Happy Bakery

Case study

Happy Bakery is an artisan bakery located in Athens. It has been a family owned bakery since 1963. The current owner is the grandson of the initial owner and has taken over the business since 2014. The past few years he is trying to incorporate technology into the business and expand.

There is a file log of codes for every product and each product has a category, for example, sweet or savoury. There is also an inventory log and a waste log.

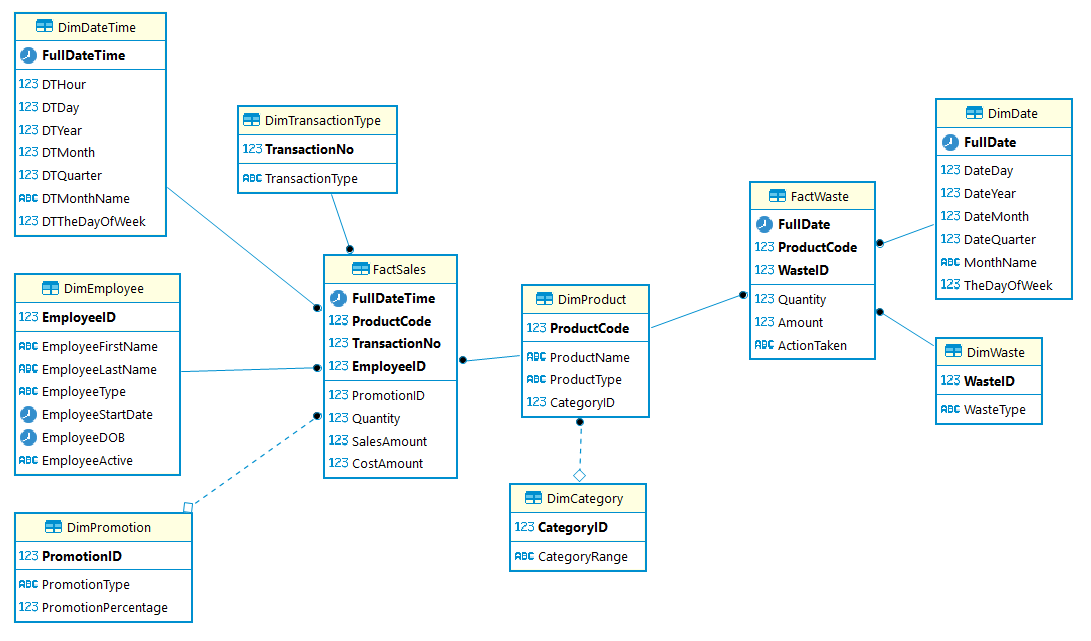
Every time the bakery introduces a new product, they bake enough for a 10 days sale and then they do a basic sales analysis to determine whether they will incorporate the product into their shop and if so, how many units they will bake per week. They always bake 20% more products than they predict that they will sell per week. Their weekly sales determine their supply orders.

Happy Bakery keeps a sales log storing data such as sales quantity, amount, datetime and more. They allow their customers to pay cash, card, Paypal or Google Pay.

The current owner is interested in finding out some business insight on sales and waste of the Bakery and asked for your help!

Below is a constellation schema for the bakery’s brand new data warehouse.

Try to query the data and have a look at the data warehouse, we will use this to complete tasks for the next 5 weeks.



Happy Bakery data warehouse query logbook

For further SQL examples and tutorial visit: <https://www.w3schools.com/sql/default.asp>

Instructions:

* During the lectures we will discuss some of these topics.
* Every topic has an example, followed by some exercises for you to do.
* Some of the solutions will be discussed during the workshops.
* Feel free to complete the tasks in advance if you are comfortable with SQL.
* You are **STRONGLY** advised to complete all the tasks before attempting the SQL queries for your coursework.

