Data Transformation

LIGHTHOUSE LABS

Steve Mitchell

Agenda

- What is Data Transformation?
- SQL Case Statement
- Casting New Data Types
- Window Functions
- User Defined Functions
- Exercises

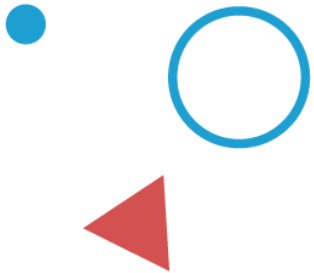
LIGHTHOUSE LABS

What is Data Transformation?

LIGHTHOUSE LABS

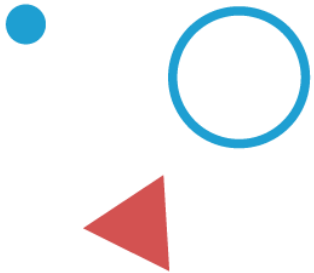
Definition

- **Process of converting data from one format or structure to another.**
- **Makes it more suitable for analysis, reporting, or further processing.**



Importance of Data Transformation

- **Clean and organized data:** Ensures data quality and consistency, making it easier to work with.
- **Enhanced decision-making:** Accurate and well-structured data helps derive meaningful insights and make informed decisions.
- **Compatibility:** Data transformation enables compatibility across different systems and databases.
- **Data normalization:** Helps in reducing redundancy and achieving data integrity.

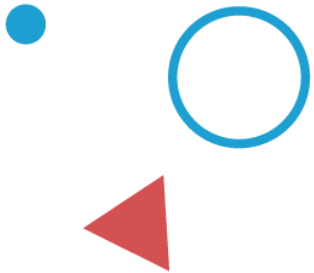


Real World Examples

1. Converting Time Zones or Timestamps / Timedeltas
2. Address Standardization
3. Combining Survey Data with different formats.
4. Hashing/Encrypting Sensitive Data

Datetime documentation:

https://www.postgresql.org/docs/current/functions-datetime.html

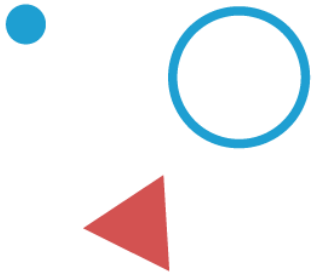


SQL CASE Statement

LIGHTHOUSE LABS

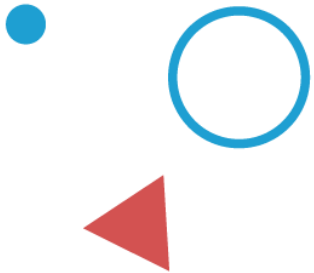
What Is a CASE Statement?

- Opening arguments in court?
- Complimenting someone’s briefcase?
- Declaring the letter capitalization for a project?
- A conditional expression in SQL (usually in SELECT).
- Used to create a new column in the result table (mostly).
- Used to filter existing columns as well.
- Result based on specific conditions.



CASE Statement Syntax

```
CASE
     WHEN condition THEN 'desired output'
     WHEN condition THEN 'desired output_2'
     WHEN condition THEN 'desired output_3'
     ..
     ..
     ELSE 'another output'
     END as case_new_column_name
```



Examples

1. **Age Group Classification**
    - **Classify people into age groups (e.g., Child, Teen, Adult, and Senior) based on their age.**

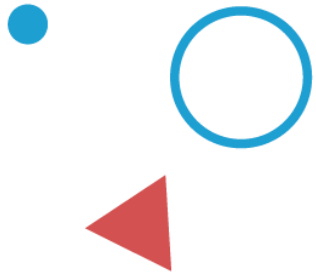
2. **Discount Calculation**
    - **Calculate a discount for customers based on their total purchase amount (e.g., 5% discount for purchases over $500, 10% discount for purchases over $1,000).**

3. **Academic Grade Evaluation**
    - **Convert numeric exam scores into letter grades (e.g., A, B, C, D, F) based on a predefined grading scale.**

