SQL DATA STANDARDIZATION DOCUMENTATION

DNA LEARNING (WEEKEND - 1)

STRING FUNCTIONS

1. TRIM() - removes trailing and leading spaces from the string
2. LTRIM() - removes trailing or leading space from the beginning of the string
3. RTRIM() - removes trailing or leading spaces from the end of the string
4. UPPER() - Capitalizes the string
5. LOWER() - Makes the string characters into small letters
6. LENGTH() - finds length of string
7. SUBSTRING() - Extracts the part of the string

   SUBSTRING('john', 1, 3) - (string, start position, no of characters to be extracted)

   Ans: joh

8. INSTR() - Gives the position of character in the string.

   INSTR(manhatoo.mackry@gmail.com, '@')

   Ans : 16

9. CHAR_LENGTH() - Finds no of characters in the string
10. CONCAT() - Adds two strings

11. REGEXP_REPLACE() - Removes unwanted characters

    REGEXP_REPLACE(728973jh89u23x534, '^[0-9]', '')

    Ans: 7289738923534

12. LPAD - Adds any character from beginning of string to make it to a specific length

    LPAD(555, 6, 0) - Given number is 555, and result length should be 6 padded up With 0

    Ans : 000555

13. RPAD - Adds any character from the end of string

    RPAD(555, 6, 9) - Given number is 555, and result length should be 6 padded up

S DHAANESH

With 9

Ans : 555999

<u>SQL DATA ANALYSIS (CUSTOMER DATASET)</u>

Dataset - https://github.com/Dhaanesh26/data_standardisation/blob/main/datasets/customer.csv

Q1. Remove extra spaces and fix casing in names

```
SELECT TRIM(UPPER(CustomerName))
```

Q2. Standardize phone number formats

```
SELECT PhoneNumber,
CONCAT('+1',
RIGHT(REGEXP_REPLACE(SUBSTRING_INDEX(PhoneNumber,'x',1),'[^0-9]', '')), 10)
```

Q3. Check email if @ is present

```
SELECT Email,
        CASE
                WHEN  INSTR(Email, '@') > 0 THEN 'Valid'
                ELSE 'Inavalid'
        END AS email_status
FROM customer
```

Q4. Query the domain name from email

```
SELECT Email, SUBSTRING(Email, INSTR(Email, '@') + 1) AS domain_name
FROM customer
```

Q5. Convert string to proper date format

```
SELECT STR_TO_DATE(CreatedDate, '%Y-%m-%d') AS standardised_signup_date
FROM customer
```

Q6. Standardize categorical data using CASE statements

```
SELECT isActive,
        CASE
                WHEN UPPER(isActive) IN ('TRUE', 'T') THEN 'True'
                ELSE 'False'
        END AS customer_status
FROM customer;
```

Q7. Consistent granularity for analysis (Extra)

```
SELECT DATE_FORMAT(CreateDate, ‘%Y-%m’) AS month,  SUM(sales) AS total_sales
FROM customer
GROUP BY DATE_FORMAT(CreateDate, ‘%Y-%m’)
ORDER BY month
```

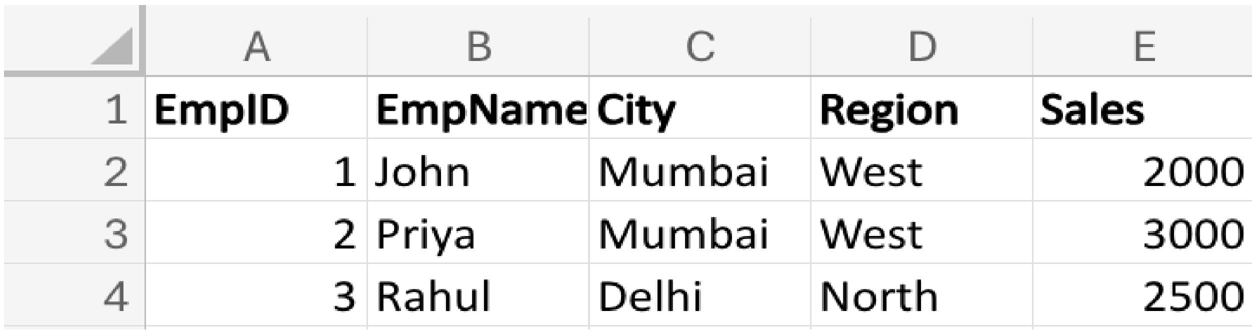
Note : Why not use AS name in group by as well?. It can be used because, GROUP BY clause is performed before the select statement, so the month Alias is not yet created to be used. Hence full expression is used in the group by clause.

THEORY

PARTITION BY

It is used in window functions to split the result into groups of rows and perform calculations within the group, without collapsing like how GROUP BY does.

EXAMPLE TABLE



	A	B	C	D	E
1	EmpID	EmpName	City	Region	Sales
2	1	John	Mumbai	West	2000
3	2	Priya	Mumbai	West	3000
4	3	Rahul	Delhi	North	2500

Q. Employee sales by city and region

```
SELECT EmpName, City, Region, Sales,
        SUM(sales) OVER (PARTITION BY City) AS total_city_sales,
        SUM(sales) OVER(PARTITION BY Region) AS total_region_sales,
        RANK() OVER(PARTITION BY City ORDER BY Sales) AS city_sales_rank
FROM employees;
```

Difference between GROUP BY & PARTITION BY

GROUP BY	**PARTITION BY**
Aggregate data into summary results	Apply window functions without collapsing
Rows are collapsed, returns one row per group	Rows are not collapsed, returns one row per row group
Aggregate functions are used (SUM, MIN, MAX, AVG)	Windows functions are used (RANK(), ROW_NUMBER()

S DHAANESH