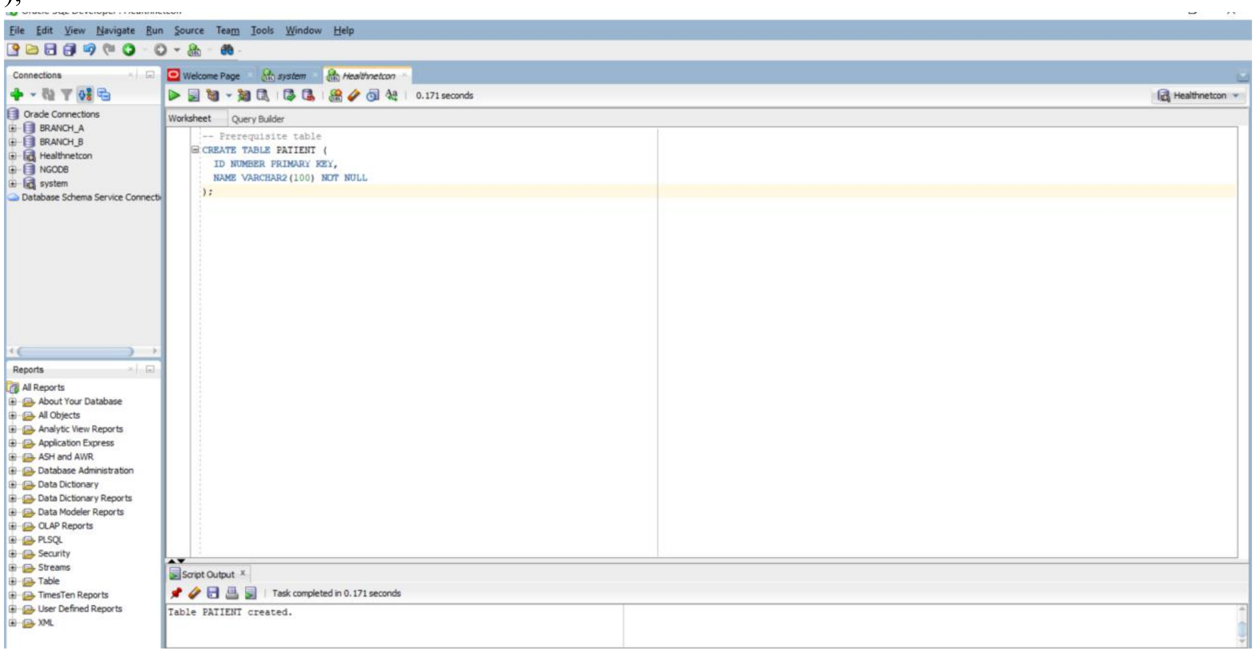
QUESTION 1

bugs

```
--1-Missing commas between columns:


--PATIENT_MED_ID NUMBER PRIMARY KEY -- missing comma after this line

--PATIENT_ID NUMBER REFERENCES PATIENT(ID)

--2-MED_NAME should be NOT NULL (currently optional).


--3-DOSE_MG CHECK syntax is wrong. It should be:

--DOSE_MG NUMBER(6,2) CHECK (DOSE_MG >= 0)

--4 CK_RX_DATES CHECK clause syntax is invalid:

---CHECK (START_DT <= END_DT WHEN BOTH NOT NULL) -- wrong

---CHECK (START_DT IS NULL OR END_DT IS NULL OR START_DT <= END_DT)

---


-- Use schema

ALTER SESSION SET CURRENT_SCHEMA = HEALTHNET;



-- Prerequisite table

CREATE TABLE PATIENT (

  ID NUMBER PRIMARY KEY,

  NAME VARCHAR2(100) NOT NULL
```

);

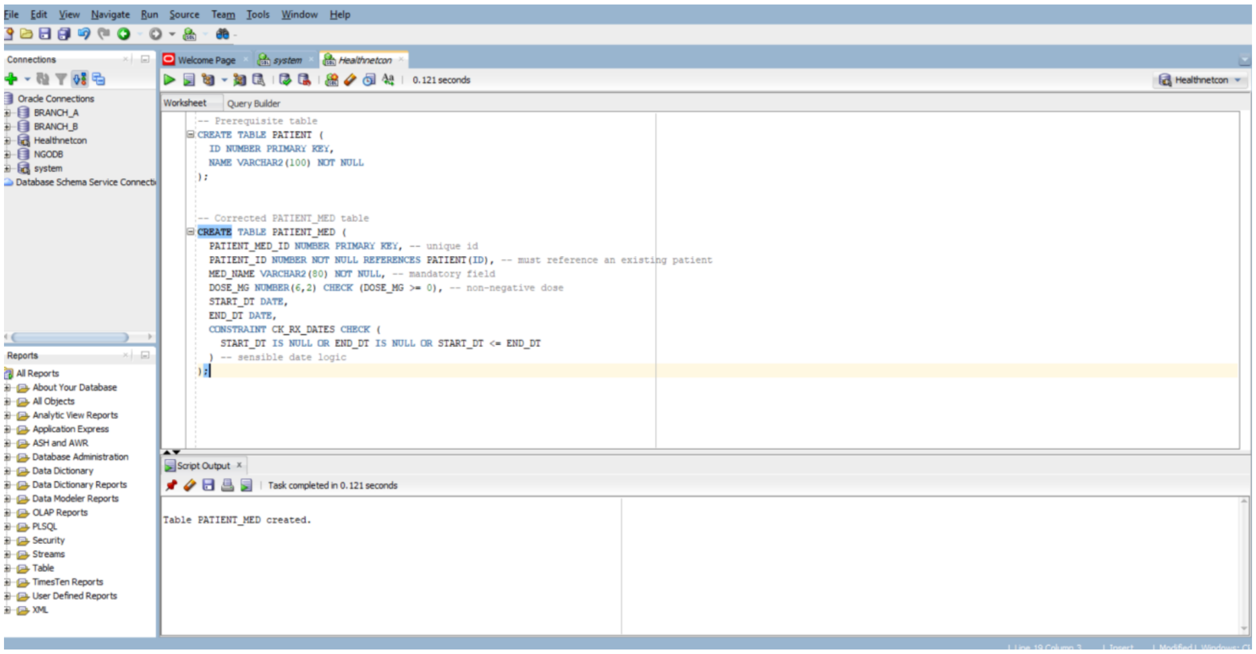


```
-- Prerequisite table
CREATE TABLE PATIENT (
  ID NUMBER PRIMARY KEY,
  NAME VARCHAR2(100) NOT NULL
);
```

```
Table PATIENT created.
```

-- 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
);
```



```sql
-- 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
);
```

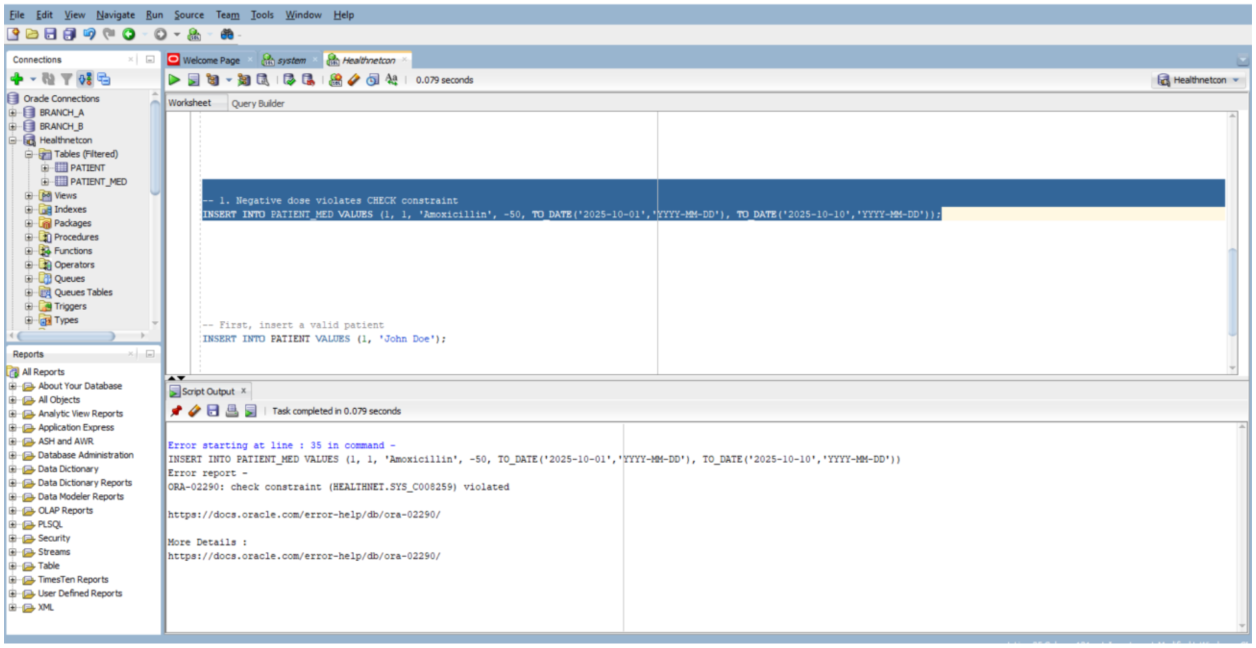
Task completed in 0.121 seconds

