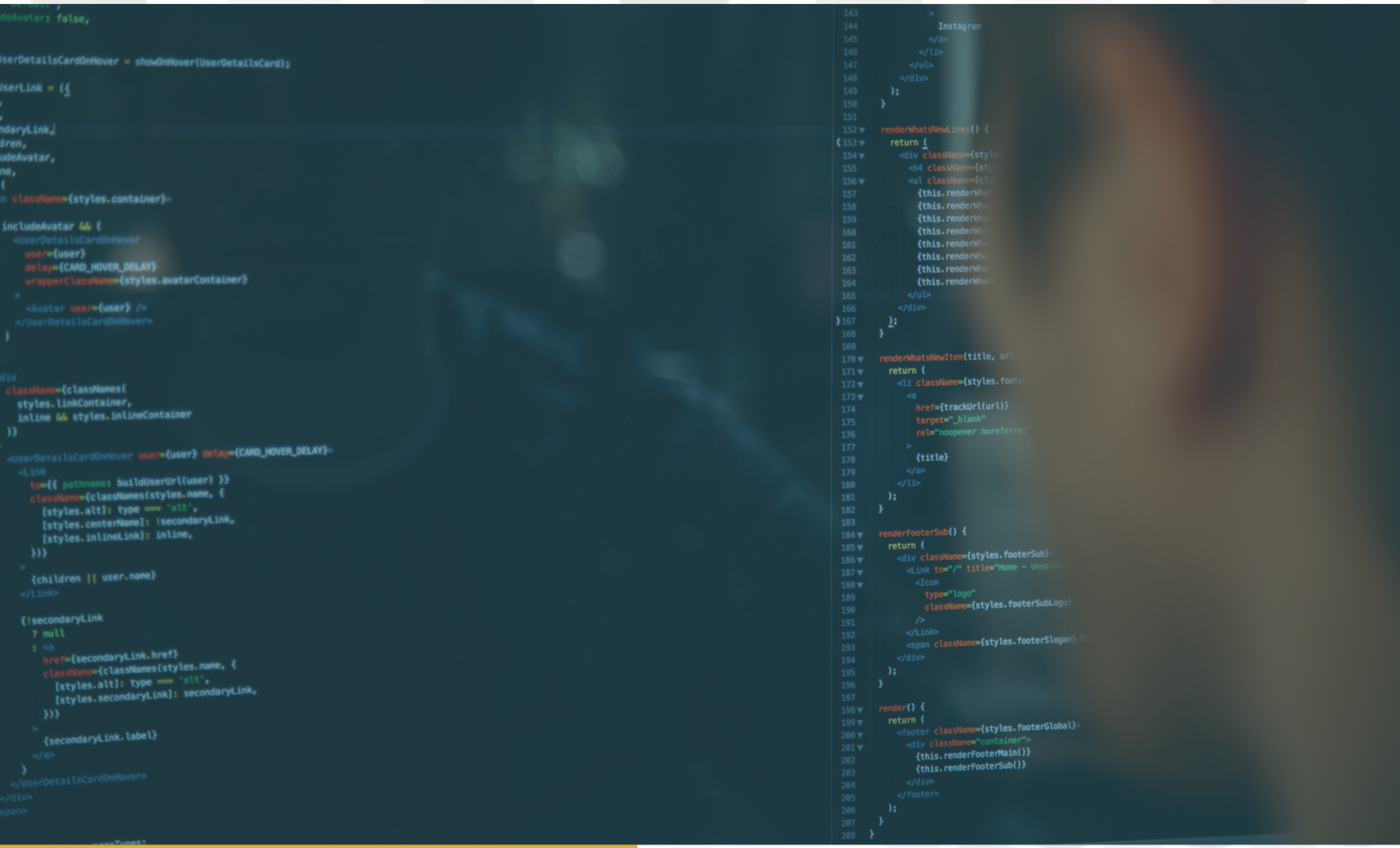


05

With, Case when, Functions & NoSQL

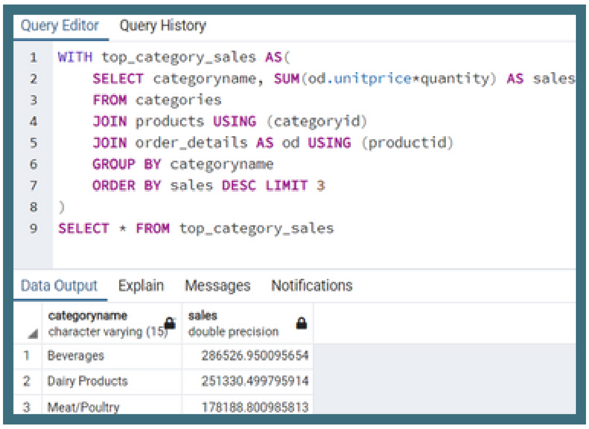
CTE-Common Table Expression

CTE defines temporary tables which you can then use in a SELECT statement. It becomes a convenient way to manage complicated queries.

