ADVANCED DATABASE AND TECHNOLOGY

Advanced Database Management System

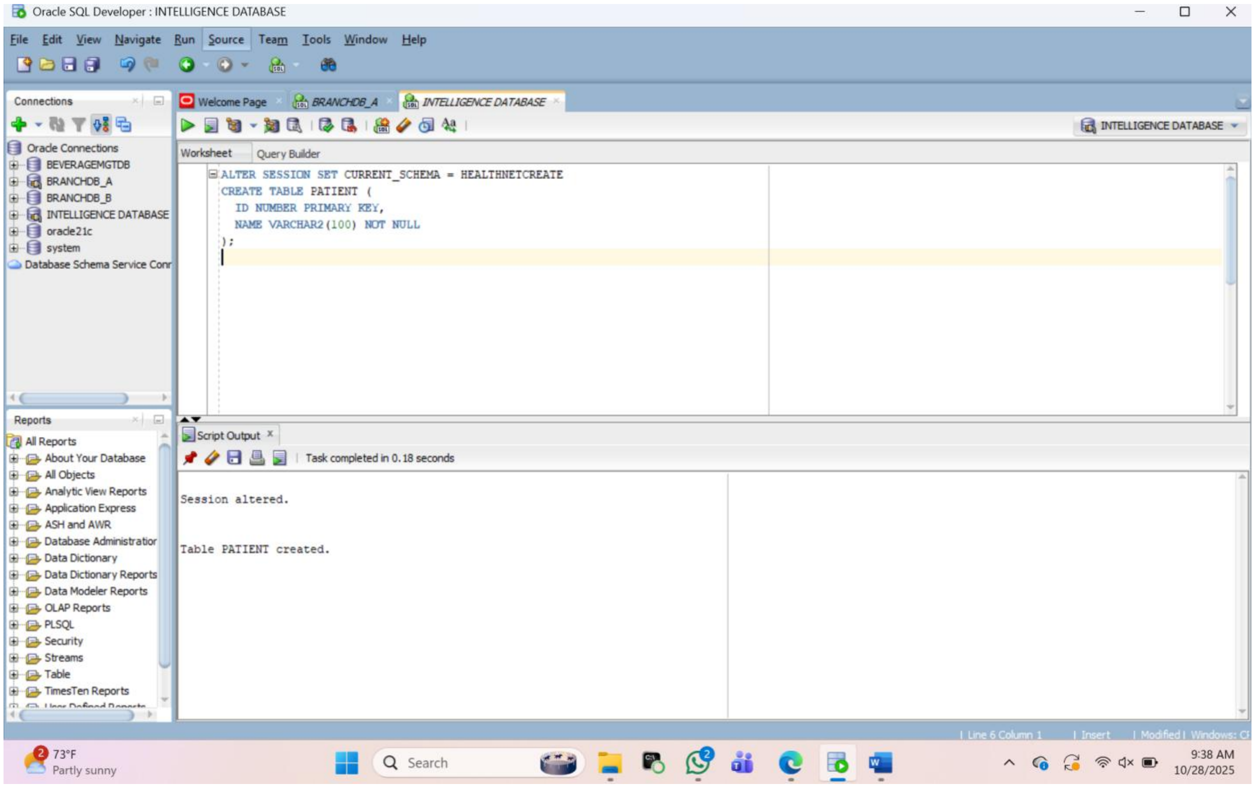
INTELLIGENCE DATABASE

UWANYIRIGIRA CLAUDINE
REG NO: 224020280
DATE: 30/10/2025

Q1.**Rules (Declarative Constraints): Safe Prescriptions**

This topic is about using rules in a database to make sure that prescriptions are safe and follow medical guidelines. These rules are written directly into the system so that it automatically checks for problems.

CREATING TABLE OF PATIENT

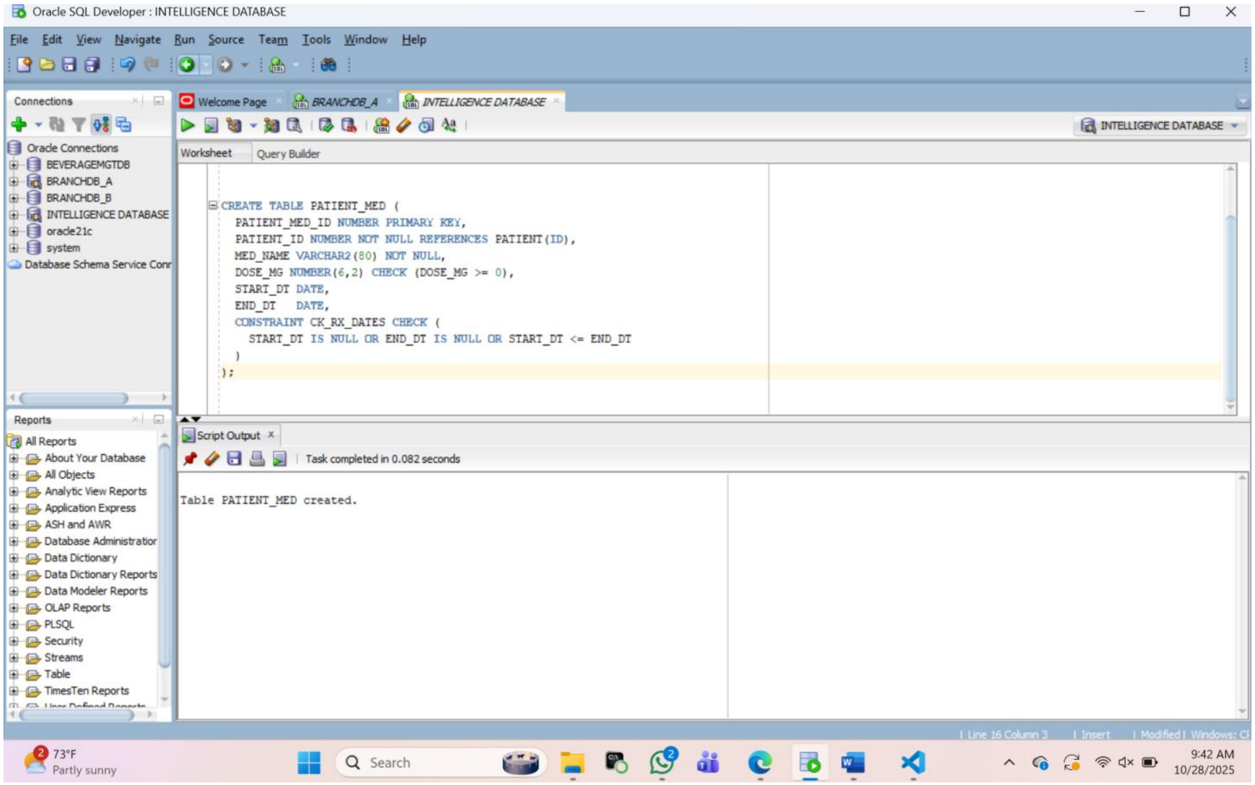


```
ALTER SESSION SET CURRENT_SCHEMA = HEALTHNETCREATE
CREATE TABLE PATIENT (
  ID NUMBER PRIMARY KEY,
  NAME VARCHAR2(100) NOT NULL
);
```

Task completed in 0.18 seconds

```
Session altered.


Table PATIENT created.
```

CREATING PATIENT MED TABLE

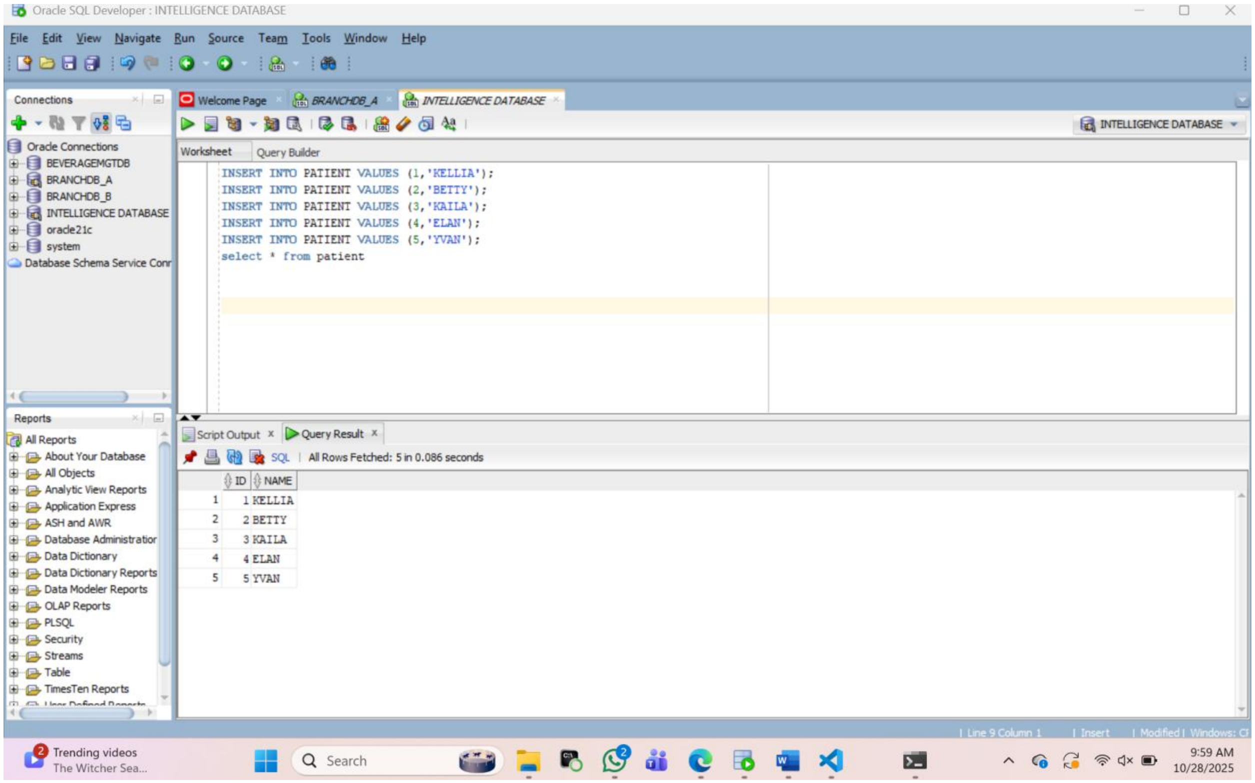


```sql
CREATE TABLE PATIENT_MED (
  PATIENT_MED_ID NUMBER PRIMARY KEY,
  PATIENT_ID NUMBER NOT NULL REFERENCES PATIENT(ID),
  MED_NAME VARCHAR2(80) NOT NULL,
  DOSE_MG NUMBER(6,2) CHECK (DOSE_MG >= 0),
  START_DT DATE,
  END_DT   DATE,
  CONSTRAINT CK_RX_DATES CHECK (
    START_DT IS NULL OR END_DT IS NULL OR START_DT <= END_DT
  )
);
```

