Conditional Expressions and Procedures Introduction

Section Overview

- CASE
- COALESCE
- NULLIF
- CAST
- Views
- Import and Export Functionality

CASE

- Use the CASE statement to only execute SQL code when certain conditions are met.

- This is very similar to IF/ELSE statement in other programming languages.

- There are two main ways to use a case STATEMENT, either CASE or a CASE expression.

  - General Syntax:

```
CASE
        WHEN condition1 THEN result1
        WHEN condition2 THEN result2
        ELSE some_other_result
END
```

Ex:

a
1
2

```
SELECT a,
CASE
        WHEN a = 1 THEN 'one'
        WHEN a = 2 THEN 'two'
        ELSE 'other'
END AS label
FROM test
```

a	label
1	one
2	two

- The CASE expression syntax first evaluates an expression then compares the result with each value in the WHEN clauses sequentially.

  CASE Expression Syntax:
  CASE expression
  WHEN value1 **THEN** result1
  WHEN value2 **THEN** result2
  ELSE some_other_result
  END

  Ex:
  SELECT a,
  CASE a
  WHEN 1 **THEN** ‘one’
  WHEN 2 **THEN** ‘two’
  ELSE ‘other’
  END AS label
  FROM test

```sql
SELECT customer_id,
    CASE
        WHEN (customer_id <= 100) THEN 'Premium'
        WHEN (CUSTOMER_ID  BETWEEN 100 AND 200) THEN 'Plus'
        ELSE 'Normal'
END as customer_class
FROM customer
```

	customer_id [PK] integer	customer_class text
1	524	Normal
2	1	Premium
3	2	Premium
4	3	Premium
5	4	Premium

```sql
SELECT customer_id,
    CASE customer_id
        WHEN 2 THEN 'Winner'
        WHEN 5 THEN 'Second Place'
        ELSE 'Normal'
END AS raffle_result
FROM customer
```

	customer_id [PK] integer	raffle_result text
1	524	Normal
2	1	Normal
3	2	Winner
4	3	Normal
5	4	Normal
6	5	Second Place

```sql
SELECT
SUM(CASE rental_rate
    WHEN 0.99 THEN 1
    ELSE 0
END) AS number_of_bargains
FROM film
```

	number_of_bargains bigint
1	341

```sql
SELECT
SUM(CASE rental_rate
    WHEN 0.99 THEN 1
    ELSE 0
END) AS bargins,
SUM(CASE rental_rate
   WHEN 2.99 THEN 1
   ELSE 0
END) AS regular,
SUM(CASE rental_rate
   WHEN 4.99 THEN 1
   ELSE 0
END) AS premium
FROM film
```

	bargins bigint	regular bigint	premium bigint
1	341	323	336

CASE – Challenge Task

1. Use CASE and the dvdrental database to re-create this table:

r bigint	pg bigint	pg13 bigint
195	194	223

```sql
SELECT
SUM (CASE rating
            WHEN 'R' THEN 1
            ELSE 0
    END) AS r,
SUM (CASE rating
            WHEN 'PG' THEN 1
            ELSE 0
    END) AS pg,
SUM (CASE rating
            WHEN 'PG-13' THEN 1
            ELSE 0
    END) AS pg13
FROM film
```

COALESCE function

- Accepts an unlimited number of arguments. It returns the first argument that is not null. If all arguments are null, the COALESCE function will return null.
  - COALESCE (arg_1, arg_2, …, arg_n)
    - SELECT COALESCE (1, 2)
      Return -> 1
    - SELECT COALESCE (NULL, 2, 3)
      Return -> 2

- The COALESCE function becomes useful when querying a table that contains null values and substituting it with another value.
  - Table of product. What's the final price?

Item	Price	Discount
A	100	20
B	300	null
C	200	10

❌:
SQL cannot subtract null
SELECT item, (price – discount) AS final
FROM table

Item	final
A	80
B	null
C	190

✔:
If the value is NULL, return 0
SELECT item, (price – COALESCE (discount, 0)) AS final
FROM table

Item	final
A	80
B	300
C	190

- Keep the COALESCE function in mind in case you encounter a table with null values that you want to perform operations on!

CAST operator

- Let's you convert from one data type into another.

- Keep in mind not every instance of a data type can be CAST to another data type, it must be reasonable to convert the data, for example '5' to an integer will work, 'five' to an integer will not.

- Two main way:
  - SELECT CAST ('5' AS INTEGER)
  - SELECT '5'::INTEGER

- Keep in mind you can then use this in a SELECT query with a column name instead of a single instance
  - SELECT CAST (date AS TIMESTAMP)
    FROM table

```
1  SELECT CAST('5' AS INTEGER) AS new_result
```

	new_result integer
1	5

```
1  SELECT '5'::INTEGER AS new_result
```

	new_result integer
1	5

```
1  SELECT CAST(inventory_id AS VARCHAR)
2  FROM rental
```

	inventory_id character varying
1	1525
2	1711
3	2452
4	2079

NULLIF function

- NULLIF function takes in 2 inputs and returns NULL if both are equal, otherwise it returns the first argument passed.
  - NULLIF (arg1, arg2)
    - NULLIF (10, 10)
      Return -> NULL
    - NULLIF (10, 12)
      Return -> 10

- This becomes very useful in cases where a NULL value would cause an error or unwanted result.

