**Question 1: Using Function to Calculate Average Monthly Payment and Demographics by Province and Year**

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION CalcAvgPaymentAndDemographics (

p\_province\_code IN VARCHAR2,

p\_year IN NUMBER

) RETURN VARCHAR2 IS

v\_avg\_payment NUMBER;

v\_total\_payment NUMBER := 0;

v\_month\_count NUMBER := 0;

v\_age\_avg\_amount NUMBER;

v\_prov\_total\_amount NUMBER;

v\_no\_data EXCEPTION;

v\_output VARCHAR2(4000);

v\_prov\_column VARCHAR2(30); -- To dynamically select the column for the province

BEGIN

-- Map province code to column name in NUMBYPROV

CASE p\_province\_code

WHEN 'NL' THEN v\_prov\_column := 'NL\_MALE';

WHEN 'PEI' THEN v\_prov\_column := 'PEI\_MALE';

WHEN 'NS' THEN v\_prov\_column := 'NS\_MALE';

WHEN 'NB' THEN v\_prov\_column := 'NB\_MALE';

WHEN 'QUE' THEN v\_prov\_column := 'QUE\_MALE';

WHEN 'ONT' THEN v\_prov\_column := 'ONT\_MALE';

WHEN 'MAN' THEN v\_prov\_column := 'MAN\_MALE';

WHEN 'SASK' THEN v\_prov\_column := 'SASK\_MALE';

WHEN 'ALTA' THEN v\_prov\_column := 'ALTA\_MALE';

WHEN 'BC' THEN v\_prov\_column := 'BC\_MALE';

WHEN 'YT' THEN v\_prov\_column := 'YT\_MALE';

WHEN 'NWT' THEN v\_prov\_column := 'NWT\_MALE';

WHEN 'NVT' THEN v\_prov\_column := 'NVT\_MALE';

WHEN 'OUT\_OF\_CANADA' THEN v\_prov\_column := 'OUT\_OF\_CANADA\_MALE';

ELSE

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid province code');

END CASE;

-- Calculate total monthly payments and number of months for the given province and year

BEGIN

EXECUTE IMMEDIATE

'SELECT SUM(' || v\_prov\_column || '), COUNT(\*) FROM NUMBYPROV WHERE "YEAR" = :1'

INTO v\_total\_payment, v\_month\_count

USING p\_year;

-- Calculate the average monthly payment

IF v\_month\_count > 0 THEN

v\_avg\_payment := v\_total\_payment / v\_month\_count;

ELSE

v\_avg\_payment := NULL;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_avg\_payment := NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error calculating average payment: ' || SQLERRM);

RETURN 'Error occurred';

END;

-- Retrieve the average amount by age group

BEGIN

EXECUTE IMMEDIATE

'SELECT AVG((UNDER\_25\_MALE + A25\_29\_MALE + A30\_34\_MALE + A35\_39\_MALE + A40\_44\_MALE + A45\_49\_MALE + A50\_54\_MALE + A55\_59\_MALE + A60\_64\_MALE) / 9)

FROM AMOUNTBYAGE WHERE "YEAR" = :1'

INTO v\_age\_avg\_amount

USING p\_year;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_age\_avg\_amount := NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error retrieving age average amount: ' || SQLERRM);

RETURN 'Error occurred';

END;

-- Retrieve the total amount by province

BEGIN

EXECUTE IMMEDIATE

'SELECT ' || v\_prov\_column || ' FROM AMOUNTBYPROV WHERE "YEAR" = :1'

INTO v\_prov\_total\_amount

USING p\_year;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_prov\_total\_amount := NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error retrieving provincial total amount: ' || SQLERRM);

RETURN 'Error occurred';

END;

-- Prepare the output

v\_output := 'Province: ' || p\_province\_code || CHR(10) ||

'Year: ' || p\_year || CHR(10) ||

'Average Monthly Payment: ' || NVL(TO\_CHAR(v\_avg\_payment), 'No data available') || CHR(10) ||

'Average Amount by Age Group: ' || NVL(TO\_CHAR(v\_age\_avg\_amount), 'No data available') || CHR(10) ||

'Total Amount by Province: ' || NVL(TO\_CHAR(v\_prov\_total\_amount), 'No data available');

RETURN v\_output;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

RETURN 'Error occurred';

END CalcAvgPaymentAndDemographics;