```
Table PATIENT_MED created.
```

Failing Inserts

-- 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'));



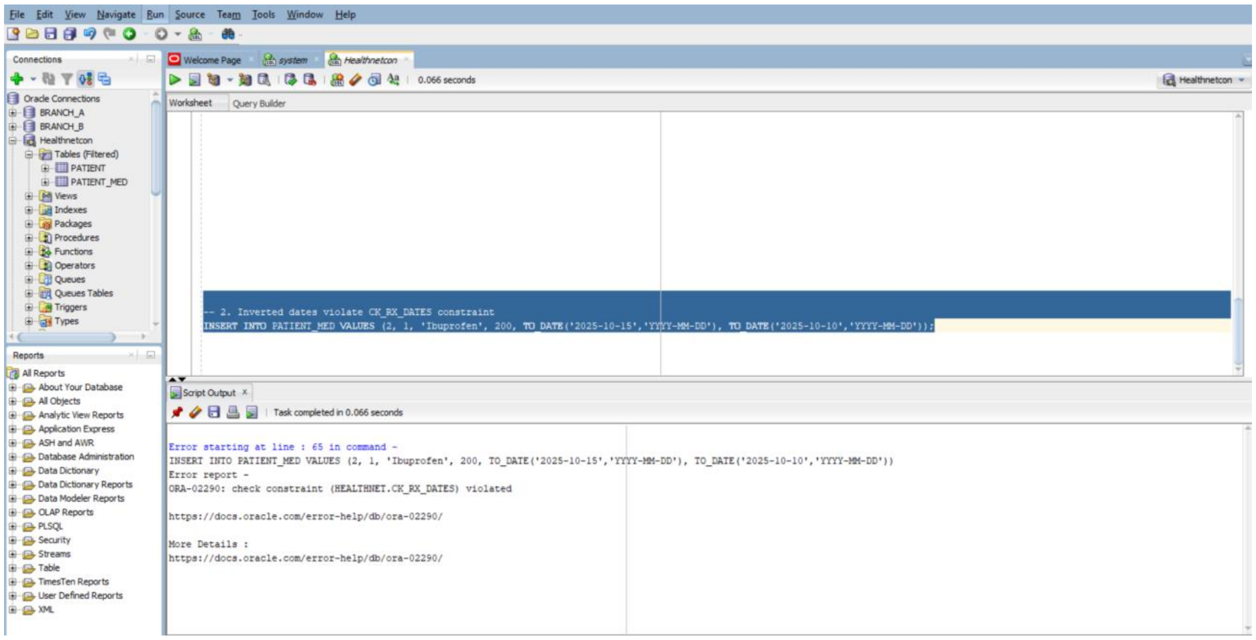
```
-- 1. Negative dose violates CHECK constraint
INSERT INTO PATIENT_MED VALUES (1, 1, 'Amoxicillin', -50, TO_DATE('2025-10-01','YYYY-MM-DD'), TO_DATE('2025-10-10','YYYY-MM-DD'));



-- First, insert a valid patient
INSERT INTO PATIENT VALUES (1, 'John Doe');
```

Task completed in 0.079 seconds

```
Error starting at line : 35 in command -
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'))
Error report -
ORA-02290: check constraint (HEALTHNET.SYS_C008259) violated

https://docs.oracle.com/error-help/db/ora-02290/

More Details :
https://docs.oracle.com/error-help/db/ora-02290/
```

-- 2. Inverted dates violate CK_RX_DATES constraint

INSERT INTO PATIENT_MED VALUES (2, 1, 'Ibuprofen', 200, TO_DATE(' 2025-12-31’,'YYYY-MM-DD'), TO_DATE('2025-10-12','YYYY-MM-DD'));



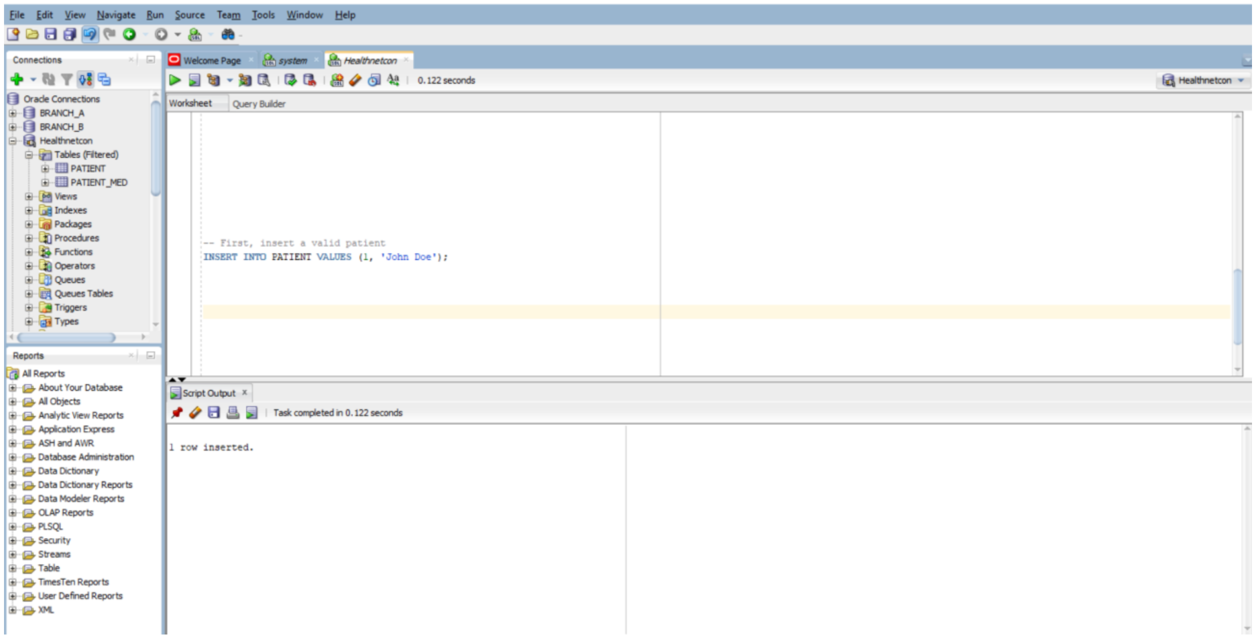
```
-- 2. Inverted dates violate CK_RX_DATES constraint
INSERT INTO PATIENT_MED VALUES (2, 1, 'Ibuprofen', 200, TO_DATE('2025-10-15','YYYY-MM-DD'), TO_DATE('2025-10-10','YYYY-MM-DD'));
```

Task completed in 0.066 seconds

```
Error starting at line : 65 in command -
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'))
Error report -
ORA-02290: check constraint (HEALTHNET.CK_RX_DATES) violated

https://docs.oracle.com/error-help/db/ora-02290/

More Details :
https://docs.oracle.com/error-help/db/ora-02290/
```

-- First, insert a valid patient

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

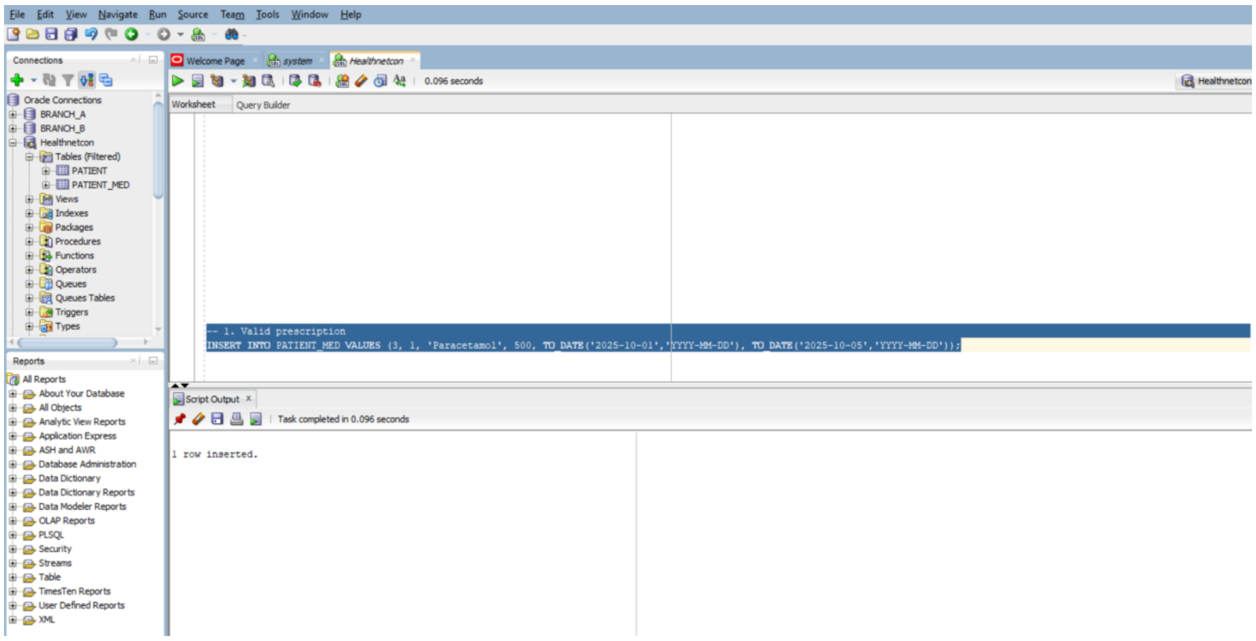


```
-- First, insert a valid patient
INSERT INTO PATIENT VALUES (1, 'John Doe');
```