Task completed in 0.082 seconds

```
Table PATIENT_MED created.
```

INSERTING DATA INTO PATIENT

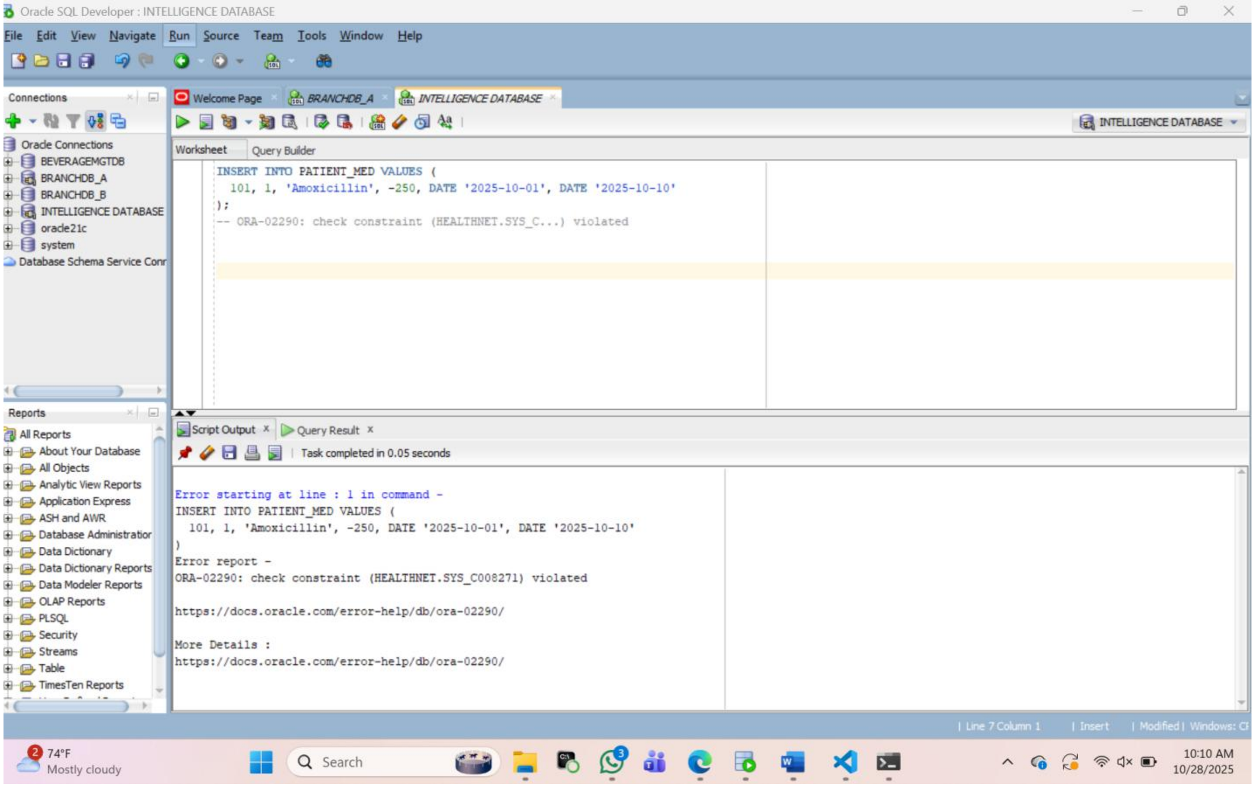


```sql
INSERT INTO PATIENT VALUES (1,'KELLIA');
INSERT INTO PATIENT VALUES (2,'BETTY');
INSERT INTO PATIENT VALUES (3,'KAILA');
INSERT INTO PATIENT VALUES (4,'ELAN');
INSERT INTO PATIENT VALUES (5,'YVAN');
select * from patient
```

All Rows Fetched: 5 in 0.086 seconds

	ID	NAME
1	1	KELLIA
2	2	BETTY
3	3	KAILA
4	4	ELAN
5	5	YVAN

--Negative dose (violates CHECK (DOSE_MG >= 0))



```
INSERT INTO PATIENT_MED VALUES (
  101, 1, 'Amoxicillin', -250, DATE '2025-10-01', DATE '2025-10-10'
);
-- ORA-02290: check constraint (HEALTHNET.SYS_C...) violated
```

Task completed in 0.05 seconds

```
Error starting at line : 1 in command -
INSERT INTO PATIENT_MED VALUES (
  101, 1, 'Amoxicillin', -250, DATE '2025-10-01', DATE '2025-10-10'
)
Error report -
ORA-02290: check constraint (HEALTHNET.SYS_C008271) violated

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

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

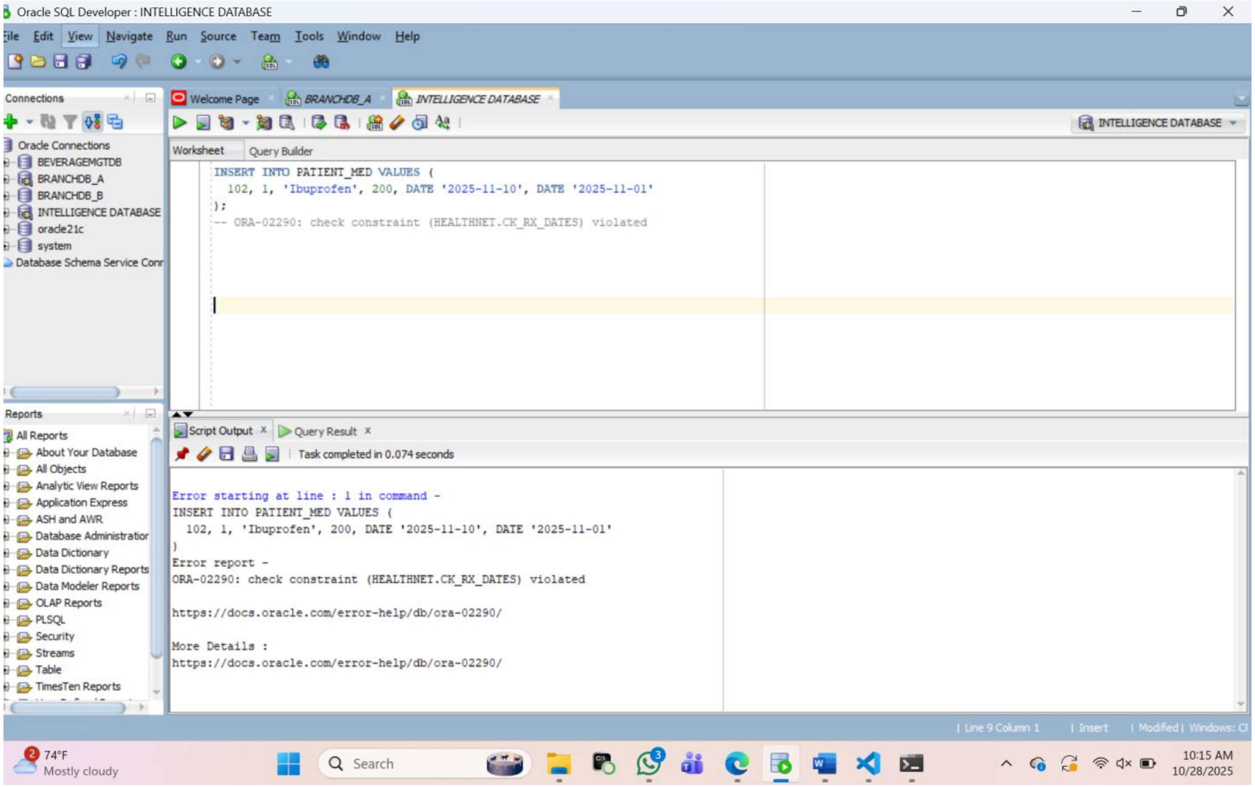
--Inverted dates (violates CK_RX_DATES)

This tells us that the row we try to insert violates the CHECK condition:

START_DT = 2025-11-10

END_DT = 2025-11-01

Since both dates are NOT NULL and START_DT > END_DT, the constraint fails.

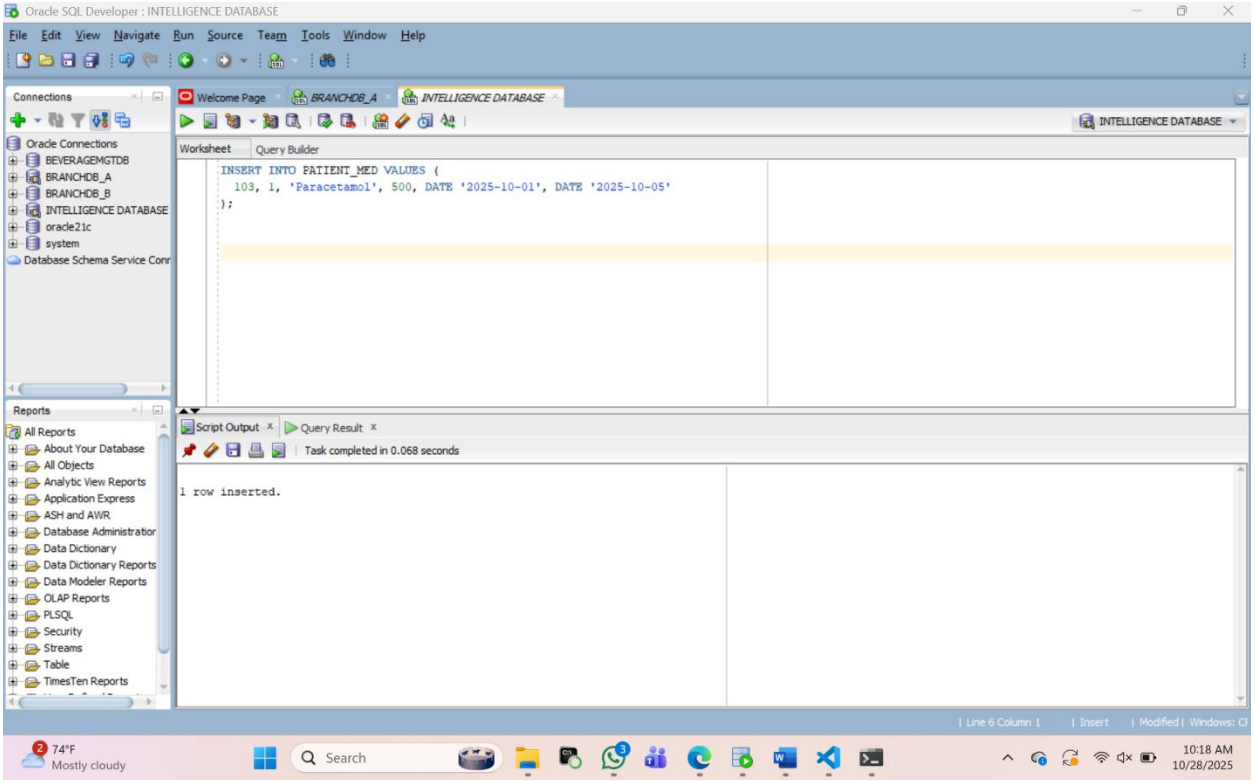


