MSc in Data Science in Data Mining

Module:

Advanced Databases Technology

Topic:

Intelligent Databases

Assignment Title:

Debugging Intelligent Databases SQL codes

Assignment no. 4

By

Full name	Reg #
Amoss Robert	224019944

Due Date: 29th October, 2025

DEBUGGED CODES

Question_1: Rules (Declarative Constraints) - Safe Prescriptions

```
-- Use schema: HEALTHNET--
ALTER SESSION SET CURRENT_SCHEMA = c##HEALTHNET;
-- prerequisite (minimal)
CREATE TABLE PATIENT (
ID NUMBER PRIMARY KEY,
NAME VARCHAR2(100) NOT NULL
);

-- BUGGY: commas, NOT NULLs, CHECK parentheses, and date rule wording
 CREATE TABLE PATIENT_MED (
  PATIENT_MED_ID NUMBER PRIMARY KEY,                    -- unique id
  PATIENT_ID NUMBER REFERENCES PATIENT(ID),             -- must reference an existing patient
  MED_NAME VARCHAR2(80) NOT NULL,                       -- should be NOT NULL
  DOSE_MG NUMBER(6,2) CHECK (DOSE_MG >= 0),             -- missing parentheses
  START_DT DATE,
  END_DT   DATE,
  CONSTRAINT CK_RX_DATES CHECK (START_DT <= END_DT)     -- corrected phrase
 );
```

```
Script Output
Task completed in 0.134 seconds

Table PATIENT_MED created.
```

Figure 1: Corrected SQL codes

```
-- FFIRST FAILED INSERT
-- TESTING AN INTEGRITY CONSTRAINT  (A MISSING PATIENT IN  PATIENT TABLE)
INSERT INTO PATIENT_MED (PATIENT_MED_ID,PATIENT_ID, MED_NAME,DOSE_MG, START_DT, END_DT)
VALUES(1,2, 'Bufen',10,DATE '2025-01-1', DATE '2025-10-10');
```

```
Script Output
Task completed in 0.132 seconds

Error starting at line : 26 in command -
INSERT INTO PATIENT_MED (PATIENT_MED_ID,PATIENT_ID, MED_NAME,DOSE_MG, START_DT, END_DT)
VALUES(1,2, 'Bufen',10,DATE '2025-01-1', DATE '2025-10-10')
Error report -
ORA-02291: integrity constraint (C##HEALTHNET.SYS_C008480) violated - parent key not found

https://docs.oracle.com/error-help/db/ora-02291/

More Details :
https://docs.oracle.com/error-help/db/ora-02291/
```

Figure 2: Testing integrity constraint (a missing patient ID in patient table)

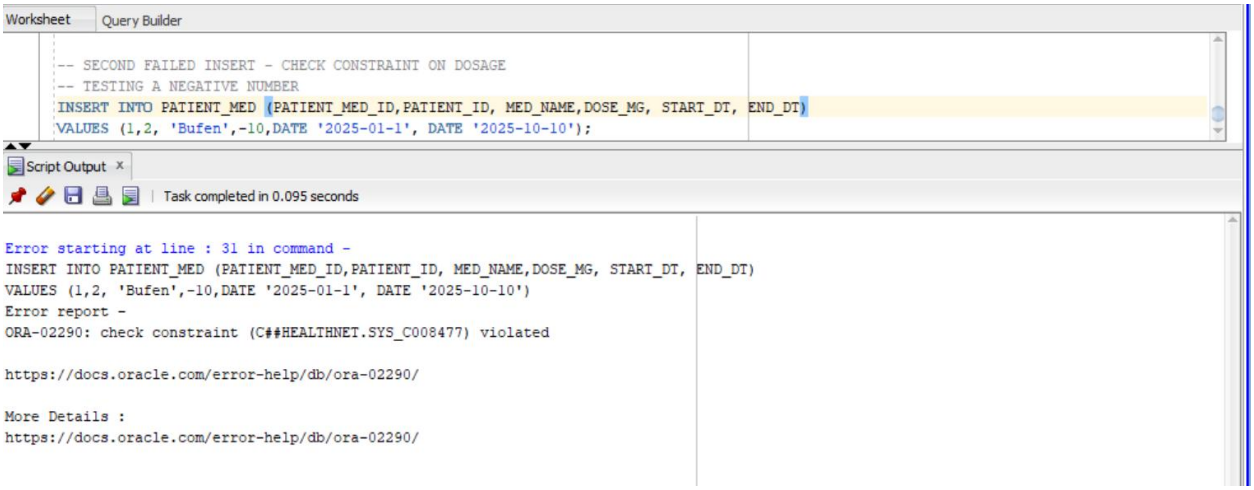


Figure 3: Testing a check constraint on dosage non-negative input

Now, I will insert a one patient record and rerun the first failed insert query into patient_med table.

