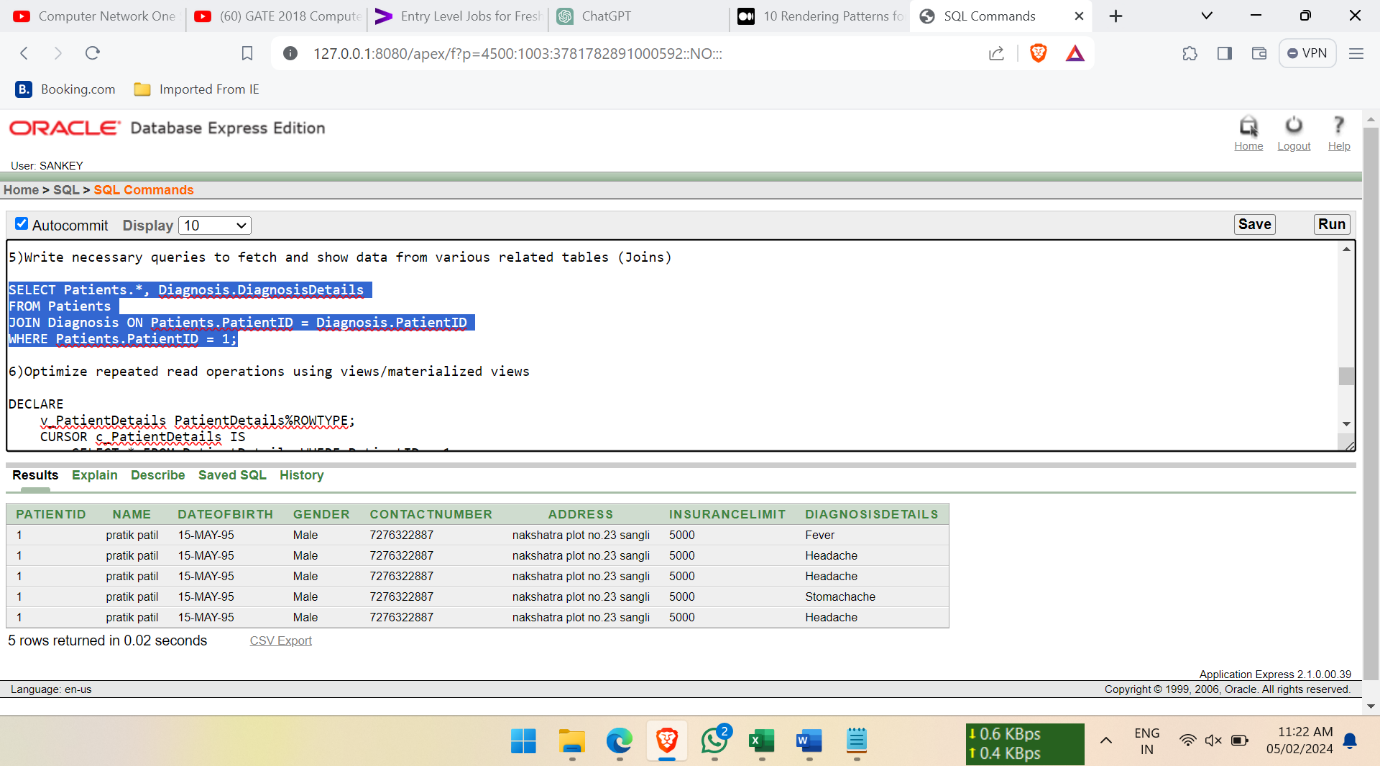
5)Write necessary queries to fetch and show data from various related tables (Joins)

SELECT Patients.\*, Diagnosis.DiagnosisDetails

FROM Patients

JOIN Diagnosis ON Patients.PatientID = Diagnosis.PatientID

WHERE Patients.PatientID = 1;



6)Optimize repeated read operations using views/materialized views

DECLARE

v\_PatientDetails PatientDetails%ROWTYPE;

CURSOR c\_PatientDetails IS

SELECT \* FROM PatientDetails WHERE PatientID = 1;

BEGIN

OPEN c\_PatientDetails;

FETCH c\_PatientDetails INTO v\_PatientDetails;

IF c\_PatientDetails%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('PatientID: ' || v\_PatientDetails.PatientID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_PatientDetails.Name);