# Task 1: Straightforward selects

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

Select ProductName from DimProduct;

Select ProductName from DimProduct where ProductType = 'Bread';

Select \* from DimPromotion where PromotionPercentage > 0.10 order by PromotionType;

Select \* from DimDateTime where FullDateTime = '2019-01-21 09:00:00.000' ;

Now your turn. Create queries to:

1. List all the active employees.
2. List all the employees that work in production and are active.

# Task 2: Wild Cards & Join Syntax

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

Select \* from DimProduct where ProductType Like 'B%';

Select \* from DimProduct where ProductType Like 'B%d';

SELECT SalesAmount, ProductType FROM FactSales, DimProduct WHERE FactSales.ProductCode = DimProduct.ProductCode and PromotionID IS NULL;

SELECT SalesAmount, ProductType FROM FactSales, DimProduct WHERE FactSales.ProductCode = DimProduct.ProductCode and ProductName like '%bread%' Order by ProductType;

Now your turn. Create queries to:

1. List all the employees that their names start with the letter ‘B’
2. List the waste transactions that were gifted.
3. List all the waste transactions that were gifted due to decorating problems.
4. List all the waste transactions that were gifted due to decorating problems that involved products, which their name starts with the letter C and ends with the letters ed.

# Task 3: Dates

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

SELECT GETDATE()

Select Year(EmployeeDOB) from DimEmployee;

Select Month(EmployeeDOB) from DimEmployee;

Select Day(EmployeeDOB) from DimEmployee;

Select DATEADD(YEAR,1, FullDateTime) from FactSales where PromotionID IS NOT NULL;

Select Datediff(Year, EmployeeDOB, GETDATE()) from DimEmployee;

Select \* from DimEmployee Where (DATEDIFF(DAY, EmployeeDOB, GETDATE()) / 365.25) > 25;

Now your turn. Create queries to:

1. List all the names and ages of non-active employees.
2. List all the sales with a promotion from the year 2020.
3. List all the sales without a promotion between September 2019 and November 2019 that happened on mid day.
4. List all the sales without a promotion from the year 2020 that their quantities are less than 20 if they happened on a Monday.

# Task 4: Outer and Self Joins

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

SELECT SalesAmount, ProductType FROM FactSales inner join DimProduct on FactSales.ProductCode = DimProduct.ProductCode where PromotionID IS NULL; // simple join of two tables where the product code match the row is returned.

SELECT SalesAmount FROM FactSales right outer join DimEmployee on FactSales.EmployeeID = DimEmployee.EmployeeID Where Quantity <2; // would tell you the sales amount of every employee

SELECT SalesAmount FROM FactSales left join DimEmployee on FactSales.EmployeeID = DimEmployee.EmployeeID Where Quantity <2;

SELECT \* FROM FactSales inner join DimProduct on FactSales.ProductCode = DimProduct.ProductCode inner join DimCategory on DimCategory.CategoryID = DimProduct.CategoryID where PromotionID IS NULL; // join of more than 1 tables

Now your turn. Create queries to:

1. List the date, category range, product name and employee first name and last name of every sale occurred in december in descending date order.
2. List all sale, transaction type, product, and waste records that have sales and waste recorded on the same days. (hint: try different joints)
3. List all sale, transaction type, product, and waste records that have at least sales recorded on the common sale-waste days. (hint: try different joints)
4. List all sale, transaction type, product, and waste records that have at least wastes recorded on the common sale-waste days. (hint: try different joints)

# Task 5: Case statements

Run the following query: What does it do?