WITH name AS (SELECT statement...

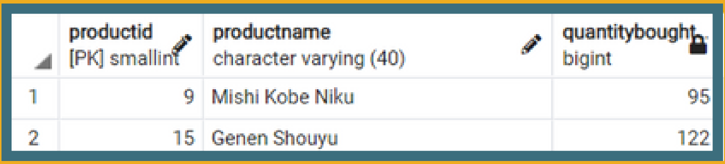
SELECT statement that includes name from WITH part

Example:

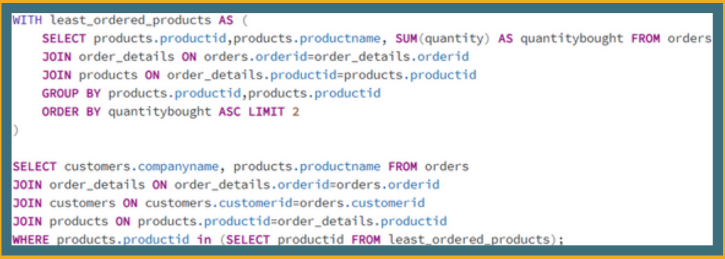


```
WITH top_category_sales AS(
    SELECT categoryname, SUM(od.unitprice*quantity) AS sales
    FROM categories
    JOIN products USING (categoryid)
    JOIN order_details AS od USING (productid)
    GROUP BY categoryname
    ORDER BY sales DESC LIMIT 3
)
SELECT * FROM top_category_sales
```

	categoryname character varying (15)	sales double precision
1	Beverages	286526.950095654
2	Dairy Products	251330.499795914
3	Meat/Poultry	178188.800985813

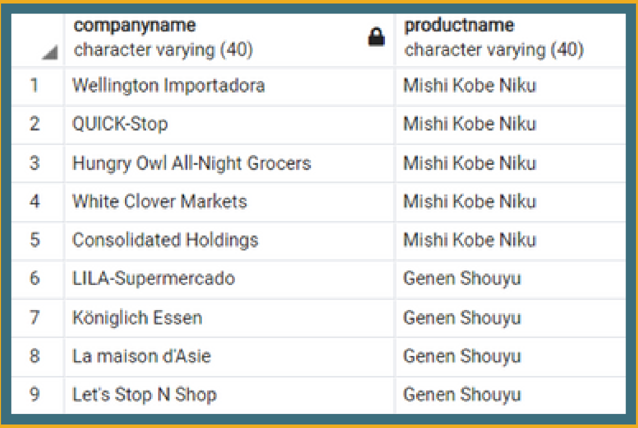


	productid [PK] smallint	productname character varying (40)	quantitybought bigint
1	9	Mishi Kobe Niku	95
2	15	Genen Shouyu	122



```
WITH least_ordered_products AS (
    SELECT products.productid,products.productname, SUM(quantity) AS quantitybought FROM orders
    JOIN order_details ON orders.orderid=order_details.orderid
    JOIN products ON order_details.productid=products.productid
    GROUP BY products.productid,products.productid
    ORDER BY quantitybought ASC LIMIT 2
)

SELECT customers.companyname, products.productname FROM orders
JOIN order_details ON order_details.orderid=orders.orderid
JOIN customers ON customers.customerid=orders.customerid
JOIN products ON products.productid=order_details.productid
WHERE products.productid in (SELECT productid FROM least_ordered_products);
```



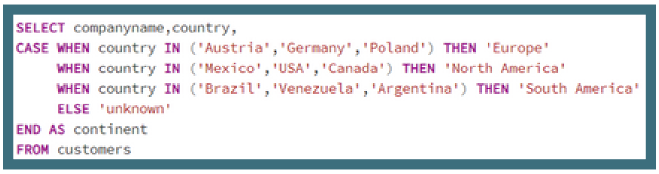
	companyname character varying (40)	productname character varying (40)
1	Wellington Importadora	Mishi Kobe Niku
2	QUICK-Stop	Mishi Kobe Niku
3	Hungry Owl All-Night Grocers	Mishi Kobe Niku
4	White Clover Markets	Mishi Kobe Niku
5	Consolidated Holdings	Mishi Kobe Niku
6	LILA-Supermercado	Genen Shouyu
7	Königlich Essen	Genen Shouyu
8	La maison d'Asie	Genen Shouyu
9	Let's Stop N Shop	Genen Shouyu