/

-- Single call to display the output

DECLARE

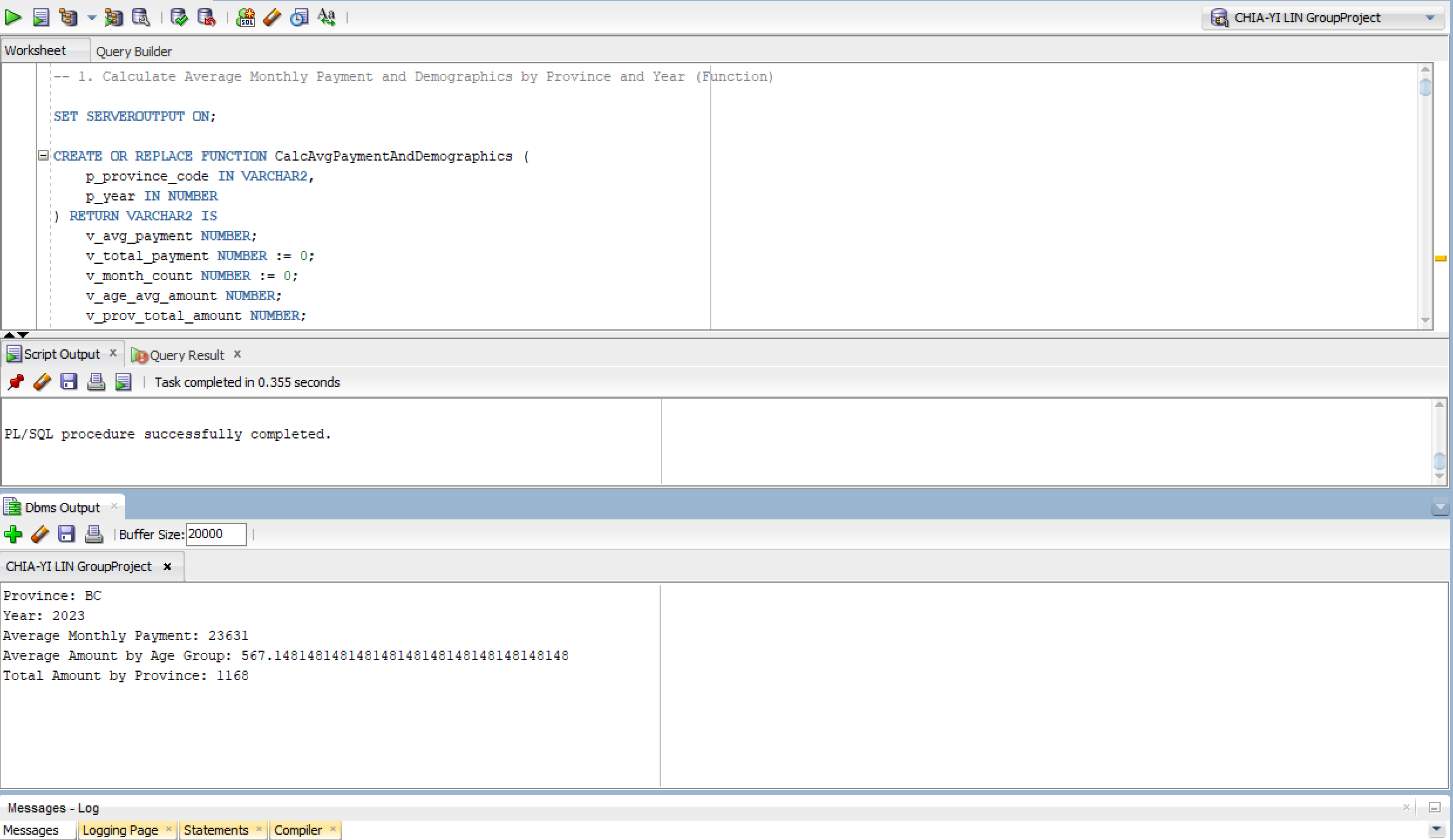
v\_result VARCHAR2(4000);

BEGIN

v\_result := CalcAvgPaymentAndDemographics('BC', 2023);

DBMS\_OUTPUT.PUT\_LINE(v\_result);

END;

/

**Question 2: Using a package, retrieve the total amount by province and gender for a given year**

SET SERVEROUTPUT ON;

-- Package specification

CREATE OR REPLACE PACKAGE DemographicPackage AS

PROCEDURE GetTotalAmount(p\_year IN NUMBER);