```
1 row inserted.
```

-- 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'));



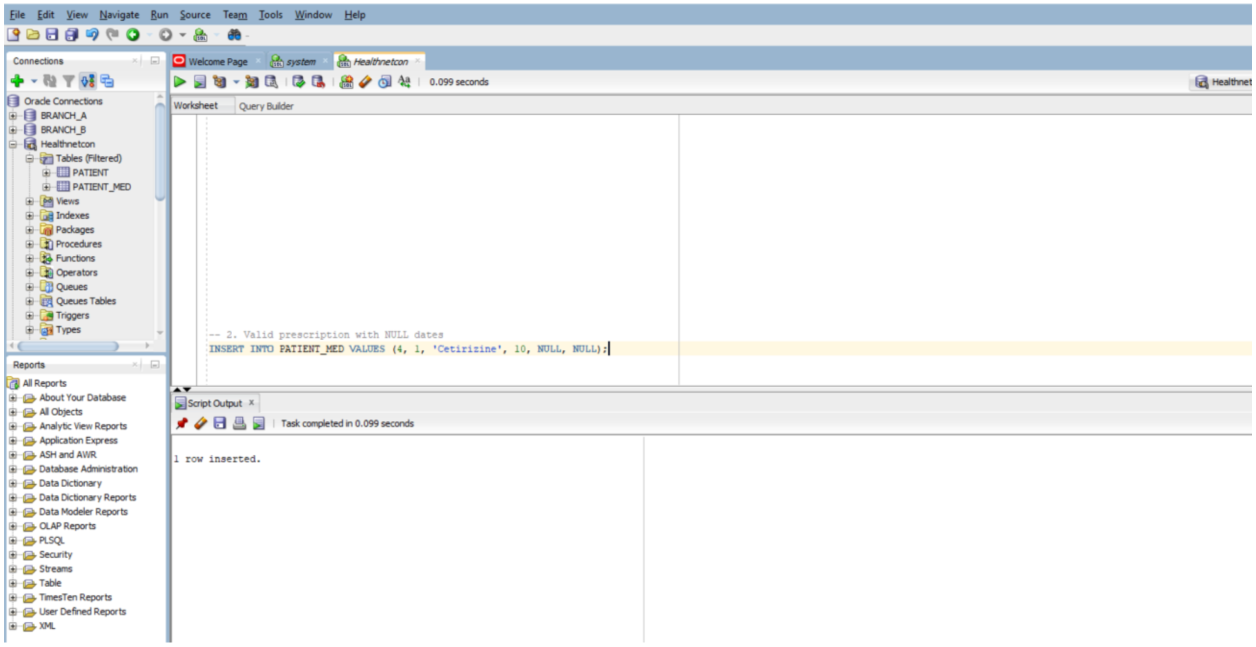
```
-- 1. Valid prescription
INSERT INTO PATIENT_MED VALUES (3, 1, 'Paracetamol', 500, TO_DATE('2025-10-01','YYYY-MM-DD'), TO_DATE('2025-10-05','YYYY-MM-DD'));
```

Task completed in 0.096 seconds

```
1 row inserted.
```

-- 2. Valid prescription with NULL dates

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

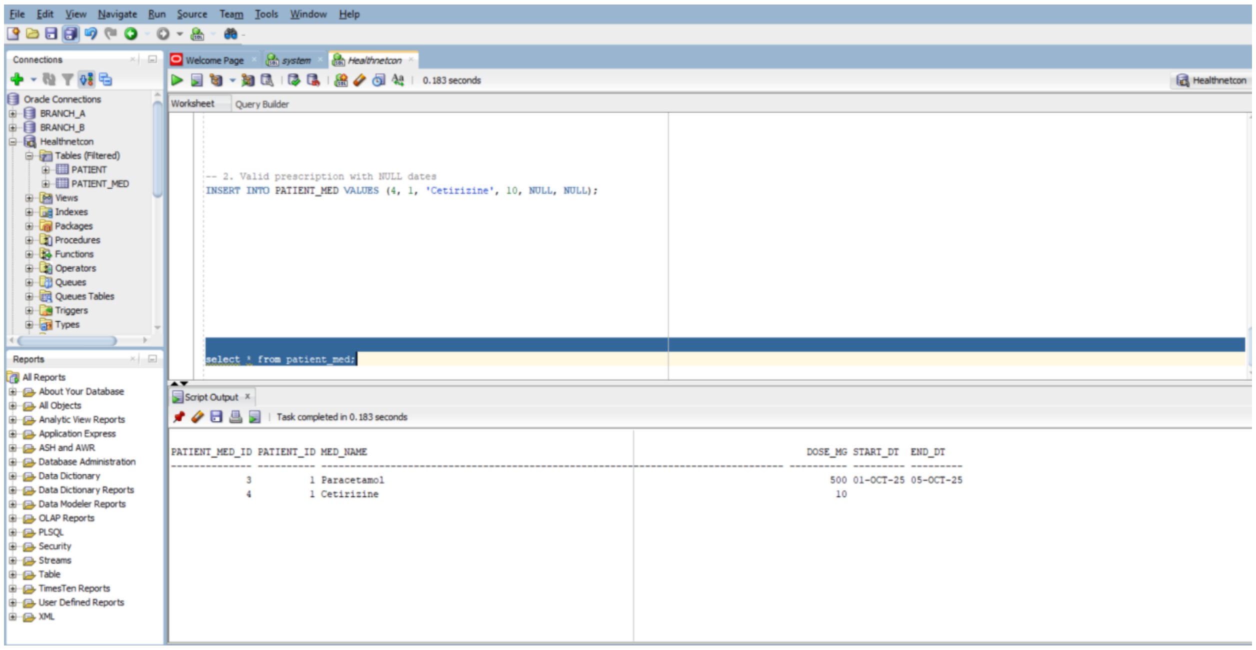


```
-- 2. Valid prescription with NULL dates
INSERT INTO PATIENT_MED VALUES (4, 1, 'Cetirizine', 10, NULL, NULL);
```

Task completed in 0.099 seconds

```
1 row inserted.
```

select * from patient_med;



```
-- 2. Valid prescription with NULL dates
INSERT INTO PATIENT_MED VALUES (4, 1, 'Cetirizine', 10, NULL, NULL);

select * from patient_med;
```

Task completed in 0.183 seconds

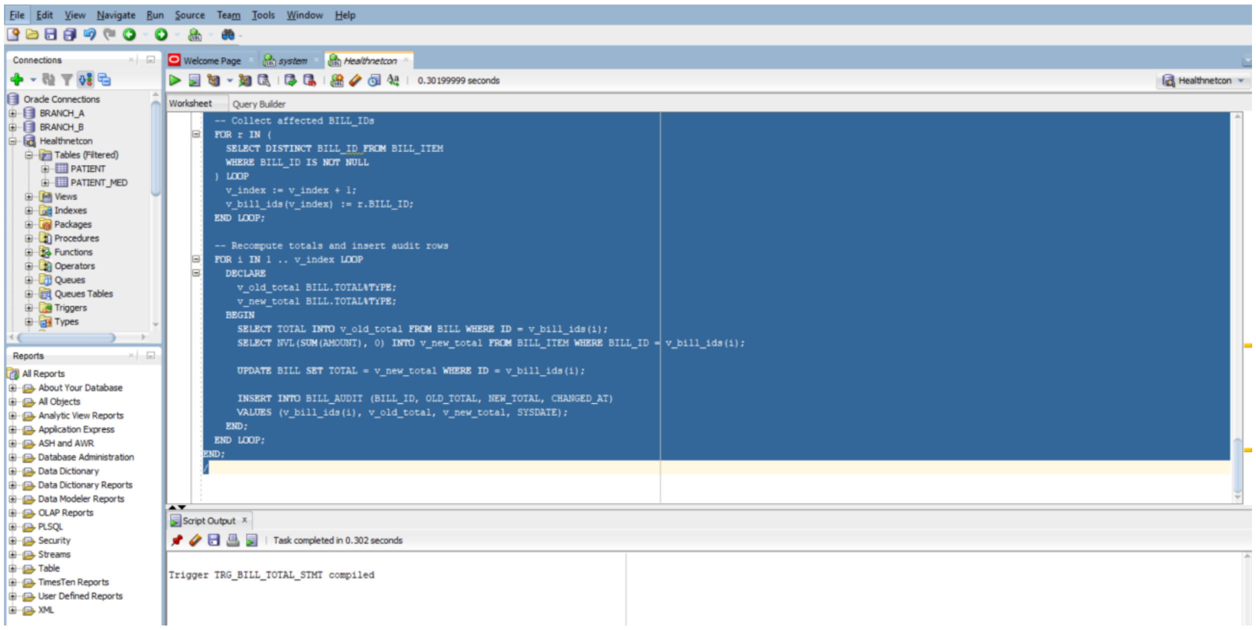
```
PATIENT_MED_ID PATIENT_ID MED_NAME                                                                            DOSE_MG START_DT  END_DT
-------------- ---------- -------------------------------------------------------------------------------- ---------- --------- ---------
             3          1 Paracetamol                                                                             500 01-OCT-25 05-OCT-25
             4          1 Cetirizine                                                                               10
```

Buggy Code	Correction	Explanation
No commas between column definitions	Added commas between each column definition	SQL requires commas to separate columns in a CREATE TABLE statement
MED_NAME VARCHAR2(80)	MED_NAME VARCHAR2(80) NOT NULL	Ensures MED_NAME is mandatory
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
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
PATIENT_ID NUMBER REFERENCES PATIENT(ID)	PATIENT_ID NUMBER NOT NULL REFERENCES PATIENT(ID)	Ensures foreign key is mandatory

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;
/
```



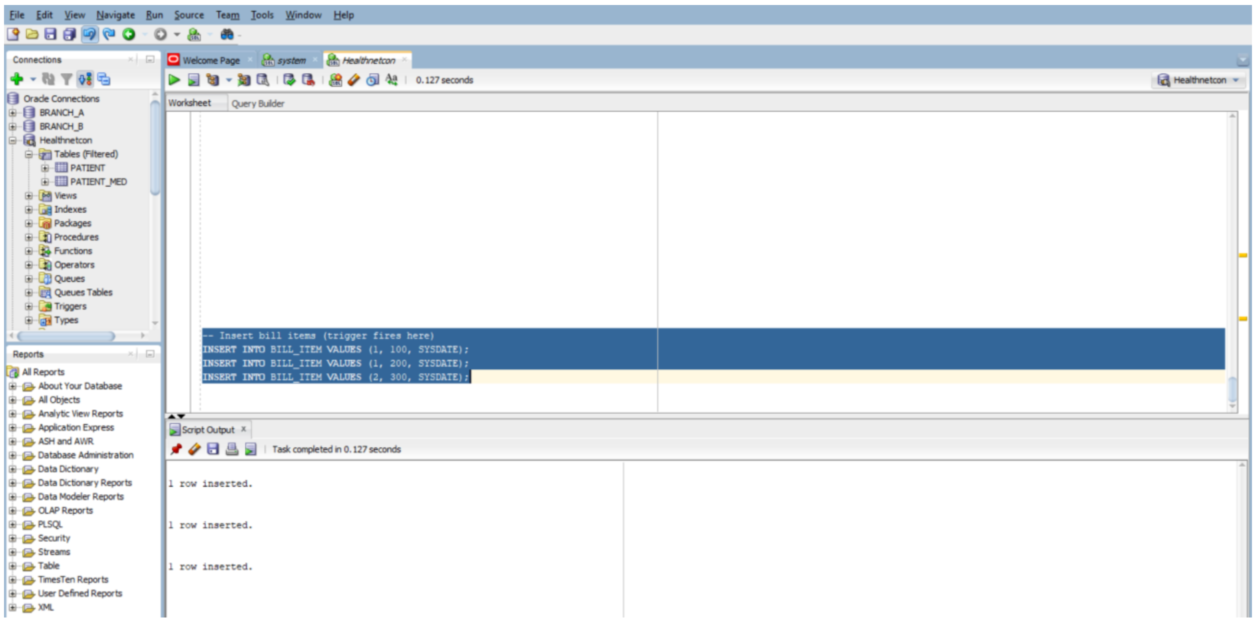
```
  -- Collect affected BILL_IDs
  FOR r IN (
    SELECT DISTINCT BILL_ID FROM BILL_ITEM
    WHERE BILL_ID IS NOT NULL
  ) LOOP
    v_index := v_index + 1;
    v_bill_ids(v_index) := r.BILL_ID;
  END LOOP;

  -- 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;
/
```