ELSE

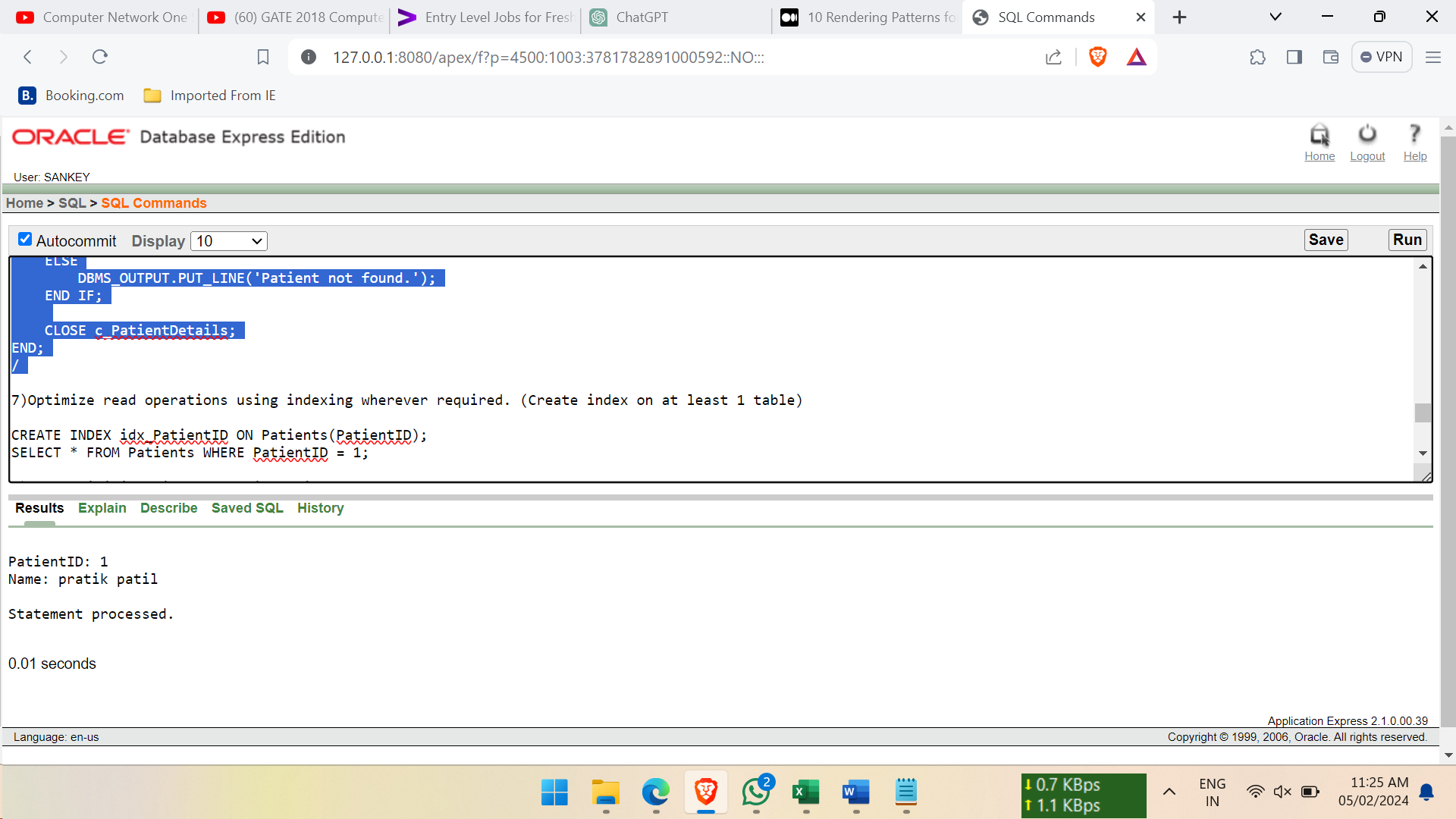
DBMS\_OUTPUT.PUT\_LINE('Patient not found.');

END IF;

CLOSE c\_PatientDetails;

END;