Case When

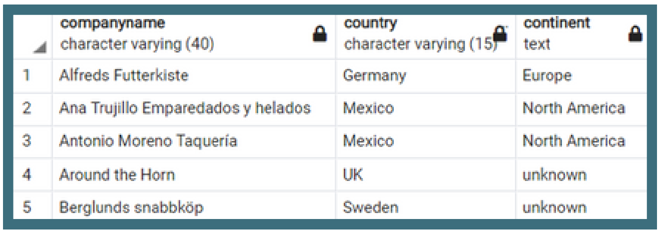
Basic syntax and use

Like IF/THEN statements in regular programming
Used in SELECT expression

```
CASE WHEN condition THEN result
     WHEN condition THEN result
     ELSE default
END
```



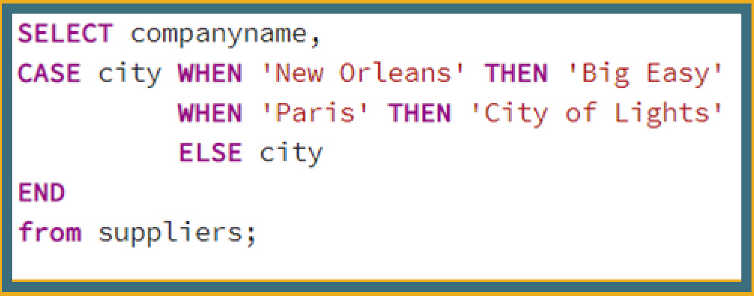
```
SELECT companyname,country,
CASE WHEN country IN ('Austria','Germany','Poland') THEN 'Europe'
     WHEN country IN ('Mexico','USA','Canada') THEN 'North America'
     WHEN country IN ('Brazil','Venezuela','Argentina') THEN 'South America'
     ELSE 'unknown'
END AS continent
FROM customers
```



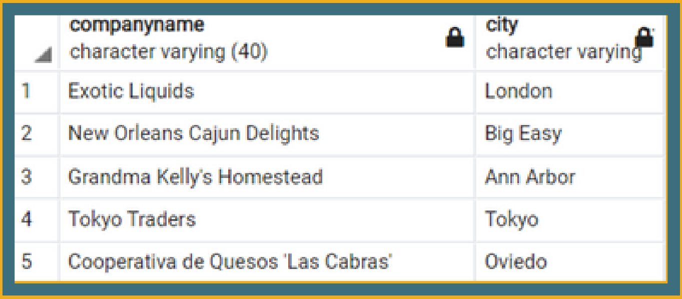
	companyname character varying (40)	country character varying (15)	continent text
1	Alfreds Futterkiste	Germany	Europe
2	Ana Trujillo Emparedados y helados	Mexico	North America
3	Antonio Moreno Taquería	Mexico	North America
4	Around the Horn	UK	unknown
5	Berglunds snabbköp	Sweden	unknown

Case When - Another Syntax

```
CASE field WHEN value THEN result
           WHEN value THEN result
           ELSE default
END
```



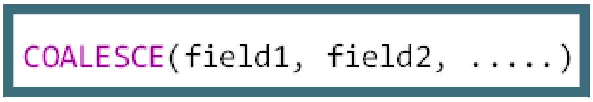
```
SELECT companyname,
CASE city WHEN 'New Orleans' THEN 'Big Easy'
          WHEN 'Paris' THEN 'City of Lights'
          ELSE city
END
from suppliers;
```



	companyname character varying (40)	city character varying
1	Exotic Liquids	London
2	New Orleans Cajun Delights	Big Easy
3	Grandma Kelly's Homestead	Ann Arbor
4	Tokyo Traders	Tokyo
5	Cooperativa de Quesos 'Las Cabras'	Oviedo

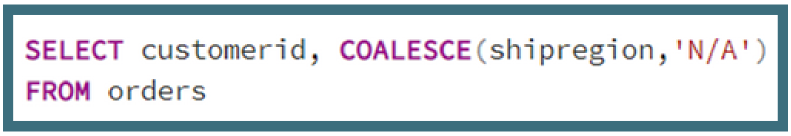
COALESCE Function

You supply a list of fields or values, it returns the first non-null value. Often used to substitute a default value for a null value.