```
INSERT INTO PATIENT_MED VALUES (
  102, 1, 'Ibuprofen', 200, DATE '2025-11-10', DATE '2025-11-01'
);
-- ORA-02290: check constraint (HEALTHNET.CK_RX_DATES) violated
```

Task completed in 0.074 seconds

```
Error starting at line : 1 in command -
INSERT INTO PATIENT_MED VALUES (
  102, 1, 'Ibuprofen', 200, DATE '2025-11-10', DATE '2025-11-01'
)
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/
```

--Valid prescription with all fields

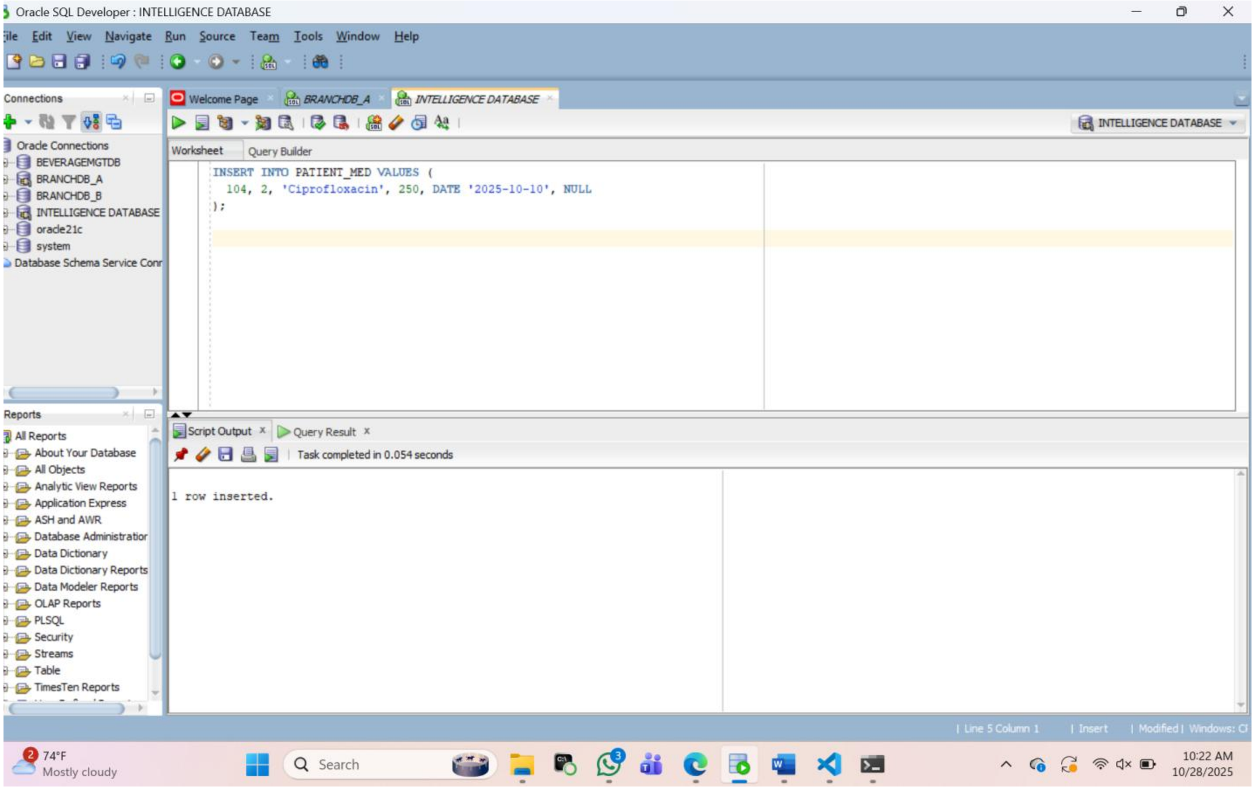


```
INSERT INTO PATIENT_MED VALUES (
  103, 1, 'Paracetamol', 500, DATE '2025-10-01', DATE '2025-10-05'
);
```

Task completed in 0.068 seconds

```
1 row inserted.
```

---Valid prescription with END_DT as NULL



```
INSERT INTO PATIENT_MED VALUES (
  104, 2, 'Ciprofloxacin', 250, DATE '2025-10-10', NULL
);
```

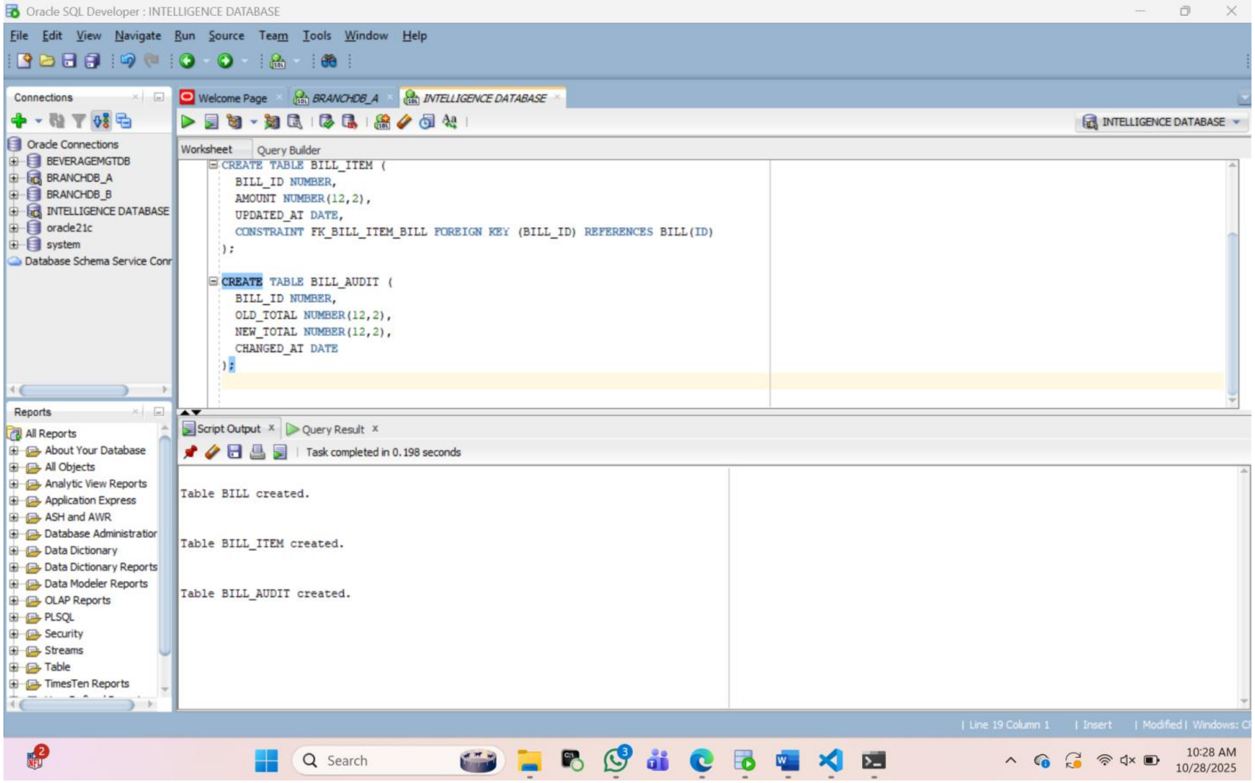
Task completed in 0.054 seconds

