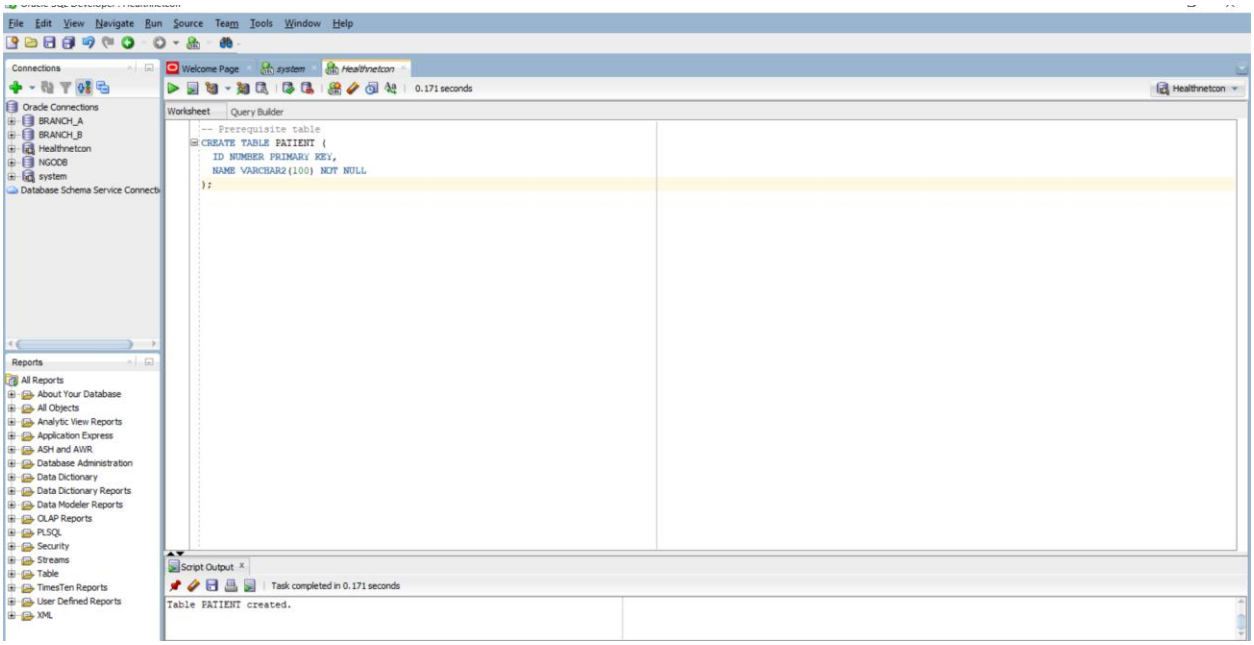
REG NO: 224020331

```
-- Prerequisite table
CREATE TABLE PATIENT (
  ID NUMBER PRIMARY KEY,
  NAME VARCHAR2(100) NOT NULL
);
```




```
-- Corrected PATIENT_MED table
CREATE TABLE PATIENT_MED (
  PATIENT_MED_ID NUMBER PRIMARY KEY, -- unique id
  PATIENT_ID NUMBER NOT NULL REFERENCES PATIENT(ID), -- must reference an existing patient
  MED_NAME VARCHAR2(80) NOT NULL, -- mandatory field
  DOSE_MG NUMBER(6,2) CHECK (DOSE_MG >= 0), -- non-negative dose
```

START_DT DATE,

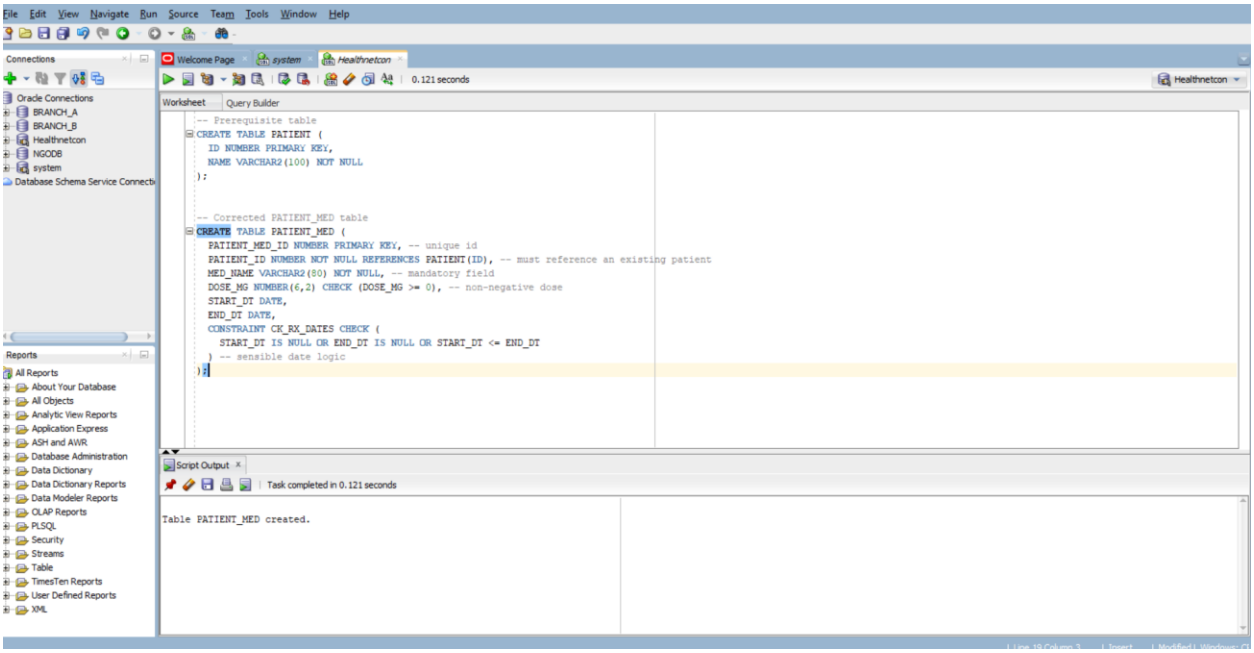
END_DT DATE,

CONSTRAINT CK_RX_DATES CHECK (

START_DT IS NULL OR END_DT IS NULL OR START_DT <= END_DT

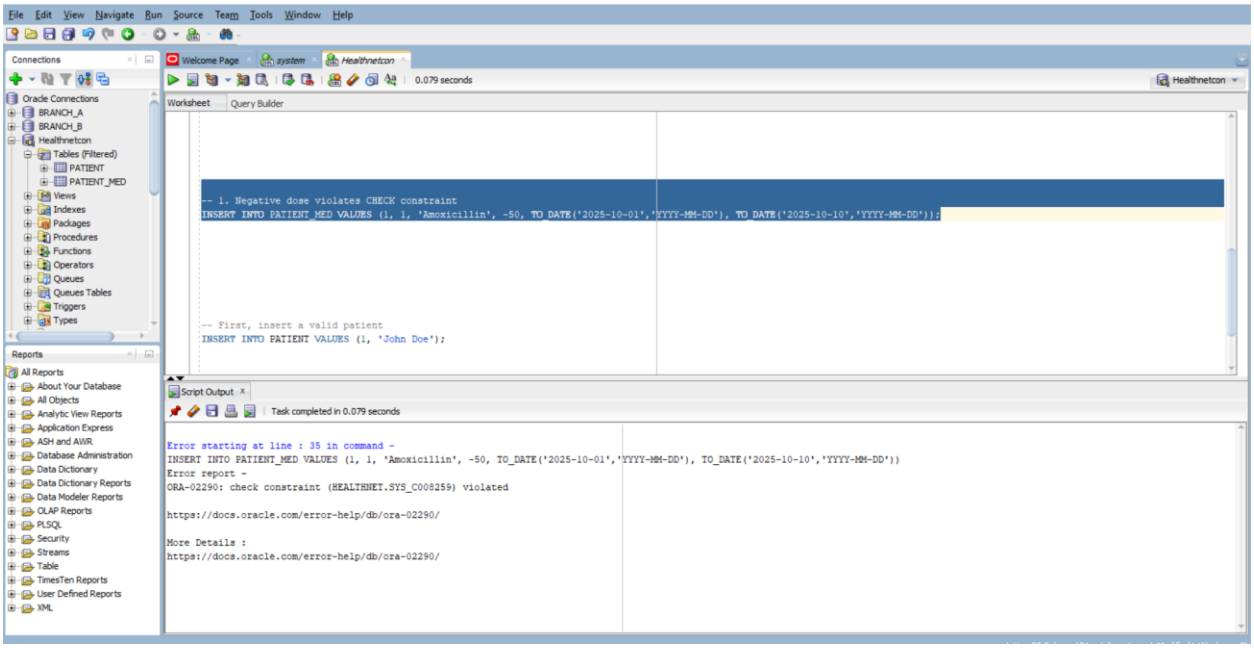
) -- sensible date logic

);



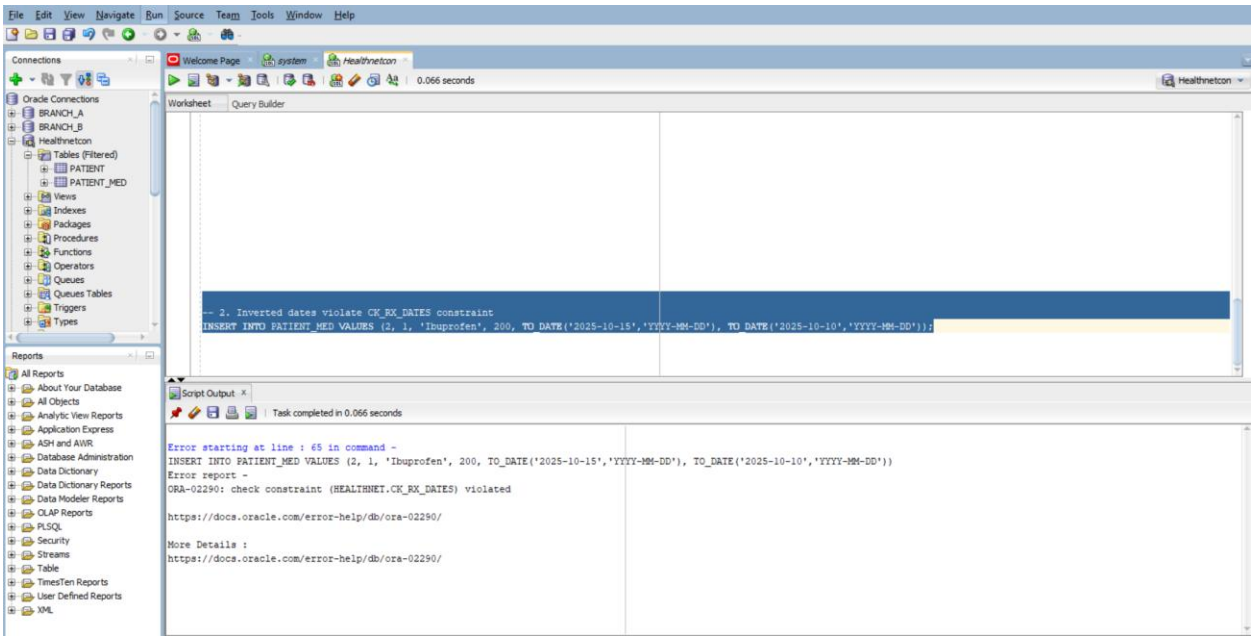
-- 1. Negative dose violates CHECK constraint