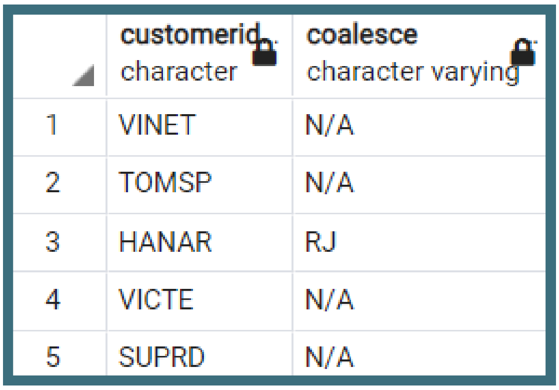


```
COALESCE(field1, field2, .....)
```

Example:



```
SELECT customerid, COALESCE(shipregion,'N/A')
FROM orders
```

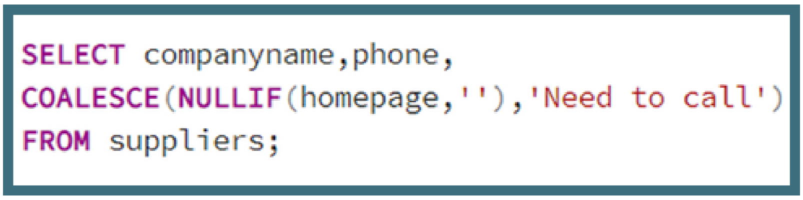


	customerid character	coalesce character varying
1	VINET	N/A
2	TOMSP	N/A
3	HANAR	RJ
4	VICTE	N/A
5	SUPRD	N/A

NULLIF Function

Used to return a null if two values are equal. Used to trigger a null in COALESCE so next value is used.

NULLIF(field1, field2)



```
SELECT companyname,phone,
COALESCE(NULLIF(homepage,''),'Need to call')
FROM suppliers;
```

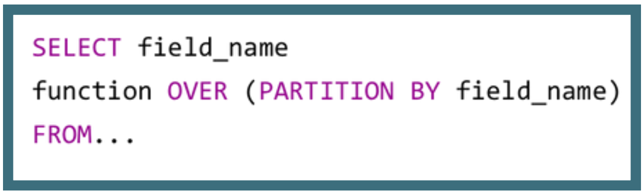
Need to Modify Some data

UPDATE suppliers
SET homepage =''
WHERE homepage IS NULL

UPDATE coustomers
SET fax =''
WHERE fax IS NULL

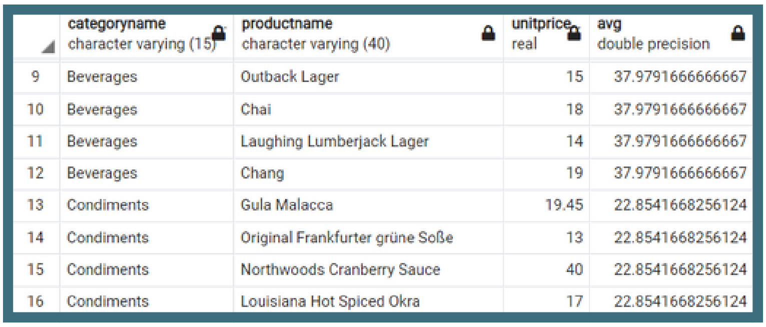
What Are Window Function?

A way to combine group by aggregation with regular select statement. The value of aggregation is calculated without combining the returned rows.

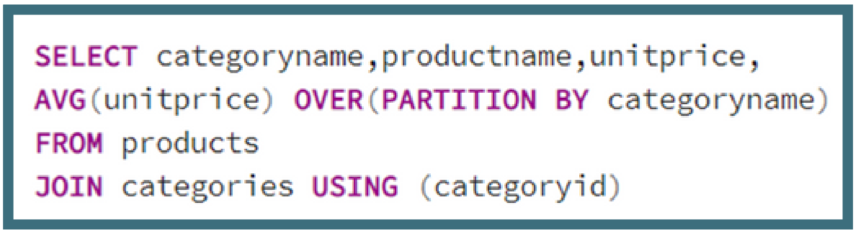


```
SELECT field_name
function OVER (PARTITION BY field_name)
FROM...
```

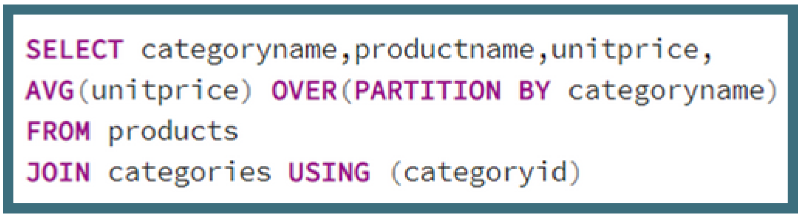
Example:



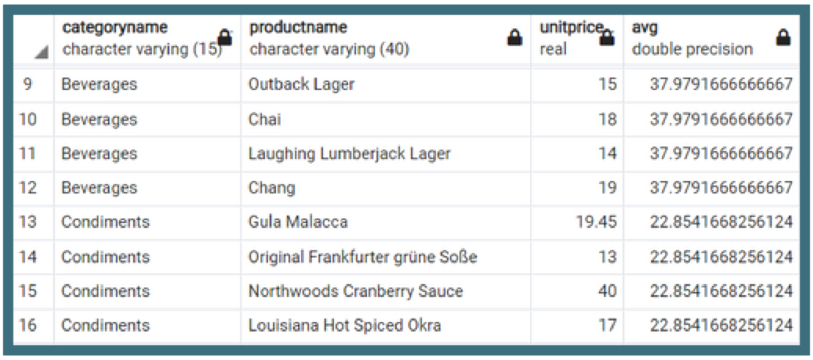
	categoryname character varying (15)	productname character varying (40)	unitprice real	avg double precision
9	Beverages	Outback Lager	15	37.9791666666667
10	Beverages	Chai	18	37.9791666666667
11	Beverages	Laughing Lumberjack Lager	14	37.9791666666667
12	Beverages	Chang	19	37.9791666666667
13	Condiments	Gula Malacca	19.45	22.8541668256124
14	Condiments	Original Frankfurter grüne Soße	13	22.8541668256124
15	Condiments	Northwoods Cranberry Sauce	40	22.8541668256124
16	Condiments	Louisiana Hot Spiced Okra	17	22.8541668256124



```
SELECT categoryname,productname,unitprice,
AVG(unitprice) OVER(PARTITION BY categoryname)
FROM products
JOIN categories USING (categoryid)
```



```
SELECT categoryname,productname,unitprice,
AVG(unitprice) OVER(PARTITION BY categoryname)
FROM products
JOIN categories USING (categoryid)
```



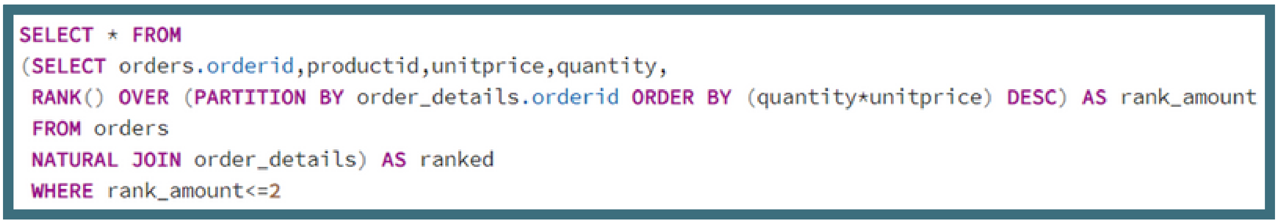
	categoryname character varying (15)	productname character varying (40)	unitprice real	avg double precision
9	Beverages	Outback Lager	15	37.9791666666667
10	Beverages	Chai	18	37.9791666666667
11	Beverages	Laughing Lumberjack Lager	14	37.9791666666667
12	Beverages	Chang	19	37.9791666666667
13	Condiments	Gula Malacca	19.45	22.8541668256124
14	Condiments	Original Frankfurter grüne Soße	13	22.8541668256124
15	Condiments	Northwoods Cranberry Sauce	40	22.8541668256124
16	Condiments	Louisiana Hot Spiced Okra	17	22.8541668256124

Nesting Queries Gives You Great Power

Someone asks about fraud detection. We want to know when an order comes in that is 5 times greater than the customer's average order.

RANK Function

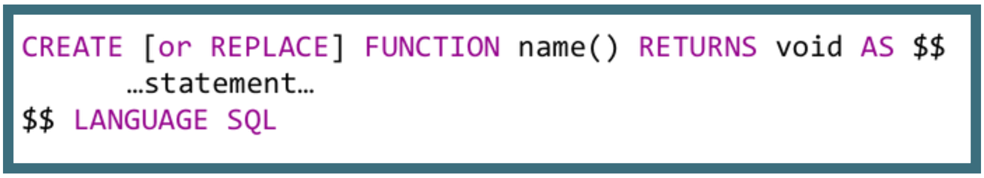
How do I join two tables and return the top 2 results from the 2nd table for each row in first table?
LIMIT won't work because it limits total rows returned.
Window function RANK() will do this top two most valuable items ordered for each orders records.



```
SELECT * FROM
(SELECT orders.orderid,productid,unitprice,quantity,
 RANK() OVER (PARTITION BY order_details.orderid ORDER BY (quantity*unitprice) DESC) AS rank_amount
 FROM orders
 NATURAL JOIN order_details) AS ranked
 WHERE rank_amount<=2
```