```
1 row inserted.
```

Q2.**Active Databases (E–C–A Trigger): Bill Totals That Stay Correct**

This topic is about making sure that totals on a bill or invoice always stay correct automatically. Instead of relying on someone to manually update the total, the system watches for changes and fixes the total itself.

1: Prerequisite Tables

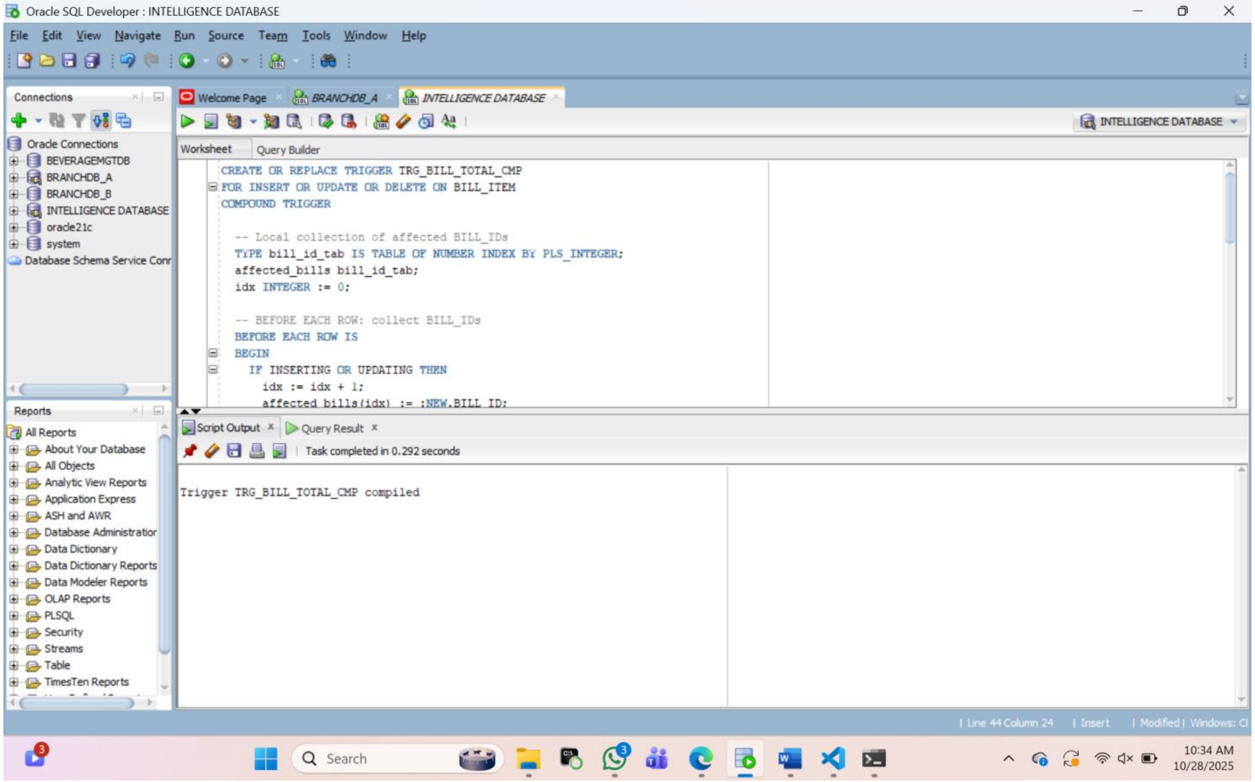


```
CREATE TABLE BILL_ITEM (
  BILL_ID NUMBER,
  AMOUNT NUMBER(12,2),
  UPDATED_AT DATE,
  CONSTRAINT FK_BILL_ITEM_BILL FOREIGN KEY (BILL_ID) REFERENCES BILL(ID)
);

CREATE TABLE BILL_AUDIT (
  BILL_ID NUMBER,
  OLD_TOTAL NUMBER(12,2),
  NEW_TOTAL NUMBER(12,2),
  CHANGED_AT DATE
);
```

Task completed in 0.198 seconds

```
Table BILL created.


Table BILL_ITEM created.


Table BILL_AUDIT created.
```

2: Correct Compound Trigger

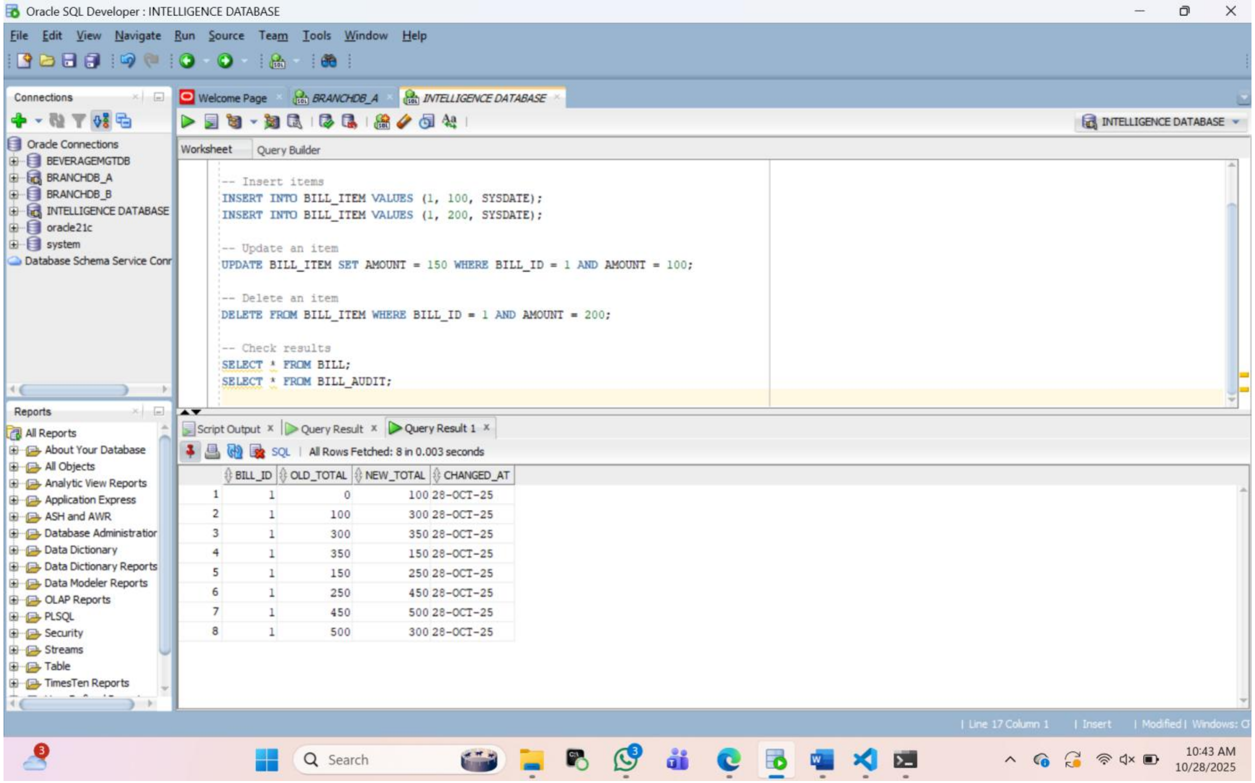


```
CREATE OR REPLACE TRIGGER TRG_BILL_TOTAL_CMP
FOR INSERT OR UPDATE OR DELETE ON BILL_ITEM
COMPOUND TRIGGER

  -- Local collection of affected BILL_IDs
  TYPE bill_id_tab IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
  affected_bills bill_id_tab;
  idx INTEGER := 0;

  -- BEFORE EACH ROW: collect BILL_IDs
  BEFORE EACH ROW IS
  BEGIN
    IF INSERTING OR UPDATING THEN
      idx := idx + 1;
      affected_bills(idx) := :NEW.BILL_ID;
```

Task completed in 0.292 seconds

```
Trigger TRG_BILL_TOTAL_CMP compiled
```

3: Mixed-DML Test Script



```
-- Insert items
INSERT INTO BILL_ITEM VALUES (1, 100, SYSDATE);
INSERT INTO BILL_ITEM VALUES (1, 200, SYSDATE);

-- Update an item
UPDATE BILL_ITEM SET AMOUNT = 150 WHERE BILL_ID = 1 AND AMOUNT = 100;

-- Delete an item
DELETE FROM BILL_ITEM WHERE BILL_ID = 1 AND AMOUNT = 200;