Task completed in 0.302 seconds

```
Trigger TRG_BILL_TOTAL_STMT compiled
```

-- 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);
```



```
-- 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);
```

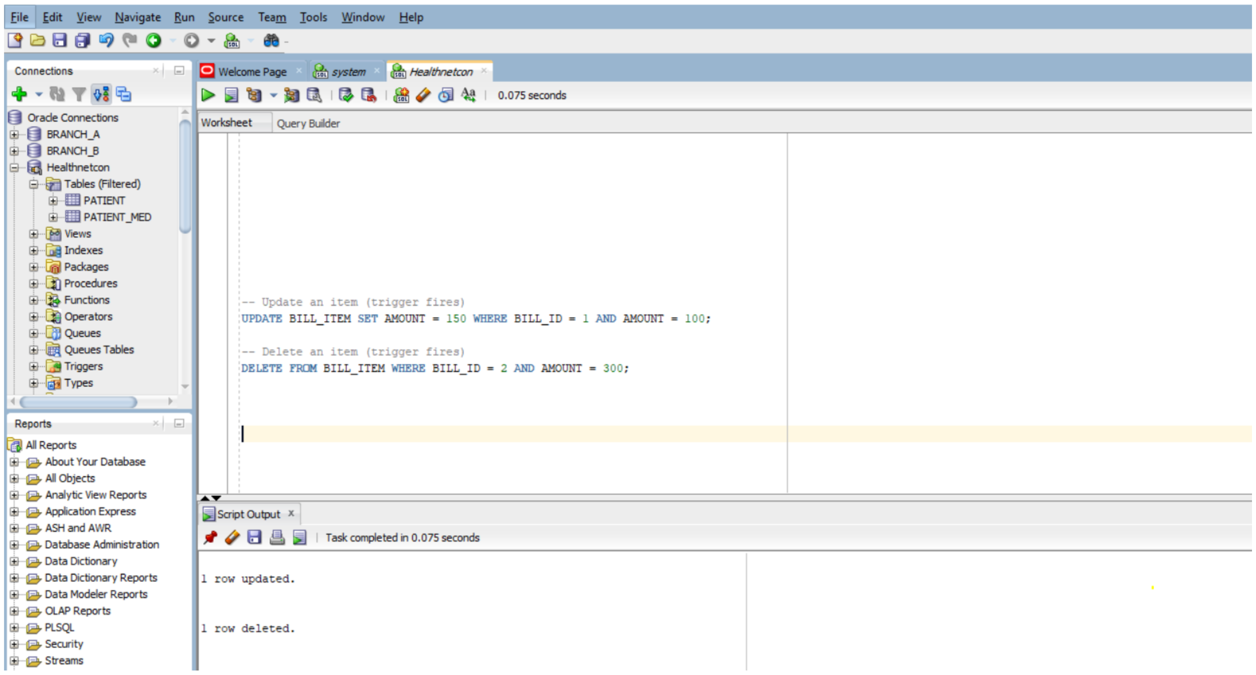
```
1 row inserted.

1 row inserted.

1 row inserted.
```

-- 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;

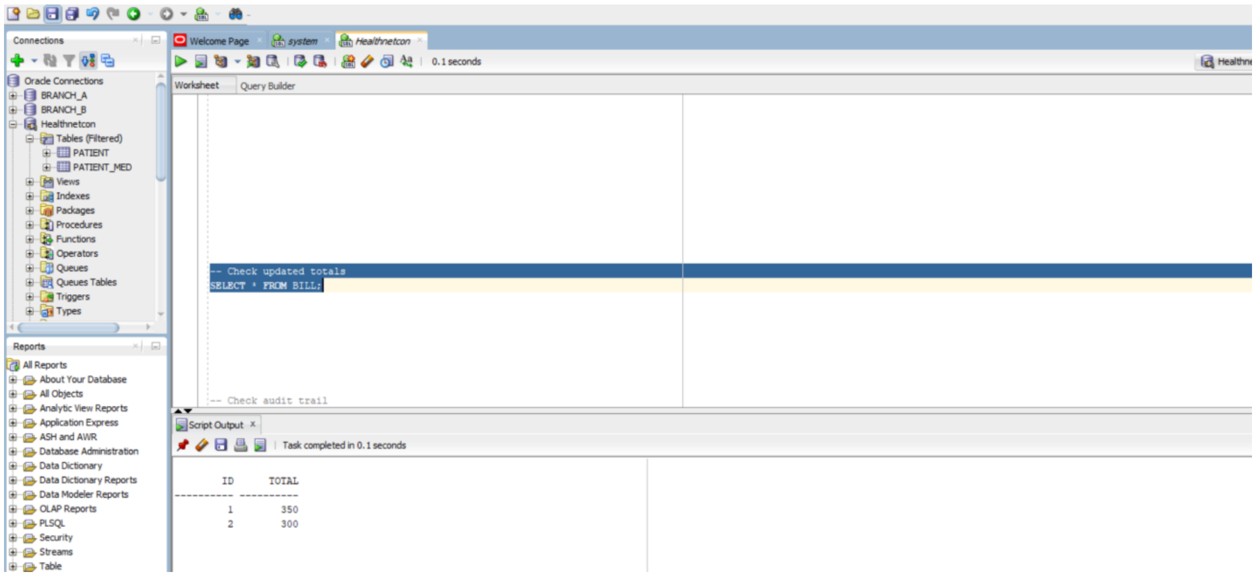


```
-- Update an item (trigger fires)
UPDATE BILL_ITEM SET AMOUNT = 150 WHERE BILL_ID = 1 AND AMOUNT = 100;

-- Delete an item (trigger fires)
DELETE FROM BILL_ITEM WHERE BILL_ID = 2 AND AMOUNT = 300;
```

```
1 row updated.

1 row deleted.
```

-- Check updated totals

SELECT * FROM BILL;

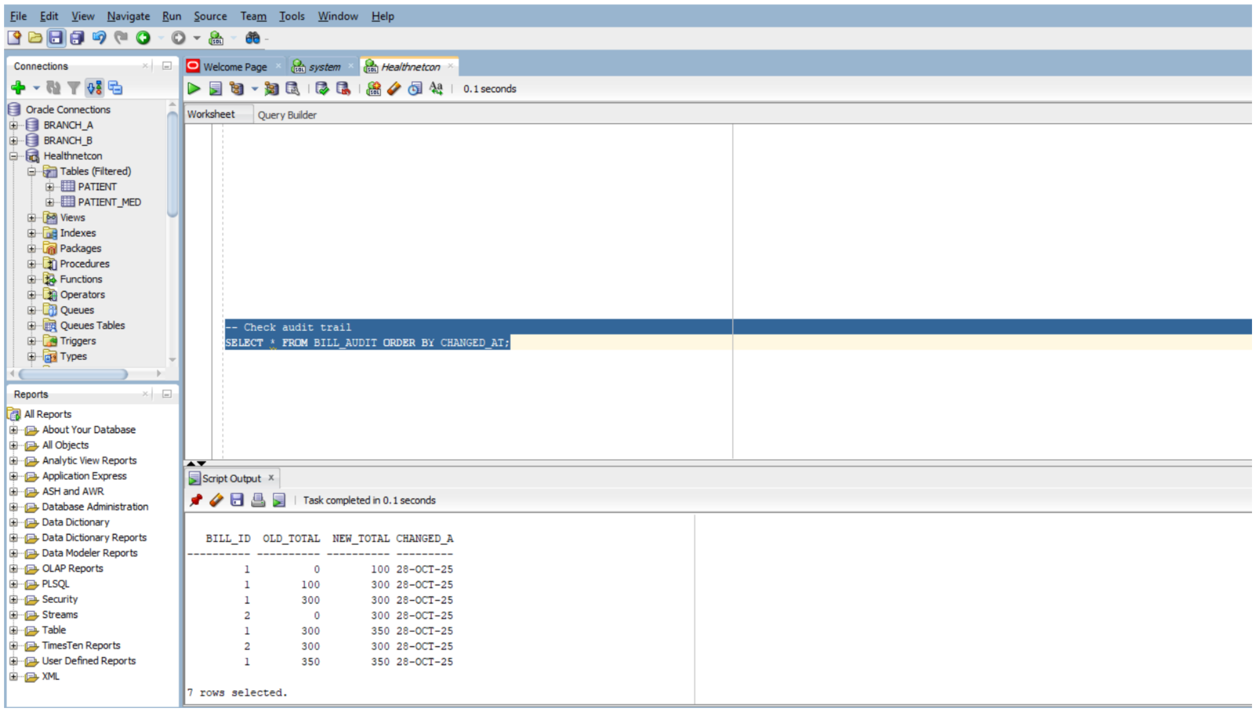


```
-- Check updated totals
SELECT * FROM BILL;