INSERT INTO PATIENT_MED VALUES (1, 1, 'Amoxicillin', -50, TO_DATE('2025-10-01','YYYY-MM-DD'), TO_DATE('2025-10-10','YYYY-MM-DD'));



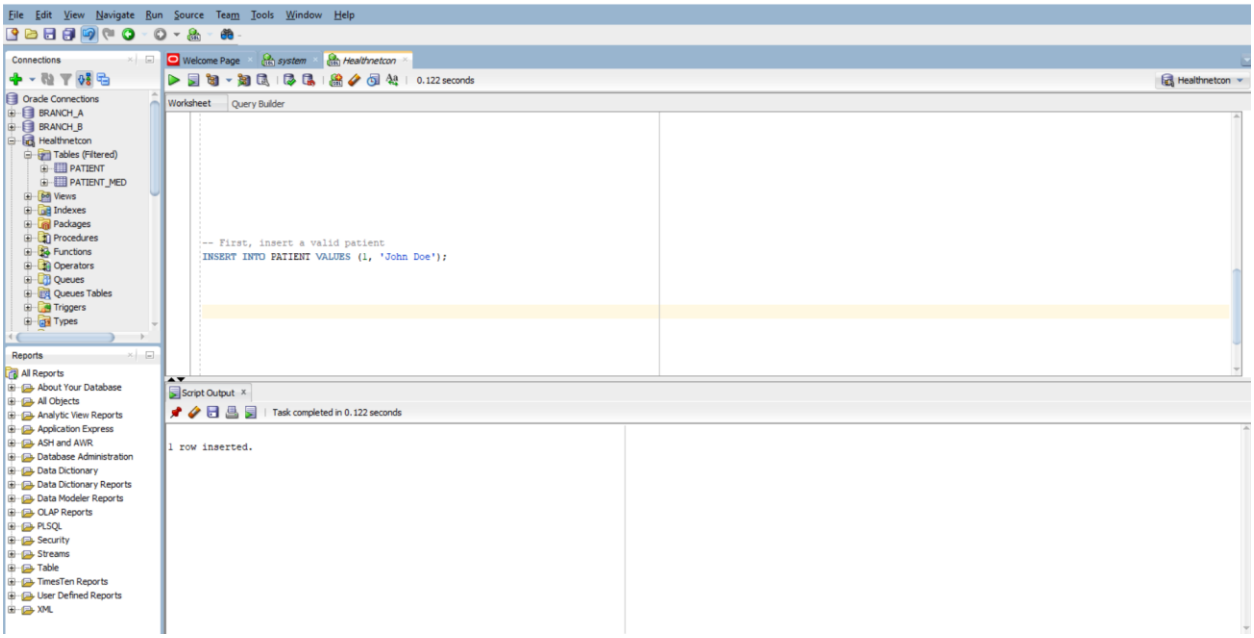
-- 2. Inverted dates violate CK_RX_DATES constraint

INSERT INTO PATIENT_MED VALUES (2, 1, 'Ibuprofen', 200, TO_DATE('2025-10-15','YYYY-MM-DD'), TO_DATE('2025-10-10','YYYY-MM-DD'));



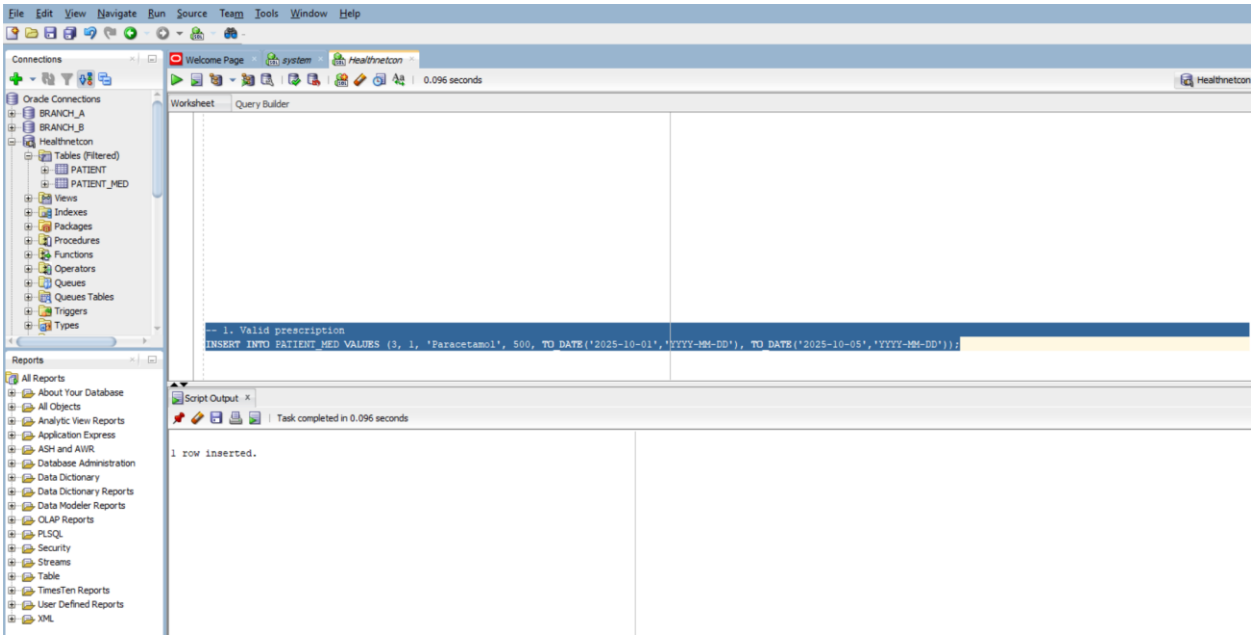
-- First, insert a valid patient

INSERT INTO PATIENT VALUES (1, 'John Doe');



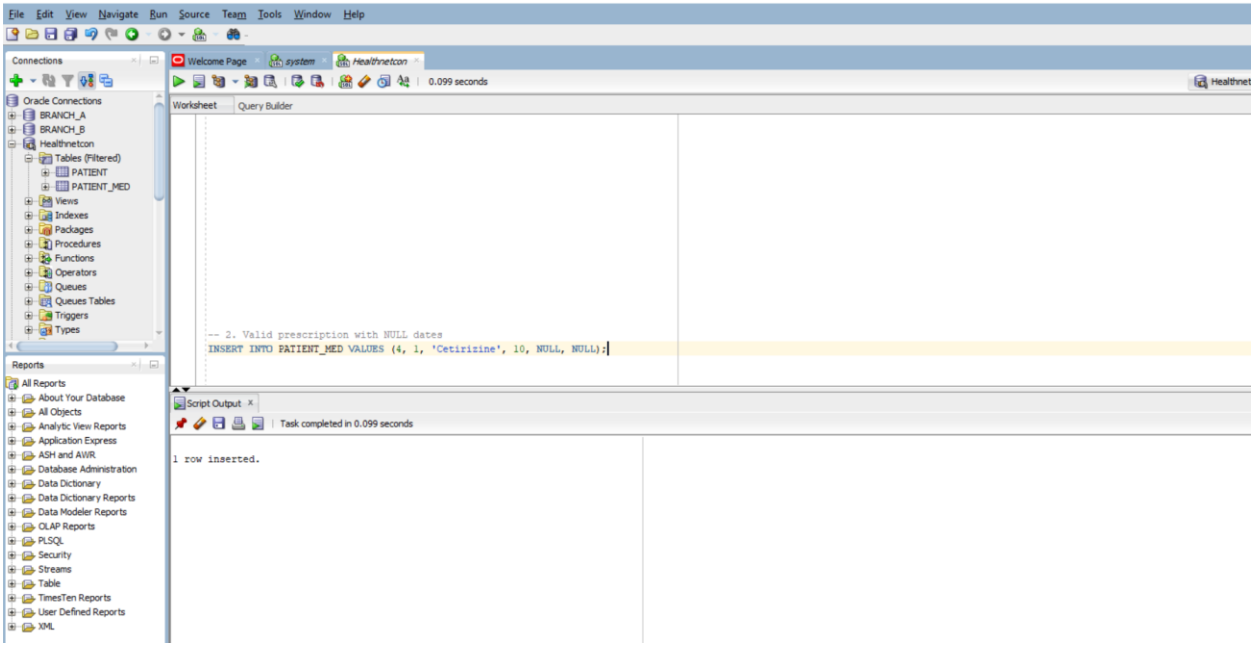
-- 1. Valid prescription

INSERT INTO PATIENT_MED VALUES (3, 1, 'Paracetamol', 500, TO_DATE('2025-10-01','YYYY-MM-DD'), TO_DATE('2025-10-05','YYYY-MM-DD'));