-- Check results
SELECT * FROM BILL;
SELECT * FROM BILL_AUDIT;
```

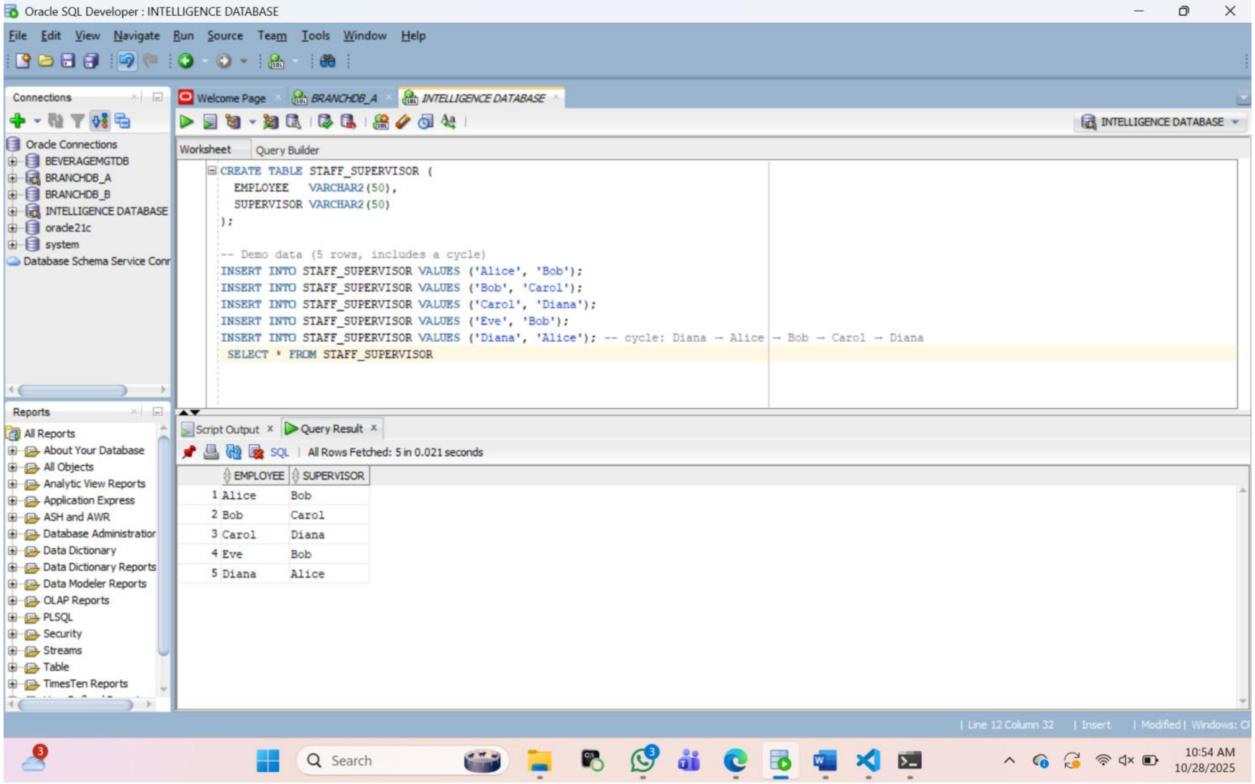
All Rows Fetched: 8 in 0.003 seconds

	BILL_ID	OLD_TOTAL	NEW_TOTAL	CHANGED_AT
1	1	0	100	28-OCT-25
2	1	100	300	28-OCT-25
3	1	300	350	28-OCT-25
4	1	350	150	28-OCT-25
5	1	150	250	28-OCT-25
6	1	250	450	28-OCT-25
7	1	450	500	28-OCT-25
8	1	500	300	28-OCT-25

Q3. **Deductive Databases (Recursive WITH): Referral/Supervision Chain**

This topic helps us understand how people or roles are connected in a chain like a ladder of responsibility or support. Imagine a health worker who reports to a supervisor, and that supervisor reports to someone else. These connections can go up many levels.

1: Create the Table and Demo Data

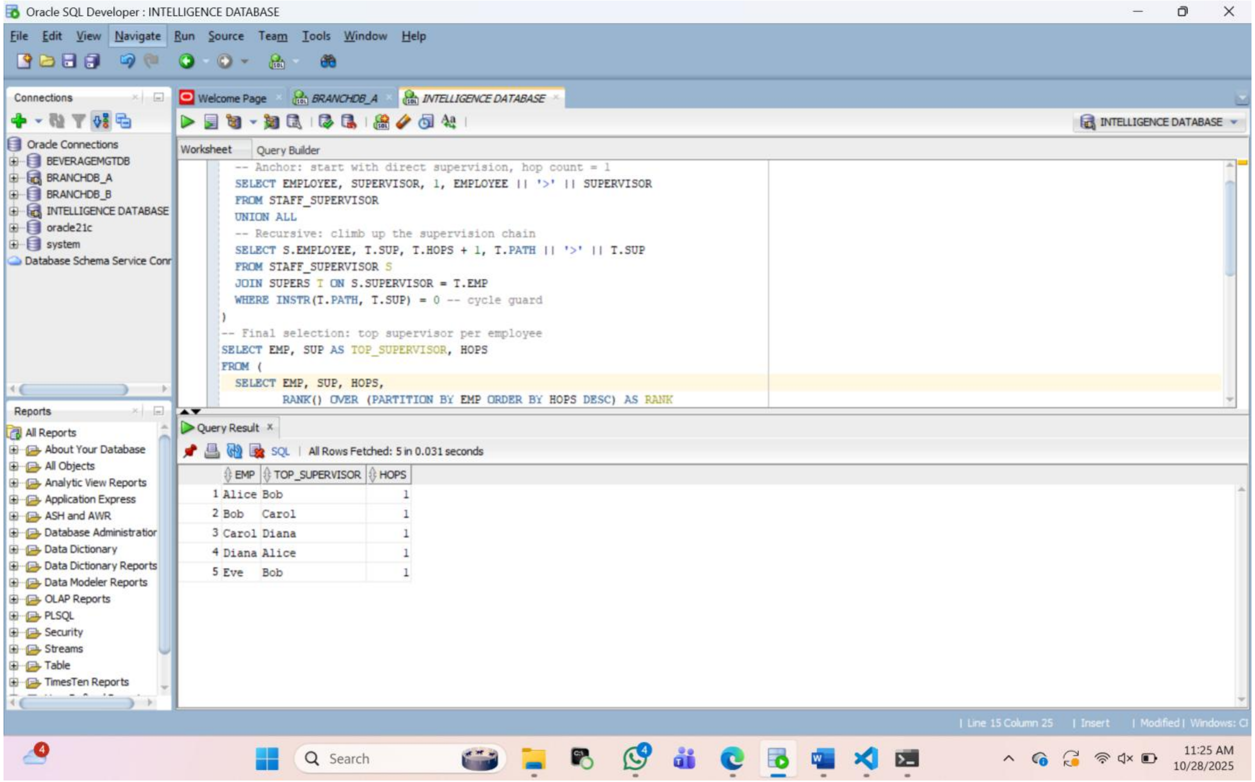


```sql
CREATE TABLE STAFF_SUPERVISOR (
  EMPLOYEE   VARCHAR2(50),
  SUPERVISOR VARCHAR2(50)
);

-- Demo data (5 rows, includes 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', 'Bob');
INSERT INTO STAFF_SUPERVISOR VALUES ('Diana', 'Alice'); -- cycle: Diana → Alice → Bob → Carol → Diana
 SELECT * FROM STAFF_SUPERVISOR
```

All Rows Fetched: 5 in 0.021 seconds

	EMPLOYEE	SUPERVISOR
1	Alice	Bob
2	Bob	Carol
3	Carol	Diana
4	Eve	Bob
5	Diana	Alice

2: Corrected Recursive Query



