**Return Query Results**

--Return sales person data using a Query

CREATE OR REPLACE FUNCTION fn\_get\_sales\_people()

RETURNS SETOF sales\_person AS

$body$

BEGIN

RETURN QUERY

SELECT \*

FROM sales\_person;

END;

$body$

LANGUAGE plpgsql

SELECT (fn\_get\_sales\_people()).\*;

**Return Specific Data from Query Using Multiple Tables**

--Get top 10 most expensive products

SELECT product.name, product.supplier, item.price

FROM item

NATURAL JOIN product

ORDER BY item.price DESC

LIMIT 10;

CREATE OR REPLACE FUNCTION fn\_get\_10\_expensive\_prods()

RETURNS TABLE (

name varchar,

supplier varchar,

price numeric

) AS

$body$

BEGIN

RETURN QUERY

SELECT product.name, product.supplier, item.price

FROM item

NATURAL JOIN product

ORDER BY item.price DESC

LIMIT 10;

END;

$body$

LANGUAGE plpgsql

SELECT (fn\_get\_10\_expensive\_prods()).\*;

**IF ELSEIF and ELSE**

Check order status with IF ELSEIF and ELSE

--Check order performance with IF ELSEIF and ELSE

CREATE OR REPLACE FUNCTION fn\_check\_month\_orders(the\_month int)

RETURNS varchar AS

$body$

--Put variables here

DECLARE

total\_orders int;

BEGIN

--Check total orders

SELECT COUNT(purchase\_order\_number)

INTO total\_orders

FROM sales\_order

WHERE EXTRACT(MONTH FROM time\_order\_taken) = the\_month;

--Use conditionals to provide different output

IF total\_orders > 5 THEN

RETURN CONCAT(total\_orders, ' Orders : Doing Good');

ELSEIF total\_orders < 5 THEN

RETURN CONCAT(total\_orders, ' Orders : Doing Bad');

ELSE

RETURN CONCAT(total\_orders, ' Orders : On Target');

END IF;

END;

$body$

LANGUAGE plpgsql

SELECT fn\_check\_month\_orders(12);

**CASE Statement**

--Do the same using the case statement

--Check order performance with IF ELSEIF and ELSE

CREATE OR REPLACE FUNCTION fn\_check\_month\_orders(the\_month int)

RETURNS varchar AS

$body$

--Put variables here

DECLARE

total\_orders int;

BEGIN

--Check total orders

SELECT COUNT(purchase\_order\_number)

INTO total\_orders

FROM sales\_order

WHERE EXTRACT(MONTH FROM time\_order\_taken) = the\_month;

-- Case executes different code depending on an exact value

-- for total\_orders or a range of values

CASE

WHEN total\_orders < 1 THEN

RETURN CONCAT(total\_orders, ' Orders : Terrible');

WHEN total\_orders > 1 AND total\_orders < 5 THEN

RETURN CONCAT(total\_orders, ' Orders : Get Better');

WHEN total\_orders = 5 THEN

RETURN CONCAT(total\_orders, ' Orders : On Target');

ELSE

RETURN CONCAT(total\_orders, ' Orders : Doing Good');

END CASE;

END;

$body$

LANGUAGE plpgsql

SELECT fn\_check\_month\_orders(11);

**Loop Statement**

LOOP

Statements

EXIT WHEN condition is true;

END LOOP;

You can also exit with EXIT; with no condition

--Sum values up to a max number using

CREATE OR REPLACE FUNCTION fn\_loop\_test(max\_num int)

RETURNS int AS

$body$

--Put variables here

DECLARE

j INT DEFAULT 1;

tot\_sum INT DEFAULT 0;

BEGIN

LOOP

tot\_sum := tot\_sum + j;

j := j + 1;

EXIT WHEN j > max\_num;

END LOOP;

RETURN tot\_sum;

END;

$body$

LANGUAGE plpgsql

SELECT fn\_loop\_test(5);