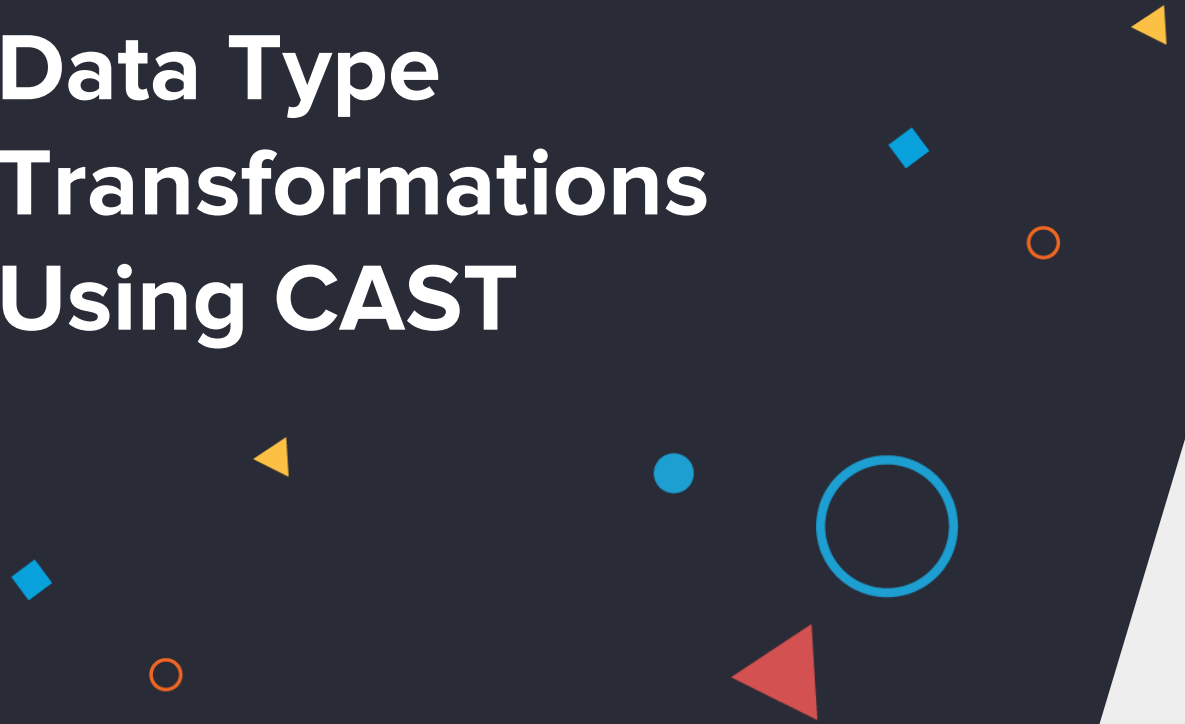
Northwind Examples

1. In the order_details table, classify each row based on its quantity: “large” (>40), “medium” (<=40 and >20), or “small” (<=20).

2. Show every row from order_details that has no discount and quantity <10 or if there is a discount, the quantity must be between 10 and 20.



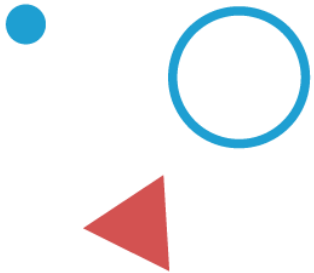
Data Type Transformations Using CAST



LIGHTHOUSE LABS

Definition of CAST

- Something on a broken leg or arm?
- A spell from Harry Potter?
- SQL function that explicitly converts an expression of one data type to another?



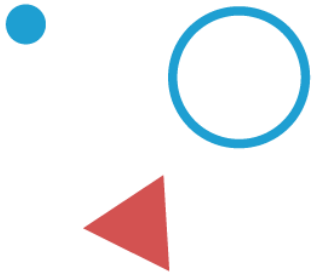
CAST Syntax

CAST Function:
CAST(column_example AS data_type_to_transform_to)

or

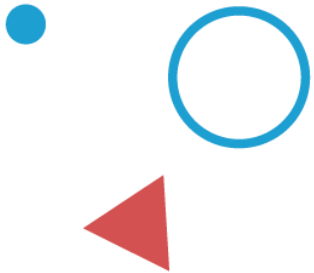
CAST Operator:
column_example::data_type_to_transform_to