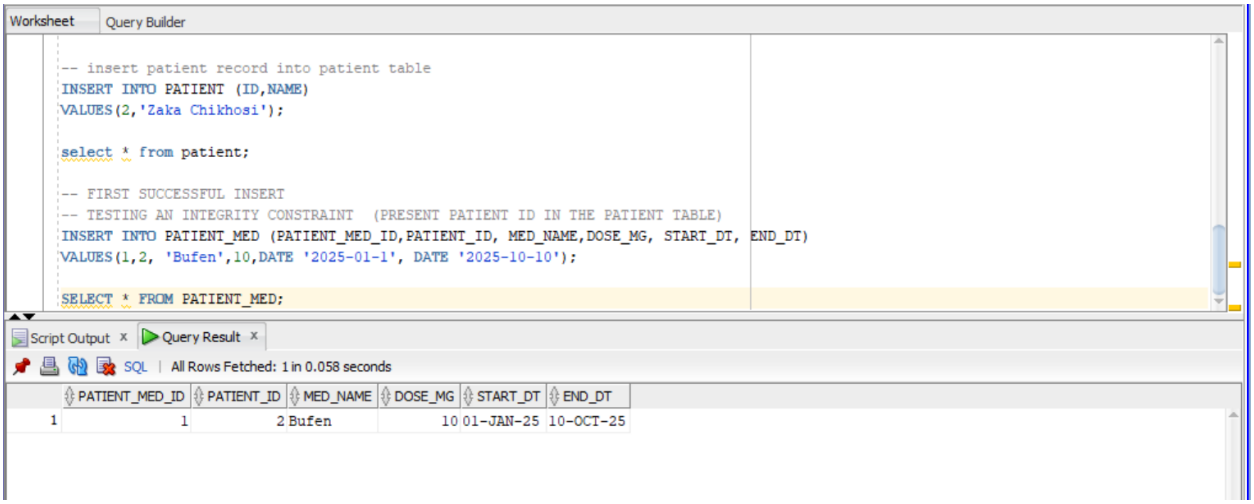


Figure 4: First Success Integrity Constraint - (Presence of patient ID in the patient Table)

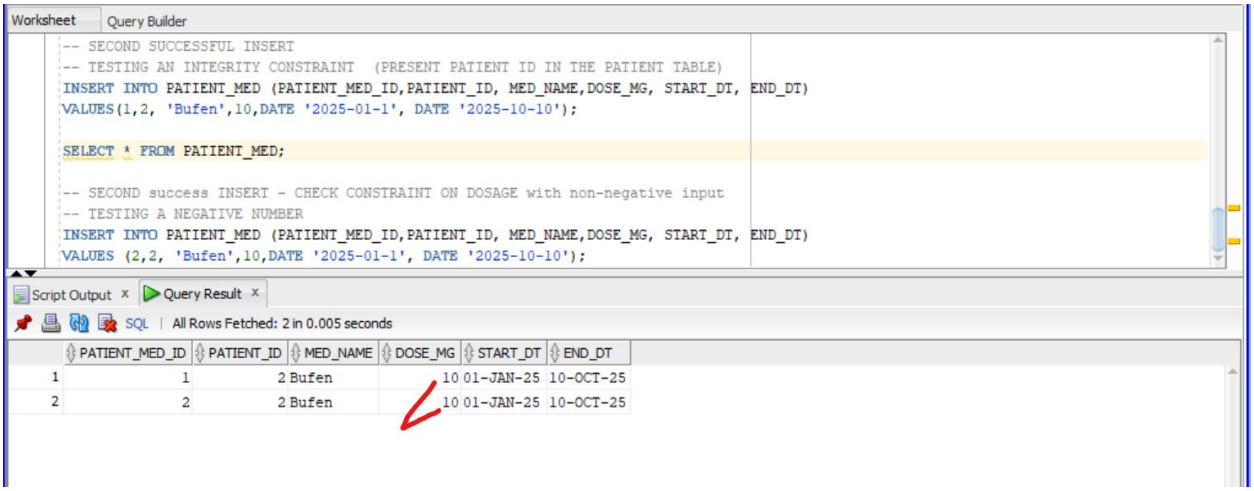


Figure 5: SECOND success INSERT - CHECK CONSTRAINT ON DOSAGE with non-negative input