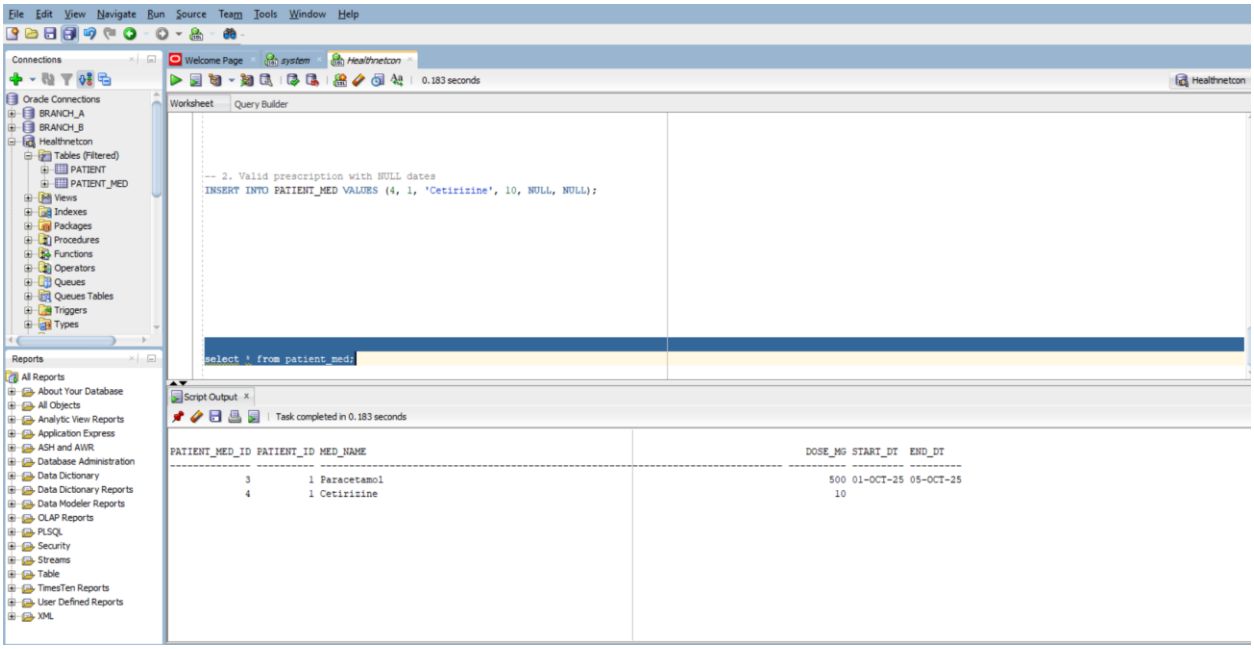


-- 2. Valid prescription with NULL dates

INSERT INTO PATIENT_MED VALUES (4, 1, 'Cetirizine', 10, NULL, NULL);



select * from patient_med;



Error Type	Buggy Code	Correction	Explanation
Missing commas	No commas between column definitions	Added commas between each column definition	SQL requires commas to separate columns in a `CREATE TABLE` statement
Missing NOT NULL	`MED_NAME VARCHAR2(80)`	`MED_NAME VARCHAR2(80) NOT NULL`	Ensures `MED_NAME` is mandatory
Malformed CHECK clause	`DOSE_MG NUMBER(6,2) CHECK DOSE_MG >= 0`	`DOSE_MG NUMBER(6,2) CHECK (DOSE_MG >= 0)`	CHECK constraints must be enclosed in parentheses
Invalid date logic	`CHECK (START_DT <= END_DT WHEN BOTH NOT NULL)`	`CHECK (START_DT IS NULL OR END_DT IS NULL OR START_DT <= END_DT)`	SQL doesn't support "WHEN BOTH NOT NULL"; use logical OR to allow NULLs
Missing NOT NULL on FK	`PATIENT_ID NUMBER REFERENCES PATIENT(ID)`	`PATIENT_ID NUMBER NOT NULL REFERENCES PATIENT(ID)`	Ensures foreign key is mandatory

Error Type	Buggy Code	Correction	Explanation

2. -- Main bill table
CREATE TABLE BILL (
  ID NUMBER PRIMARY
KEY,
  TOTAL NUMBER(12,2)
);

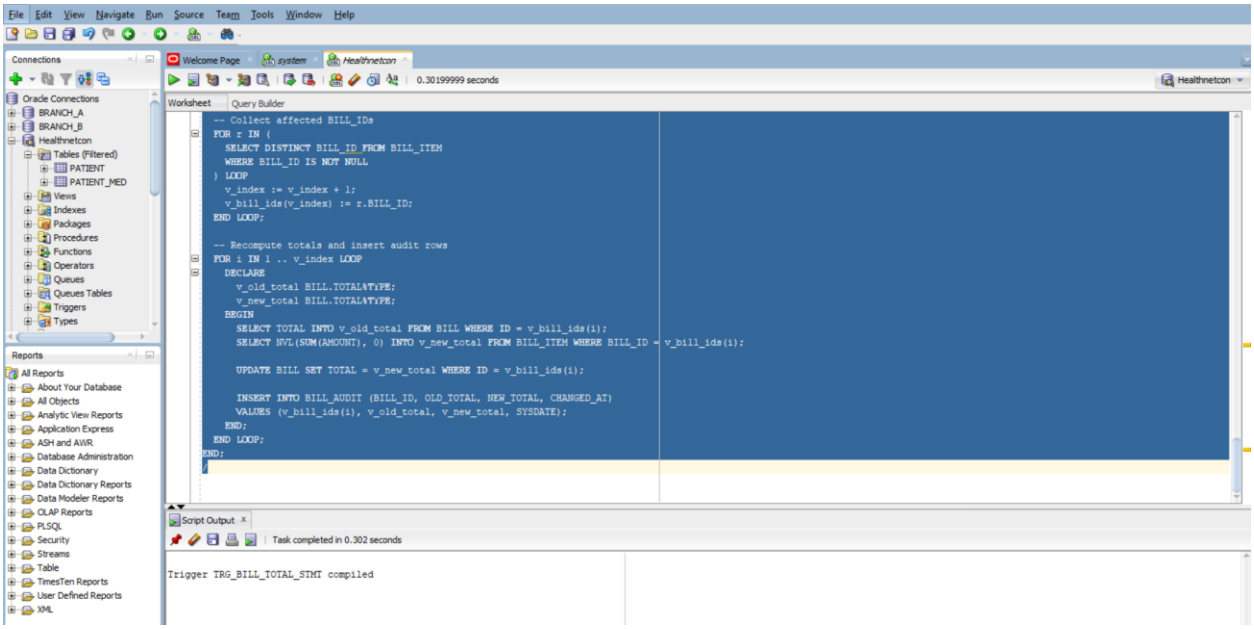
-- Items linked to bills
CREATE TABLE BILL_ITEM
(
  BILL_ID NUMBER,
  AMOUNT NUMBER(12,2),
  UPDATED_AT DATE,
  CONSTRAINT
FK_BILL_ITEM_BILL
FOREIGN KEY (BILL_ID)
REFERENCES BILL(ID)
);

-- Audit log for changes
CREATE TABLE
BILL_AUDIT (
  BILL_ID NUMBER,
  OLD_TOTAL
NUMBER(12,2),
  NEW_TOTAL
NUMBER(12,2),
  CHANGED_AT DATE
);

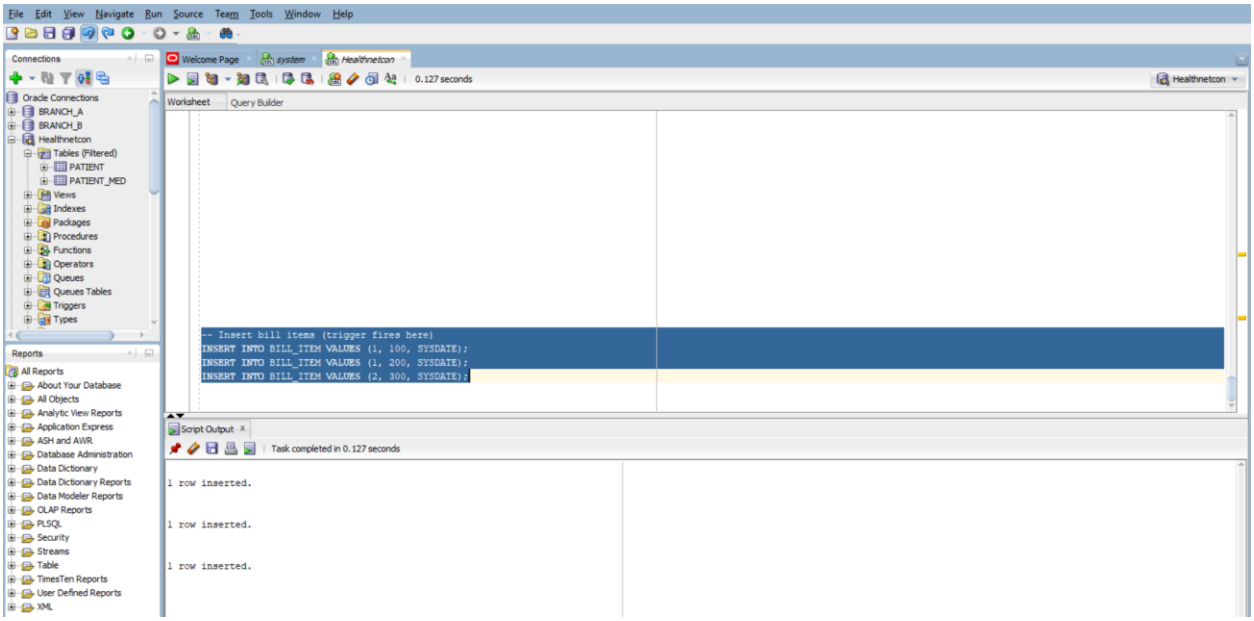
Correct Compound Trigger: `TRG_BILL_TOTAL_CMP`: it updates `BILL.TOTAL` once per statement and logs changes into `BILL_AUDIT`, avoiding mutating-table errors and redundant updates.

```
CREATE OR REPLACE TRIGGER TRG_BILL_TOTAL_STMT
AFTER INSERT OR UPDATE OR DELETE ON BILL_ITEM
DECLARE
  TYPE bill_id_table IS TABLE OF BILL_ITEM.BILL_ID%TYPE INDEX BY PLS_INTEGER;
  v_bill_ids bill_id_table;
  v_index PLS_INTEGER := 0;
BEGIN
  -- Collect affected BILL_IDs
  FOR r IN (
    SELECT DISTINCT BILL_ID FROM BILL_ITEM
    WHERE BILL_ID IS NOT NULL
  ) LOOP
    v_index := v_index + 1;
    v_bill_ids(v_index) := r.BILL_ID;
  END LOOP;

  -- Recompute totals and insert audit rows
  FOR i IN 1 .. v_index LOOP
    DECLARE
      v_old_total BILL.TOTAL%TYPE;
      v_new_total BILL.TOTAL%TYPE;
    BEGIN
      SELECT TOTAL INTO v_old_total FROM BILL WHERE ID = v_bill_ids(i);
      SELECT NVL(SUM(AMOUNT), 0) INTO v_new_total FROM BILL_ITEM WHERE BILL_ID = v_bill_ids(i);
```

```
    UPDATE BILL SET TOTAL = v_new_total WHERE ID = v_bill_ids(i);


    INSERT INTO BILL_AUDIT (BILL_ID, OLD_TOTAL, NEW_TOTAL, CHANGED_AT)
    VALUES (v_bill_ids(i), v_old_total, v_new_total, SYSDATE);
  END;
 END LOOP;
END;
/
```




```
-- Insert bill items (trigger fires here)
INSERT INTO BILL_ITEM VALUES (1, 100, SYSDATE);
INSERT INTO BILL_ITEM VALUES (1, 200, SYSDATE);
INSERT INTO BILL_ITEM VALUES (2, 300, SYSDATE);
```