https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-cast/



Common Data Types

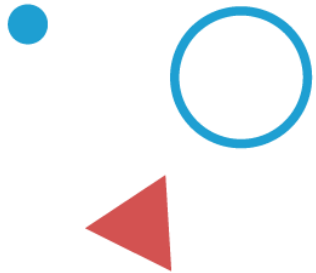
1. INTEGER
2. FLOAT
3. DATE
4. TIMESTAMP
5. VARCHAR/TEXT
6. BOOLEAN



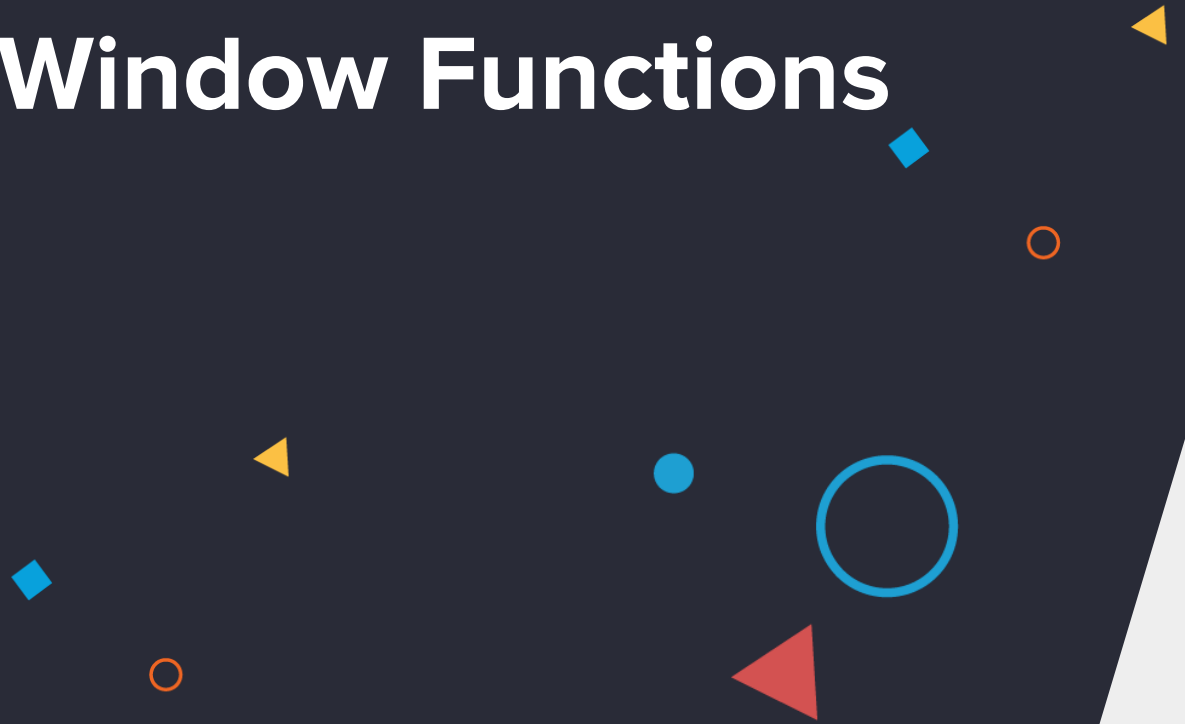
Northwind Examples

1. In the orders table, convert the orderdate to a string.

2. In the products table, show unitsinstock as a percentage of the grand total unitsinstock.



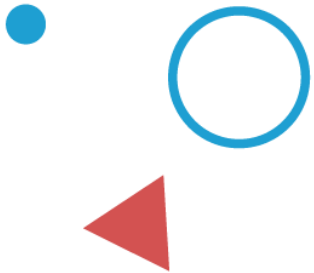
Window Functions



LIGHTHOUSE LABS

Definition of a Window Function

- **Perform calculations across a set of table rows related to the current row.**
- **Returns a single value for each row in the result set.**

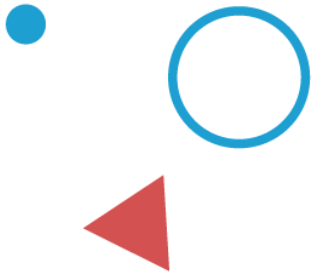


Definition of a Window Function

- **Similar to how aggregation functions work with GROUP BY, but each row gets a result.**

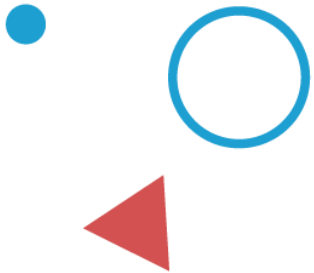
Purpose of Window Functions

- Allow for air to freely flow from inside/out or outside/in?
- Allow for user to look outside?
- Allow for advanced calculations and data manipulation.
- Enable insights that would be difficult or inefficient to obtain using standard aggregate functions.