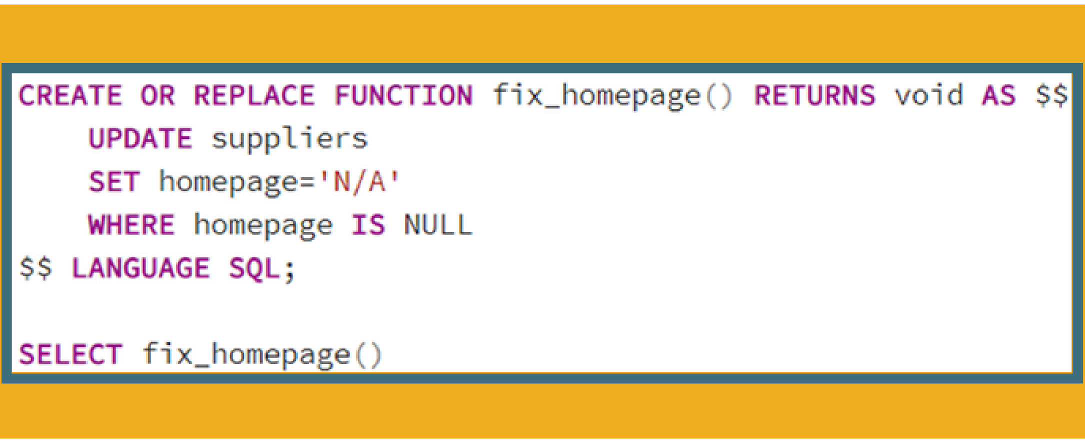
CREATE or REPLACE FUNCTION

Syntax For Simplest Function



```
CREATE [or REPLACE] FUNCTION name() RETURNS void AS $$
       …statement…
$$ LANGUAGE SQL
```

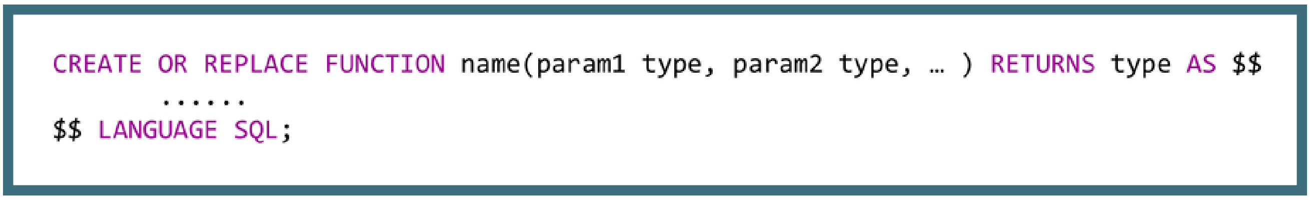
Example:
Write a function called fix...homepage() that updates all suppliers with null in homepage field to 'N/A'



```
CREATE OR REPLACE FUNCTION fix_homepage() RETURNS void AS $$
    UPDATE suppliers
    SET homepage='N/A'
    WHERE homepage IS NULL
$$ LANGUAGE SQL;

SELECT fix_homepage()
```

Function Parameters



```
CREATE OR REPLACE FUNCTION name(param1 type, param2 type, … ) RETURNS type AS $$
       ......
$$ LANGUAGE SQL;
```

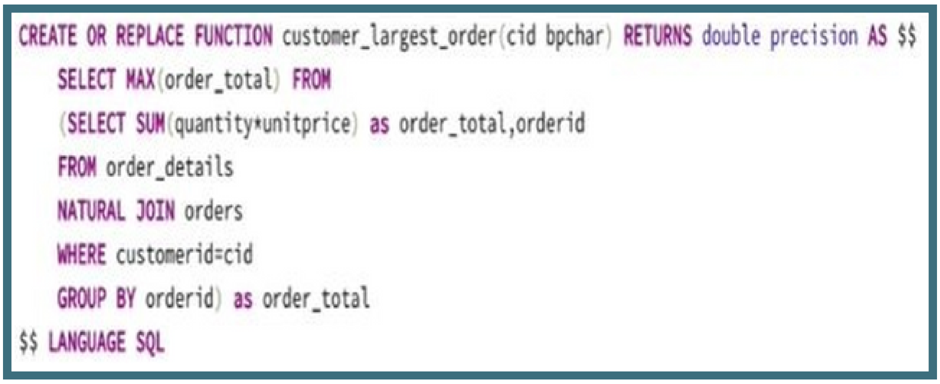
Danger

if you name the parameter the same as column names the function might get confused Ex : parameter named customerid and query uses customerid=customerid. You need a different name or use positional notation $1

2 Ways To Reference Parameters

- By name : param1, param2
- By position: $1, $2
- In older postgresql version, by position is the only way to do this