-- Check audit trail
```

```
        ID      TOTAL
---------- ----------
         1        350
         2        300
```

-- Check audit trail

SELECT * FROM BILL_AUDIT ORDER BY CHANGED_AT;



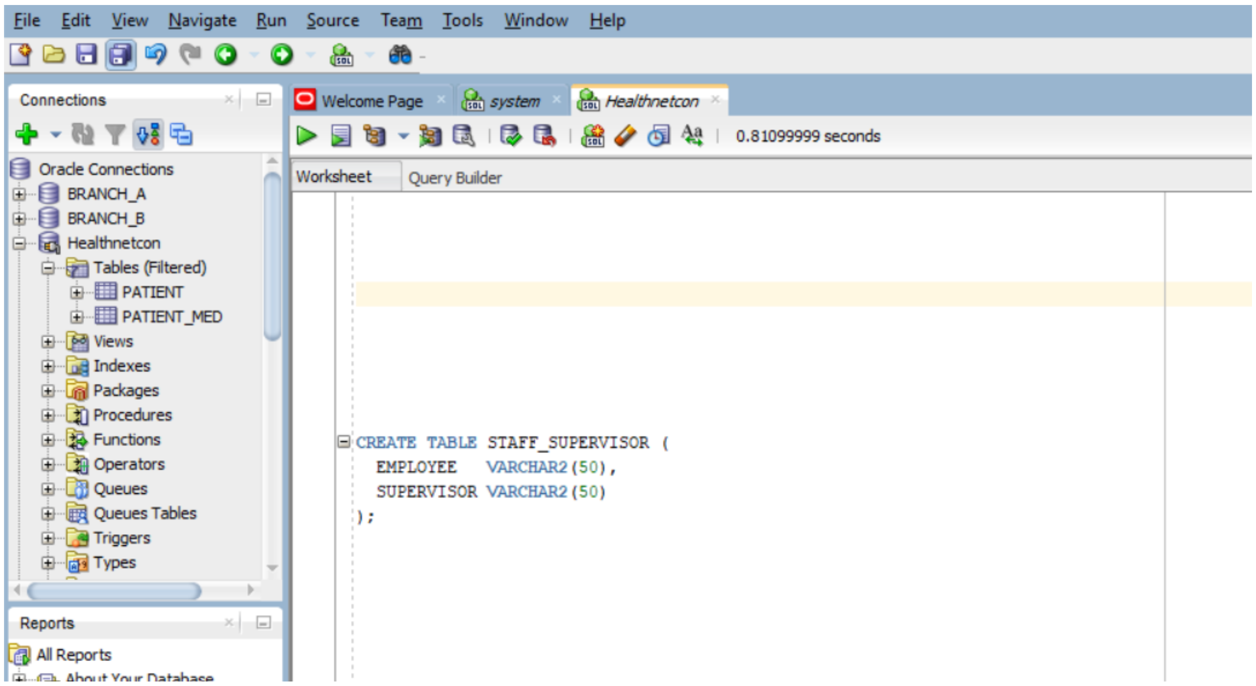
```
-- Check audit trail
SELECT * FROM BILL_AUDIT ORDER BY CHANGED_AT;
```

Task completed in 0.1 seconds

```
   BILL_ID  OLD_TOTAL  NEW_TOTAL CHANGED_A
---------- ---------- ---------- ---------
         1          0        100 28-OCT-25
         1        100        300 28-OCT-25
         1        300        300 28-OCT-25
         2          0        300 28-OCT-25
         1        300        350 28-OCT-25
         2        300        300 28-OCT-25
         1        350        350 28-OCT-25

7 rows selected.
```

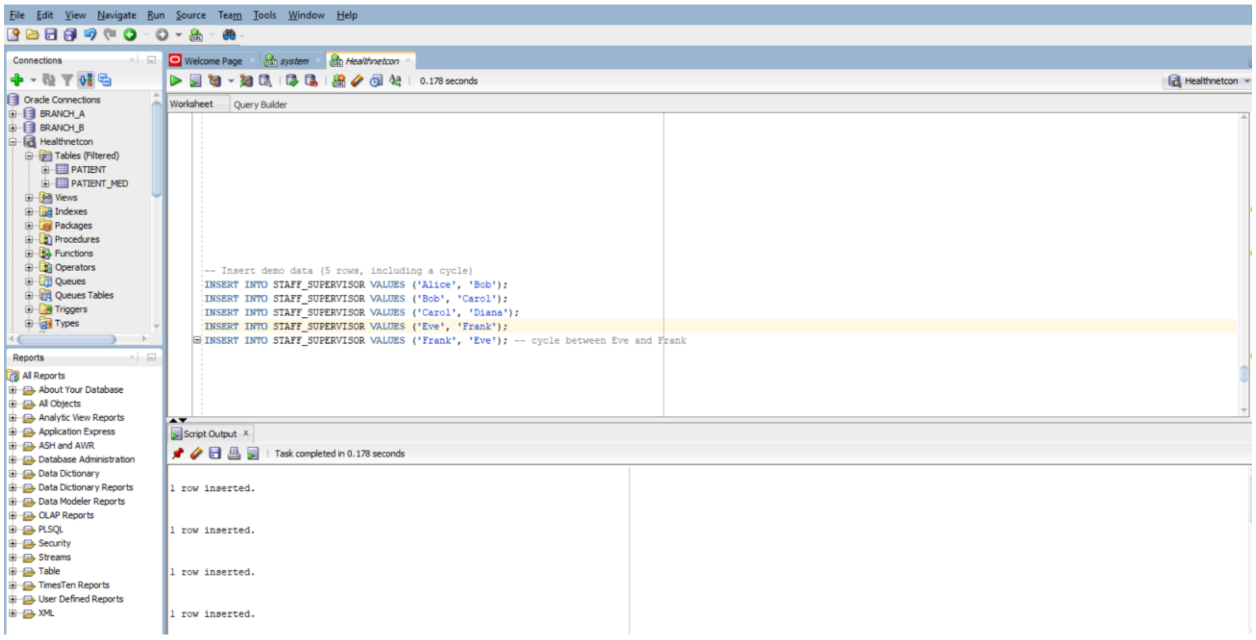
- BILL.TOTAL for ID 1 should reflect the sum of its items (e.g., 150 + 200 = 350).
- BILL.TOTAL for ID 2 should be 0 after deletion.
- BILL_AUDIT should show old and new totals for each change.

3.



```sql
CREATE TABLE STAFF_SUPERVISOR (
  EMPLOYEE    VARCHAR2(50),
  SUPERVISOR VARCHAR2(50)
);
```

Inserting rows



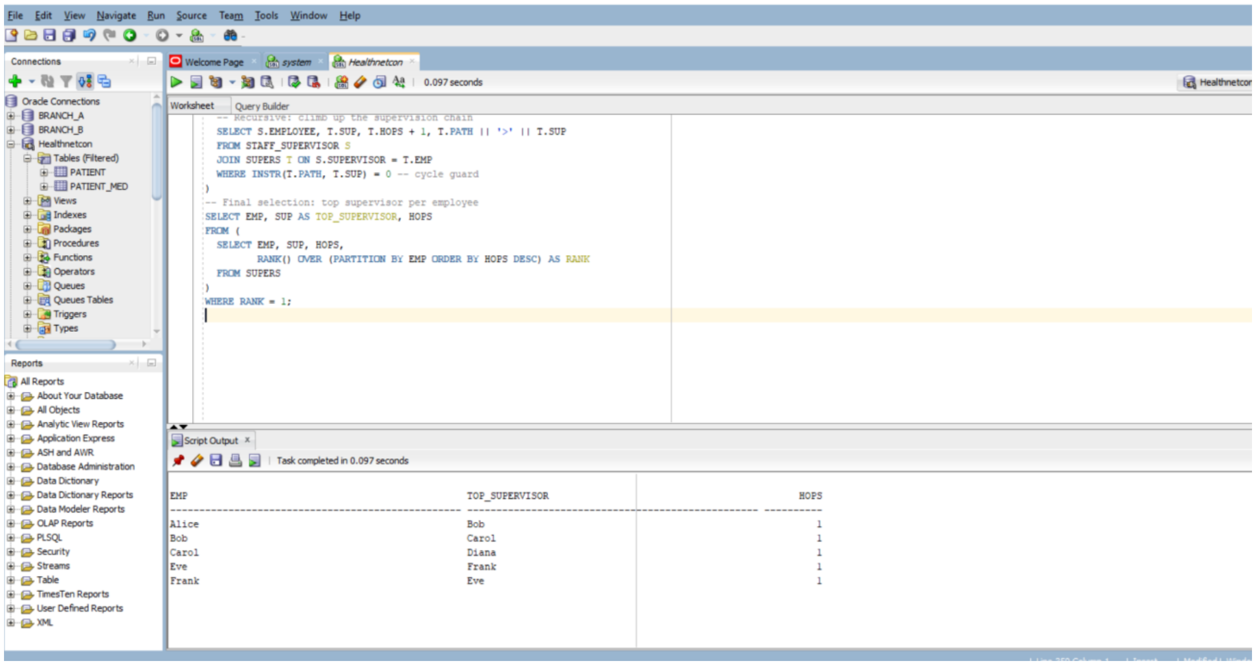
```sql
-- Insert demo data (5 rows, including a cycle)
INSERT INTO STAFF_SUPERVISOR VALUES ('Alice', 'Bob');
INSERT INTO STAFF_SUPERVISOR VALUES ('Bob', 'Carol');
INSERT INTO STAFF_SUPERVISOR VALUES ('Carol', 'Diana');
INSERT INTO STAFF_SUPERVISOR VALUES ('Eve', 'Frank');
INSERT INTO STAFF_SUPERVISOR VALUES ('Frank', 'Eve'); -- cycle between Eve and Frank
```

Task completed in 0.178 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

-- 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;
```



```
  -- 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;
```