Question_2: Active Databases (E-C-A Trigger) - Bill Totals

```
/
-- Students: replace with a statement-level trigger named TRG_BILL_TOTAL_STMT
-- (or a compound trigger named TRG_BILL_TOTAL_CMP) that:
-- 1) collects affected BILL_IDs; 2) recomputes once per bill; 3) inserts an audit row.

-- Compound trigger
CREATE OR REPLACE TRIGGER TRG_BILL_TOTAL_CMP
FOR INSERT OR UPDATE OR DELETE ON BILL_ITEM
COMPOUND TRIGGER

 -- Declare a collection to store affected bill IDs
 TYPE t_bill_ids IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
 g_bill_ids t_bill_ids;
 g_count INTEGER := 0;

 -- Before or after each row
 AFTER EACH ROW IS
 BEGIN

   -- Handle :NEW and :OLD safely
   IF INSERTING OR UPDATING THEN
     g_count := g_count + 1;
     g_bill_ids(g_count) := :NEW.BILL_ID;
   ELSIF DELETING THEN
     g_count := g_count + 1;
     g_bill_ids(g_count) := :OLD.BILL_ID;
   END IF;
 END AFTER EACH ROW;

 -- After statement finishes, recompute totals once per affected bill
 AFTER STATEMENT IS
 BEGIN
   FOR i IN 1 .. g_count LOOP
     DECLARE
       v_bill_id NUMBER := g_bill_ids(i);
       v_old_total NUMBER(12,2);

       v_new_total NUMBER(12,2);
     BEGIN
       -- Get the old total
       SELECT NVL(TOTAL, 0) INTO v_old_total FROM BILL WHERE ID = v_bill_id;

       -- Compute new total
       SELECT NVL(SUM(AMOUNT), 0)
         INTO v_new_total
         FROM BILL_ITEM
        WHERE BILL_ID = v_bill_id;

       -- Update the BILL table
       UPDATE BILL
          SET TOTAL = v_new_total
        WHERE ID = v_bill_id;


       -- Record the change in the audit table
       INSERT INTO BILL_AUDIT (BILL_ID, OLD_TOTAL, NEW_TOTAL, CHANGED_AT)
       VALUES (v_bill_id, v_old_total, v_new_total, SYSDATE);
     EXCEPTION
       WHEN NO_DATA_FOUND THEN
         NULL; -- Ignore if BILL row doesn't exist
     END;
   END LOOP;
 END AFTER STATEMENT;
END TRG_BILL_TOTAL_CMP;
/
```

```
Script Output
Task completed in 1.049 seconds

Trigger TRG_BILL_TOTAL_CMP compiled
```

Figure 6: Corrected and compiled TRG_BILL_CMP

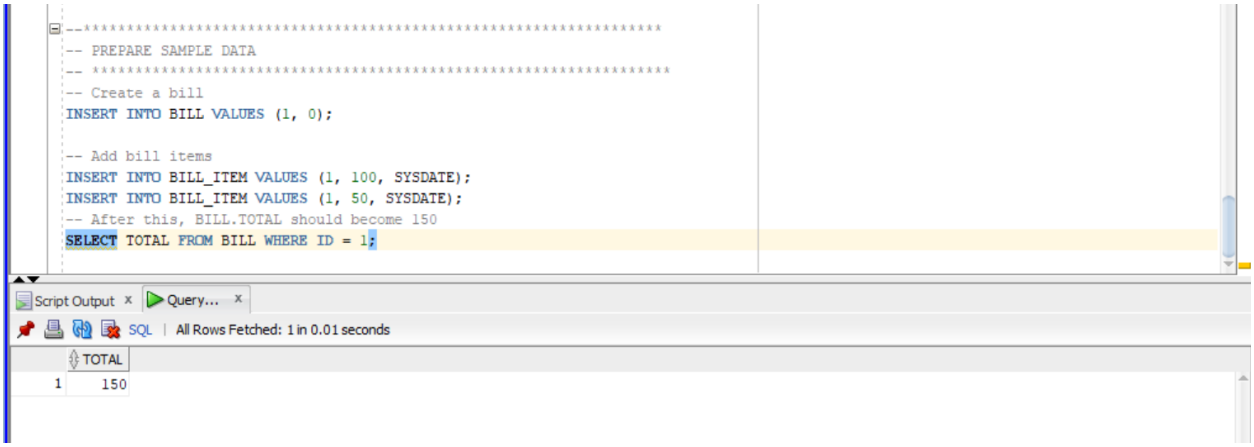


Figure 7: Result of testing TRG_BILL_CMP by inserting new item in Bill table

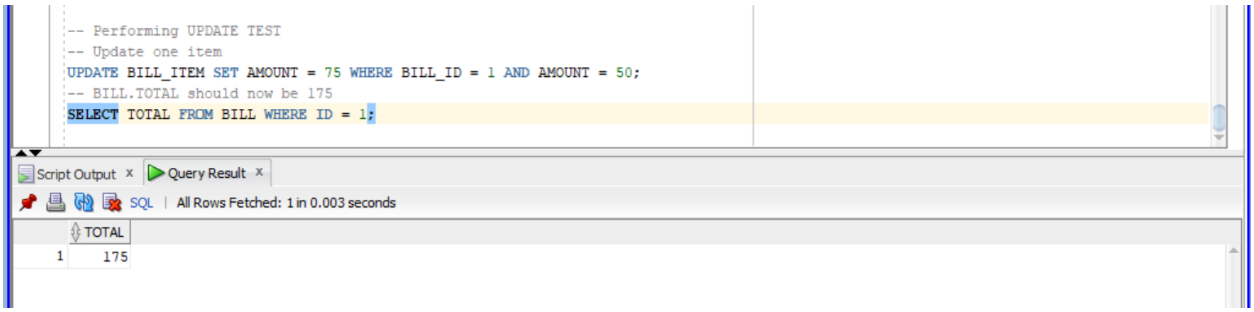


Figure 8: Result of testing TRG_BILL_CMP by updating existing item in Bill ITEM table