Window Function Syntax

```
WINDOW_FUNCTION(arg1, arg2, …) OVER (
      PARTITION BY column1
      ORDER BY column 2
)
```

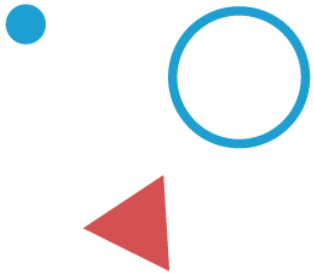


Window Functions - Components

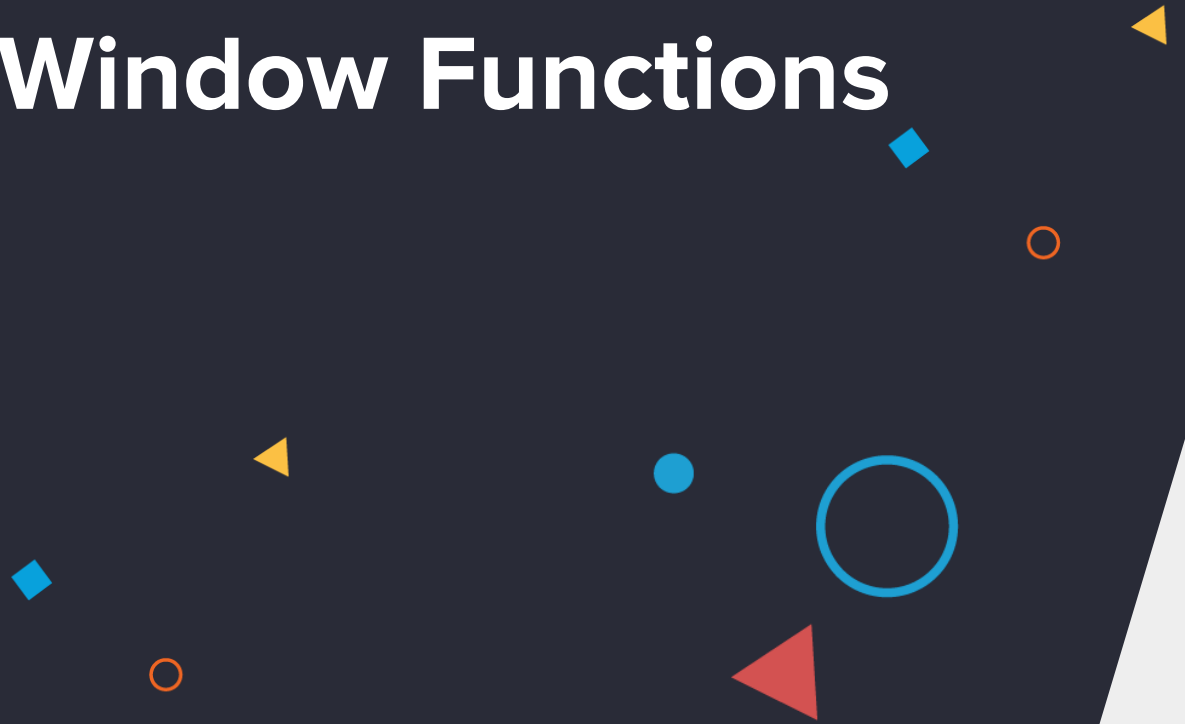
- OVER()
  - The OVER() clause defines the window or range of rows to be considered for the calculation.

- PARTITION BY
  - The PARTITION BY clause divides the result set into partitions to which the window function is applied.

- ORDER BY
  - The ORDER BY clause within the OVER() clause specifies the order in which the rows in the partition are processed by the window function.



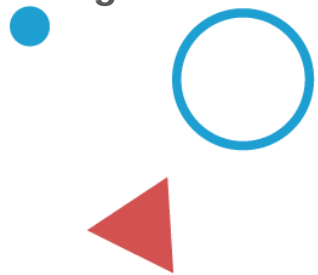
Window Functions



LIGHTHOUSE LABS

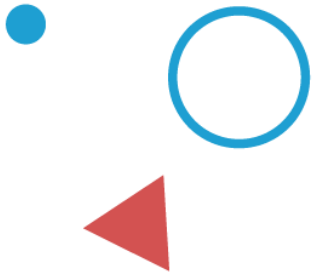
Window Functions - Ranking Functions

- ROW_NUMBER()
  - Assigns a unique, sequential integer to each row within the result set.
- RANK()
  - Assigns a unique rank to each row within the result set, leaving gaps in rank values when there are equal rank values.
- DENSE_RANK()
  - Assigns a unique rank to each row within the result set, without leaving gaps in rank values when there are equal rank values.
- PERCENT_RANK()
  - Calculates the relative rank of each row within the result set as a percentage, ranging from 0 to 1.