SELECT

EmployeeDOB, EmployeeFirstName, EmployeeLastName,

CASE

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000321' AS DATETIME) AND CAST('19000419' AS DATETIME) THEN 'Ares'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000420' AS DATETIME) AND CAST('19000520' AS DATETIME) THEN 'Taurus'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000521' AS DATETIME) AND CAST('19000620' AS DATETIME) THEN 'Gemini'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000621' AS DATETIME) AND CAST('19000722' AS DATETIME) THEN 'Cancer'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000723' AS DATETIME) AND CAST('19000822' AS DATETIME) THEN 'Leo'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000823' AS DATETIME) AND CAST('19000922' AS DATETIME) THEN 'Virgo'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000923' AS DATETIME) AND CAST('19001022' AS DATETIME) THEN 'Libra'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19001023' AS DATETIME) AND CAST('19001121' AS DATETIME) THEN 'Scorpio'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19001122' AS DATETIME) AND CAST('19001221' AS DATETIME) THEN 'Sagittarius'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19010101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19001222' AS DATETIME) AND CAST('19010119' AS DATETIME) THEN 'Capricorn'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000120' AS DATETIME) AND CAST('19000219' AS DATETIME) THEN 'Aquarius'

WHEN DATEADD(YEAR,-DATEDIFF(YEAR,'19000101',EmployeeDOB),EmployeeDOB) BETWEEN CAST('19000220' AS DATETIME) AND CAST('19000320' AS DATETIME) THEN 'Pisces'

END

FROM DimEmployee Where EmployeeLastName Like 'B%' or EmployeeLastName Like 'C%';

Now your turn.

Create a case statement that returns a list of all the bakery’s products and their range, as well as the word “sweet” if the product is sweet and “savoury” if the product is savoury. (*Tip: Run a query that returns all the products and categories first, so as you know how to group the sweet and savoury products.* )

# Task 6: Non-Correlated & Correlated Subquery

Run the following query - Sub query is executed independently – does not need main query

SELECT \* FROM DimProduct inner join FactSales on FactSales.ProductCode = DimProduct.ProductCode WHERE Quantity > (SELECT AVG(Quantity) from FactSales);

Run the following query - Sub query is part of the outer query and won’t run independently

SELECT Quantity, CostAmount, SalesAmount FROM FactSales fs WHERE Quantity > (SELECT AVG(Quantity) FROM FactSales WHERE fs.Quantity > 35 )

Now your turn.

1. List the details of the sales that have more than the average cost amount.
2. List the details of the sales that have more than the average sales amount for 2019.
3. List the details of the sales that have equal or less than average quantity.
4. List the details of the sales that have more or equal than the average SalesAmount of the top 25 sales in SalesAmount.

# Task 7: Aggregate Functions

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

SELECT top (10) ROUND(costamount, 2) AS RoundedCost FROM FactSales ;

SELECT Count(\*) from FactWaste WHERE Month(FullDate) =10;

SELECT Sum(Amount) AS [Total Number of Waste] from FactWaste;

SELECT AVG(Amount) AS [Average Waste] from FactWaste;

SELECT MAX(Amount) AS [The Most Costly Waste] from FactWaste;

SELECT MIN(Amount) AS [The Least Costly Waste] from FactWaste;

Now your turn.

1. What is the average cost of all the sales during 2019?
2. What is the average profit(Amount-Cost) of all the sales during 2019?
3. What is the most profitable sale?
4. What is the least costly sale?
5. How many sales exist for december 2020?
6. Round the average sale amount in 3 decimal places.

# Task 8: Changing data types

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

You can’t add an Int (EmployeeID) to a String (EmployeeLastName) so convert EmployeeID to a Char

SELECT CONVERT(CHAR, EmployeeID ) + EmployeeLastName as [Employee Profile] from DimEmployee;

SELECT CAST(EmployeeID as CHAR) + EmployeeLastName as [Employee Profile] from DimEmployee;

Now your turn.