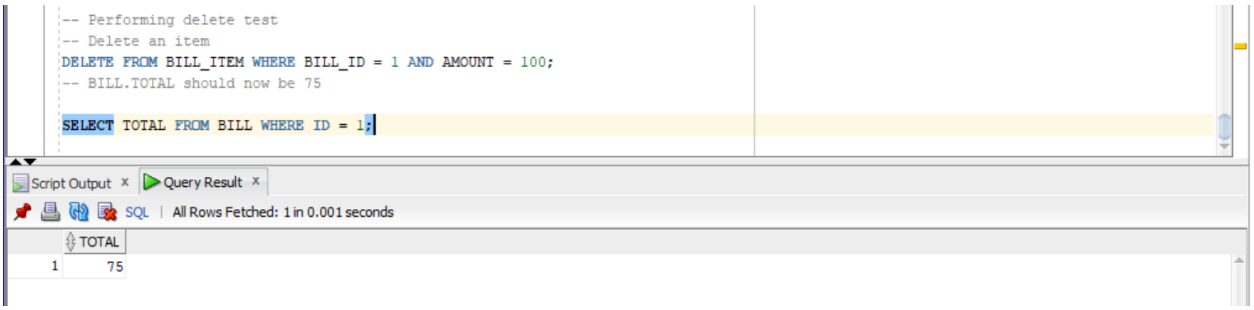


Figure 9: Result of testing TRG_BILL_CMP by deleting an existing item in Bill ITEM table