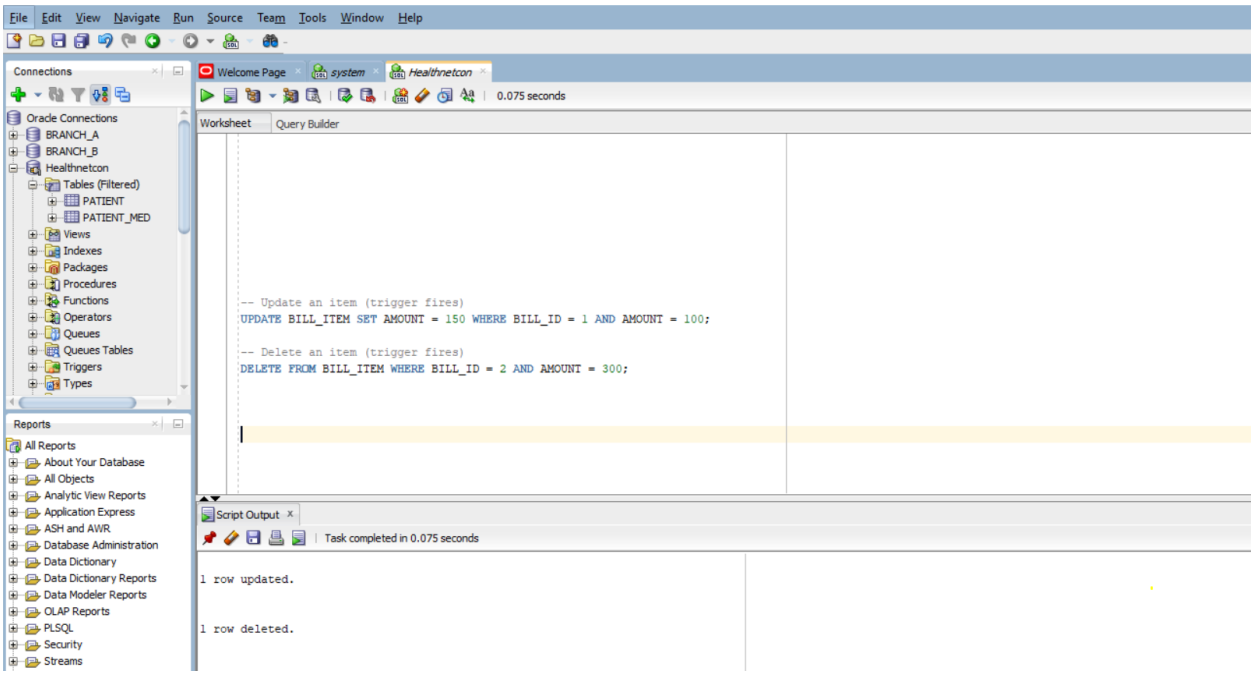


-- Update an item (trigger fires)

UPDATE BILL_ITEM SET AMOUNT = 150 WHERE BILL_ID = 1 AND AMOUNT = 100;

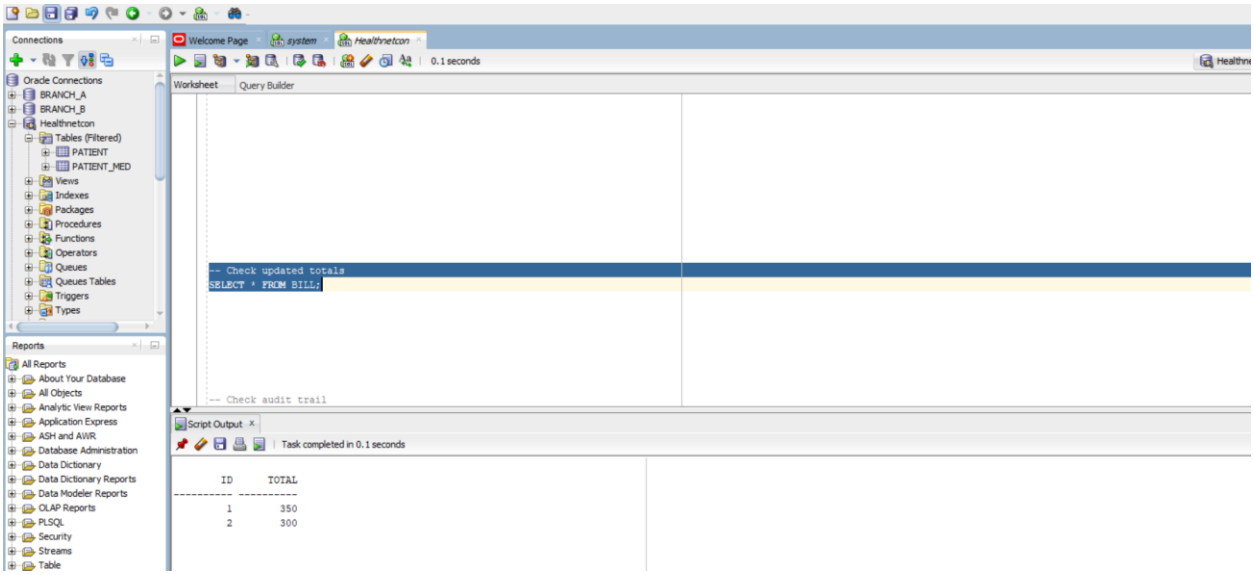
-- Delete an item (trigger fires)

DELETE FROM BILL_ITEM WHERE BILL_ID = 2 AND AMOUNT = 300;



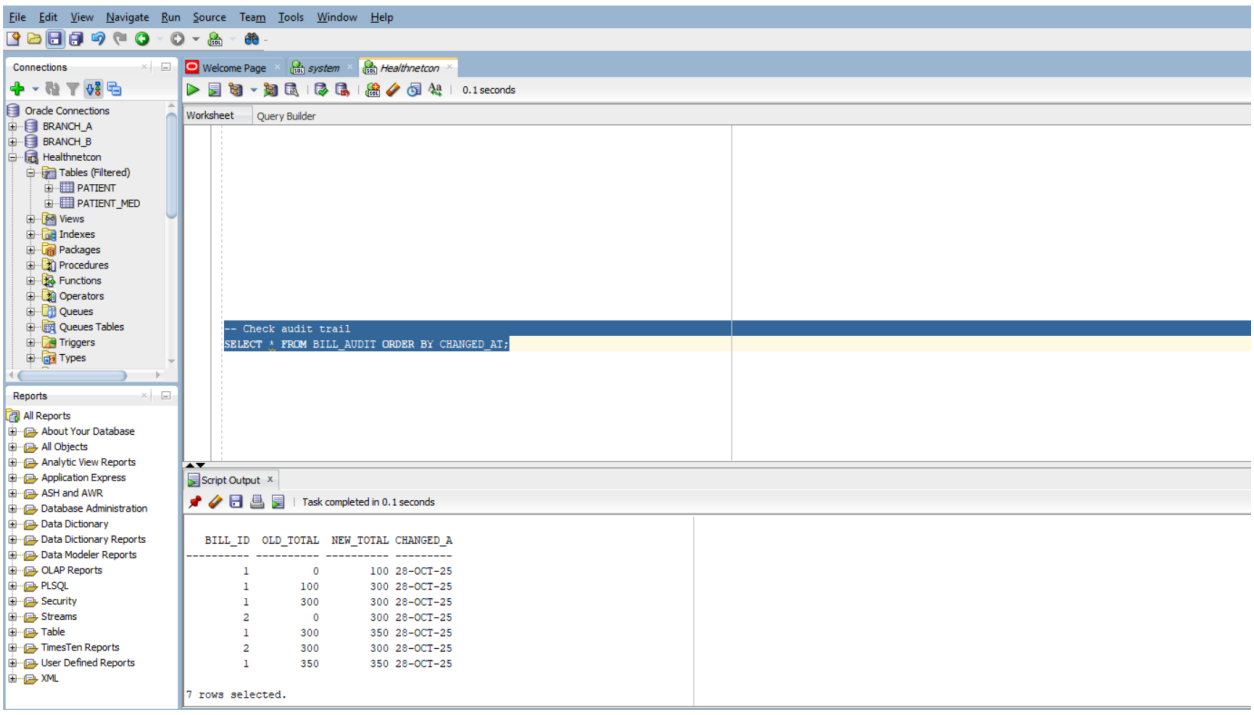
-- Check updated totals

SELECT * FROM BILL;