1. List each employee’s first name with the year of their birth in one column.
2. List each employee’s first name with the day and month of their employment’s start date in one column.
3. Identify the error and the difference between:

|  |  |
| --- | --- |
| DECLARE @string VARCHAR(10);  SET @string = 1;  SELECT @string + ' is a string.' | DECLARE @notastring INT;  SET @notastring = '1';  SELECT @notastring + ' is not a string.' |

# Task 9: Date and Language formats

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

Dates

DECLARE @yourDateString NVARCHAR(10) = '12-09-2018';

SELECT @yourDateString + ' = the input.';

SET DATEFORMAT dmy;

SELECT CONVERT(DATE, @yourDateString) AS [DMY-Interpretation-of-input-format];

SET DATEFORMAT mdy;

SELECT CONVERT(DATE, @yourDateString) AS [MDY-Interpretation-of-input-format];

Languages

SET LANGUAGE Greek;

SELECT DATENAME(month, GETDATE()) AS 'Month Name';

SET LANGUAGE us\_english;

SELECT DATENAME(month, GETDATE()) AS 'Month Name' ;

GO

Now your turn.

1. List the date of birth of all the employees in dmy and mdy formats.
2. List the month of birth of the top 3 employees, order ascending on their last name.
3. List the month of birth of the top 3 employees in any language that you like, ordering ascending on their last name.

# Task 10: String Functions

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

SELECT UPPER (EmployeeFirstName) FROM DimEmployee;

SELECT LOWER (EmployeeFirstName) FROM DimEmployee;

SELECT Len (EmployeeFirstName) FROM DimEmployee;

SELECT TRIM (EmployeeFirstName) FROM DimEmployee;

SELECT CONCAT(EmployeeFirstName, ' ', EmployeeLastName) As FullName FROM DimEmployee;

SELECT CONCAT(EmployeeFirstName, ' ', EmployeeLastName) As [Full Name] FROM DimEmployee;

SELECT REVERSE(CONCAT(EmployeeFirstName, ' ', EmployeeLastName)) as [Full Name] FROM DimEmployee;

Now your turn.

1. Concatenate the product name and product type in 1 column and name it CompleteProduct.
2. Concatenate the product name, product type and category range in 1 column and name it full product description.

# Task 11: Group By & Having

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

HAVING specifies a search condition for a group or an aggregate and can be used only with the SELECT statement. HAVING is typically used with a GROUP BY clause. When GROUP BY is not used, there is an implicit single, aggregated group.

SELECT Count(FactSales.TransactionNo) As [Number of Transactions], Month(FullDateTime) As MONTH from DimTransactionType inner join FactSales on FactSales.TransactionNo = DimTransactionType.TransactionNo where year(FullDateTime) = 2019 group by Month(FullDateTime);

SELECT EmployeeLastName, EmployeeFirstName, Count(SalesAmount) from FactSales inner join DimEmployee on FactSales.EmployeeID = DimEmployee.EmployeeID WHERE PromotionID IS NOT NULL GROUP BY EmployeeLastName, EmployeeFirstName Having Count(SalesAmount) > 43 ORDER by Count(SalesAmount) desc;

Now your turn.

1. List the amount of waste per month that is gifted.
2. List how many different waste actions occurred per month in 2020.
3. What is the minimum and maximum amount of waste for January 2019 per different actions.
4. What is the amount of waste and the quantity of waste per month when the quantity of waste is at least 30.

# Task 12: Union

Try the following query and have a look at the results. Does it do what you expect? It combines result sets of two or more SELECT statements into a single result set.

Select Sum(Quantity) AS Sales, 'At most 25' AS [Age] from DimEmployee inner join FactSales on FactSales.EmployeeID = DimEmployee.EmployeeID where UPPER(EmployeeActive) = 'YES' and Datediff(Year, EmployeeDOB, GETDATE()) <= 25

UNION

Select Sum(Quantity) AS Sales, 'At least 26' AS [Age] from DimEmployee inner join FactSales on FactSales.EmployeeID = DimEmployee.EmployeeID where UPPER(EmployeeActive) = 'YES' and Datediff(Year, EmployeeDOB, GETDATE()) > 25 ;

Now your turn.

1. List the total sales amount and the total cost amount of all the sales per type of employee.
2. List the total sales amount and the total cost amount of all the sales per type of employee for the months January, February and March 2019.

