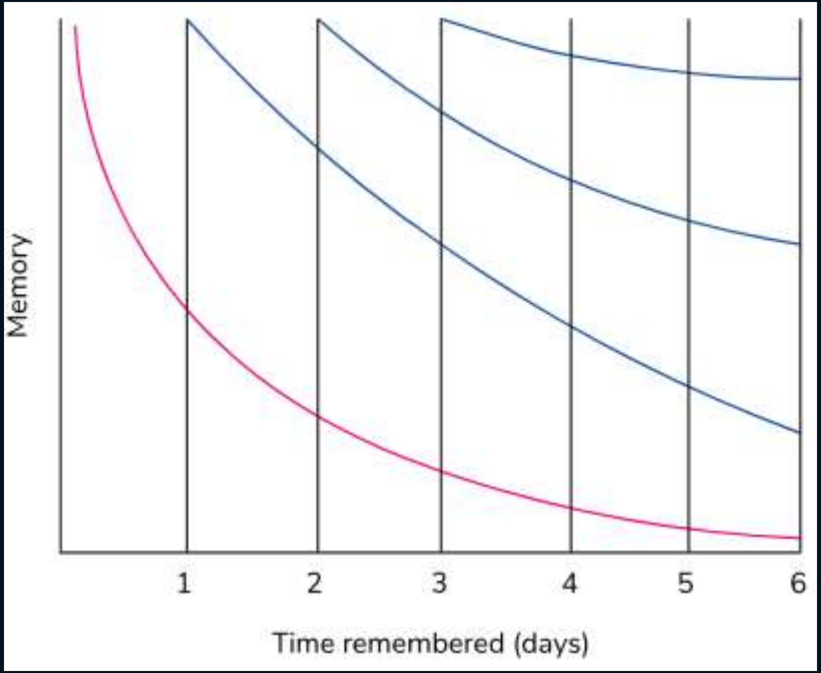
correlation one

TECH FOR JOBS

Support Session 8

Ahmad Albaqsami

The Forgetting Curve



Learners forget:

- 50% of information in 1 hour
- 70% in 24 hours
- 90% in a week

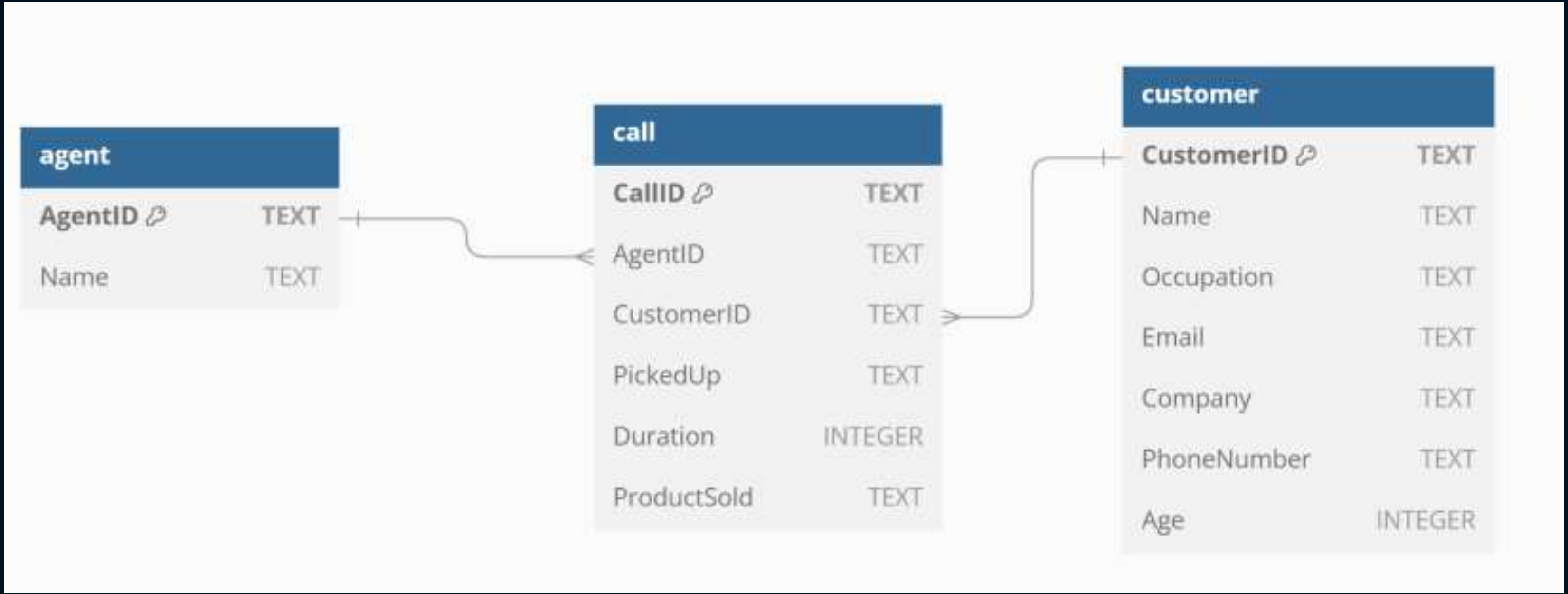
-Hermann Ebbinghaus: Memory loss follows exponential decay.