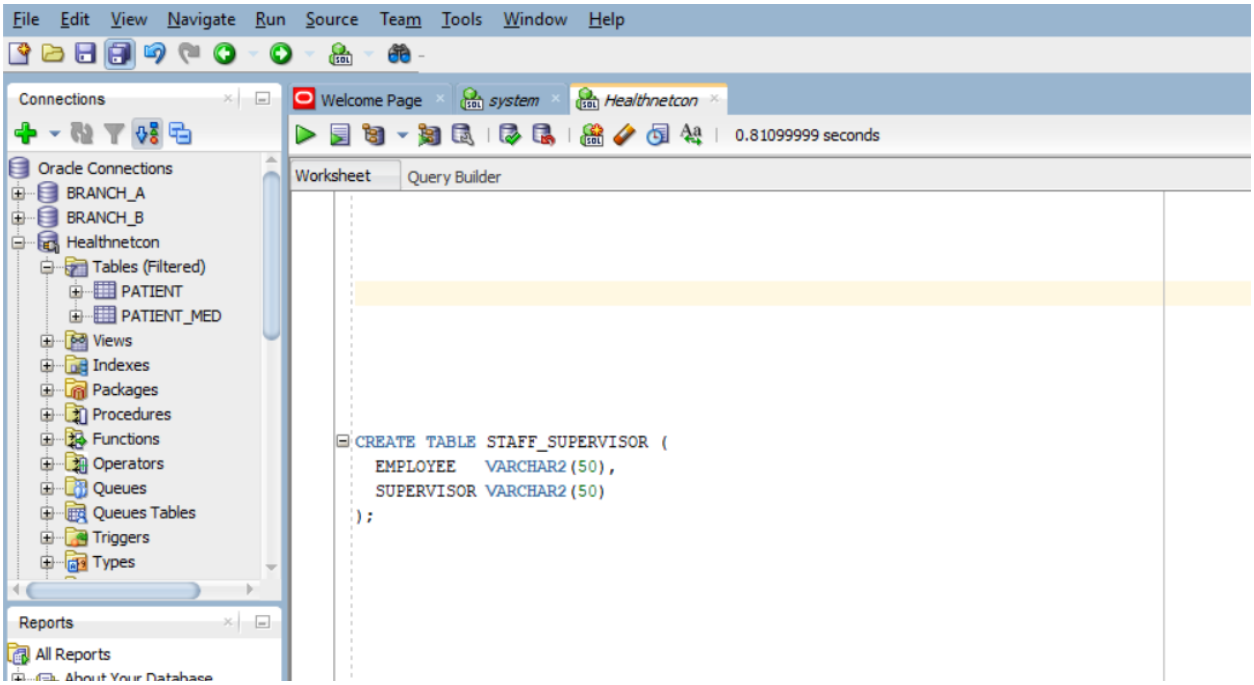
-- Check audit trail

SELECT * FROM BILL_AUDIT ORDER BY CHANGED_AT;

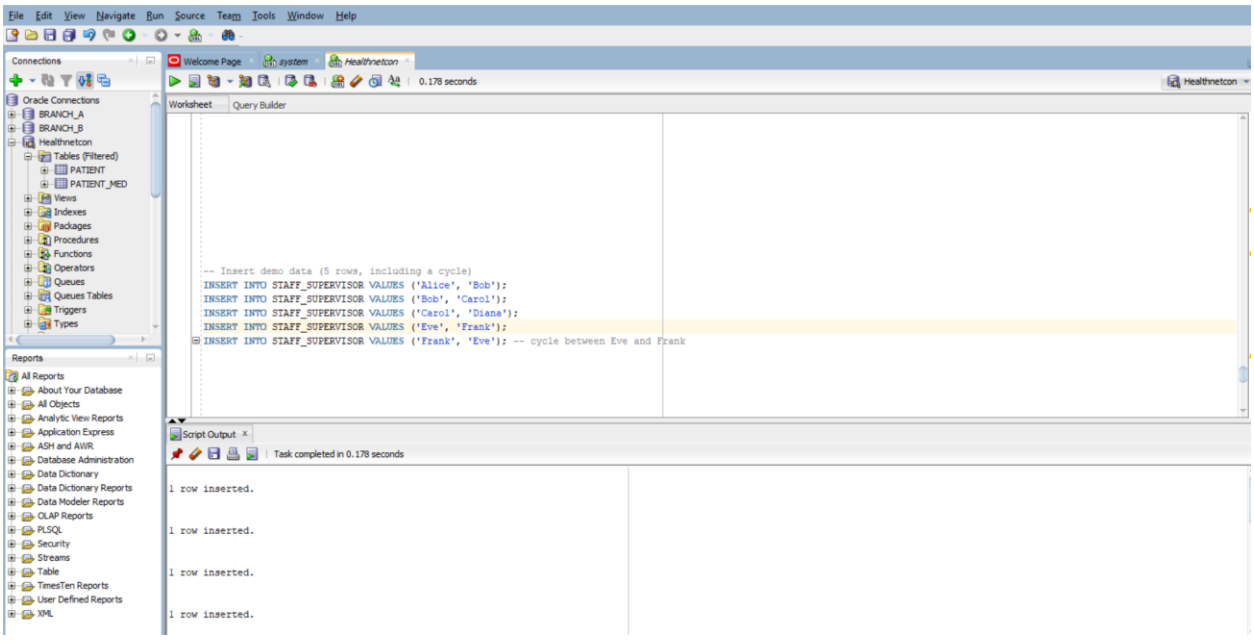


- `BILL.TOTAL` for ID 1 should reflect the sum of its items (e.g., 150 + 200 = 350).
- `BILL.TOTAL` for ID 2 should be 0 after deletion.
- `BILL_AUDIT` should show old and new totals for each change.

3.

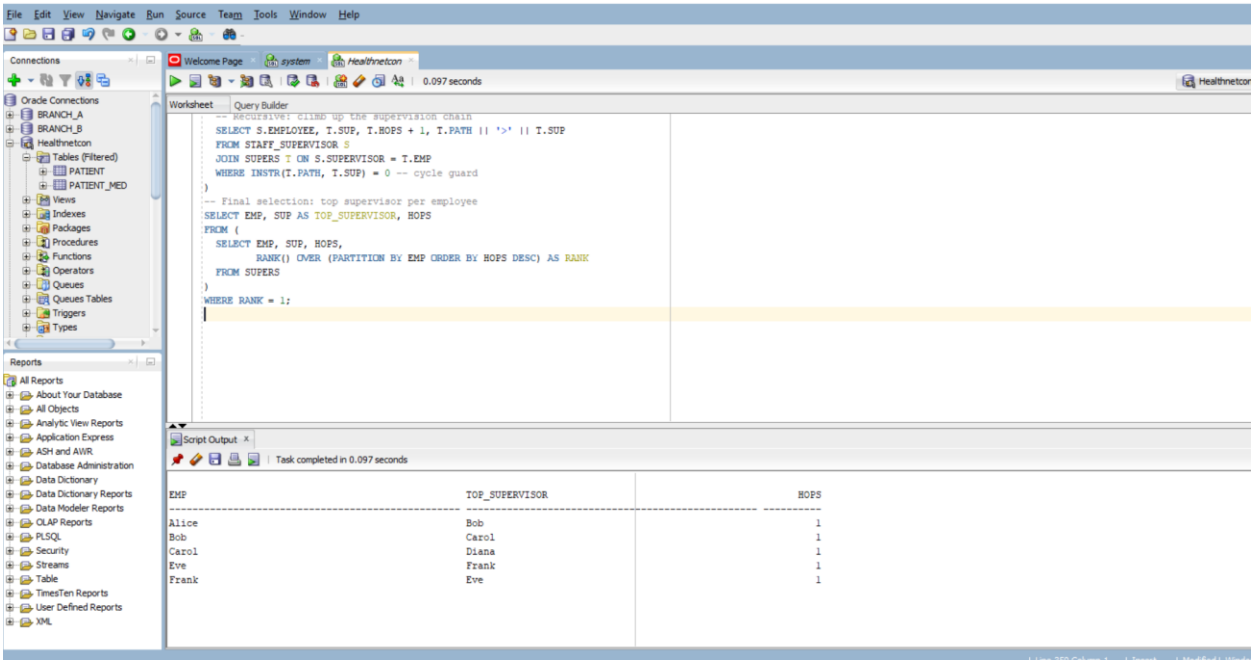


Inserting rows



-- Corrected recursive query

```
WITH SUPERS (EMP, SUP, HOPS, PATH) AS (
  -- Anchor: start with direct supervision, hop count = 1
  SELECT EMPLOYEE, SUPERVISOR, 1, EMPLOYEE || '>' || SUPERVISOR
  FROM STAFF_SUPERVISOR
  UNION ALL
  -- Recursive: climb up the supervision chain
  SELECT S.EMPLOYEE, T.SUP, T.HOPS + 1, T.PATH || '>' || T.SUP
  FROM STAFF_SUPERVISOR S
  JOIN SUPERS T ON S.SUPERVISOR = T.EMP
  WHERE INSTR(T.PATH, T.SUP) = 0 -- cycle guard
)
-- Final selection: top supervisor per employee
SELECT EMP, SUP AS TOP_SUPERVISOR, HOPS
FROM (
  SELECT EMP, SUP, HOPS,
        RANK() OVER (PARTITION BY EMP ORDER BY HOPS DESC) AS RANK
  FROM SUPERS
)
WHERE RANK = 1;
```



Bug	Fix
Anchor hop count was `0`	Set to `1` to reflect first supervision step
Join direction was reversed	Corrected to climb up: `S.SUPERVISOR = T.EMP`
Cycle guard was naive	Improved with `INSTR(PATH, T.SUP) = 0`
Scalar subquery with `MAX(HOPS or` the **number of steps** it takes to reach an employee's **top supervisor** by following the chain of supervision)	Replaced with `RANK()` analytic function for clarity and correctness