- Given this table calculate the ratio of Department A to Department B

Name	Department
Lauren	A
Vinton	A
Claire	B

```sql
SELECT (
SUM (CASE WHEN department = 'A' THEN 1 ELSE 0 END)/
SUM (CASE WHEN department = 'B' THEN 1 ELSE 0 END)
) AS department_ratio
FROM depts
```

	department_ratio bigint
1	2

```sql
DELETE FROM depts
WHERE department = 'B'
```

```sql
SELECT * FROM depts
```

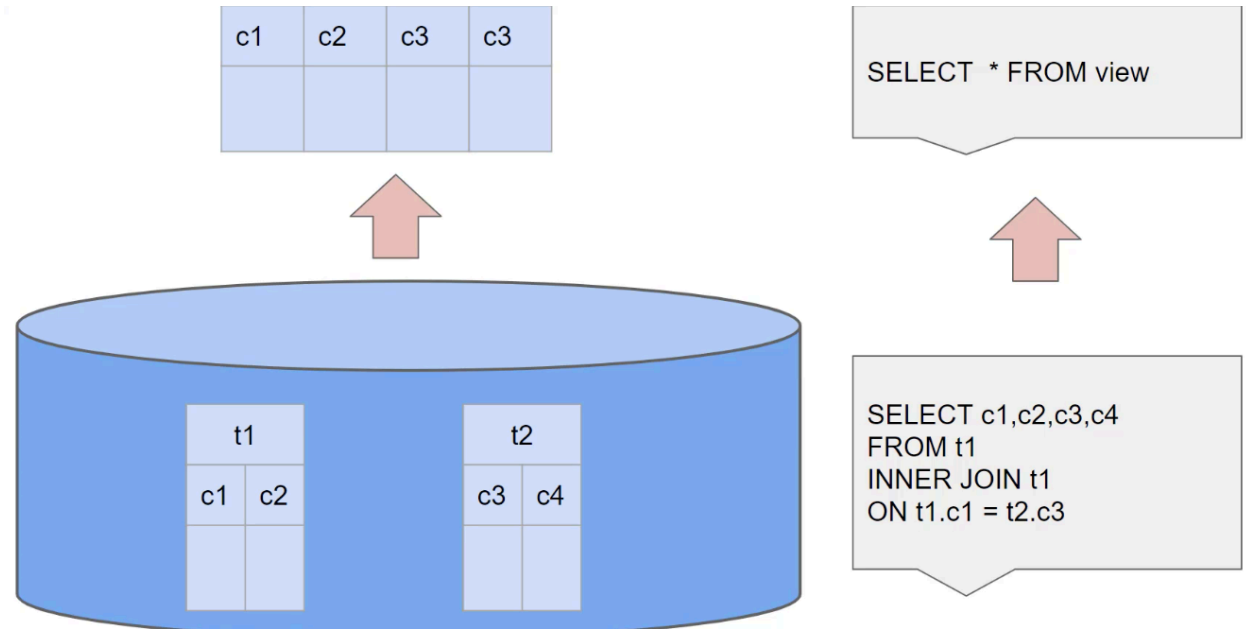
	first_name character varying (50)	department character varying (50)
1	Lauren	A
2	Vinton	A

```sql
SELECT (
SUM (CASE WHEN department = 'A' THEN 1 ELSE 0 END)/
NULLIF(SUM (CASE WHEN department = 'B' THEN 1 ELSE 0 END), 0)
) AS department_ratio
FROM depts
```

	department_ratio bigint
1	[null]

Views

- Often there are specific combinations of tables and conditions that you find yourself using quite often for a project.
- Instead of having to perform the same query over and over again as a starting point, you can create a VIEW to quickly see this query with a simple call.



- A view is a database object that is of a stored query.
- A view can be accessed as a virtual table in PostgreSQL.
- Notice that a view does not store data physically, it simply stores the query.
- You can also update and alter the view.

```
CREATE VIEW customer_info AS
SELECT first_name, last_name, address
FROM customer
INNER JOIN address
ON customer.address_id = address.address_id
```

```
SELECT * FROM customer_info
```

	first_name character varying (45)	last_name character varying (45)	address character varying (50)
1	Jared	Ely	1003 Qinhuangdao Street
2	Mary	Smith	1913 Hanoi Way
3	Patricia	Johnson	1121 Loja Avenue
4	Linda	Williams	692 Joliet Street

```sql
CREATE OR REPLACE VIEW customer_info AS
SELECT first_name, last_name, address, district
FROM customer
INNER JOIN address
ON customer.address_id = address.address_id
```

```sql
SELECT * FROM customer_info
```

	first_name character varying (45)	last_name character varying (45)	address character varying (50)	district character varying (20)
1	Jared	Ely	1003 Qinhuangdao Street	West Java
2	Mary	Smith	1913 Hanoi Way	Nagasaki
3	Patricia	Johnson	1121 Loja Avenue	California
4	Linda	Williams	692 Joliet Street	Attika

Drop view:

```sql
DROP VIEW IF EXISTS customer_info
```

Change name of the view:

```sql
ALTER VIEW customer_info RENAME TO c_info
```