- Mitigation strategies:
  - Spaced repetition enhances retention.
  - Overlearning reinforces memory.

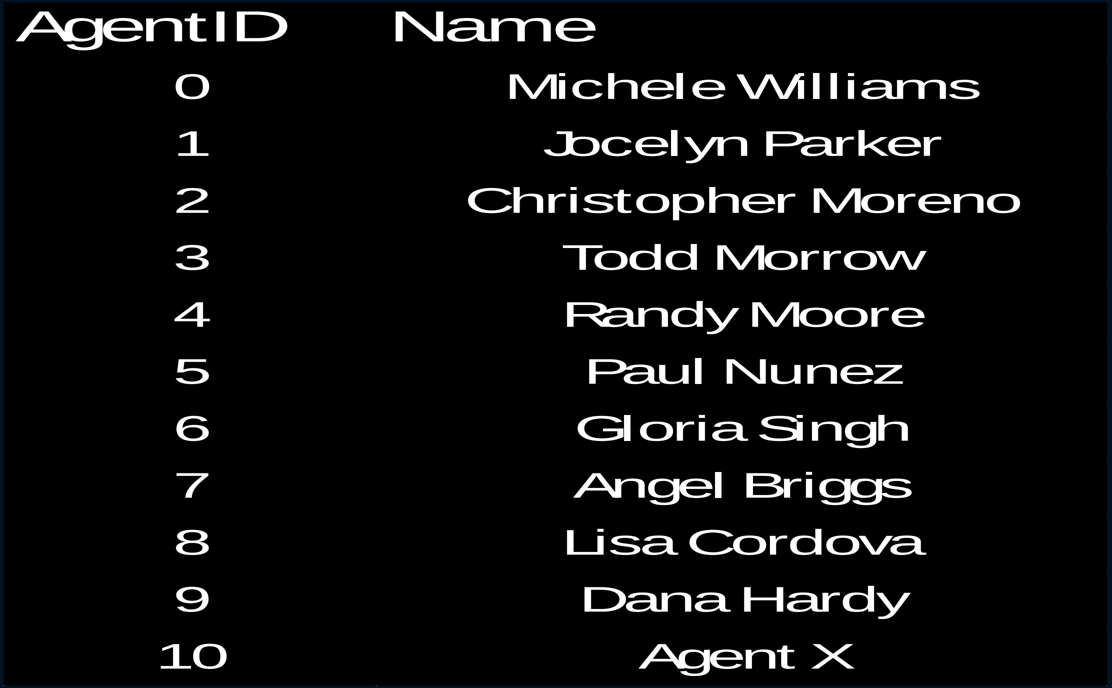
Agenda

- SQL Aggregation and Subqueries part 1
- SQL Aggregation and Subqueries part 2

Case Study


agent
AgentID TEXT
Name TEXT
call
CallID TEXT
AgentID TEXT
CustomerID TEXT
PickedUp TEXT
Duration INTEGER
ProductSold TEXT
customer
CustomerID TEXT
Name TEXT
Occupation TEXT
Email TEXT
Company TEXT
PhoneNumber TEXT
Age INTEGER

Agent Table (11 records)



AgentID	Name
0	Michele Williams
1	Jocelyn Parker
2	Christopher Moreno
3	Todd Morrow
4	Randy Moore
5	Paul Nunez
6	Gloria Singh
7	Angel Briggs
8	Lisa Cordova
9	Dana Hardy
10	Agent X

Customer Table (1000 records)