Diana

└— Carol

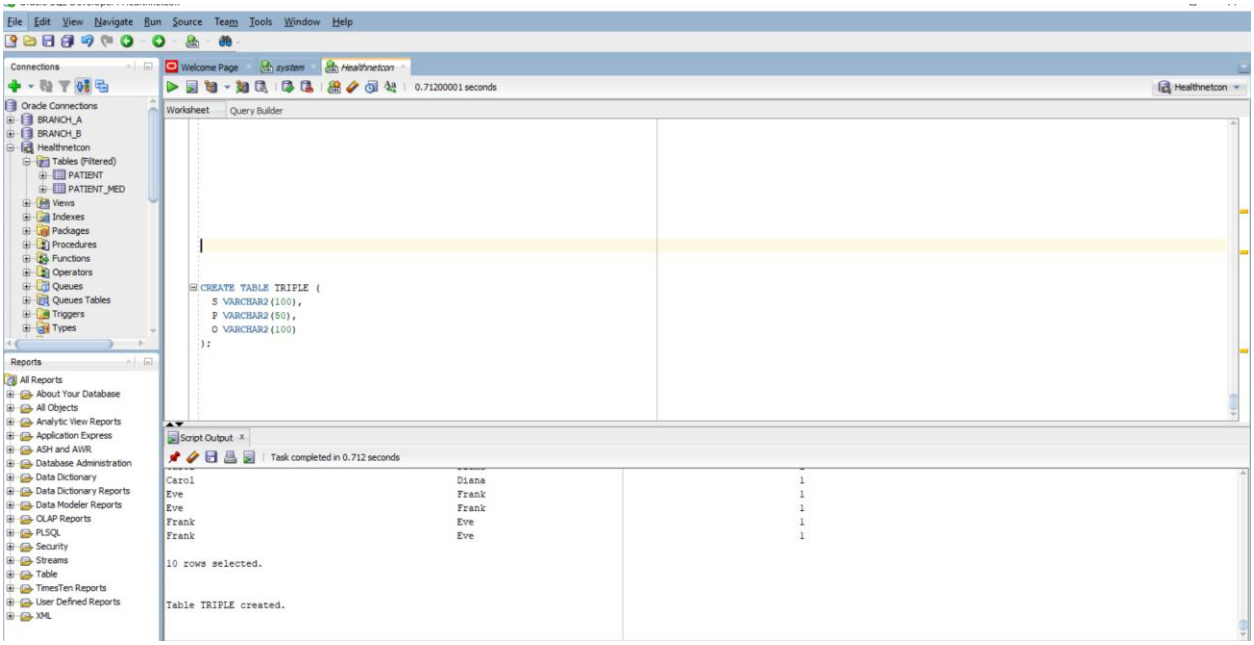
└— Bob

└— Alice

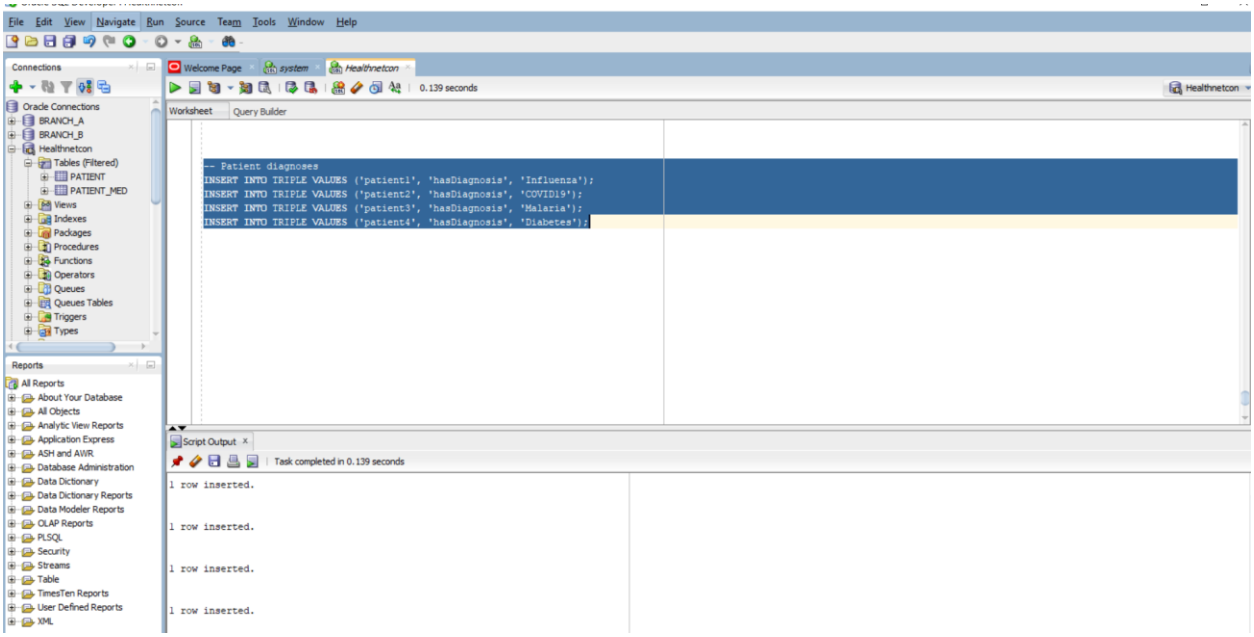
Eve ↔ Frank (cycle)

4.

```
CREATE TABLE TRIPLE (
  S VARCHAR2(100),
  P VARCHAR2(50),
  O VARCHAR2(100)
);
```




```
-- Patient diagnoses
INSERT INTO TRIPLE VALUES ('patient1', 'hasDiagnosis', 'Influenza');
INSERT INTO TRIPLE VALUES ('patient2', 'hasDiagnosis', 'COVID19');
INSERT INTO TRIPLE VALUES ('patient3', 'hasDiagnosis', 'Malaria');
INSERT INTO TRIPLE VALUES ('patient4', 'hasDiagnosis', 'Diabetes');
```



-- Taxonomy edges

INSERT INTO TRIPLE VALUES ('Influenza', 'isA', 'ViralInfection');

INSERT INTO TRIPLE VALUES ('COVID19', 'isA', 'ViralInfection');

INSERT INTO TRIPLE VALUES ('Malaria', 'isA', 'ParasiticInfection');

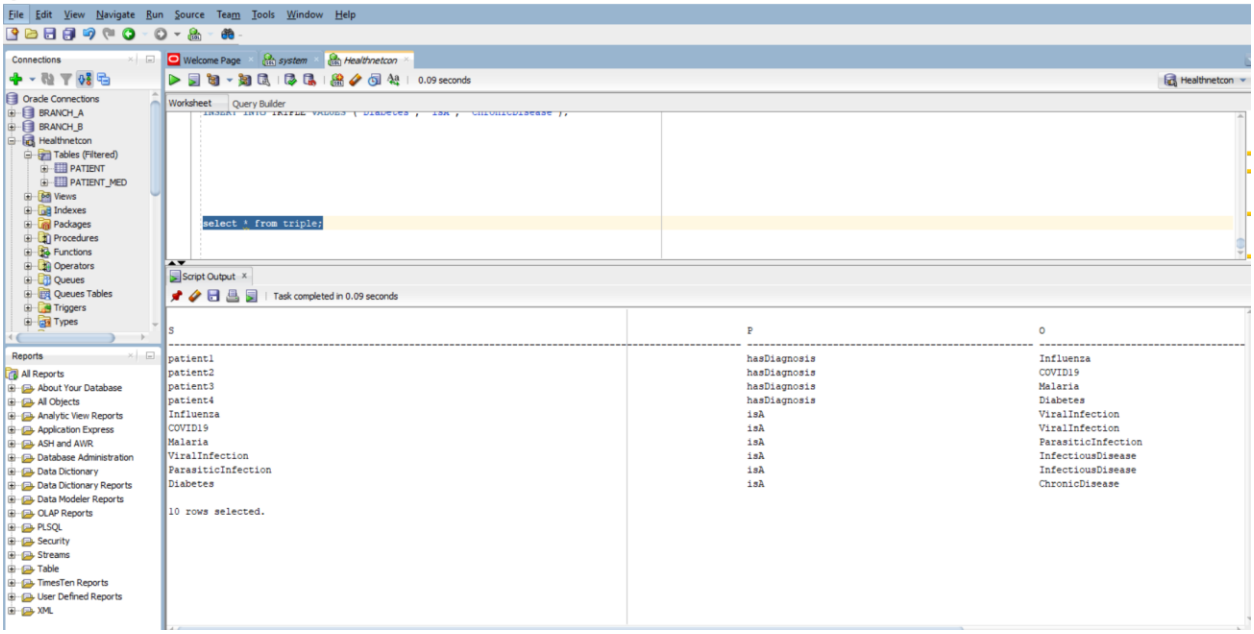
INSERT INTO TRIPLE VALUES ('ViralInfection', 'isA', 'InfectiousDisease');

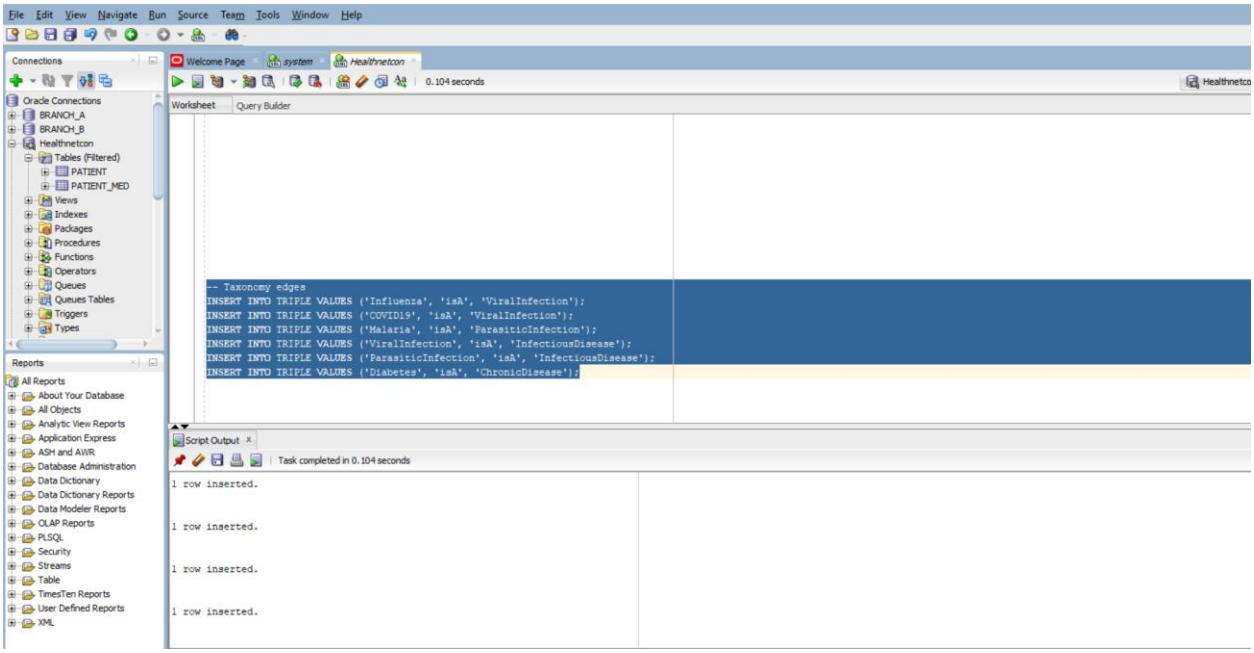
INSERT INTO TRIPLE VALUES ('ParasiticInfection', 'isA', 'InfectiousDisease');