END DemographicPackage;

/

-- Package body

CREATE OR REPLACE PACKAGE BODY DemographicPackage AS

PROCEDURE GetTotalAmount(p\_year IN NUMBER) IS

v\_amount NUMBER;

v\_province\_code VARCHAR2(30);

CURSOR c\_provinces IS

SELECT 'NL' AS province FROM DUAL UNION ALL

SELECT 'PEI' FROM DUAL UNION ALL

SELECT 'NS' FROM DUAL UNION ALL

SELECT 'NB' FROM DUAL UNION ALL

SELECT 'QUE' FROM DUAL UNION ALL

SELECT 'ONT' FROM DUAL UNION ALL

SELECT 'MAN' FROM DUAL UNION ALL

SELECT 'SASK' FROM DUAL UNION ALL

SELECT 'ALTA' FROM DUAL UNION ALL

SELECT 'BC' FROM DUAL UNION ALL

SELECT 'YT' FROM DUAL UNION ALL

SELECT 'NWT' FROM DUAL UNION ALL

SELECT 'NVT' FROM DUAL UNION ALL

SELECT 'OUT\_OF\_CANADA' FROM DUAL;

BEGIN

FOR r IN c\_provinces LOOP

v\_province\_code := r.province;

-- Male total

BEGIN

EXECUTE IMMEDIATE

'SELECT ' || v\_province\_code || '\_MALE FROM NUMBYPROV WHERE "YEAR" = :1'

INTO v\_amount

USING p\_year;

DBMS\_OUTPUT.PUT\_LINE('Total Male Amount for ' || v\_province\_code || ' in ' || p\_year || ': ' || NVL(v\_amount, 0));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No male data found for ' || v\_province\_code || ' in ' || p\_year);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error retrieving male amount for ' || v\_province\_code || ': ' || SQLERRM);

END;

-- Female total

BEGIN

EXECUTE IMMEDIATE

'SELECT ' || v\_province\_code || '\_FEMALE FROM NUMBYPROV WHERE "YEAR" = :1'

INTO v\_amount

USING p\_year;

DBMS\_OUTPUT.PUT\_LINE('Total Female Amount for ' || v\_province\_code || ' in ' || p\_year || ': ' || NVL(v\_amount, 0));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No female data found for ' || v\_province\_code || ' in ' || p\_year);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error retrieving female amount for ' || v\_province\_code || ': ' || SQLERRM);

END;

END LOOP;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

END GetTotalAmount;

END DemographicPackage;

/

-- Example call to the package procedure

BEGIN

DemographicPackage.GetTotalAmount(2023);

END;

/

A screenshot of a computer

Description automatically generated

**Question 3: Using Function and Procedure to Calculate Average Monthly CPP Payment Per Person Each Year.**

CREATE OR REPLACE FUNCTION average\_payment\_per\_person(

p\_year IN NUMBER

) RETURN NUMBER

IS

lv\_payment\_amount NUMBER := 0;

lv\_person\_number NUMBER := 0;

lv\_payment\_per\_person NUMBER := 0;

BEGIN

-- Fetch the total payment amount and total number of people for the given year

SELECT

SUM(a.NL\_MALE + a.PEI\_MALE + a.NS\_MALE + a.NB\_MALE + a.QUE\_MALE + a.ONT\_MALE +

a.MAN\_MALE + a.SASK\_MALE + a.ALTA\_MALE + a.BC\_MALE + a.YT\_MALE + a.NWT\_MALE +

a.NVT\_MALE + a.OUT\_OF\_CANADA\_MALE + a.ALL\_LOCATIONS\_MALE +

a.NL\_FEMALE + a.PEI\_FEMALE + a.NS\_FEMALE + a.NB\_FEMALE + a.QUE\_FEMALE +

a.ONT\_FEMALE + a.MAN\_FEMALE + a.SASK\_FEMALE + a.ALTA\_FEMALE + a.BC\_FEMALE +

a.YT\_FEMALE + a.NWT\_FEMALE + a.NVT\_FEMALE + a.OUT\_OF\_CANADA\_FEMALE +

a.ALL\_LOCATIONS\_FEMALE + a.NL\_BOTH + a.PEI\_BOTH + a.NS\_BOTH + a.NB\_BOTH +

a.QUE\_BOTH + a.ONT\_BOTH + a.MAN\_BOTH + a.SASK\_BOTH + a.ALTA\_BOTH + a.BC\_BOTH +

a.YT\_BOTH + a.NWT\_BOTH + a.NVT\_BOTH + a.OUT\_OF\_CANADA\_BOTH + a.ALL\_LOCATIONS\_BOTH) AS total\_payment\_amount,