CustomerID	Name	Occupation	Email	Company	PhoneNumber	Age
0	David Melton	Unemployed	DMelton@zoho.com	Morris, Winters and Ramirez	409-093-0748	16
1	Michael Gonzalez	Student	Gonzalez_Michael@yahoo.com	Hernandez and Sons	231-845-0673	19
2	Amanda Wilson	Student	Amanda.Wilson75@verizon.com	Mooney, West and Hansen	844-276-4552	18
3	Robert Thomas	Engineer, structural	RThomas@xfinity.com	Johnson-Gordon	410-404-8000	25
…	…	…	…	…	…	…

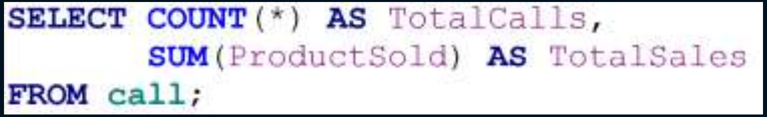
Call Table (9940 records)

CallID	AgentID	CustomerID	PickUp	Duration	ProductSold
0	10	179	0	0	0
1	5	691	1	116	0
2	10	80	1	165	0
3	6	629	1	128	0
4	8	318	1	205	0
5	7	319	1	225	1
…	…	…	…	…	…

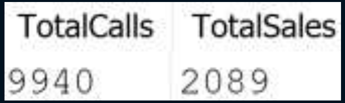
Introduction to Aggregate Functions

- Aggregate functions perform calculations on a set of values and return a single summarized value.

- **Common Aggregate Functions**:
    - COUNT(): Counts rows in a table/column.
    - SUM(): Calculates the sum of values.
    - AVG(): Computes the average value.
    - MIN(): Finds the minimum value.
    - MAX(): Finds the maximum value.



```sql
SELECT COUNT(*) AS TotalCalls,
       SUM(ProductSold) AS TotalSales
FROM call;
```



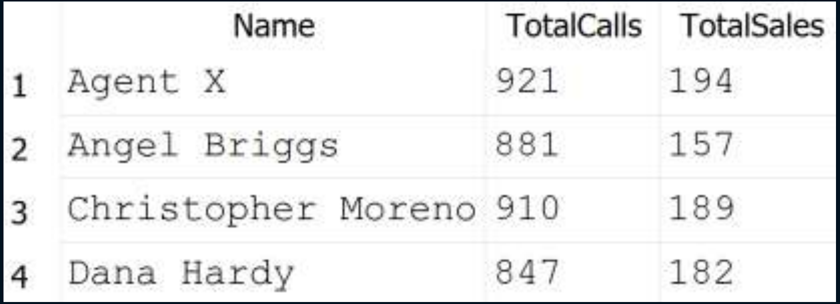
TotalCalls	TotalSales
9940	2089

Evaluating Individual Agent Performance (GROUP BY)

- **GROUP BY**: Groups rows by unique values in a column.
- Syntax:

```
GROUP BY column_name
```

-
  ```
  SELECT agent.name,
          COUNT(*) AS TotalCalls,
          SUM(call.ProductSold) AS TotalSales
  FROM call
  JOIN agent ON agent.AgentID = call.AgentID
  GROUP BY agent.name
  ORDER BY agent.name;
  ```

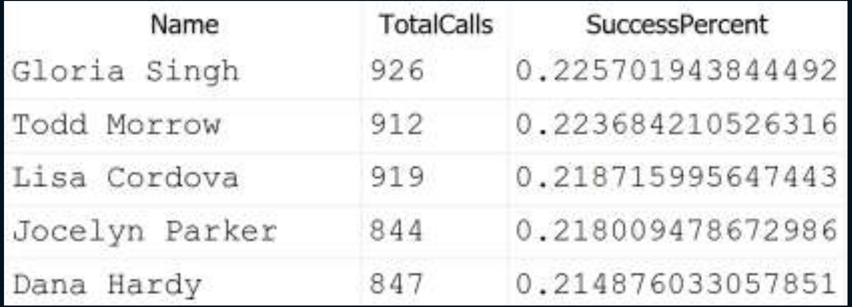


	Name	TotalCalls	TotalSales
1	Agent X	921	194
2	Angel Briggs	881	157
3	Christopher Moreno	910	189
4	Dana Hardy	847	182

Calculating Success Rates (AVG())