# Task 13: Group By extensions or subtotal operators

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

## Rollup

Creates a group for each combination of column expressions and "rolls up" the results into subtotals and grand totals.

Select Sum(SalesAmount) AS TotalSales , MONTH(FullDateTime) As PerMonth, Year(FullDateTime) As PerYear from FactSales

Group by Rollup(MONTH(FullDateTime), YEAR(FullDateTime));

Now your turn.

1. List the amount of waste per month that is gifted grouped per month and year, as well as their grand totals.
2. What is the amount of waste and the quantity of waste per month and year when the quantity of waste is at least 30, as well as their grand totals.

## Cube

Creates groups for all possible combinations of columns. For GROUP BY CUBE (a, b) the results have groups for unique values of (a, b), (NULL, b), (a, NULL), and (NULL, NULL).

Select Sum(SalesAmount) AS TotalSales , MONTH(FullDateTime) As PerMonth, Year(FullDateTime) As PerYear from FactSales Group by Cube(MONTH(FullDateTime), YEAR(FullDateTime));

Now your turn.

1. List the amount of waste per month that is gifted grouped per month and year, as well as their cubed grand totals.
2. What is the amount of waste and the quantity of waste per month and year when the quantity of waste is at least 30, as well as their cubed grand totals.

## Grouping sets

It gives you the ability to combine multiple GROUP BY clauses into one GROUP BY clause. The results are the equivalent of UNION ALL of the specified groups.

Try this first:

Select Sum(SalesAmount) AS TotalSales , MONTH(FullDateTime) As PerMonth, Year(FullDateTime) As PerYear from FactSales Group by GROUPING SETS (Rollup(MONTH(FullDateTime), YEAR(FullDateTime)), Cube(MONTH(FullDateTime), YEAR(FullDateTime)));

The results are the same as this query that returns a union of the two GROUP BY statements.

Select Sum(SalesAmount) AS TotalSales , MONTH(FullDateTime) As PerMonth, Year(FullDateTime) As PerYear from FactSales Group by Rollup(MONTH(FullDateTime), YEAR(FullDateTime))

UNION ALL

Select Sum(SalesAmount) AS TotalSales , MONTH(FullDateTime) As PerMonth, Year(FullDateTime) As PerYear from FactSales Group by Cube(MONTH(FullDateTime), YEAR(FullDateTime));

Now your turn.

1. List the amount of waste per month that is gifted grouped per month and year, as well as their cubed grand totals using grouping sets..
2. What is the amount of waste and the quantity of waste per month and year when the quantity of waste is at least 30, as well as their cubed grand totals, using grouping sets..

# Task 14: Over and PARTITION BY

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

We can use the PARTITION BY clause with the Over clause to specify the column on which we need to perform aggregation.

SELECT Month(FullDate) As [Month], Year(FullDate) As [Year], ActionTaken, Min(Amount) OVER(PARTITION BY ActionTaken) As MinAmount, AVG(Amount) OVER(PARTITION BY ActionTaken) As AverageAmount from FactWaste Order By Year(FullDate), Month(FullDate), ActionTaken;

SELECT ROW\_NUMBER() OVER(PARTITION BY EmployeeType ORDER BY EmployeeLastName) AS "Row Number", EmployeeLastName, EmployeeFirstName, EmployeeType from DimEmployee Where EmployeeActive <> 'No';

Now your turn.

1. List the Sum, average min and maximum of sales per promotion type, print the date in dmy format and order it by year.
2. List the profit (SalesAmount-CostAmount) and row number of sales per month, print the date in dmy format and order it by month for the year 2020.

# Task 15: Pivot

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output.