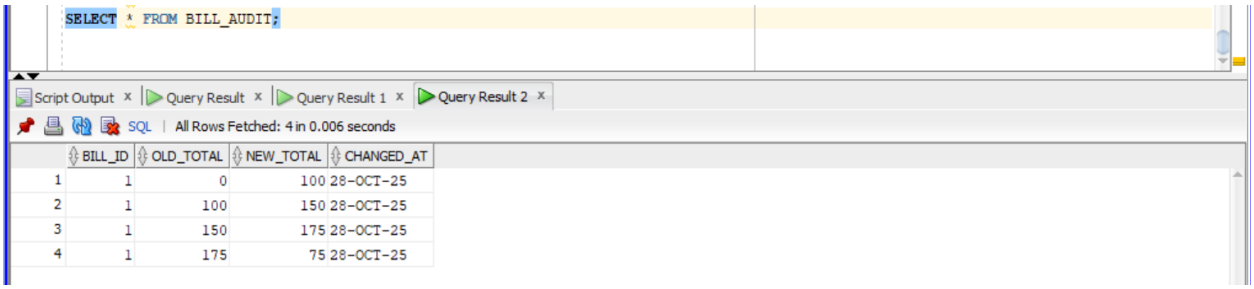


Figure 10: Audit Traces after previous insert, update and delete operations

Question_3: Deductive Databases (Recursive WITH): Referral/Supervision Chain

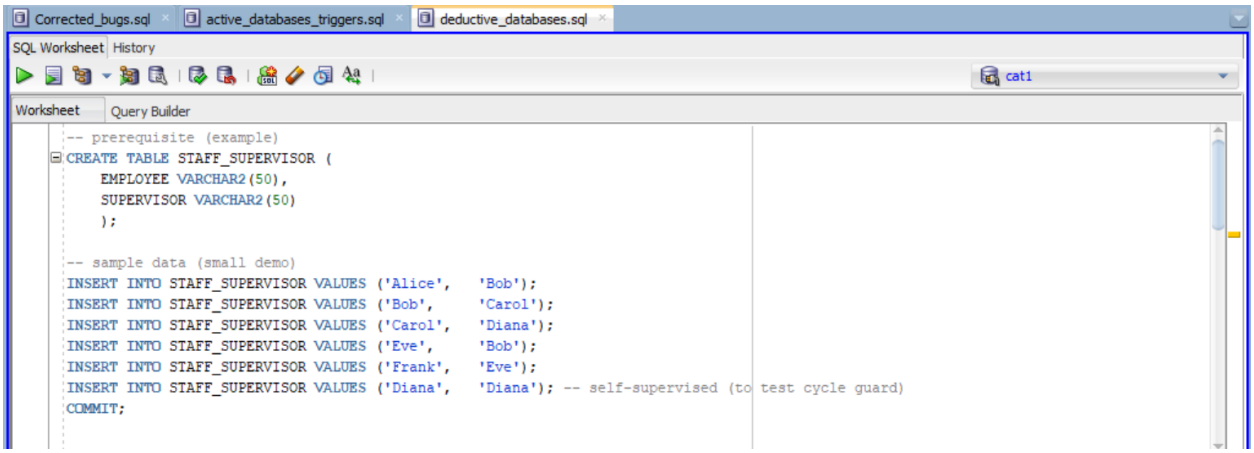
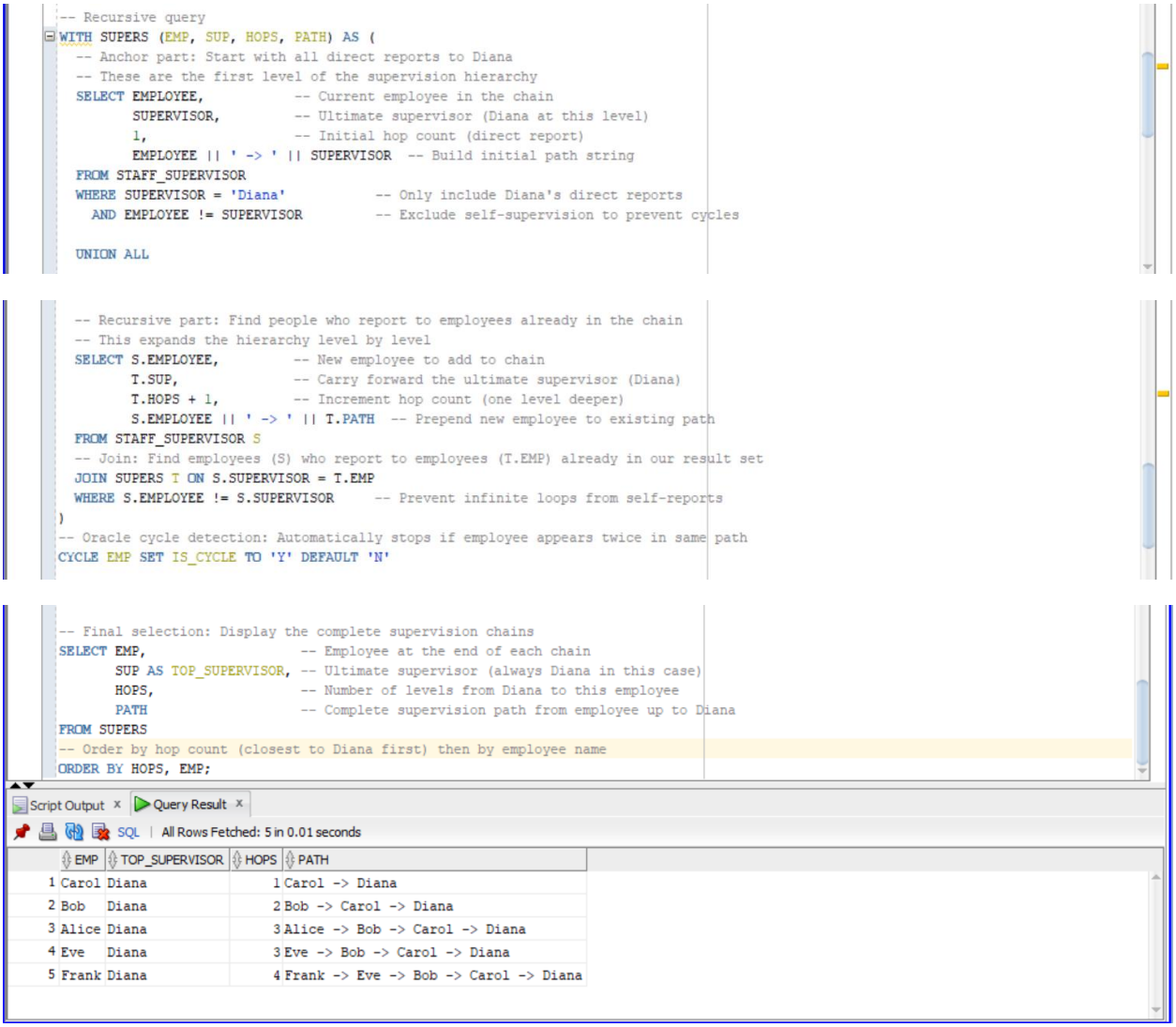