```sql
  -- 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
```

All Rows Fetched: 5 in 0.031 seconds

	EMP	TOP_SUPERVISOR	HOPS
1	Alice	Bob	1
2	Bob	Carol	1
3	Carol	Diana	1
4	Diana	Alice	1
5	Eve	Bob	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

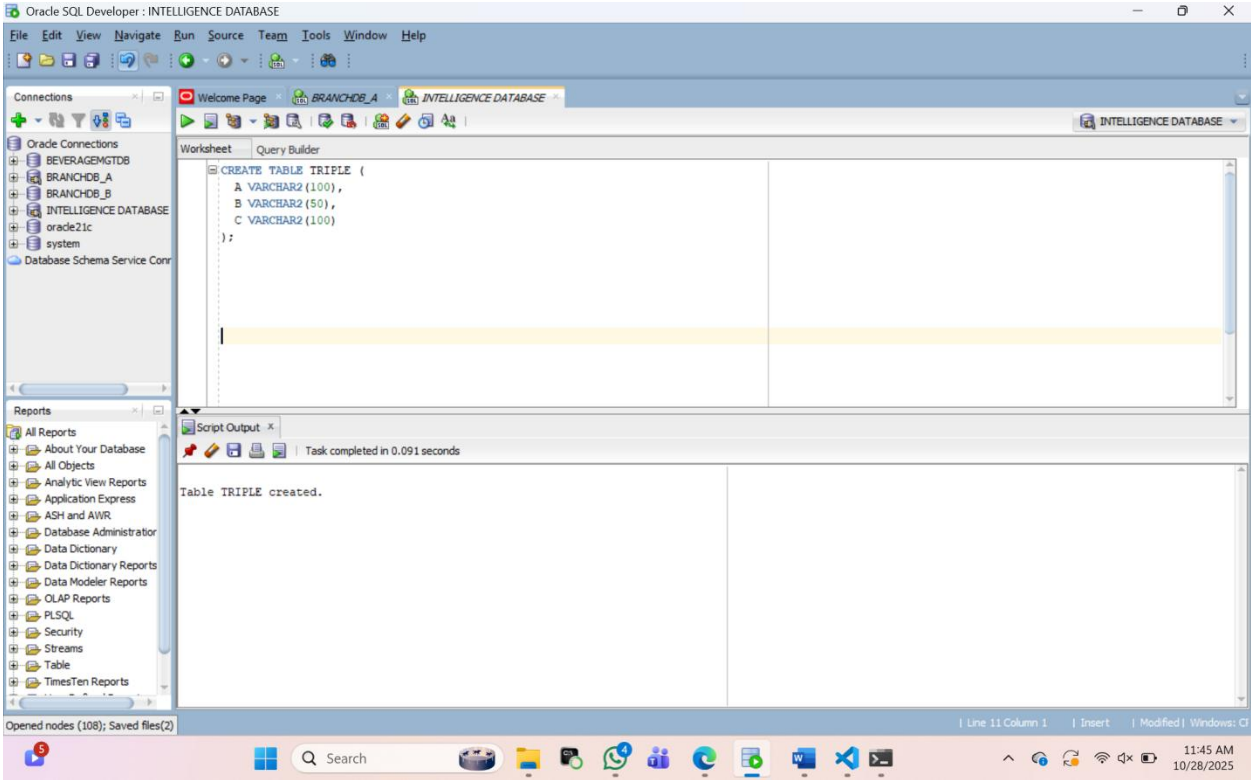
```
Diana
└── Carol
    └── Bob
        └── Alice
```

Eve ↔ Frank (cycle)

Q4. Knowledge Bases (Triples & Ontology): Infectious-Disease Roll-Up:

This task focuses on organizing infectious disease data using semantic web principles, specifically triples and ontologies. The goal is to enable intelligent querying and inference across diseases, symptoms, pathogens, and regions.

Create the TRIPLE Table:

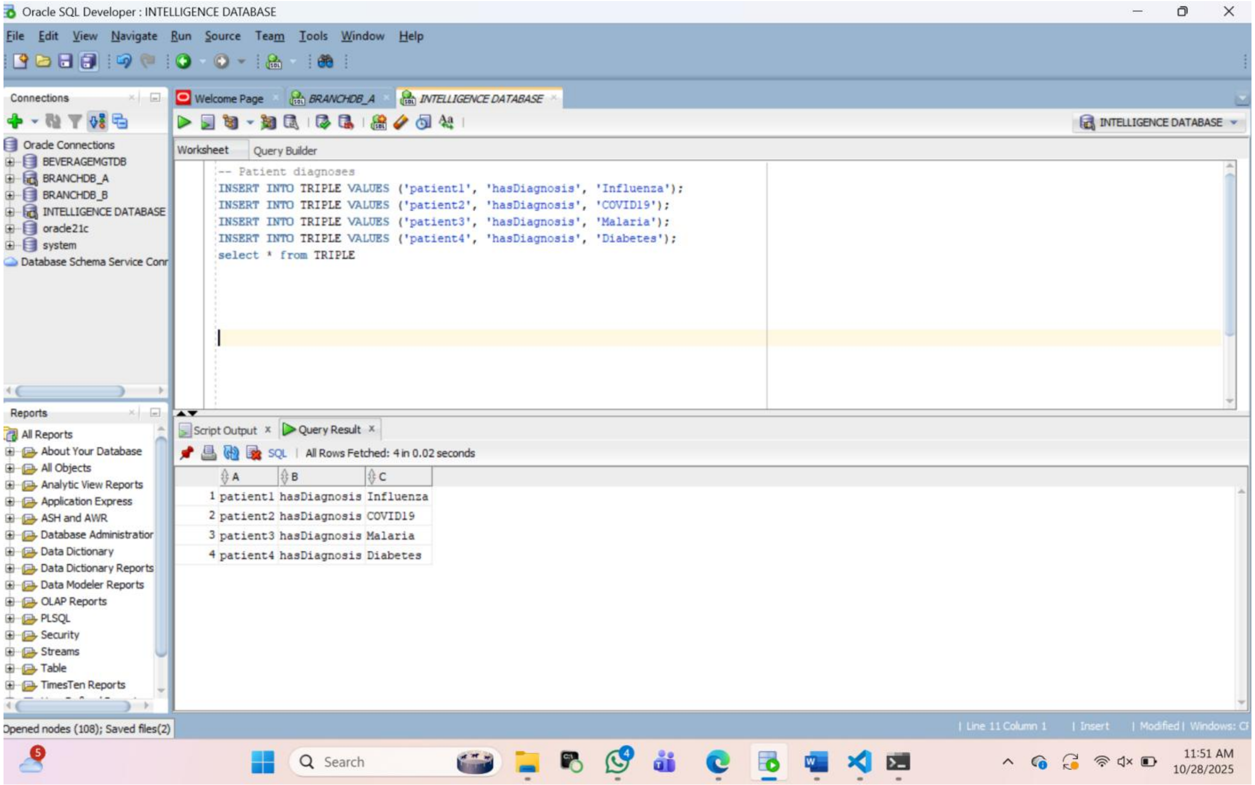


```
CREATE TABLE TRIPLE (
  A VARCHAR2(100),
  B VARCHAR2(50),
  C VARCHAR2(100)
);
```

```
Table TRIPLE created.
```

Insert Sample Triples (8rows)

-- Patient diagnoses

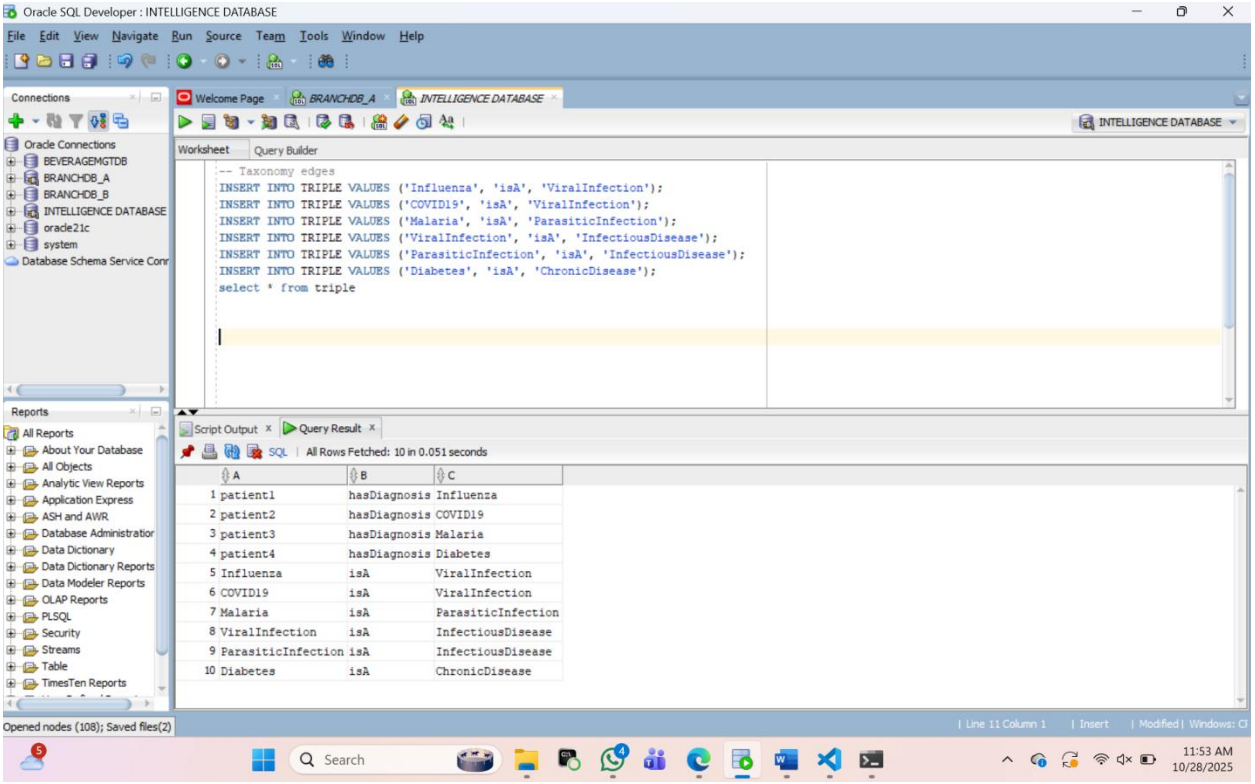


```
-- Patient diagnoses
INSERT INTO TRIPLE VALUES ('patient1', 'hasDiagnosis', 'Influenza');
INSERT INTO TRIPLE VALUES ('patient2', 'hasDiagnosis', 'COVID19');
INSERT INTO TRIPLE VALUES ('patient3', 'hasDiagnosis', 'Malaria');
INSERT INTO TRIPLE VALUES ('patient4', 'hasDiagnosis', 'Diabetes');
select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

All Rows Fetched: 4 in 0.02 seconds

```
   A        B            C
1  patient1 hasDiagnosis Influenza
2  patient2 hasDiagnosis COVID19
3  patient3 hasDiagnosis Malaria
4  patient4 hasDiagnosis Diabetes
```