INSERT INTO TRIPLE VALUES ('Diabetes', 'isA', 'ChronicDisease');

Check inserted rows;

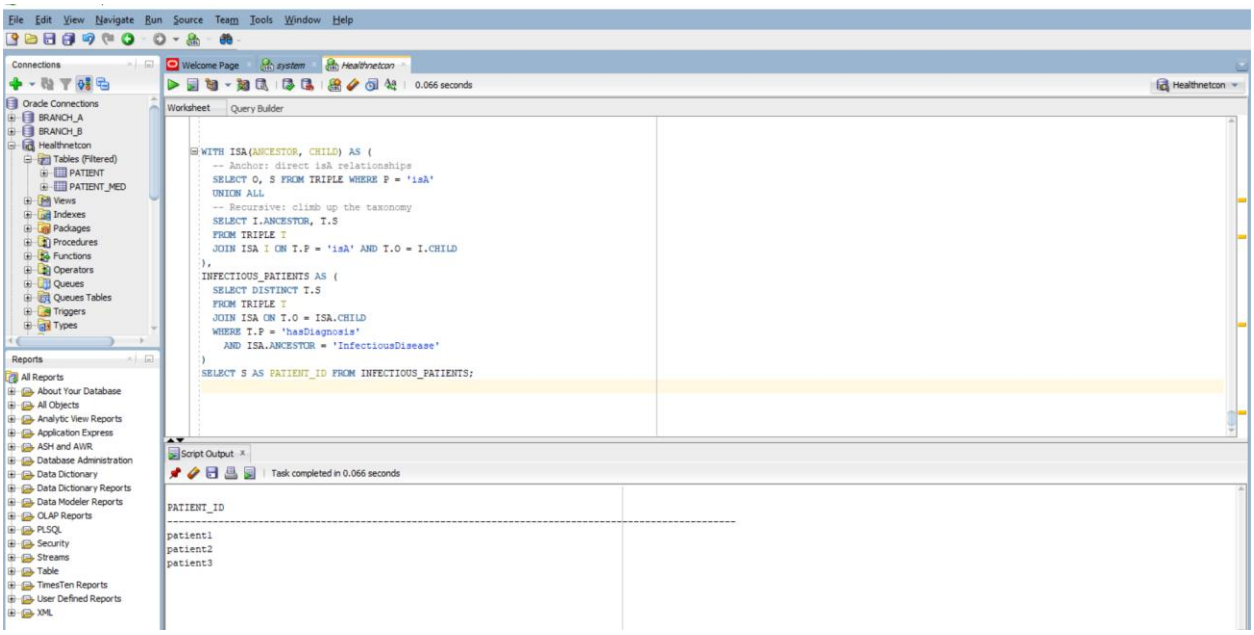
select * from triple;






```
WITH ISA(ANCESTOR, CHILD) AS (
  -- Anchor: direct isA relationships
  SELECT O, S FROM TRIPLE WHERE P = 'isA'
  UNION ALL
```

```
  -- Recursive: climb up the taxonomy
  SELECT I.ANCESTOR, T.S
  FROM TRIPLE T
  JOIN ISA I ON T.P = 'isA' AND T.O = I.CHILD
),
INFECTIOUS_PATIENTS AS (
  SELECT DISTINCT T.S
  FROM TRIPLE T
  JOIN ISA ON T.O = ISA.CHILD
  WHERE T.P = 'hasDiagnosis'
    AND ISA.ANCESTOR = 'InfectiousDisease'
)
SELECT S AS PATIENT_ID FROM INFECTIOUS_PATIENTS;
```



- Represent facts in a flexible, searchable format

- Link concepts together (like diseases to categories)

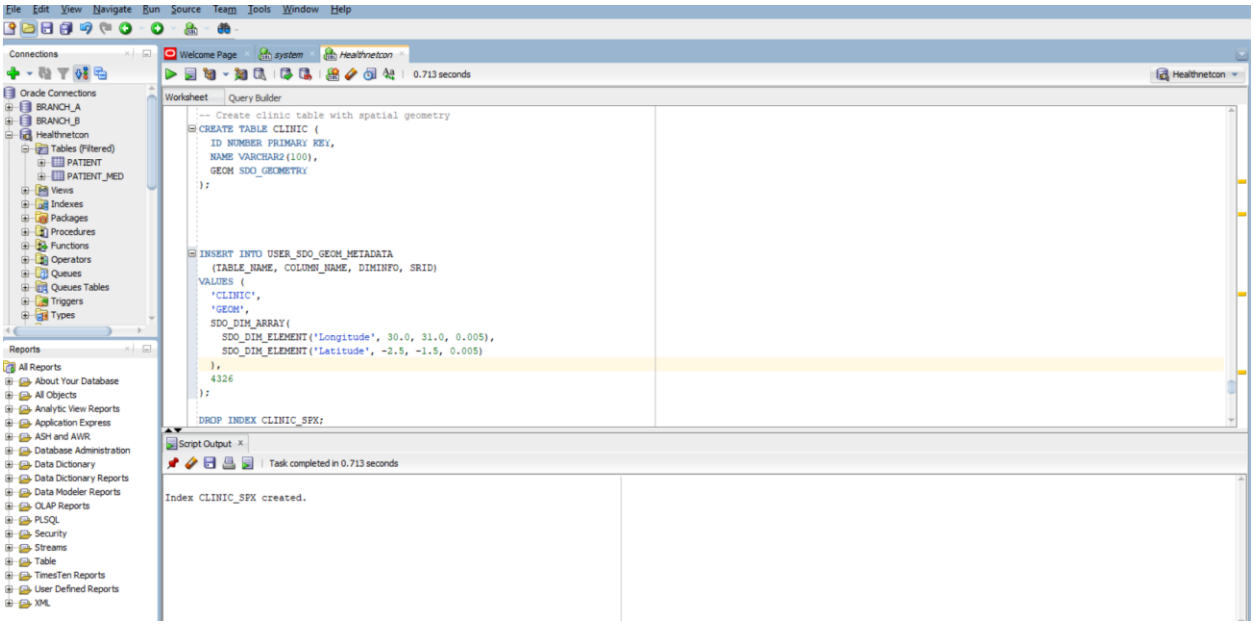
- Enable reasoning and inference (e.g., if Influenza is an InfectiousDisease, then patient1 has an InfectiousDisease)

5.