SELECT ProductName, Pancake, Loaf, Bread, Dansih, Tart, [Sweet Bagel], [Savoury Bagel]

FROM (

SELECT ProductType, ProductName, CategoryID

FROM DimProduct) up

PIVOT (Count(CategoryID) FOR ProductType IN (Pancake, Loaf, Bread, Dansih, Tart, [Sweet Bagel], [Savoury Bagel])) AS pvt

ORDER BY ProductName

Pivot more than 1 tables

SELECT ProductType, Breads, Bagels, Doughnuts, Pies, Pastries

FROM (

SELECT ProductType, ProductName, CategoryRange

FROM DimProduct inner join DimCategory on DimCategory.CategoryID = DimProduct.CategoryID) up

PIVOT (Count(ProductName) FOR CategoryRange IN (Breads, Bagels, Doughnuts, Pies, Pastries)) AS pvt

ORDER BY ProductType

Now your turn.

1. Pivot the FactWaste table the new columns must be the different types of action taken and you must sum the amount per month in the rows. (Tip: *Find the different types of actions taken first*)

# Task 16: Ranking

The following examples show the four ranking functions used in the same query. Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

The DENSE\_RANK() is a window function that assigns a rank to each row within a partition of a result set. Unlike the RANK() function, the DENSE\_RANK() function returns consecutive rank values. Rows in each partition receive the same ranks if they have the same values.

SELECT EmployeeFirstName, EmployeeLastName

,ROW\_NUMBER() OVER (ORDER BY quantity) AS "Row Number"

,RANK() OVER (ORDER BY quantity) AS Rank

,DENSE\_RANK() OVER (ORDER BY quantity) AS "Dense Rank"

,PromotionType

,PromotionPercentage

FROM DimEmployee inner join FactSales on DimEmployee.EmployeeID = FactSales.EmployeeID inner join DimPromotion on FactSales.PromotionID = DimPromotion.PromotionID

where FactSales.PromotionID IS NOT NULL;

SELECT EmployeeFirstName, EmployeeLastName

,ROW\_NUMBER() OVER (ORDER BY PromotionPercentage) AS "Row Number"

,RANK() OVER (ORDER BY PromotionPercentage) AS Rank

,DENSE\_RANK() OVER (ORDER BY PromotionPercentage) AS "Dense Rank"

,PromotionType

,PromotionPercentage

FROM DimEmployee inner join FactSales on DimEmployee.EmployeeID = FactSales.EmployeeID inner join DimPromotion on FactSales.PromotionID = DimPromotion.PromotionID

where FactSales.PromotionID IS NOT NULL;

# Task 17: Offset Functions

The OFFSET and FETCH clauses are the options of the ORDER BY clause. They allow you to limit the number of rows to be returned by a query.

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

There are two categories of offset functions in an SQL server namely, functions whose offset is relative to the current row (LAG and LEAD functions) and functions whose offset is relative to the start or the end of the window frame, such as First\_Value and Last\_Value.

## LAG and LEAD

The LAG and LEAD functions allow you to return a value expression from a row in a window partition that is given an offset before (LAG) or after (LEAD) of the current row.

This example returns the SalesAmount for each employee as we as the values of the previous and next sales of that employee.

select EmployeeID, FullDateTime, ProductCode, SalesAmount,

LAG(SalesAmount) Over (PARTITION by EmployeeID order by FullDateTime, ProductCode) as preSalesAmount,

Lead(SalesAmount) Over (PARTITION by EmployeeID order by FullDateTime, ProductCode) as postSalesAmount

from FactSales

## FIRST\_VALUE and LAST\_VALUE

Given an ordered set of rows, FIRST\_VALUE returns the value of the specified expression with respect to the first row in the window frame. The LAST\_VALUE function returns the value of the expression with respect to the last row in the frame.

FIRST\_VALUE() function example to return product categories with the lowest sales volumes in 2019 and 2020.