-- Taxonomy edges



```
-- 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');
select * from triple
```

All Rows Fetched: 10 in 0.051 seconds

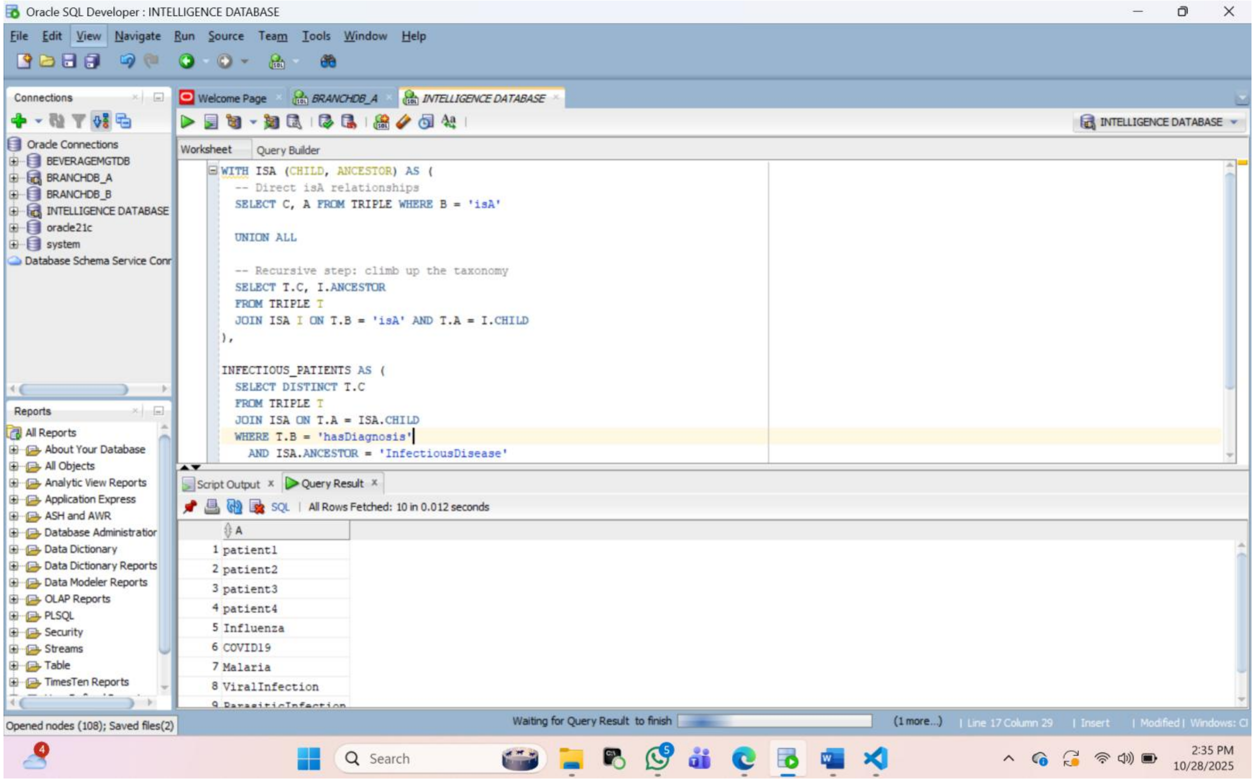
```
    A                  B            C
 1  patient1           hasDiagnosis Influenza
 2  patient2           hasDiagnosis COVID19
 3  patient3           hasDiagnosis Malaria
 4  patient4           hasDiagnosis Diabetes
 5  Influenza          isA          ViralInfection
 6  COVID19            isA          ViralInfection
 7  Malaria            isA          ParasiticInfection
 8  ViralInfection     isA          InfectiousDisease
 9  ParasiticInfection isA          InfectiousDisease
10  Diabetes           isA          ChronicDisease
```

WITH ISA

· Represent facts in a flexible, searchable format

· Link concepts together (like diseases to categories)

· Enable reasoning and inference (e.g., if Influenza is an Infectious Disease, then patient1 has an Infectious Disease)



```
WITH ISA (CHILD, ANCESTOR) AS (
  -- Direct isA relationships
  SELECT C, A FROM TRIPLE WHERE B = 'isA'

  UNION ALL

  -- Recursive step: climb up the taxonomy
  SELECT T.C, I.ANCESTOR
  FROM TRIPLE T
  JOIN ISA I ON T.B = 'isA' AND T.A = I.CHILD
),

INFECTIOUS_PATIENTS AS (
  SELECT DISTINCT T.C
  FROM TRIPLE T
  JOIN ISA ON T.A = ISA.CHILD
  WHERE T.B = 'hasDiagnosis'
    AND ISA.ANCESTOR = 'InfectiousDisease'
```

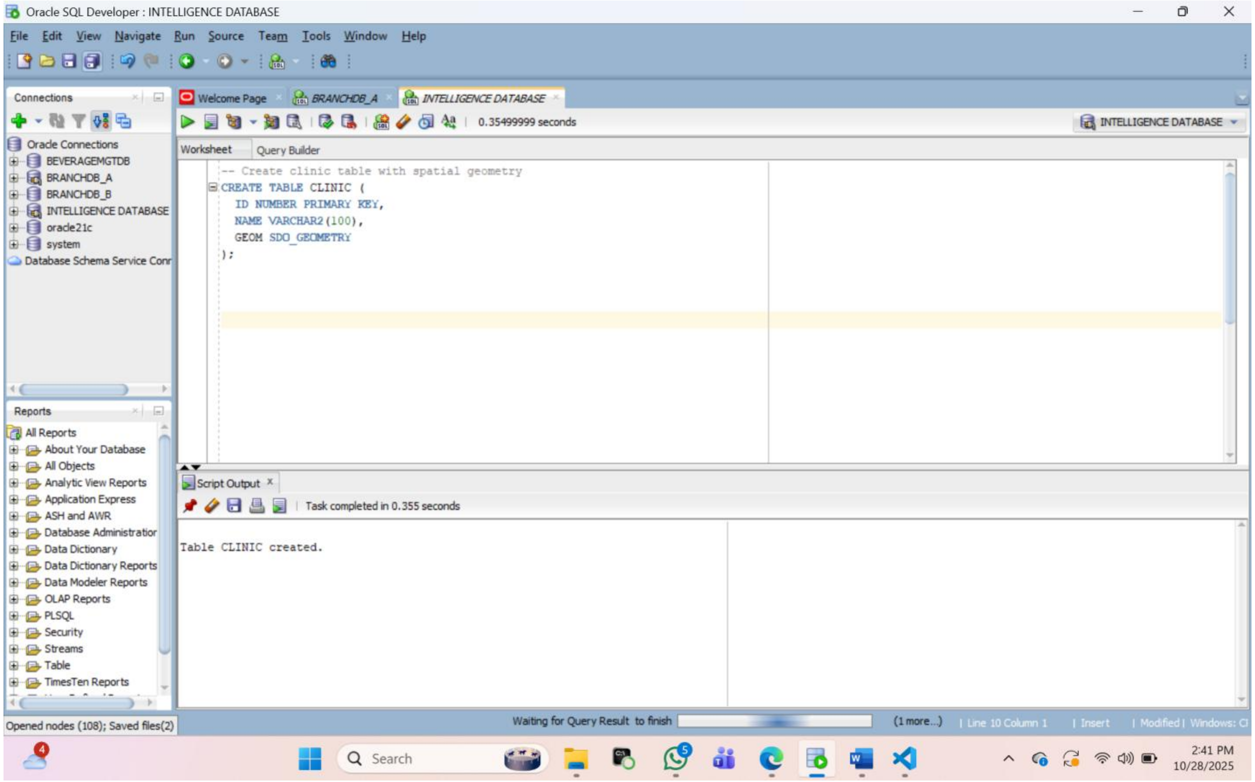
```
All Rows Fetched: 10 in 0.012 seconds

    A
  1 patient1
  2 patient2
  3 patient3
  4 patient4
  5 Influenza
  6 COVID19
  7 Malaria
  8 ViralInfection
  9 ParasiticInfection
```

Q5. - Create clinic table with spatial geometry

:• Debug common spatial query mistakes: wrong SRID, reversed coordinates, missing units