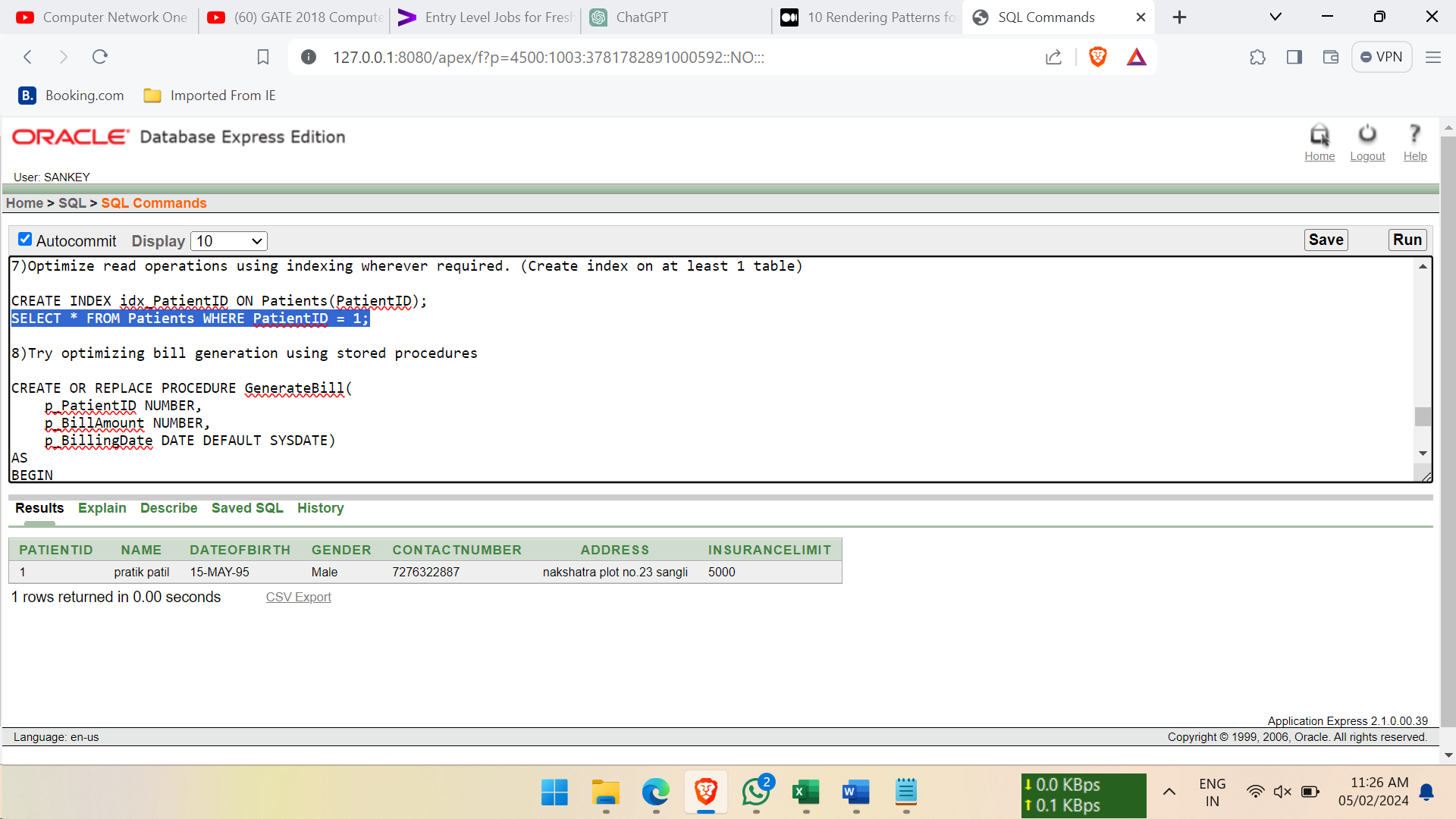
/



7)Optimize read operations using indexing wherever required. (Create index on at least 1 table)

CREATE INDEX idx\_PatientID ON Patients(PatientID);

SELECT \* FROM Patients WHERE PatientID = 1;



8)Try optimizing bill generation using stored procedures

CREATE OR REPLACE PROCEDURE GenerateBill(

p\_PatientID NUMBER,

p\_BillAmount NUMBER,

p\_BillingDate DATE DEFAULT SYSDATE)

AS

BEGIN

IF p\_PatientID IS NULL OR p\_BillAmount IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid input parameters for bill generation.');

END IF;

FOR i IN 1..10

LOOP

INSERT INTO Bills (PatientID, BillAmount, BillingDate)

VALUES (p\_PatientID, p\_BillAmount, p\_BillingDate);

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bill generated successfully.');

EXCEPTION

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

9) Add necessary triggers to indicate when patients medical insurance limit has expired.

CREATE OR REPLACE TRIGGER CheckInsuranceLimit

BEFORE UPDATE ON Patients

FOR EACH ROW

BEGIN

IF :new.InsuranceLimit <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Patient insurance limit expired');

END IF;

END;

/