- Use AVG() to calculate the percentage of successful calls:

```
SELECT agent.name,
       COUNT(*) AS TotalCalls,
       AVG(call.ProductSold) AS SuccessPercent
FROM call
JOIN agent ON agent.AgentID = call.AgentID
GROUP BY agent.name
ORDER BY SuccessPercent DESC;
```

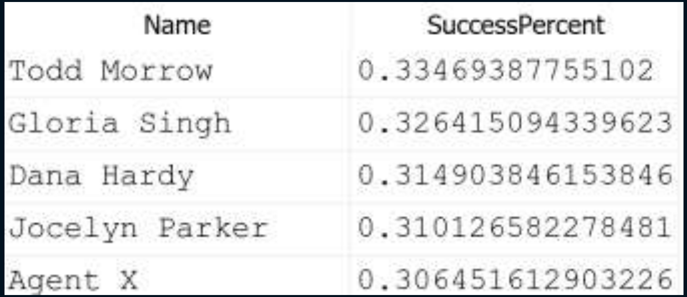


Name	TotalCalls	SuccessPercent
Gloria Singh	926	0.225701943844492
Todd Morrow	912	0.223684210526316
Lisa Cordova	919	0.218715995647443
Jocelyn Parker	844	0.218009478672986
Dana Hardy	847	0.214876033057851

Filtering by Call and Customer Attributes

- Filter queries using **WHERE** with conditions:

```
SELECT agent.name,
       AVG(call.ProductSold) AS SuccessPercent
FROM call
JOIN agent ON agent.AgentID = call.AgentID
JOIN customer ON customer.CustomerID = call.CustomerID
WHERE call.PickedUp = 1 AND customer.age >= 18
GROUP BY agent.name
ORDER BY SuccessPercent DESC;
```

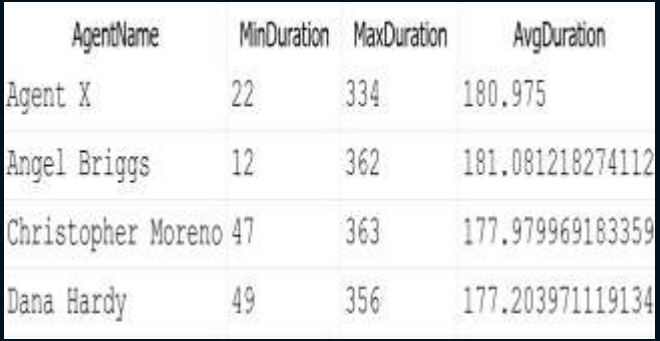


Name	SuccessPercent
Todd Morrow	0.33469387755102
Gloria Singh	0.326415094339623
Dana Hardy	0.314903846153846
Jocelyn Parker	0.310126582278481
Agent X	0.306451612903226

Call Duration Statistics

- Find average, minimum, and maximum call durations:

```
SELECT agent.Name AS AgentName,
             MIN(Call.Duration) AS MinDuration,
             MAX(Call.Duration) AS MaxDuration,
             AVG(Call.Duration) AS AvgDuration
FROM call
JOIN agent ON agent.AgentID = call.AgentID
WHERE call.Duration > 0
GROUP BY agent.Name;
```