Task completed in 0.097 seconds

```
EMP                                                TOP_SUPERVISOR                                            HOPS
-------------------------------------------------- -------------------------------------------------- ----------
Alice                                              Bob                                                          1
Bob                                                Carol                                                        1
Carol                                              Diana                                                        1
Eve                                                Frank                                                        1
Frank                                              Eve                                                          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)

QUESTION 4 :

bugs

--The direction of recursion is wrong:

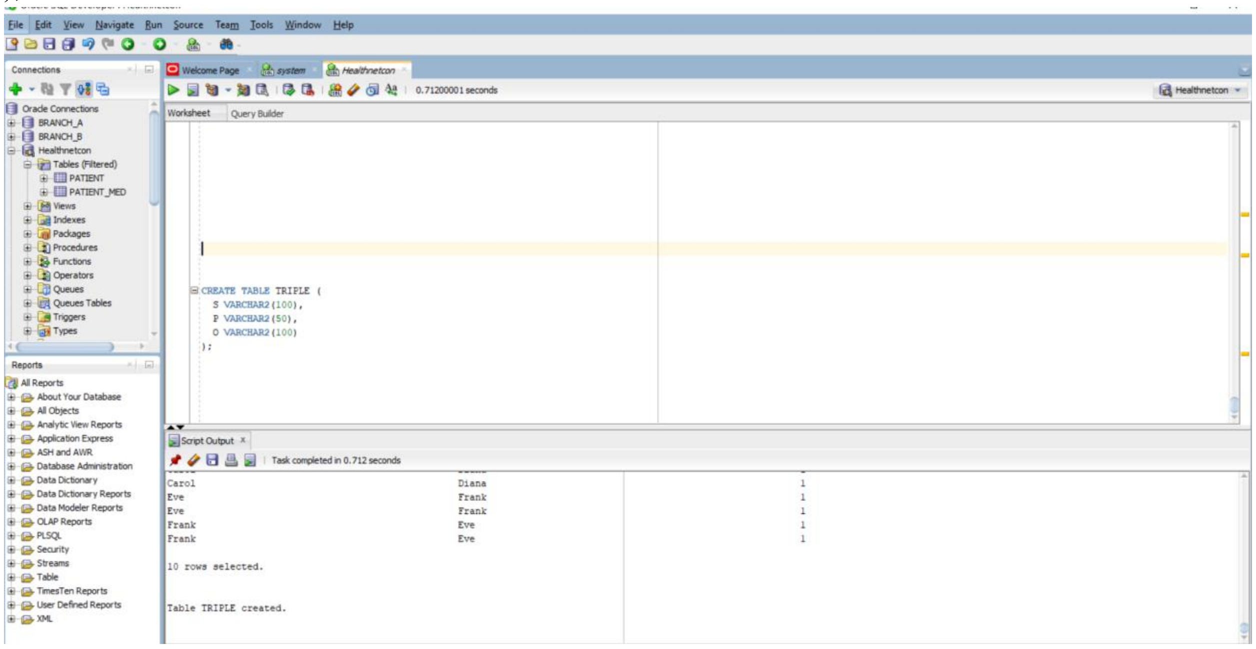
--CHILD/ANCESTOR are reversed in recursion.

--The base case starts from CHILD but should start from ANCESTOR.

--The final filter compares the wrong column (ISA.CHILD = 'InfectiousDisease'), it should be ISA.ANCESTOR = 'InfectiousDisease'.

-

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



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

Task completed in 0.712 seconds

```
Carol                                   Diana                                        1
Eve                                     Frank                                        1
Eve                                     Frank                                        1
Frank                                   Eve                                          1
Frank                                   Eve                                          1

10 rows selected.


Table TRIPLE created.
```

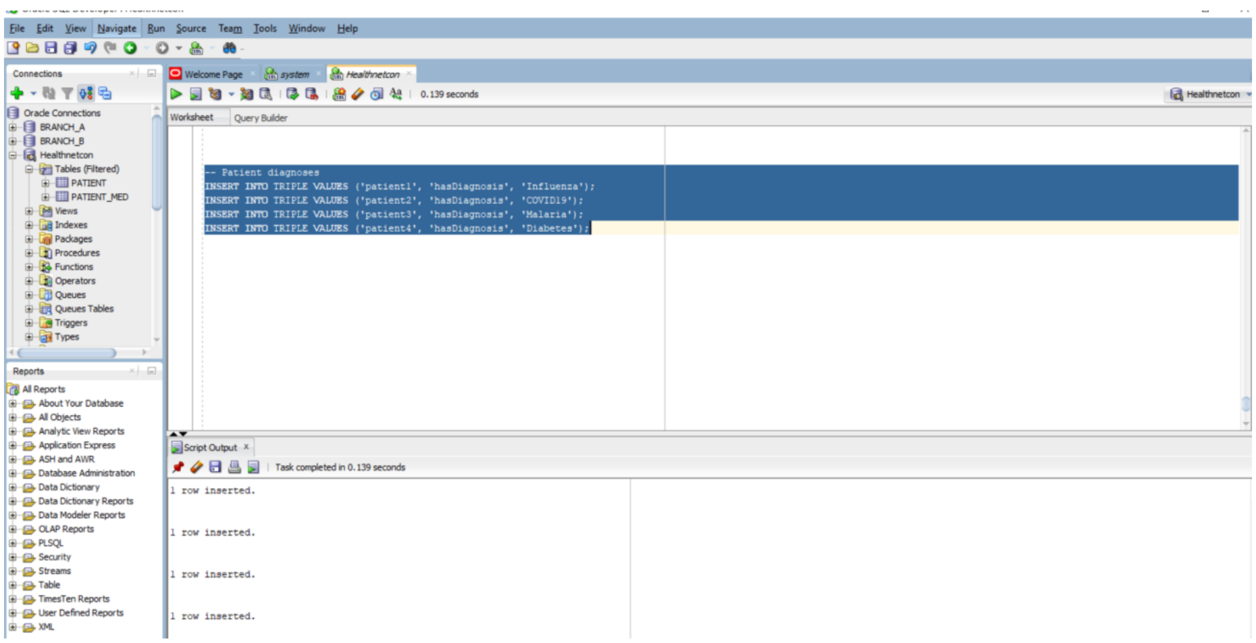
-- Patient diagnoses

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');



```
-- 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');
```

Task completed in 0.139 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

-- Taxonomy edges

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

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

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

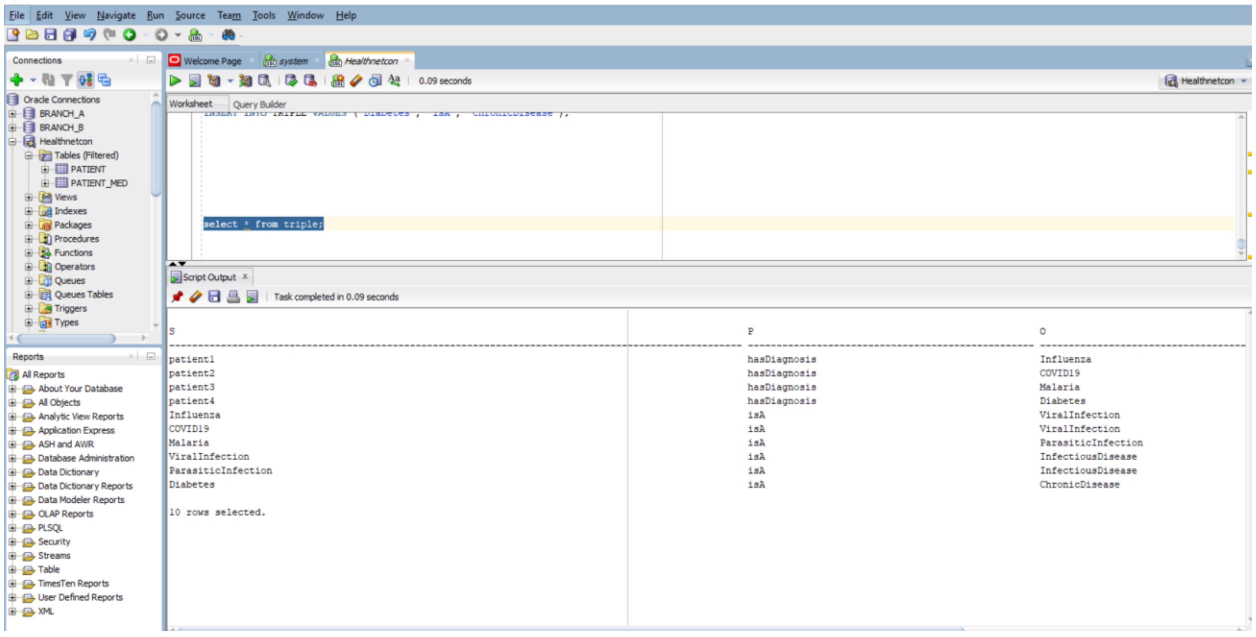
INSERT INTO TRIPLE VALUES ('ViralInfection', 'isA', 'InfectiousDisease');

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

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

Check inserted rows;

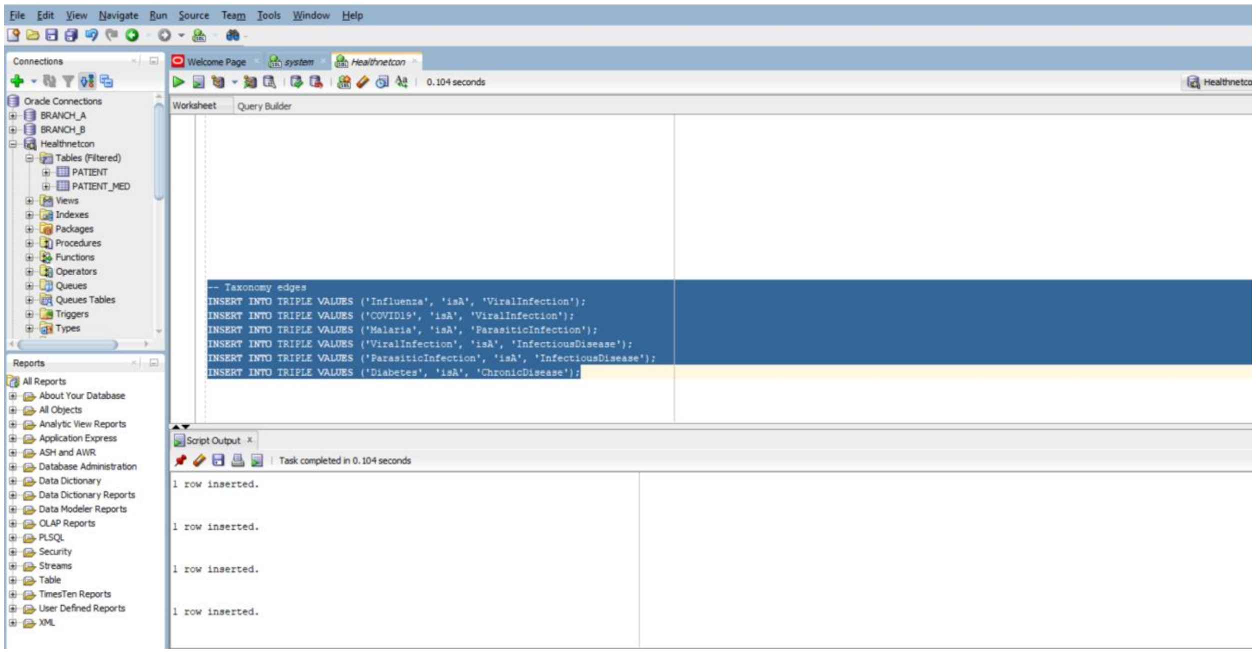
select * from triple;



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

select * from triple;
```