Example:
Find the largest order amount given a specific customer



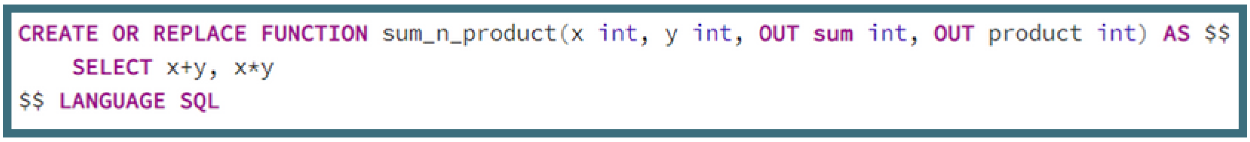
```
CREATE OR REPLACE FUNCTION customer_largest_order(cid bpchar) RETURNS double precision AS $$
    SELECT MAX(order_total) FROM
    (SELECT SUM(quantity*unitprice) as order_total,orderid
    FROM order_details
    NATURAL JOIN orders
    WHERE customerid=cid
    GROUP BY orderid) as order_total
$$ LANGUAGE SQL
```

IN and OUT

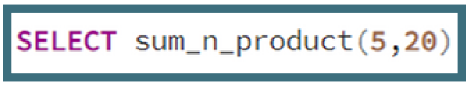
Using IN, OUT, INOUT (both input and output), and VARIADIC (covered with arrays) CREATE FUNCTION name (IN x int, IN y int, OUT sum int) Example: create a function to both add and multiple two numbers

IN and OUT

Example:
create a function to both add and multiple
two numbers



```
CREATE OR REPLACE FUNCTION sum_n_product(x int, y int, OUT sum int, OUT product int) AS $$
    SELECT x+y, x*y
$$ LANGUAGE SQL
```

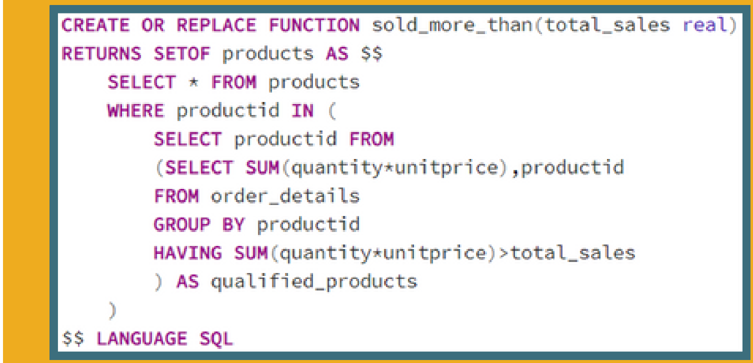


```
SELECT sum_n_product(5,20)
```

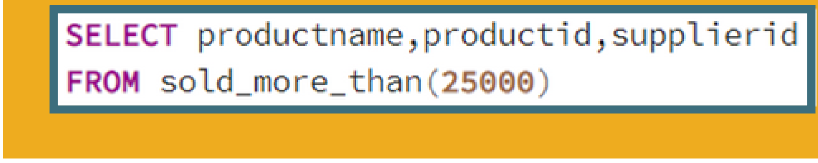
Way to Return Multiple Columns

CREATE FUNCTION name (x int, OUT sum int, OUT product int) RETURN SETOF record AS

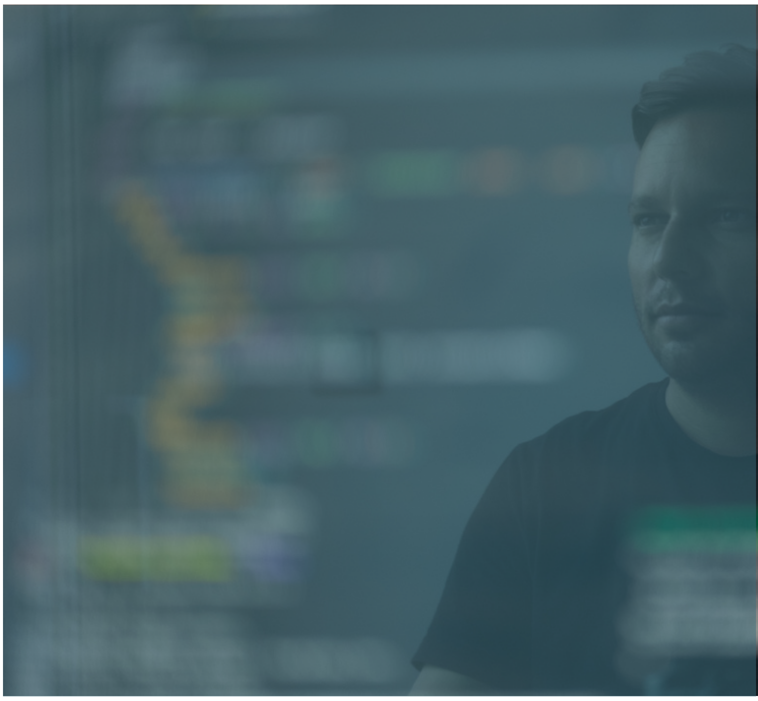
Example:
Let's return all products that have total sales greater than some input value



```
CREATE OR REPLACE FUNCTION sold_more_than(total_sales real)
RETURNS SETOF products AS $$
    SELECT * FROM products
    WHERE productid IN (
        SELECT productid FROM
        (SELECT SUM(quantity*unitprice),productid
        FROM order_details
        GROUP BY productid
        HAVING SUM(quantity*unitprice)>total_sales
        ) AS qualified_products
    )
$$ LANGUAGE SQL
```



```
SELECT productname,productid,supplierid
FROM sold_more_than(25000)
```