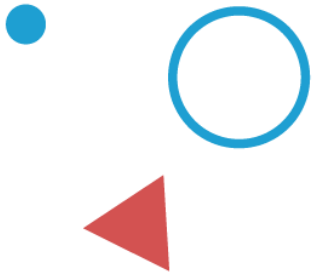
Window Functions - Aggregates

- **SUM(): Calculates the sum of a column's values over the specified window.**
- **AVG(): Calculates the average of a column's values over the specified window.**
- **MIN(): Returns the minimum value in the specified window.**
- **MAX(): Returns the maximum value in the specified window.**
- **COUNT(): Counts the number of rows in the specified window.**



Window Functions - Navigation

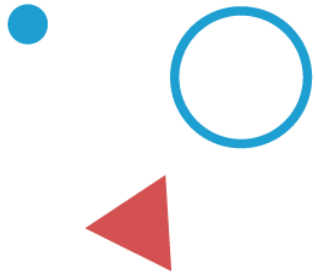
- **FIRST_VALUE(): Returns the first value in the specified window.**
- **LAST_VALUE(): Returns the last value in the specified window.**
- **NTH_VALUE(): Return the n-th value in the specified window.**
- **LEAD(): Returns the value in the specified column for the row that is a certain number of rows ahead of the current row within the partition.**
- **LAG(): Returns the value in the specified column for the row that is a certain number of rows behind the current row within the partition.**



Window Functions - Distribution

- **CUME_DIST(): Calculates the cumulative distribution of a value in the specified window, returning a relative value between 0 and 1.**

- **NTILE(n): Divides the result set into a specified number (n) of approximately equal buckets and assigns a bucket number to each row.**



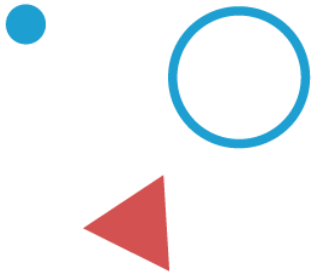
User Defined Functions

LIGHTHOUSE LABS

Definition

- **User-Defined Functions (UDFs) are custom functions created by the user**

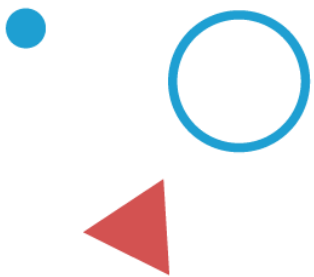
- **Perform specific tasks that may not be readily available or easily achievable using built-in functions.**



Let’s build one together!

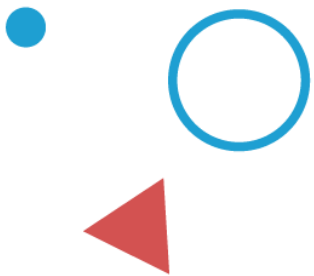
```
CREATE OR REPLACE FUNCTION calculate_total_price(input_order_id INTEGER)
RETURNS NUMERIC AS $$
DECLARE
  total_price NUMERIC := 0;
BEGIN
  SELECT SUM(unitprice * quantity * (1 - Discount))
  INTO total_price
  FROM order_details
  WHERE orderid = input_order_id;

  RETURN total_price;
END;
$$ LANGUAGE plpgsql;
```



Now Try This!

```
SELECT orderid, calculate_total_price(orderid) AS TotalPrice
FROM orders;
```



SQL Exercises

LIGHTHOUSE LABS

EXERCISES

1. Assign a “Low”, “Medium”, or “High” price label to each product based on its unit price.
2. Classify customers as "Local" or "International" based on their country.
3. Assign a discount category to each order based on the average discount applied to its order details.
4. Convert the order date to a string.
5. Calculate the VAT (Value Added Tax) for each product's unit price assuming a VAT rate of 20%. (HARD)
    - BONUS: Round the result to 2 decimal places.
6. Calculate the total price of each order, including VAT, and format the result as a string with a currency symbol. (HARD)
7. Assign a row number to each product based on its unit price in ascending order.
8. Rank employees by the total number of orders they have managed. (HARD)
9. Assign a dense rank to customers based on the total revenue they have generated. (HARD)
10. Assign a row number to each product within its category based on its unit price in ascending order.
11. Take a look at the Other Window Functions, experiment on some queries, share your results with your peers in the discord. Explain the insight of that query or how it can be useful. (Build 2-3 unique queries) (HARD + CREATIVE)