Supplementary Material and Practice Exercise

CASE, COALESCE, and NULLIF: Conditional Expressions

- SQL also provides basic conditional constructs to determine the correct result. CASE provides a general mechanism for specifying conditional results. SQL also provides the COALESCE and NULLIF statements to deal with NULL values, but these have equivalent CASE statements.

CASE:ValueList

- The simplest form of the CASE statement determines if a value matches any values from a list and returns the corresponding result.

```
CASE <targetexpression>

WHEN <candidateexpression> THEN <resultexpression>

WHEN <candidateexpression> THEN <resultexpression> ...

WHEN <candidateexpression> THEN <resultexpression>

[ELSE <resultexpression>]  END
```

- CASE finds the first WHEN clause where <candidateexpression> = <targetexpression> and returns the value of the corresponding <resultexpression>. If no matches are found, the value of the <resultexpression> for the ELSE clause is returned. If the ELSE clause is not specified, an implicit ELSE NULL is added to the CASE statement

Assigning goodness values for the food groups.

```
SELECT name,
CASE foodgroup
WHEN 'Vegetable' THEN 'Good'
WHEN 'Fruit' THEN 'Good'
WHEN 'Milk' THEN 'Acceptable'
WHEN 'Bread' THEN 'Acceptable'
WHEN 'Meat' THEN 'Bad'
END AS quality
FROM ingredients;
```

CASE: Conditional List

- CASE also provides another powerful format where one can specify the conditional expressions in WHEN expressions.

```
CASE
WHEN Boolean_expression1 THEN expression1

[[WHEN Boolean_expression2 THEN expression2] [...]]

[ELSE expression]  END
```

❖ To show the power of the CASE statement, let's put together an order for ingredients. The amount that we want to order is based on the current inventory. If that inventory is below a threshold, then we want to place an order to raise it to the threshold; otherwise, we want to order a percentage of our inventory. The exact amount is based on the type of the food item because some will spoil more quickly than others.

```
SELECT name,
FLOOR (CASE
            WHEN inventory < 20 THEN 20 - inventory
            WHEN foodgroup = 'Milk' THEN inventory * 0.05
            WHEN foodgroup IN ('Meat', 'Bread') THEN inventory *
            0.10 WHEN foodgroup = 'Vegetable' AND unitprice <=
            0.03 THEN inventory * 0.10
            WHEN foodgroup = 'Vegetable' THEN inventory * 0.03
            WHEN foodgroup = 'Fruit' THEN inventory * 0.04
            WHEN foodgroup IS NULL THEN inventory * 0.07 ELSE
      0   END)
      AS size, vendorid
FROM ingredients
WHERE inventory < 1000 AND vendorid IS NOT NULL
ORDER BY vendorid, size;
```

NULLIF

❖ NULLIF takes two values and returns NULL if they are equal or the first value if the two values

are not equal. You can think of it as "NULL IF equal."

```
NULLIF(<value1>,<value2>)
```

❖ The World Health Organization (WHO) has declared that Meat is no longer a food group.

```
SELECT ingredientid, name, unit, unitprice,
NULLIF(foodgroup, 'Meat') AS foodgroup, inventory, vendorid  FROM
ingredients;
```

COALESCE

❖ COALESCE takes a list of values and returns the first non-NULL value.

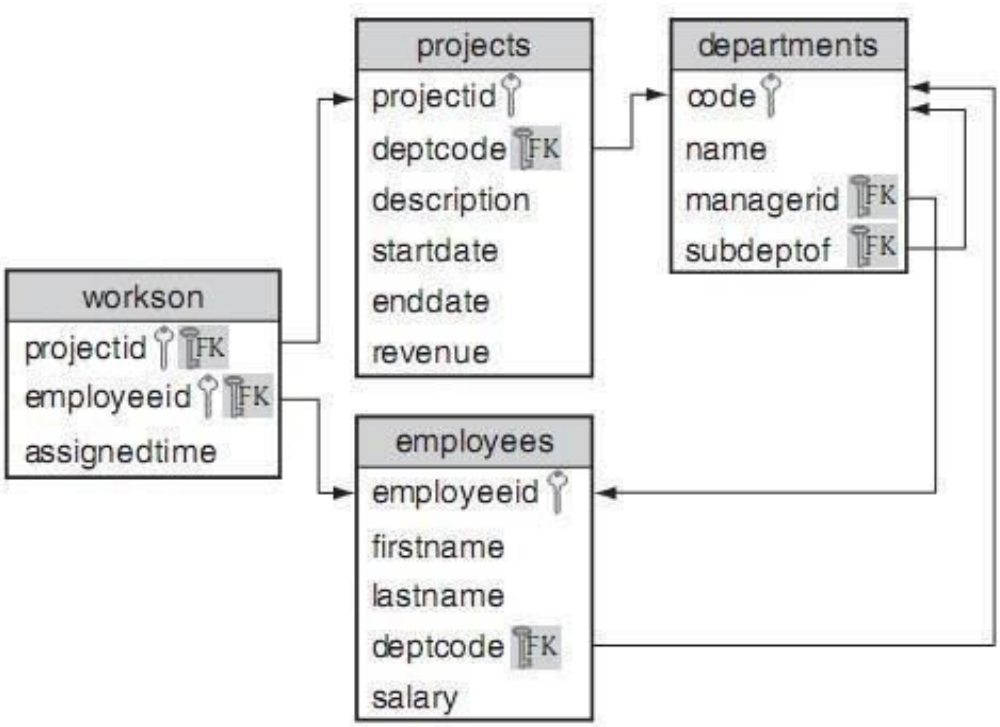
```
COALESCE(<value1>, <value2>, : : :, <valueN>)
```

- One practical use for COALESCE is providing a substitute for NULL values in the results. For example, if we want to display all of our items with a price, we would have to handle the ones with a NULL price.

```
SELECT name, price, COALESCE(price, 0.00) AS "no nulls"  FROM

items;
```

EXERCISES



Write a single SQL query for each of the following, based on employees database.

1. List all employee names as one field called name.
2. List all the department codes assigned to a project. Remove all duplicates.
3. Find the project ID and duration of each project.
4. Find the project ID and duration of each project. If the project has not finished, report its execution time as of now. [Hint: Getdate() gives current date]
5. Find the IDs of employees assigned to a project that is more than 20 hours per week. Write three queries using 20, 40, and 60 hour work weeks.
6. For each employee assigned to a task, output the employee ID with the following:
    - 'part time' if assigned time is < 0.33

    - 'split time' if assigned time is >= 0.33 and < 0.67
    - 'full time' if assigned time is >= 0.67
7. We need to create a list of abbreviated project names. Each abbreviated name concatenates the first three characters of the project description, a hyphen, and the department code. All characters must be uppercase (e.g., EMP-ADMIN).
8. For each project, list the ID and year the project started. Order the results in ascending order by year.
9. If every employee is given a 5% raise, find the last name and new salary of the employees who will make more than $50,000.
10. For all the employees in the HDWRE department, list their ID, first name, last name, and salary after a 10% raise. The salary column in the result should be named Next Year.
11. Create a neatly formatted directory of all employees, including their department code and name. The list should be sorted first by department code, then by last name, then by first name.