SUM(n.NL\_MALE + n.PEI\_MALE + n.NS\_MALE + n.NB\_MALE + n.QUE\_MALE + n.ONT\_MALE +

n.MAN\_MALE + n.SASK\_MALE + n.ALTA\_MALE + n.BC\_MALE + n.YT\_MALE + n.NWT\_MALE +

n.NVT\_MALE + n.OUT\_OF\_CANADA\_MALE + n.NL\_FEMALE + n.PEI\_FEMALE + n.NS\_FEMALE +

n.NB\_FEMALE + n.QUE\_FEMALE + n.ONT\_FEMALE + n.MAN\_FEMALE + n.SASK\_FEMALE +

n.ALTA\_FEMALE + n.BC\_FEMALE + n.YT\_FEMALE + n.NWT\_FEMALE + n.NVT\_FEMALE +

n.OUT\_OF\_CANADA\_FEMALE) AS total\_person\_number

INTO lv\_payment\_amount, lv\_person\_number

FROM AMOUNTBYPROV a

JOIN NUMBYPROV n ON a.year = n.year

WHERE a.year = p\_year;

-- Calculate the average payment per person

IF lv\_person\_number > 0 THEN

lv\_payment\_per\_person := ROUND(lv\_payment\_amount \* 10000 / lv\_person\_number, 2); -- Round to 2 decimal places

ELSE

lv\_payment\_per\_person := 0; -- Avoid division by zero

END IF;

RETURN lv\_payment\_per\_person;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error in average\_payment\_per\_person: ' || SQLERRM);

RETURN NULL;

END;

/

CREATE OR REPLACE PROCEDURE average\_payment\_person\_yrs(

p\_start\_year IN NUMBER,

p\_end\_year IN NUMBER

) IS

lv\_average\_payment NUMBER := 0;

BEGIN

-- Loop through each year from start year to end year

FOR i IN p\_start\_year..p\_end\_year LOOP

-- Calculate the average payment per person for the current year

lv\_average\_payment := average\_payment\_per\_person(i);

-- Display the result

DBMS\_OUTPUT.PUT\_LINE('Year ' || i || ': Average Payment Per Person = ' || lv\_average\_payment);

END LOOP;

END;

/

EXEC average\_payment\_person\_yrs(2010, 2023);

A screenshot of a computer

Description automatically generated

**Question 4: Using Trigger to ensure that the data columns inserted to table AMOUNTBYAGE- ALL\_AGES\_MALE, ALL\_AGES\_FEMALE, ALL\_AGES\_BOTH can be matched with data in table AMOUNTBYPROV.**

CREATE OR REPLACE TRIGGER trg\_validate\_before\_insert

BEFORE INSERT ON AMOUNTBYAGE

FOR EACH ROW

DECLARE

v\_total\_male NUMBER;

v\_total\_female NUMBER;

v\_total\_both NUMBER;

BEGIN

-- Retrieve the totals from AMOUNTBYPROV for the corresponding year

SELECT ALL\_LOCATIONS\_MALE, ALL\_LOCATIONS\_FEMALE, ALL\_LOCATIONS\_BOTH

INTO v\_total\_male, v\_total\_female, v\_total\_both

FROM AMOUNTBYPROV

WHERE YEAR = :NEW.YEAR;

-- Check if ALL\_AGES\_MALE matches the total male amount

IF :NEW.ALL\_AGES\_MALE <> v\_total\_male THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Mismatch in ALL\_AGES\_MALE. Expected: ' || v\_total\_male || ', Found: ' || :NEW.ALL\_AGES\_MALE);

END IF;

-- Check if ALL\_AGES\_FEMALE matches the total female amount

IF :NEW.ALL\_AGES\_FEMALE <> v\_total\_female THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Mismatch in ALL\_AGES\_FEMALE. Expected: ' || v\_total\_female || ', Found: ' || :NEW.ALL\_AGES\_FEMALE);

END IF;