Figure 11: Created Staff_Supervisor Table and Inserted sample data



```
-- Recursive query
WITH SUPERS (EMP, SUP, HOPS, PATH) AS (
  -- Anchor part: Start with all direct reports to Diana
  -- These are the first level of the supervision hierarchy
  SELECT EMPLOYEE,           -- Current employee in the chain
         SUPERVISOR,         -- Ultimate supervisor (Diana at this level)
         1,                  -- Initial hop count (direct report)
         EMPLOYEE || ' -> ' || SUPERVISOR  -- Build initial path string
  FROM STAFF_SUPERVISOR
  WHERE SUPERVISOR = 'Diana'           -- Only include Diana's direct reports
    AND EMPLOYEE != SUPERVISOR         -- Exclude self-supervision to prevent cycles

  UNION ALL

  -- Recursive part: Find people who report to employees already in the chain
  -- This expands the hierarchy level by level
  SELECT S.EMPLOYEE,         -- New employee to add to chain
         T.SUP,              -- Carry forward the ultimate supervisor (Diana)
         T.HOPS + 1,         -- Increment hop count (one level deeper)
         S.EMPLOYEE || ' -> ' || T.PATH  -- Prepend new employee to existing path
  FROM STAFF_SUPERVISOR S
  -- Join: Find employees (S) who report to employees (T.EMP) already in our result set
  JOIN SUPERS T ON S.SUPERVISOR = T.EMP
  WHERE S.EMPLOYEE != S.SUPERVISOR     -- Prevent infinite loops from self-reports
)
-- Oracle cycle detection: Automatically stops if employee appears twice in same path
CYCLE EMP SET IS_CYCLE TO 'Y' DEFAULT 'N'

-- Final selection: Display the complete supervision chains
SELECT EMP,                  -- Employee at the end of each chain
       SUP AS TOP_SUPERVISOR, -- Ultimate supervisor (always Diana in this case)
       HOPS,                 -- Number of levels from Diana to this employee
       PATH                  -- Complete supervision path from employee up to Diana
FROM SUPERS
-- Order by hop count (closest to Diana first) then by employee name
ORDER BY HOPS, EMP;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.01 seconds

	EMP	TOP_SUPERVISOR	HOPS	PATH
1	Carol	Diana	1	Carol -> Diana
2	Bob	Diana	2	Bob -> Carol -> Diana
3	Alice	Diana	3	Alice -> Bob -> Carol -> Diana
4	Eve	Diana	3	Eve -> Bob -> Carol -> Diana
5	Frank	Diana	4	Frank -> Eve -> Bob -> Carol -> Diana

Figure 12: Corrected recursive Query and Results showing Staff and his/her top supervisor

Question_4: Knowledge Bases (Triples & Ontology): Infectious-Disease Roll-Up

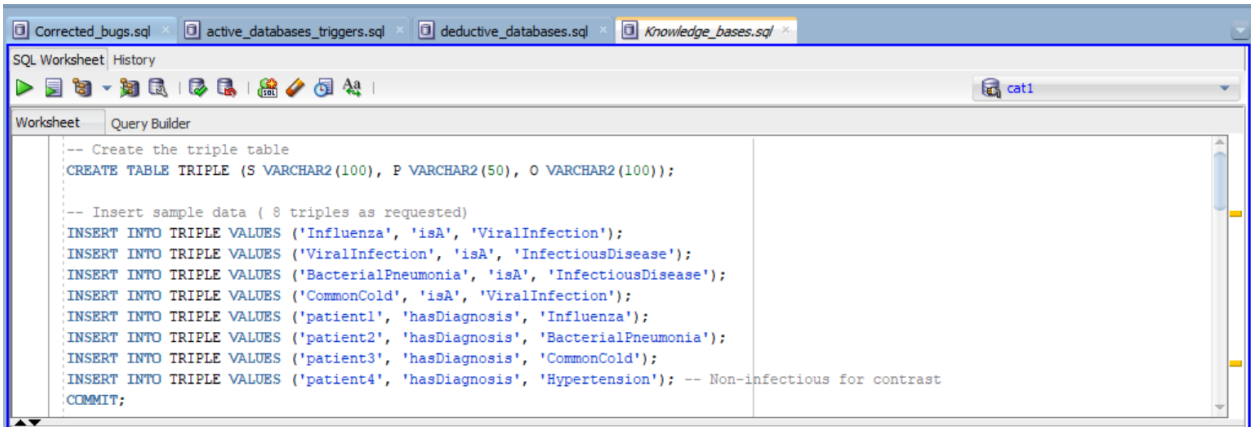


Figure 13: Table created and inserted triples

```
-- DEBUGGED SOLUTION
WITH ISA(CHILD, ANCESTOR) AS (
  /* ANCHOR: Start with direct parent-child relationships in taxonomy
     Example: ('Influenza', 'isA', 'ViralInfection') -> CHILD='Influenza', ANCESTOR='ViralInfection' */
  SELECT S AS CHILD, O AS ANCESTOR
  FROM TRIPLE
  WHERE P = 'isA'

  UNION ALL

  /* RECURSIVE: Build transitive closure - if X isA Y and Y isA Z, then X isA Z
     Fixed bug: Original had T.O = I.ANCESTOR (wrong direction) */
  SELECT T.S AS CHILD, I.ANCESTOR
  FROM TRIPLE T
  JOIN ISA I ON T.O = I.CHILD  -- Correct: T's parent (O) is I's child
  WHERE T.P = 'isA'
),
```