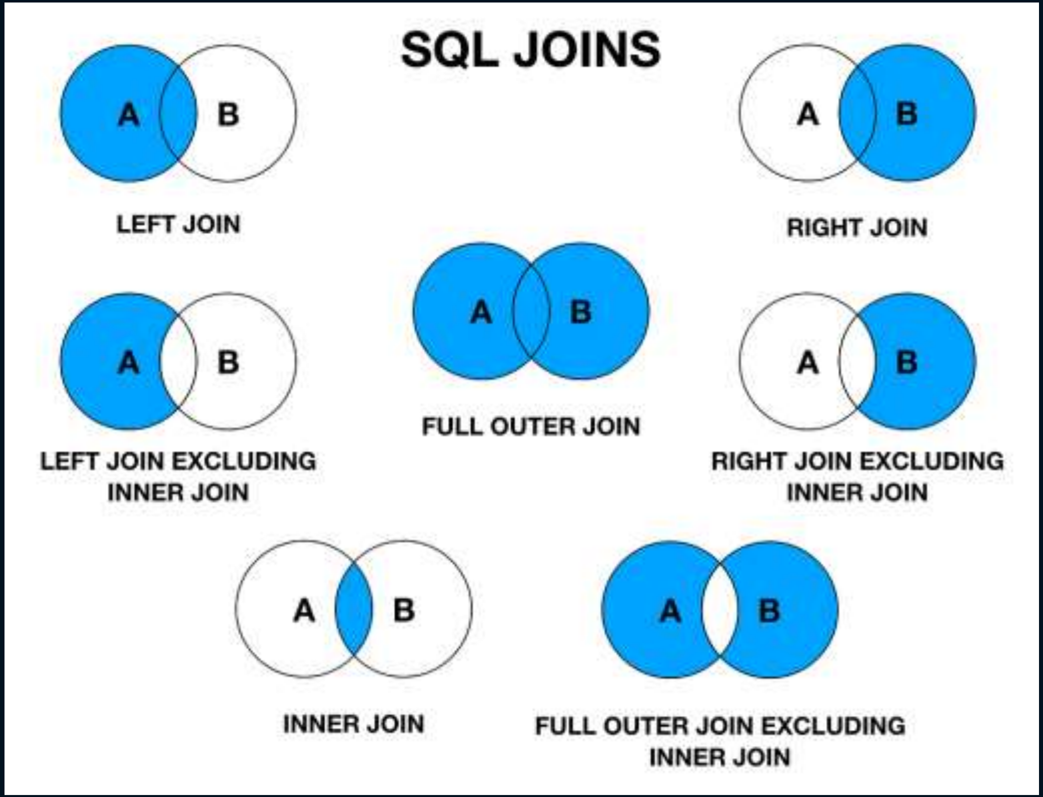


AgentName	MinDuration	MaxDuration	AvgDuration
Agent X	22	334	180.975
Angel Briggs	12	362	181.081218274112
Christopher Moreno	47	363	177.979969183359
Dana Hardy	49	356	177.203971119134

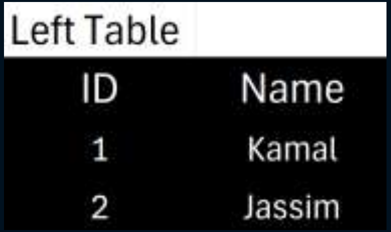
SQL Joins Overview

- JOIN Types:
  - (INNER) JOIN: Returns matching rows in both tables.
  - LEFT (OUTER) JOIN: Returns all rows from the left table and matching rows from the right table.
  - RIGHT (OUTER) JOIN: Returns all rows from the right table and matching rows from the left table.
  - FULL (OUTER) JOIN: Returns all rows when there is a match in either table.

SQL Joins

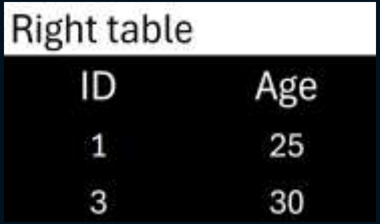

SQL JOINS
A
B
LEFT JOIN
RIGHT JOIN
FULL OUTER JOIN
LEFT JOIN EXCLUDING INNER JOIN
RIGHT JOIN EXCLUDING INNER JOIN
INNER JOIN
FULL OUTER JOIN EXCLUDING INNER JOIN

SQL Join exercise



Left Table

ID	Name
1	Kamal
2	Jassim



Right table

ID	Age
1	25
3	30

ID	Name	Age
1	Kamal	25

ID	Name	Age
1	Kamal	25
3	Null	30

ID	Name	Age
1	Kamal	25
2	Jassim	Null

ID	Name	Age
1	Kamal	25
2	Jassim	Null
3	Null	30

SQL Join exercise

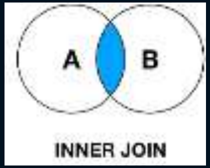
Left Table

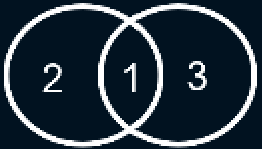
ID	Name
1	Kamal
2	Jassim

Right table

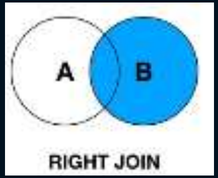
ID	Age
1	25
3	30

ID	Name	Age
1	Kamal	25

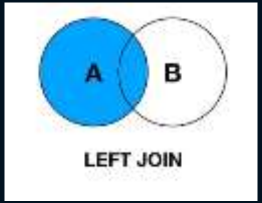




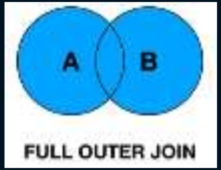
ID	Name	Age
1	Kamal	25
3	Null	30



ID	Name	Age
1	Kamal	25
2	Jassim	Null



ID	Name	Age
1	Kamal	25
2	Jassim	Null
3	Null	30



Advanced Filtering with IN

- Use IN for conditional filtering:

```
SELECT *
FROM customer
WHERE age IN (18, 20, 25, 30, 35);
```