-- Check if ALL\_AGES\_BOTH matches the total for both genders

IF :NEW.ALL\_AGES\_BOTH <> v\_total\_both THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Mismatch in ALL\_AGES\_BOTH. Expected: ' || v\_total\_both || ', Found: ' || :NEW.ALL\_AGES\_BOTH);

END IF;

END;

/

INSERT INTO AMOUNTBYPROV (YEAR, NL\_MALE, PEI\_MALE, NS\_MALE, NB\_MALE, QUE\_MALE, ONT\_MALE, MAN\_MALE, SASK\_MALE, ALTA\_MALE, BC\_MALE, YT\_MALE, NWT\_MALE, NVT\_MALE, OUT\_OF\_CANADA\_MALE, ALL\_LOCATIONS\_MALE, NL\_FEMALE, PEI\_FEMALE, NS\_FEMALE, NB\_FEMALE, QUE\_FEMALE, ONT\_FEMALE, MAN\_FEMALE, SASK\_FEMALE, ALTA\_FEMALE, BC\_FEMALE, YT\_FEMALE, NWT\_FEMALE, NVT\_FEMALE, OUT\_OF\_CANADA\_FEMALE, ALL\_LOCATIONS\_FEMALE, NL\_BOTH, PEI\_BOTH, NS\_BOTH, NB\_BOTH, QUE\_BOTH, ONT\_BOTH, MAN\_BOTH, SASK\_BOTH, ALTA\_BOTH, BC\_BOTH, YT\_BOTH, NWT\_BOTH, NVT\_BOTH, OUT\_OF\_CANADA\_BOTH, ALL\_LOCATIONS\_BOTH)

VALUES (

2024,

1170, 1120, 1180, 1175, 1205, 1220, 1185, 1205, 1195, 1175, 1275, 1205, 1300, 900, 1200, -- ALL\_LOCATIONS\_MALE from AMOUNTBYPROV

1040, 1020, 1050, 1050, 1085, 1115, 1085, 1075, 1080, 1070, 1235, 1050, 1030, 865, 1090, -- ALL\_LOCATIONS\_FEMALE from AMOUNTBYPROV

1110, 1070, 1115, 1110, 1140, 1165, 1135, 1140, 1140, 1120, 1250, 1145, 1165, 885, 1140 -- ALL\_LOCATIONS\_BOTH from AMOUNTBYPROV

);

INSERT INTO AMOUNTBYAGE (

YEAR, UNDER\_25\_MALE, A25\_29\_MALE, A30\_34\_MALE, A35\_39\_MALE, A40\_44\_MALE, A45\_49\_MALE, A50\_54\_MALE, A55\_59\_MALE, A60\_64\_MALE, ALL\_AGES\_MALE,

UNDER\_25\_FEMALE, A25\_29\_FEMALE, A30\_34\_FEMALE, A35\_39\_FEMALE, A40\_44\_FEMALE, A45\_49\_FEMALE, A50\_54\_FEMALE, A55\_59\_FEMALE, A60\_64\_FEMALE, ALL\_AGES\_FEMALE,

UNDER\_25\_BOTH, A25\_29\_BOTH, A30\_34\_BOTH, A35\_39\_BOTH, A40\_44\_BOTH, A45\_49\_BOTH, A50\_54\_BOTH, A55\_59\_BOTH, A60\_64\_BOTH, ALL\_AGES\_BOTH

)

VALUES (

2024, 896, 882, 1022, 1115, 1143, 1173, 1200, 1230, 1242, 1200, -- ALL\_AGES\_MALE from AMOUNTBYAGE

872, 864, 965, 1033, 1087, 1086, 1106, 1107, 1109, 1090, -- ALL\_AGES\_FEMALE from AMOUNTBYAGE

883, 872, 990, 1068, 1111, 1124, 1147, 1166, 1178, 1140 -- ALL\_AGES\_BOTH from AMOUNTBYAGE

);

**Successful update:**

**A screenshot of a computer

Description automatically generated**

**Fail to update:**

* **ALL\_AGES\_MALE mismatch with ALL\_LOCATIONS\_MALE**

**A screenshot of a computer

Description automatically generated**

* **ALL\_AGES\_FEMALE mismatch with ALL\_LOCATIONS\_FEMALE**

**A screenshot of a computer

Description automatically generated**

* **ALL\_AGES\_BOTH mismatch with ALL\_LOCATIONS\_BOTH**

**A screenshot of a computer

Description automatically generated**