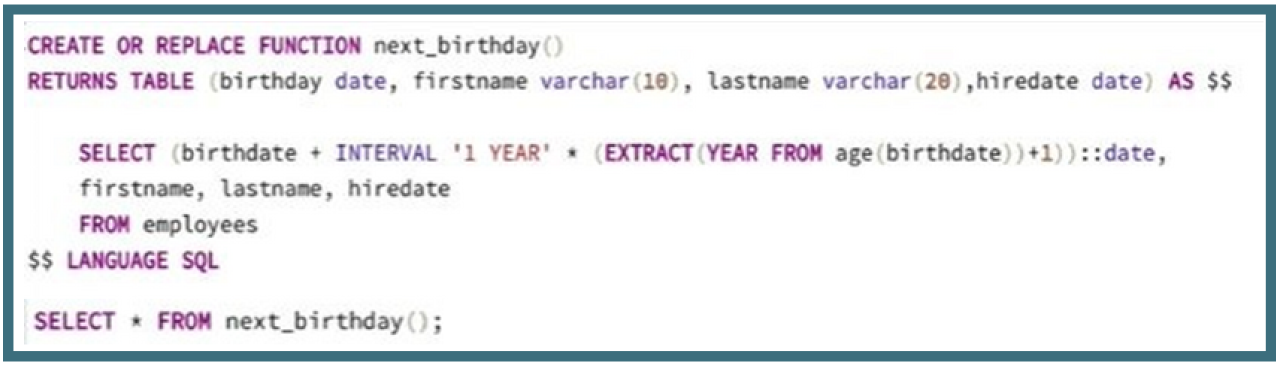


Another Way To Return A Set

RETURN TABLE (columns)
CREATE FUNCTION name (params) RETURN
TABLE (params)

Must list out all the return parameters.

Example:
Create a function called next_birthday that return all employees next birthday, first and last name, and hiredate



```
CREATE OR REPLACE FUNCTION next_birthday()
RETURNS TABLE (birthday date, firstname varchar(10), lastname varchar(20),hiredate date) AS $$

    SELECT (birthdate + INTERVAL '1 YEAR' * (EXTRACT(YEAR FROM age(birthdate))+1))::date,
    firstname, lastname, hiredate
    FROM employees
$$ LANGUAGE SQL

SELECT * FROM next_birthday();
```

SQL Database (relational)
VS
NoSQL (non-relational)

	Relation Database	NoSQL Database
Optimal Workload	Relational databases are designed for transactional applications and online transaction processing (OLTP) applications that are very consistent and suitable for online analytical processing (OLAP).	The key-value database, documents, graphics, and in NoSQL memory are designed for OLTP for a number of data access patterns that include low latency applications. The NoSOL search database is designed for semi-structured data analysis.
Data Model	The relational model normalizes data into tables consisting of rows and columns. Schemas strictly define tables, rows, columns, indexes, relationships between tables, and other database elements. The database enforces referential integrity in the relationships between tables.	NoSQL databases provides various data models, includingdocuments, graphics, key values, in memory, and searching.
ACID Property	Relational databases provide atomicity, consistency, isolation, and durability (ACID) properties	NoSQL databases often exchange by reducing some of the ACID properties of relational databases to more flexible data models that can be developed horizontally.
Performance	Performance generally depends on the disk subsystem. Optimizing queries, indexes, and table structures are often needed to achieve peak performance.	Performance is generally a function of hardware cluster size, network latency, and application calls.

Scale	Relational databases can generally be scaled by increasing hardware computing capabilities or developing scales by adding replicas to read-only workloads	NoSQL databases can generally be partitioned because key-value access patterns can be scaled by using a distributed architecture to increase throughput that provides consistent performance on an unlimited scale.
API	Requests to store and retrieve data are communicated using queries that correspond to structured query (SQL) language. This query is parsed and executed by a relational database.	The object-based API allows application developers to easily store and retrieve data structures in memory. Partition keys allow applications to search for key-value pairs, column sets, or semi-structured documents that contain serial application objects and attributes.

7 Popular NoSQL Databases

- MongoDB
- CouchDB
- Cassandra
- Redis
- Riak
- Neo4j
- OrientDB

Picture Source

- unsplash.com
- pexels.com
- pixabay.com