Task completed in 0.09 seconds

```
S                                                                                      P                                  O
-------------------------------------------------------------------------------------- ---------------------------------- ----------------------------------
patient1                                                                               hasDiagnosis                       Influenza
patient2                                                                               hasDiagnosis                       COVID19
patient3                                                                               hasDiagnosis                       Malaria
patient4                                                                               hasDiagnosis                       Diabetes
Influenza                                                                              isA                                ViralInfection
COVID19                                                                                isA                                ViralInfection
Malaria                                                                                isA                                ParasiticInfection
ViralInfection                                                                         isA                                InfectiousDisease
ParasiticInfection                                                                     isA                                InfectiousDisease
Diabetes                                                                               isA                                ChronicDisease

10 rows selected.
```

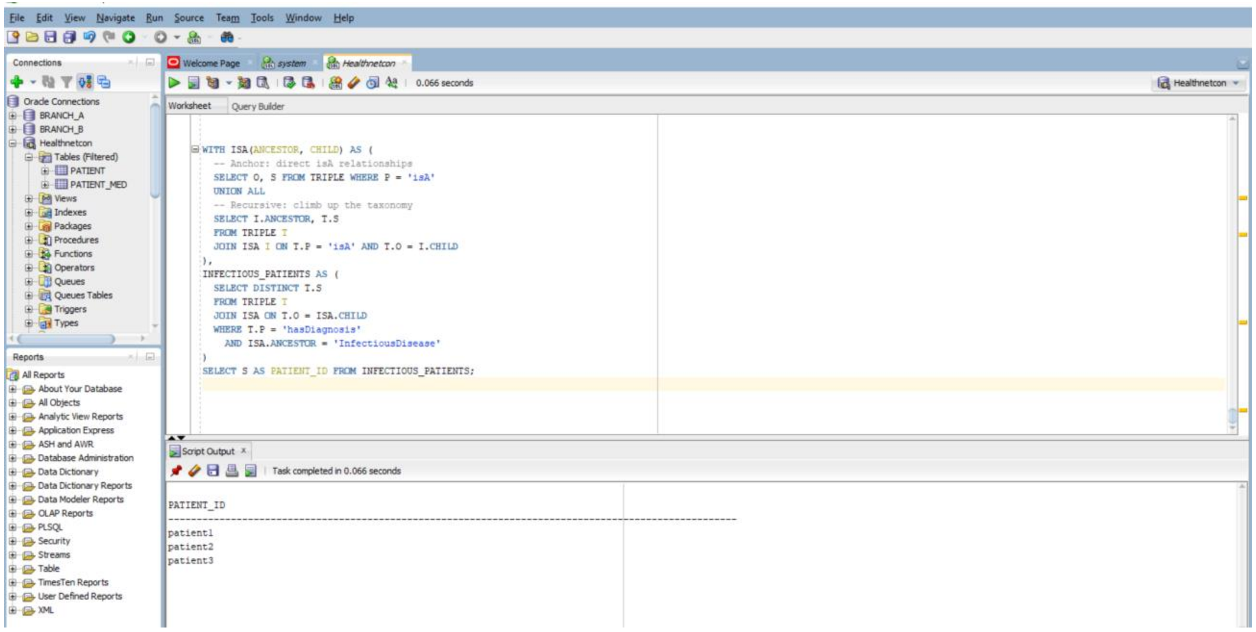


```
-- Taxonomy edges
INSERT INTO TRIPLE VALUES ('Influenza', 'isA', 'ViralInfection');
INSERT INTO TRIPLE VALUES ('COVID19', 'isA', 'ViralInfection');
INSERT INTO TRIPLE VALUES ('Malaria', 'isA', 'ParasiticInfection');
INSERT INTO TRIPLE VALUES ('ViralInfection', 'isA', 'InfectiousDisease');
INSERT INTO TRIPLE VALUES ('ParasiticInfection', 'isA', 'InfectiousDisease');
INSERT INTO TRIPLE VALUES ('Diabetes', 'isA', 'ChronicDisease');
```

Task completed in 0.104 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

```
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;
```



```
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;
```

Task completed in 0.066 seconds

```
PATIENT_ID
------------------------------------------------------------------------------------------------------
patient1
patient2
patient3
```

- 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)

QUESTION 5 : Spatial Database task for Oracle using SDO_GEOMETRY.

Bugs

--Wrong SRID – should be 4326 (WGS84) instead of 3857.Lat/Lon order swapped – Oracle expects (X=longitude, Y=latitude).

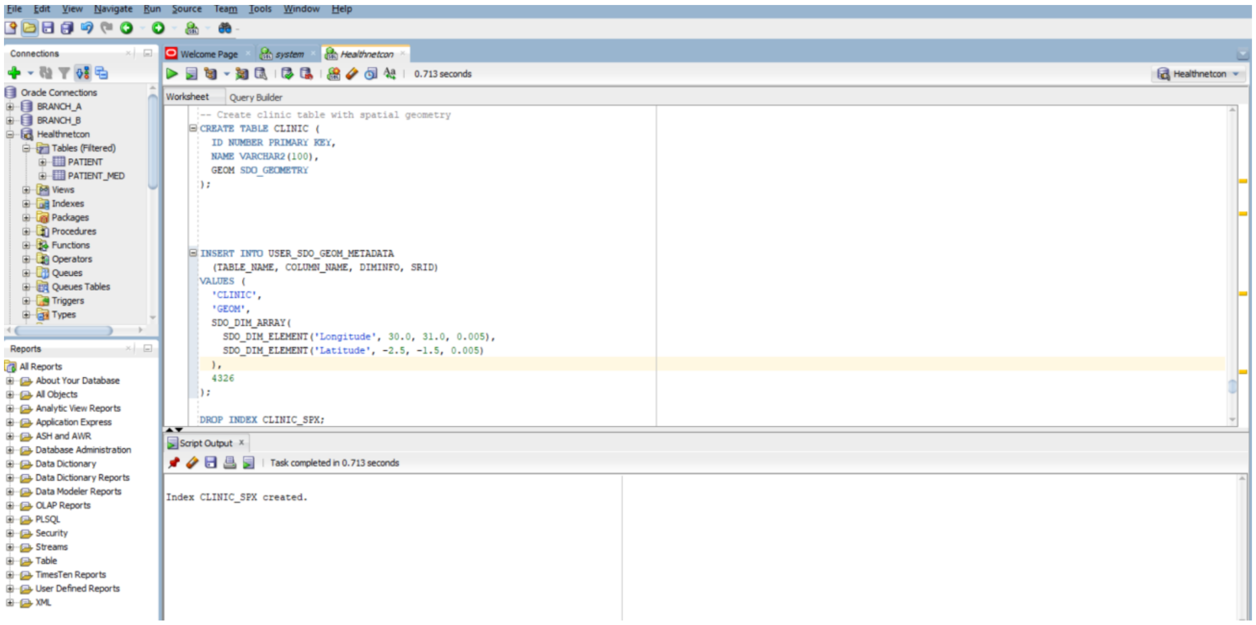
--distance units missing – need 'unit=KM'.

-- placeholder :AMB_POINT – must define the ambulance location as a

-- SDO_GEOMETRY point.

```
-- 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
);
```



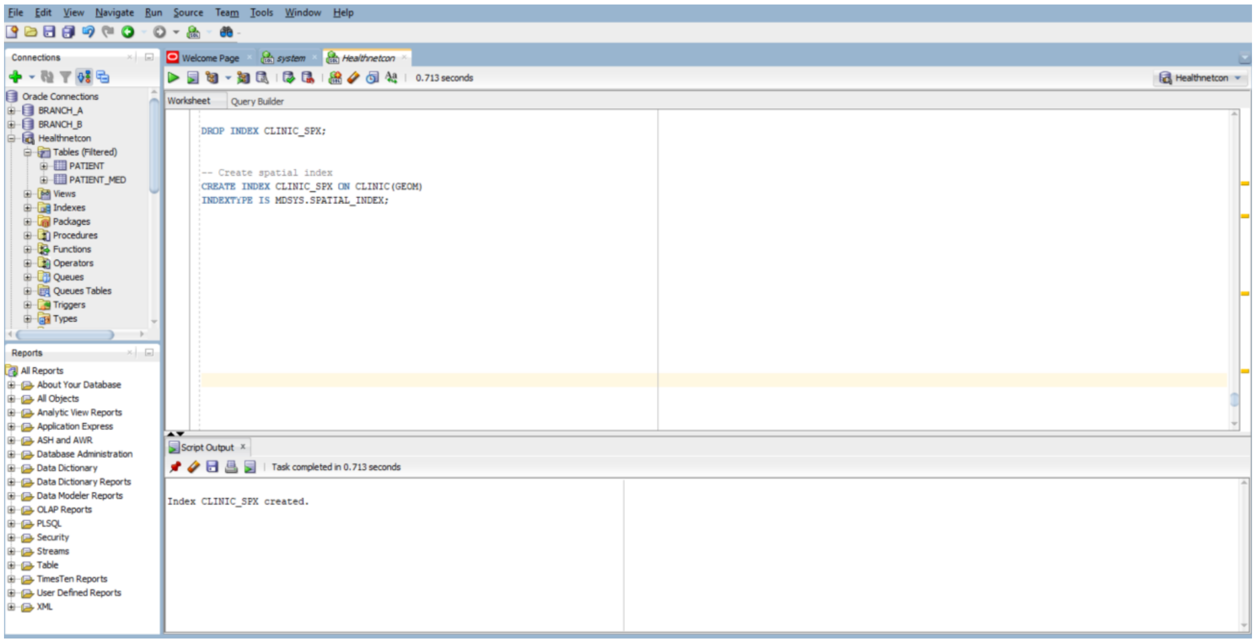
```
-- Create clinic table with spatial geometry
CREATE TABLE CLINIC (
  ID NUMBER PRIMARY KEY,
  NAME VARCHAR2(100),
  GEOM SDO_GEOMETRY
);



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
);

DROP INDEX CLINIC_SPX;
```