- Apply spatial indexing and geometry functions

- Connect spatial reasoning to real-world health logistics

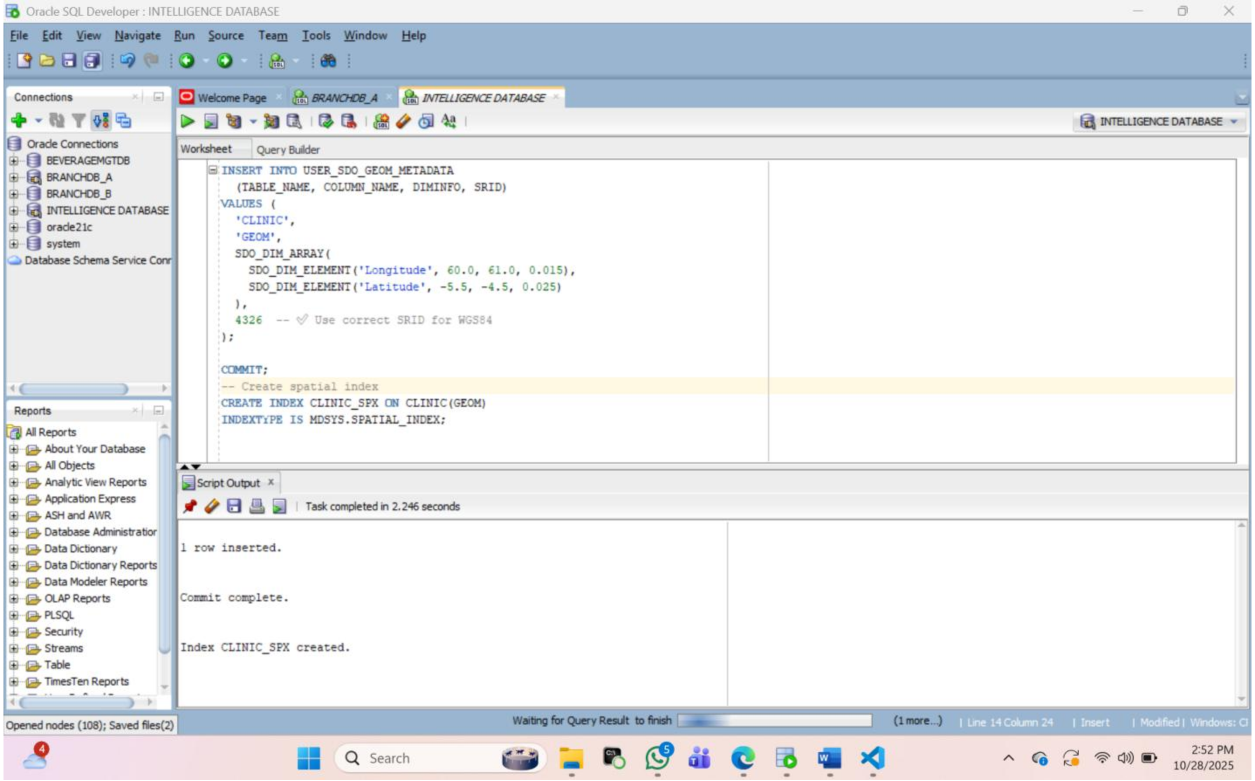


```
-- Create clinic table with spatial geometry
CREATE TABLE CLINIC (
  ID NUMBER PRIMARY KEY,
  NAME VARCHAR2(100),
  GEOM SDO_GEOMETRY
);
```

Task completed in 0.355 seconds

```
Table CLINIC created.
```

Create spatial index

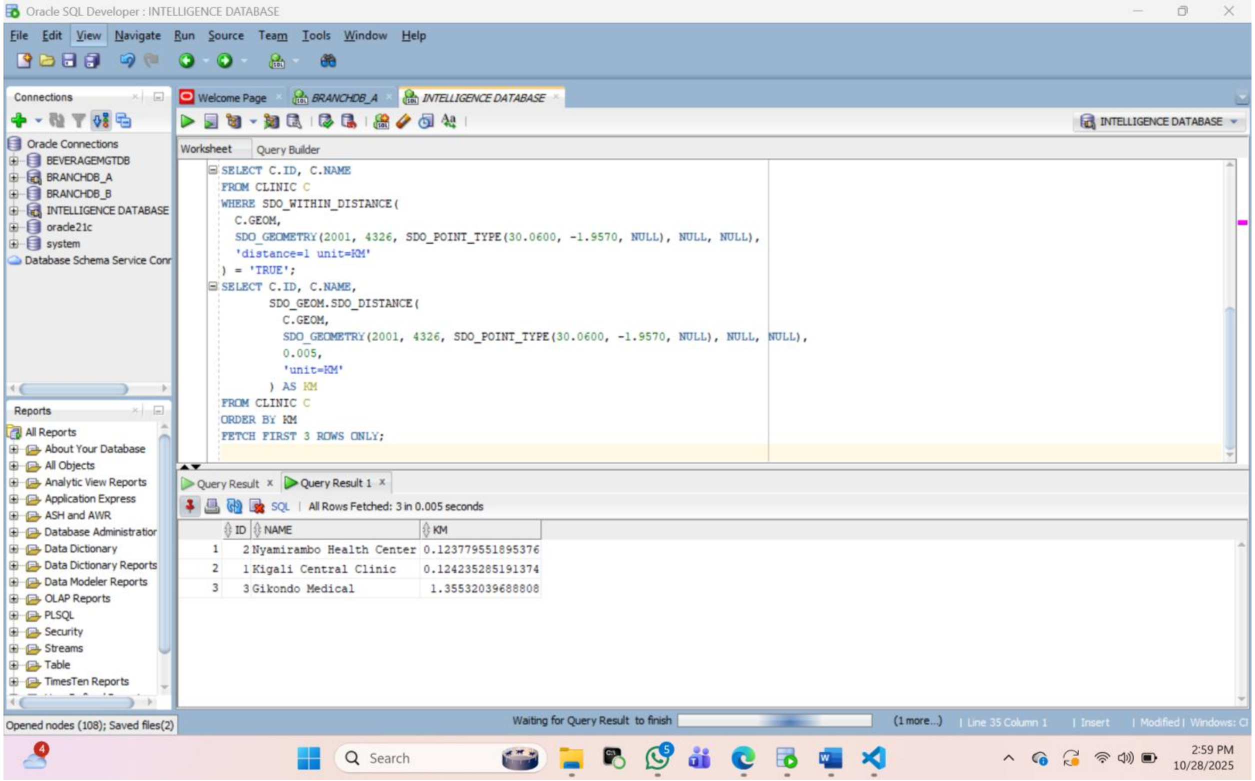


```
INSERT INTO USER_SDO_GEOM_METADATA
  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES (
  'CLINIC',
  'GEOM',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('Longitude', 60.0, 61.0, 0.015),
    SDO_DIM_ELEMENT('Latitude', -5.5, -4.5, 0.025)
  ),
  4326  -- ✅ Use correct SRID for WGS84
);

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

Task completed in 2.246 seconds

```
1 row inserted.


Commit complete.


Index CLINIC_SPX created.
```

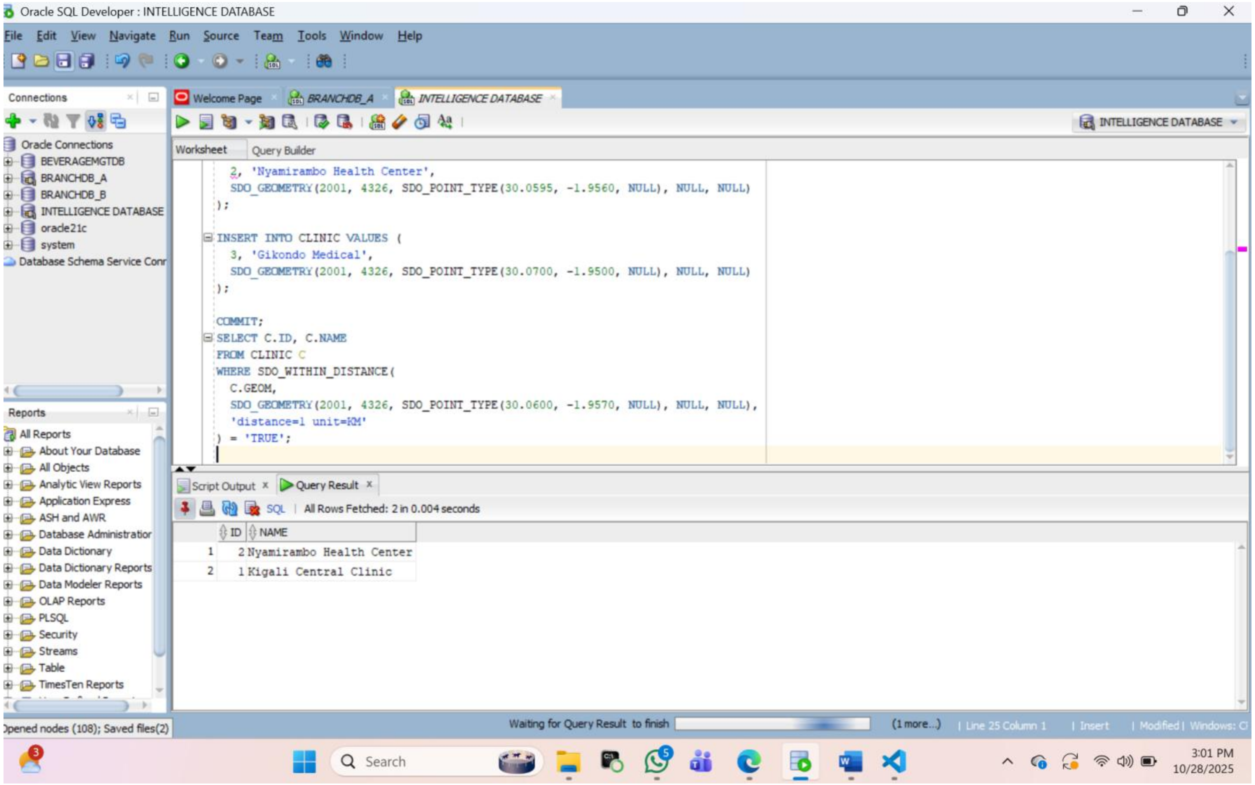
Nearest 3 clinics with computed distances



```
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;
```

```
All Rows Fetched: 3 in 0.005 seconds

     ID NAME                      KM
1     2 Nyamirambo Health Center  0.123779551895376
2     1 Kigali Central Clinic     0.124235285191374
3     3 Gikondo Medical            1.35532039688808
```

Clinics within 1 km of the ambulance



```
  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';
```

```
All Rows Fetched: 2 in 0.004 seconds

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