Select ProductType, YEAR(FullDateTime) AS [Year], Quantity,

FIRST\_VALUE(ProductType) OVER (

PARTITION BY YEAR(FullDateTime) ORDER BY Quantity

) As LowestSalesVolume

From DimProduct INNER JOIN FactSales on DimProduct.ProductCode = FactSales.ProductCode

Where YEAR(FullDateTime) BETWEEN 2019 and 2020;

This example demonstrates how to return along with each employee’s sale, the current sale amount as well as the sa;le amounts from the employee’s first and last sales.

You do not need to use the range as ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW since this is the default range of rows but I added it here to demonstrate a good practise of “cheaper” or lighter query runtimes.

select EmployeeID, FullDateTime, ProductCode, SalesAmount,

FIRST\_VALUE(SalesAmount) Over (PARTITION by EmployeeID order by FullDateTime, ProductCode ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as FIRSTSale,

LAST\_VALUE(SalesAmount) Over (PARTITION by EmployeeID order by FullDateTime, ProductCode) as LastSale

from FactSales

# Task 18: Rank distribution Functions

Rank distribution analytical functions calculate an aggregate value based on a group of rows. Unlike aggregate functions, however, analytic functions can return multiple rows for each group. Use these functions to compute moving averages, running totals or percentages within a group.

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

## CUME\_DIST

The CUME\_DIST() calculates the cumulative distribution of a value within a group of values.

CUME\_DIST() function to calculate the sales percentile for each sales staff in 2021 and 2020.

SELECT EmployeeFirstName, EmployeeLastName, YEAR(FactSales.FullDateTime) AS [YEAR],

Round(CUME\_DIST () OVER (PARTITION BY YEAR(FactSales.FullDateTime) ORDER BY SalesAmount),3) AS CumeDist

FROM DimEmployee inner join FactSales on DimEmployee.EmployeeID = FactSales.EmployeeID

Where YEAR(FactSales.FullDateTime) IN (2021, 2020)

ORDER BY YEAR(FactSales.FullDateTime) DESC;

## PERCENT\_RANK

The PERCENT\_RANK() function is similar to the CUME\_DIST() function. The PERCENT\_RANK() function evaluates the relative standing of a value within a partition of a result set.

PERCENT\_RANK() to calculate the sales percentile for each staff in 2021 and 2020.

SELECT EmployeeFirstName, EmployeeLastName, YEAR(FactSales.FullDateTime) AS [YEAR],

Round(PERCENT\_RANK () OVER (PARTITION BY YEAR(FactSales.FullDateTime) ORDER BY SalesAmount),3) AS CumeDist

FROM DimEmployee inner join FactSales on DimEmployee.EmployeeID = FactSales.EmployeeID

Where YEAR(FactSales.FullDateTime) IN (2021, 2020)

ORDER BY YEAR(FactSales.FullDateTime) DESC;

# Task 19: Logical Functions

## IIF

Returns one of two values, depending on whether the Boolean expression evaluates to true or false.

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

DECLARE @a INT = 45, @b INT = 40;

SELECT [Result] = IIF( @a > @b, 'TRUE', 'FALSE' );

DECLARE @P INT = NULL, @S INT = NULL;

SELECT [Result] = IIF( 45 > 30, @P, @S );

SELECT IIF (40 > (Select Top (1) Quantity from FactWaste Order By Quantity asc), 'High', 'Low' )

## CHOOSE

Returns the item at the specified index from a list of values.

Try the following queries and have a look at the results. What are the differences between them? Do they do what you expect?

SELECT CHOOSE ( 3, 'Food', 'Sleep', 'Shopping', 'Gym' ) AS Result;

SELECT EmployeeType, EmployeeStartDate, CHOOSE(MONTH(EmployeeStartDate),'Winter','Winter', 'Spring','Spring','Spring','Summer','Summer',

'Summer','Autumn','Autumn','Autumn','Winter') AS Quarter\_Hired

FROM DimEmployee

WHERE YEAR(EmployeeStartDate) > 2015

ORDER BY YEAR(EmployeeStartDate);