```
INFECTIOUS_PATIENTS AS (
  /* Find patients whose diagnosis ultimately rolls up to InfectiousDisease
     Fixed bug: Original compared ISA.CHILD instead of ISA.ANCESTOR */
  SELECT DISTINCT T.S AS PATIENT_ID
  FROM TRIPLE T
  JOIN ISA ON T.O = ISA.CHILD  -- Patient's diagnosis is the child in taxonomy
  WHERE T.P = 'hasDiagnosis'
    AND ISA.ANCESTOR = 'InfectiousDisease'  -- And it has InfectiousDisease as ancestor
)
-- Final result: Patients with infectious diseases
SELECT PATIENT_ID
FROM INFECTIOUS_PATIENTS
ORDER BY PATIENT_ID;
```

Figure 14: Corrected Tripple

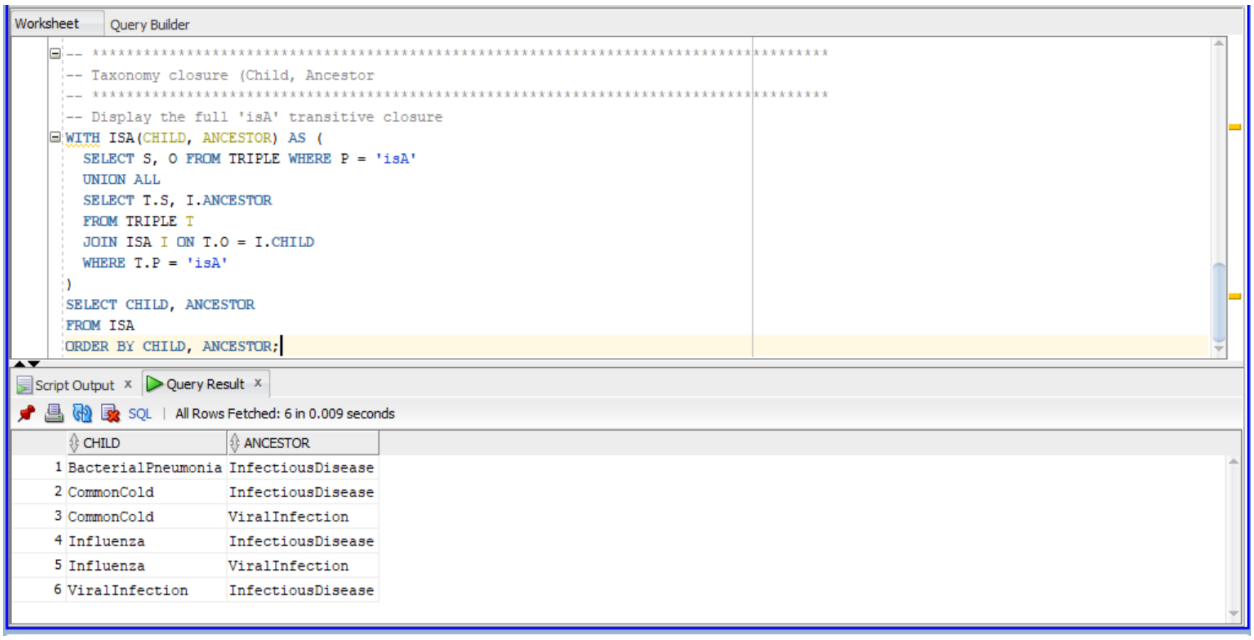


Figure 15: Closure view (Child, Ancestor)

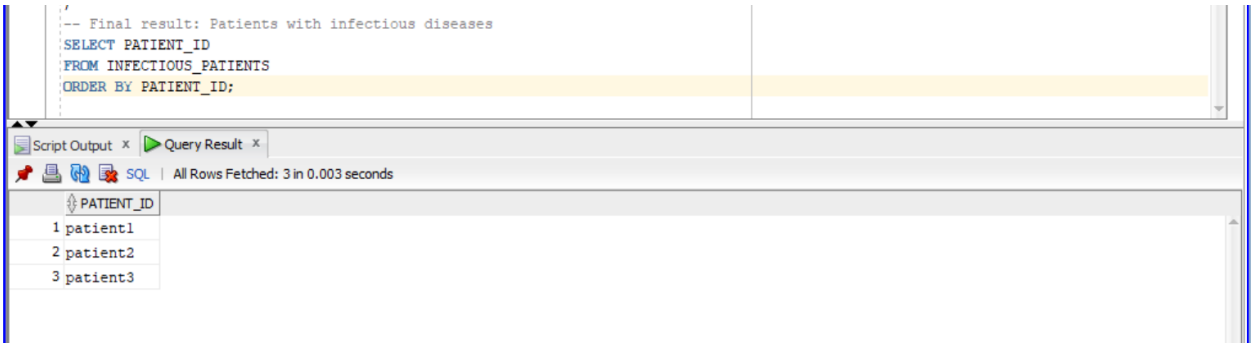


Figure 16: Final patient IDs

Question_5: Spatial Databases (Geography & Distance): Radius & Nearest-3

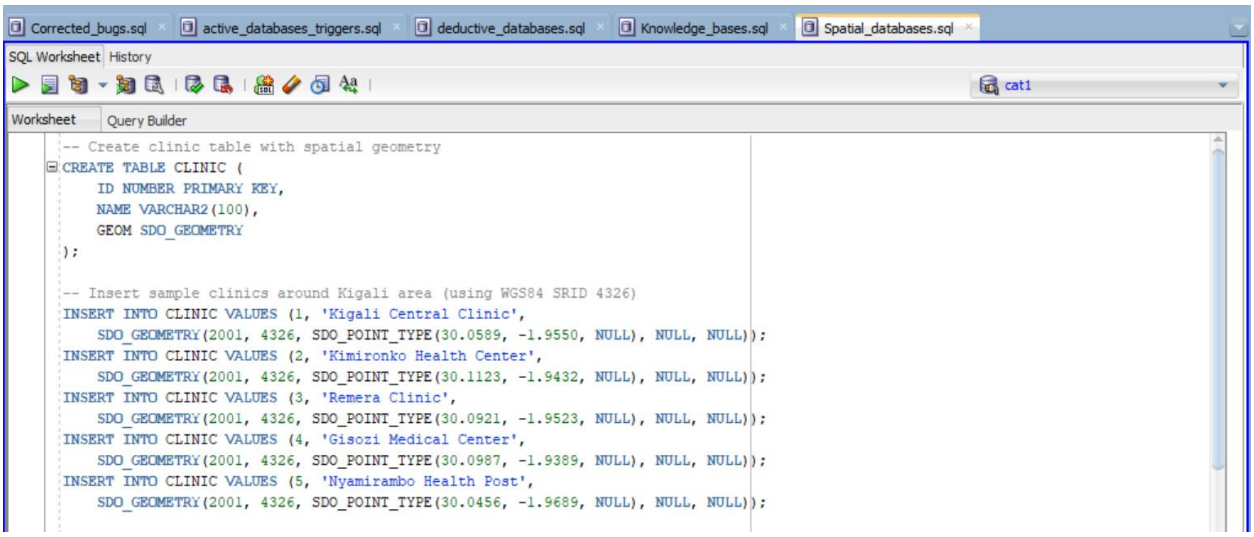


Figure 17: Table created and inserted data

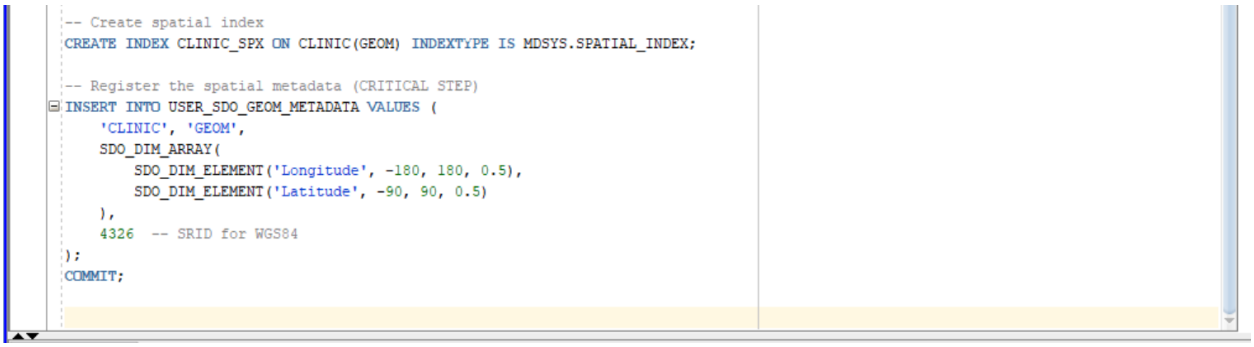


Figure 18: Spatial Index