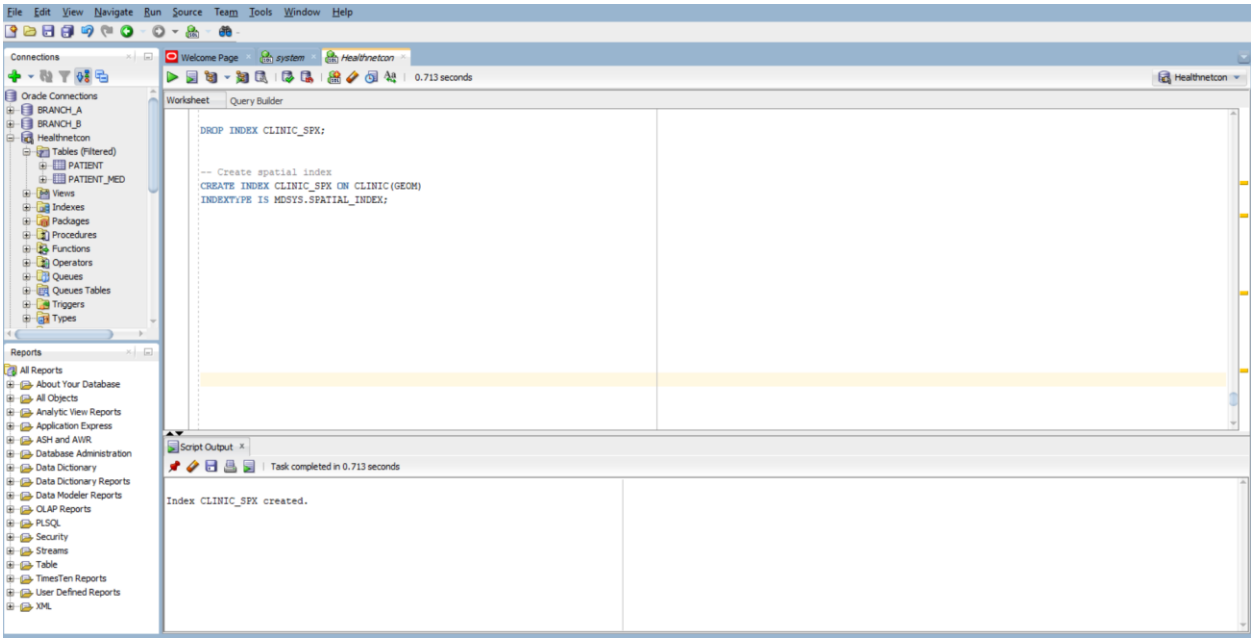
```
-- Create clinic table with spatial geometry
CREATE TABLE CLINIC (
  ID NUMBER PRIMARY KEY,
  NAME VARCHAR2(100),
  GEOM SDO_GEOMETRY
);
```

```
INSERT INTO USER_SDO_GEOM_METADATA
  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES (
  'CLINIC',
  'GEOM',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('Longitude', 30.0, 31.0, 0.005),
```

```
  SDO_DIM_ELEMENT('Latitude', -2.5, -1.5, 0.005)
 ),
 4326
);
```




```
-- Create spatial index
CREATE INDEX CLINIC_SPX ON CLINIC(GEOM)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```



-- Ambulance is at (30.0600, -1.9570)

INSERT INTO CLINIC VALUES (

1, 'Kigali Central Clinic',

SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0610, -1.9575, NULL), NULL, NULL)

);

INSERT INTO CLINIC VALUES (

2, 'Nyamirambo Health Center',

SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0595, -1.9560, NULL), NULL, NULL)

);

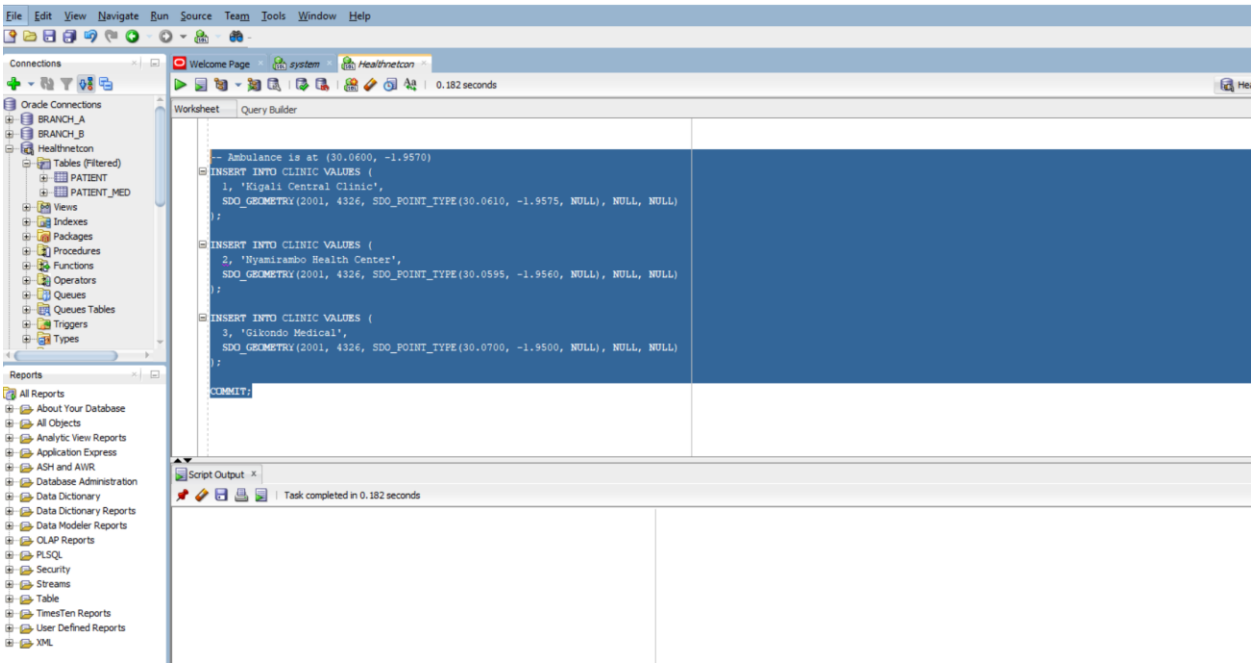
INSERT INTO CLINIC VALUES (

3, 'Gikondo Medical',

SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0700, -1.9500, NULL), NULL, NULL)

);

COMMIT;



SELECT C.ID, C.NAME

FROM CLINIC C

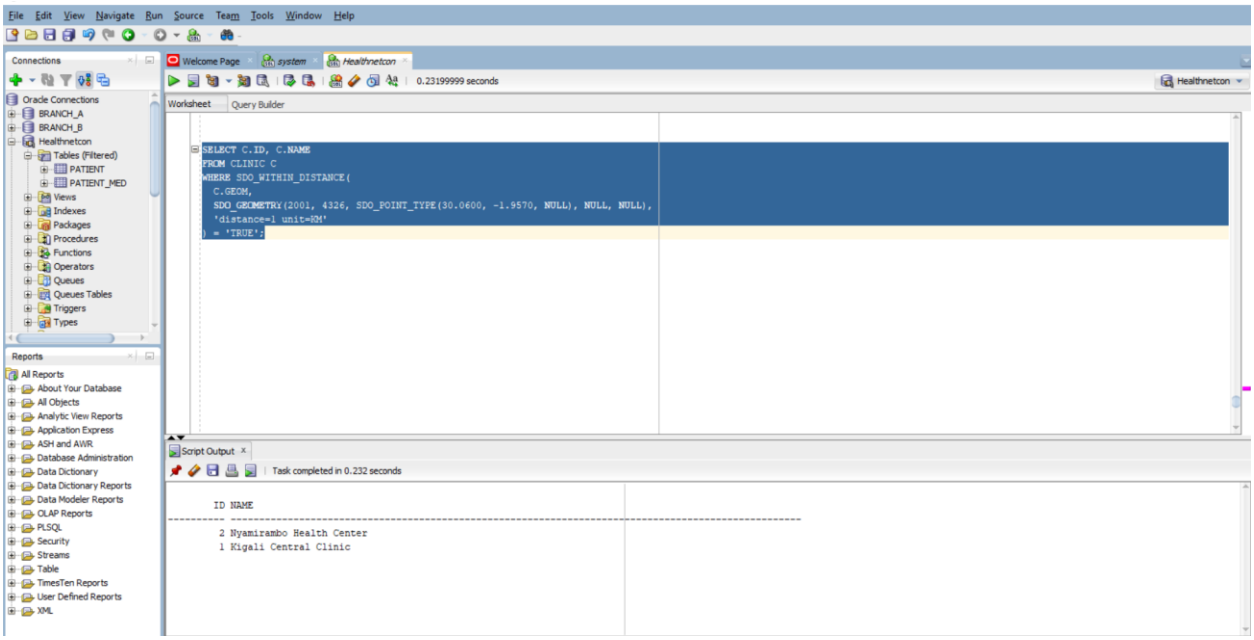
WHERE SDO_WITHIN_DISTANCE(

C.GEOM,

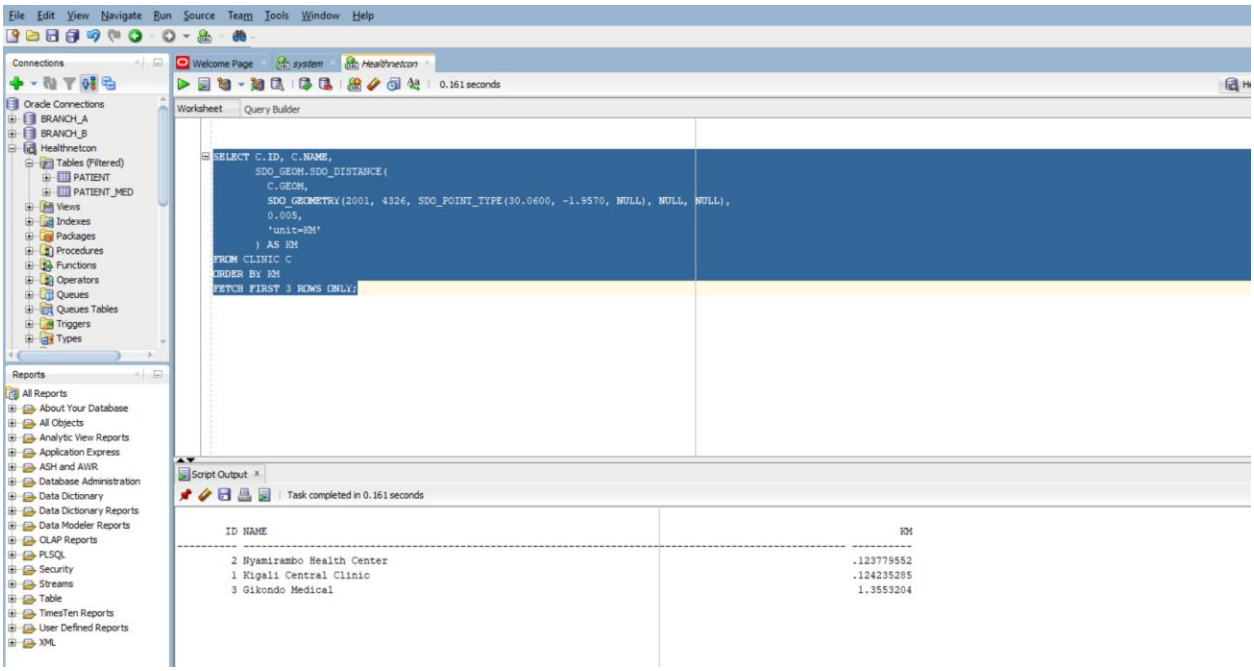
SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0600, -1.9570, NULL), NULL, NULL),

'distance=1 unit=KM'

) = 'TRUE';




```
SELECT C.ID, C.NAME,
      SDO_GEOM.SDO_DISTANCE(
        C.GEOM,
        SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0600, -1.9570, NULL), NULL, NULL),
        0.005,
        'unit=KM'
      ) AS KM
FROM CLINIC C
ORDER BY KM
FETCH FIRST 3 ROWS ONLY;
```


File Edit View Navigate Run Source Team Tools Window Help
Connections
Oracle Connections
BRANCH_A
BRANCH_B
Healthnetcon
Tables (Filtered)
PATIENT
PATIENT_MED
Views
Indexes
Packages
Procedures
Functions
Operators
Queues
Queues Tables
Triggers
Types
Reports
All Reports
About Your Database
All Objects
Analytic View Reports
Application Express
ASH and AWR
Database Administration
Data Dictionary
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
PLSQL
Security
Streams
Table
TimesTen Reports
User Defined Reports
XML
Welcome Page
system
Healthnetcon
0.161 seconds
Worksheet
Query Builder
SELECT C.ID, C.NAME,
SDO_GEOM.SDO_DISTANCE(
C.GEOM,
SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0600, -1.9570, NULL), NULL, NULL),
0.005,
'unit=KM'
) AS KM
FROM CLINIC C
ORDER BY KM
FETCH FIRST 3 ROWS ONLY;
Script Output
Task completed in 0.161 seconds
ID NAME KM
2 Nyamirambo Health Center .123779552
1 Kigali Central Clinic .124235285
3 Gikondo Medical 1.3553204