- Use IN with subqueries:

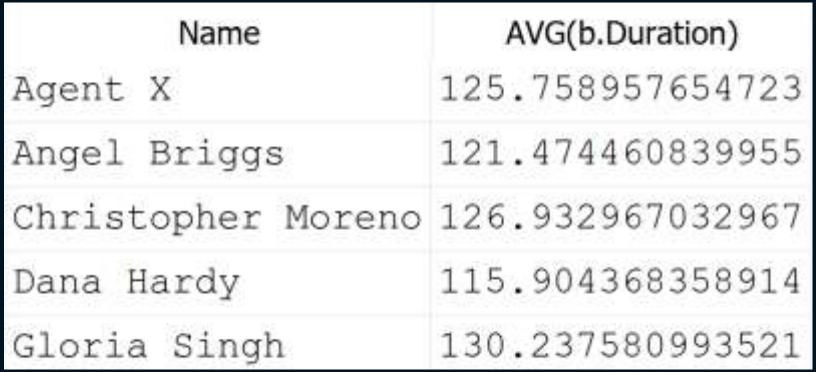
```
SELECT COUNT(*)
FROM call
WHERE AgentID IN (SELECT AgentID FROM agent);
```

Implicit JOIN

- A JOIN statement without using the JOIN keyword.

```
SELECT a.column_from_table_a, b.column_from_table_b
FROM table_a AS a, table_b AS b
WHERE a.shared_column = b.shared_column;

SELECT a.name, AVG(b.Duration)
FROM agent a, call b
WHERE a.AgentID = b.AgentID
GROUP BY a.name;
```



Name	AVG(b.Duration)
Agent X	125.75895765472З
Angel Briggs	121.474460839955
Christopher Moreno	126.932967032967
Dana Hardy	115.904368358914
Gloria Singh	130.237580993521

Subqueries

- A query nested within another query to make it relational to the dataset.

```
SELECT name, CustomerID, age
FROM customer
WHERE age >= (SELECT AVG(age) FROM customer);
```

Note: Subqueries maintain **query integrity** when data changes.

Subquery Exercise

- Get the name, call ID, and duration for calls with duration greater than the average (excluding 0).
- **Step 1: Inner Query**: Calculate the average duration (excluding 0):

```
SELECT AVG(b.Duration)
FROM call b
WHERE b.Duration > 0;
```

- **Step 2**: **Outer Query**: Filter calls using the result:

```
SELECT a.name, b.CallID, b.Duration
FROM agent a, call b
WHERE a.AgentID = b.AgentID AND b.Duration >= (SELECT AVG(b.Duration)
FROM call b WHERE b.Duration > 0);
```

Common Table Expressions (CTEs)

- A temporary result set used within a query to simplify and reuse code.

```
WITH temporary_table AS
(subquery)
SELECT *
FROM temporary_table;
```

- **Chained CTEs**:

```
WITH temp1 AS (subquery1),
        temp2 AS (subquery2)
SELECT * FROM ... ;
```

CTE Example

- Original query

```
SELECT *
FROM (SELECT *
          FROM customer
          FULL JOIN call ON customer.CustomerID = call.CustomerID
          WHERE Name is NULL) customer_outer_call
FULL JOIN (SELECT *
                FROM agent
                FULL JOIN call ON agent.AgentID = call.AgentID
                WHERE Name IS NULL) agent_outer_call
ON customer_outer_call.Name = agent_outer_call.Name;
```

CTE Example

- Refactored

```
WITH customer_outer_call AS
(    SELECT *
     FROM customer
     FULL JOIN call ON customer.CustomerID = call.CustomerID
     WHERE Name IS NULL),
agent_outer_call AS
(    SELECT *
     FROM agent
     FULL JOIN call ON agent.AgentID = call.AgentID
     WHERE Name IS NULL)
SELECT *
FROM customer_outer_call
FULL JOIN agent_outer_call ON customer_outer_call.Name = agent_outer_call.Name;
```