Task completed in 0.713 seconds

```
Index CLINIC_SPX created.
```

```
-- Create spatial index
CREATE INDEX CLINIC_SPX ON CLINIC(GEOM)
```

INDEXTYPE IS MDSYS.SPATIAL_INDEX;



```
DROP INDEX CLINIC_SPX;


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

Task completed in 0.713 seconds

```
Index CLINIC_SPX created.
```

-- 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',

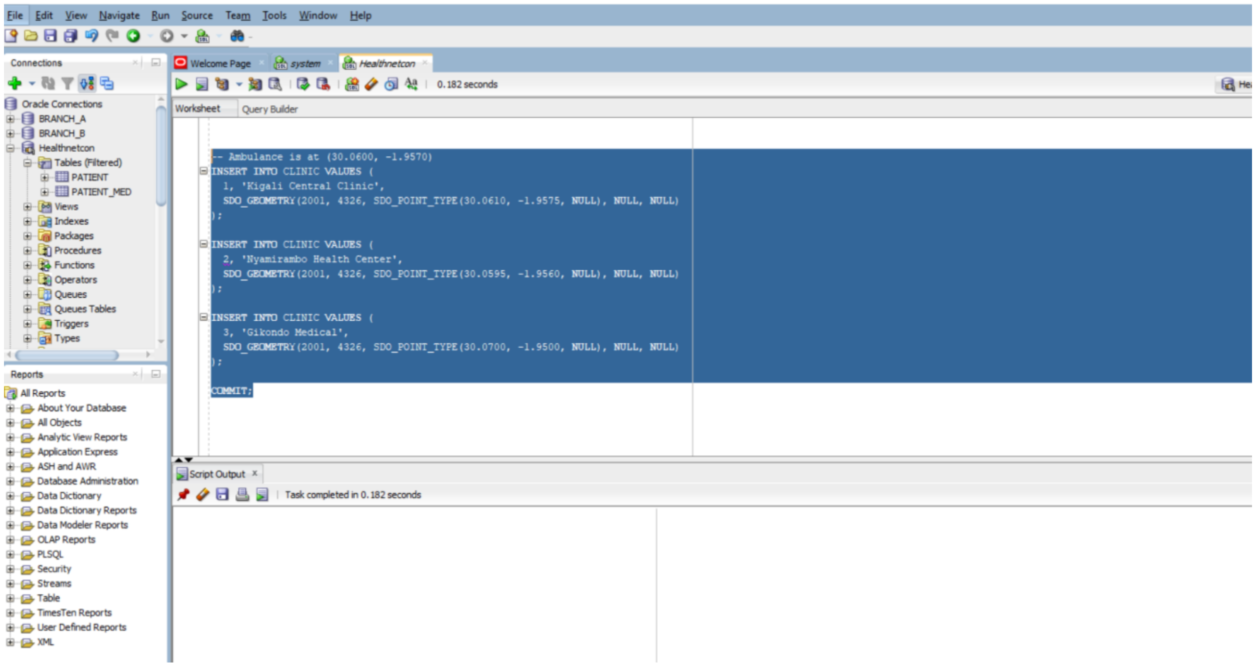
  SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0595, -1.9560, NULL), NULL, NULL)

);

INSERT INTO CLINIC VALUES (

  3, 'Gikondo Medical',

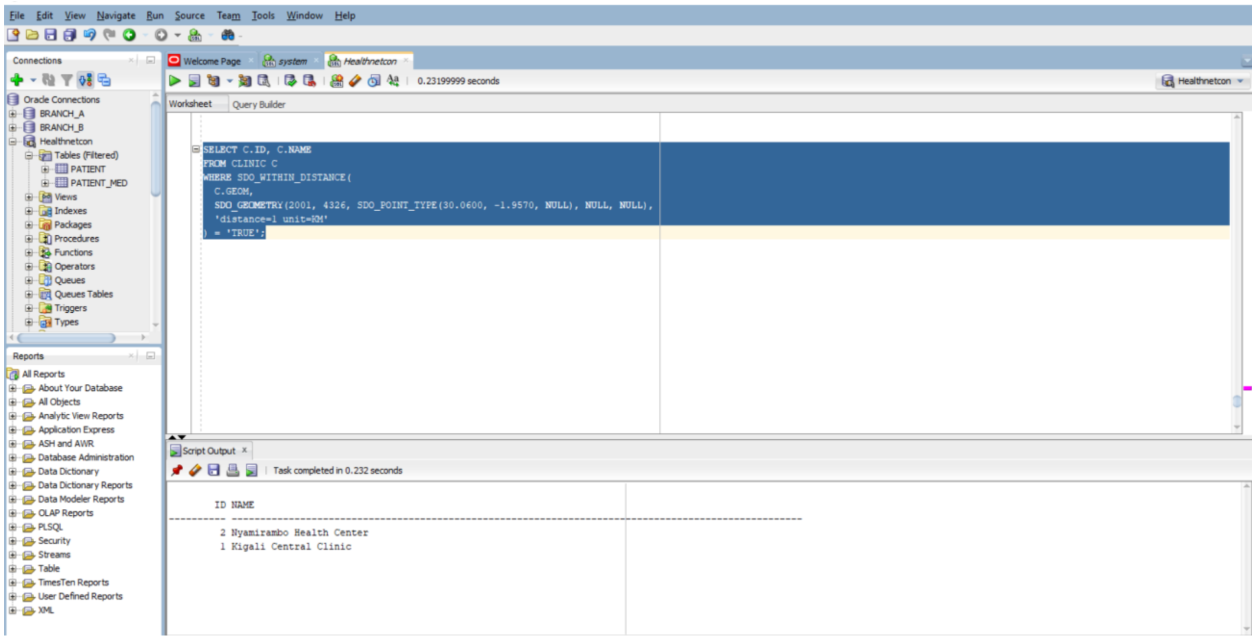
```
  SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0700, -1.9500, NULL), NULL, NULL)
);

COMMIT;
```



```
-- 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;
```

```
Task completed in 0.182 seconds
```

```
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';
```

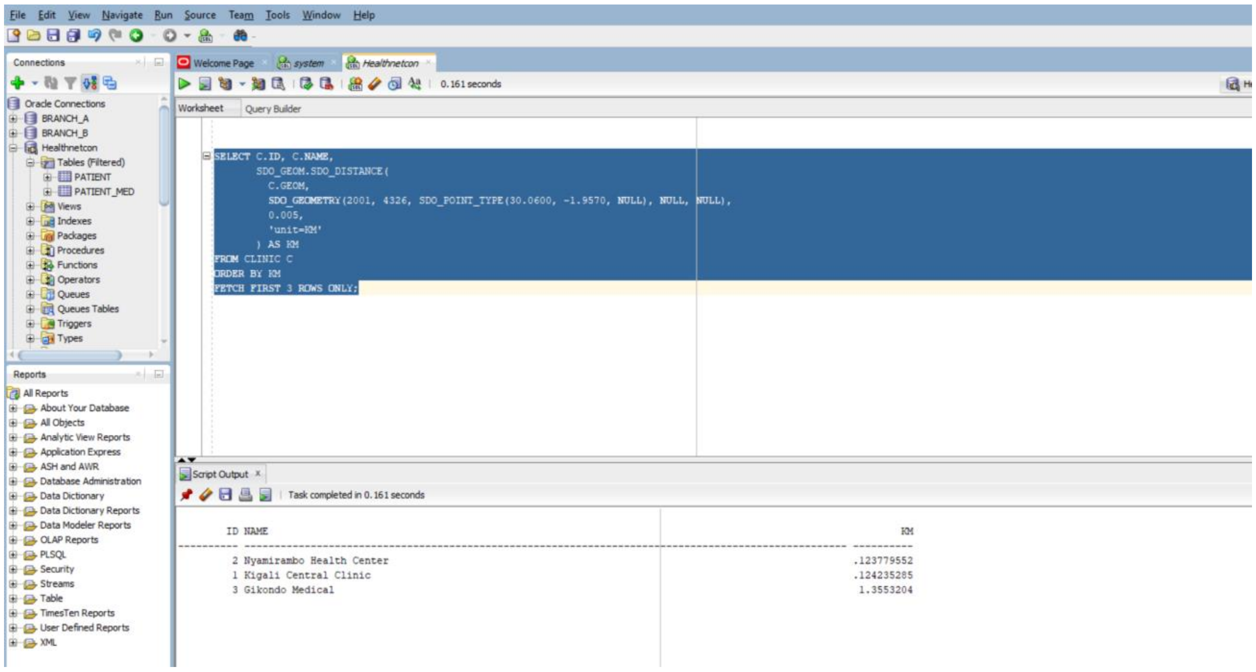


```
SELECT C.ID, C.NAME
FROM CLINIC C
WHERE SDO_WITHIN_DISTANCE(
  C.GEOM,
  SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0600, -1.9570, NULL), NULL, NULL),
  'distance=1 unit=KM'
) = 'TRUE';
```

Task completed in 0.232 seconds

```
        ID NAME
---------- ----------------------------------------------------------------------------------------------------
         2 Nyamirambo Health Center
         1 Kigali Central Clinic
```

```
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;
```



```sql
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;
```

Task completed in 0.161 seconds

```
        ID NAME                                                                                                   KM
---------- ---------------------------------------------------------------------------------------------------- ----------
         2 Nyamirambo Health Center                                                                              .123779552
         1 Kigali Central Clinic                                                                                 .124235285
         3 Gikondo Medical                                                                                        1.3553204
```