```sql
-- Query 1: Clinics within 1 km radius (FIXED)
SELECT C.ID, C.NAME
FROM CLINIC C
WHERE SDO_WITHIN_DISTANCE(
    C.GEOM,
    SDO_GEOMETRY(
        2001,
        4326,  -- FIXED: WGS84 SRID (geographic coordinates)
        SDO_POINT_TYPE(30.0600, -1.9570, NULL),  -- FIXED: Longitude first, then Latitude
        NULL, NULL
    ),
    'distance=1 unit=KM'  -- FIXED: Explicitly specify kilometers
) = 'TRUE'
ORDER BY C.ID;
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 1 in 0.008 seconds

	ID	NAME
1	1	Kigali Central Clinic

Figure 19: Fixed query to list clinics within 1 kilometer

```sql
-- Query 2: Nearest 3 clinics with distances in kilometers (FIXED)
SELECT C.ID, C.NAME,
       ROUND(SDO_GEOM.SDO_DISTANCE(
           C.GEOM,
           SDO_GEOMETRY(2001, 4326, SDO_POINT_TYPE(30.0600, -1.9570, NULL), NULL, NULL),
           0.005,        -- Tolerance for WGS84 coordinate system
           'unit=KM'     -- FIXED: Return distance in kilometers
       ), 3) AS DISTANCE_KM
FROM CLINIC C
ORDER BY DISTANCE_KM
FETCH FIRST 3 ROWS ONLY;
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

SQL | All Rows Fetched: 3 in 0.01 seconds

	ID	NAME	DISTANCE_KM
1	1	Kigali Central Clinic	0.253
2	5	Nyamirambo Health Post	2.073
3	3	Remera Clinic	3.609

Figure 20: Three clinics with distances in kilometers

```sql
-- Verify clinic locations and spatial setup
SELECT ID, NAME,
       SDO_UTIL.TO_WKTGEOMETRY(GEOM) AS LOCATION
FROM CLINIC;

-- Verify spatial index is valid
SELECT INDEX_NAME, STATUS
FROM USER_INDEXES
WHERE INDEX_NAME = 'CLINIC_SPX';
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 5 in 0.008 seconds

	ID	NAME	LOCATION
1	1	Kigali Central Clinic	POINT (30.0589 -1.955)
2	2	Kimironko Health Center	POINT (30.1123 -1.9432)
3	3	Remera Clinic	POINT (30.0921 -1.9523)
4	4	Gisozi Medical Center	POINT (30.0987 -1.9389)
5	5	Nyamirambo Health Post	POINT (30.0456 -1.9689)

Figure 21: Verifying clinics locations