Best Practice

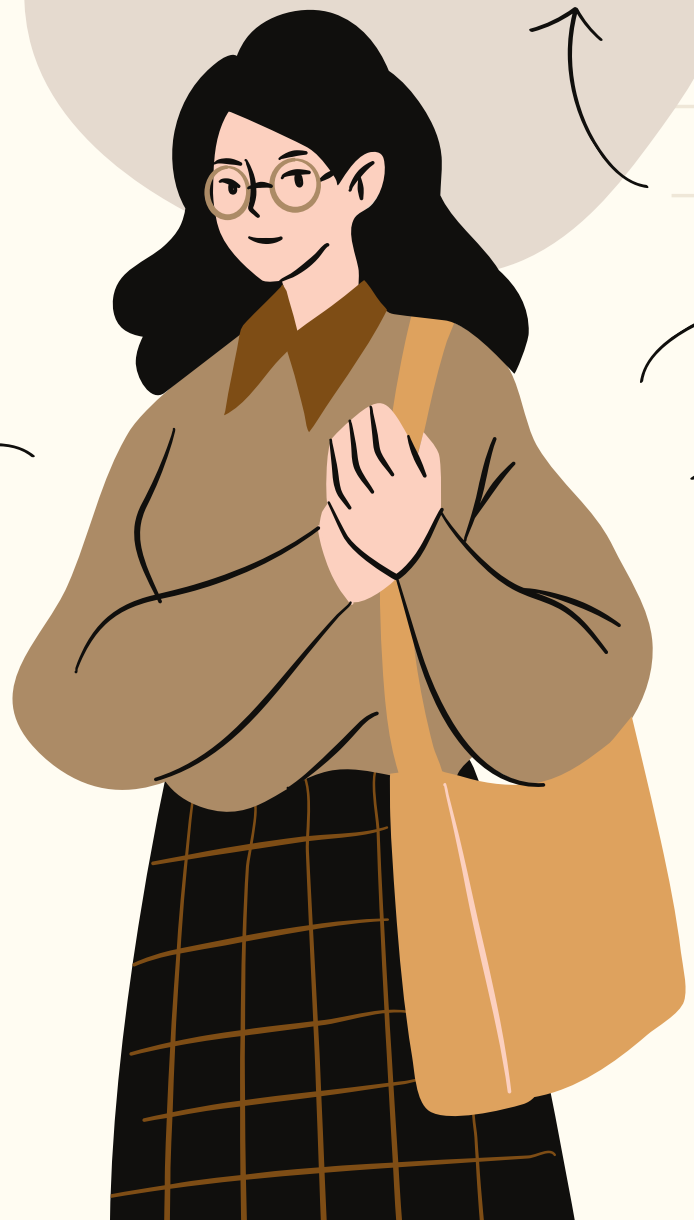
"Top 10 SQL Tips for Writing Efficient Queries"

Prefer INNER JOIN

Avoid DISTINCT & ORDER BY

Select only what you need

Avoid leading %



Use IN instead of OR

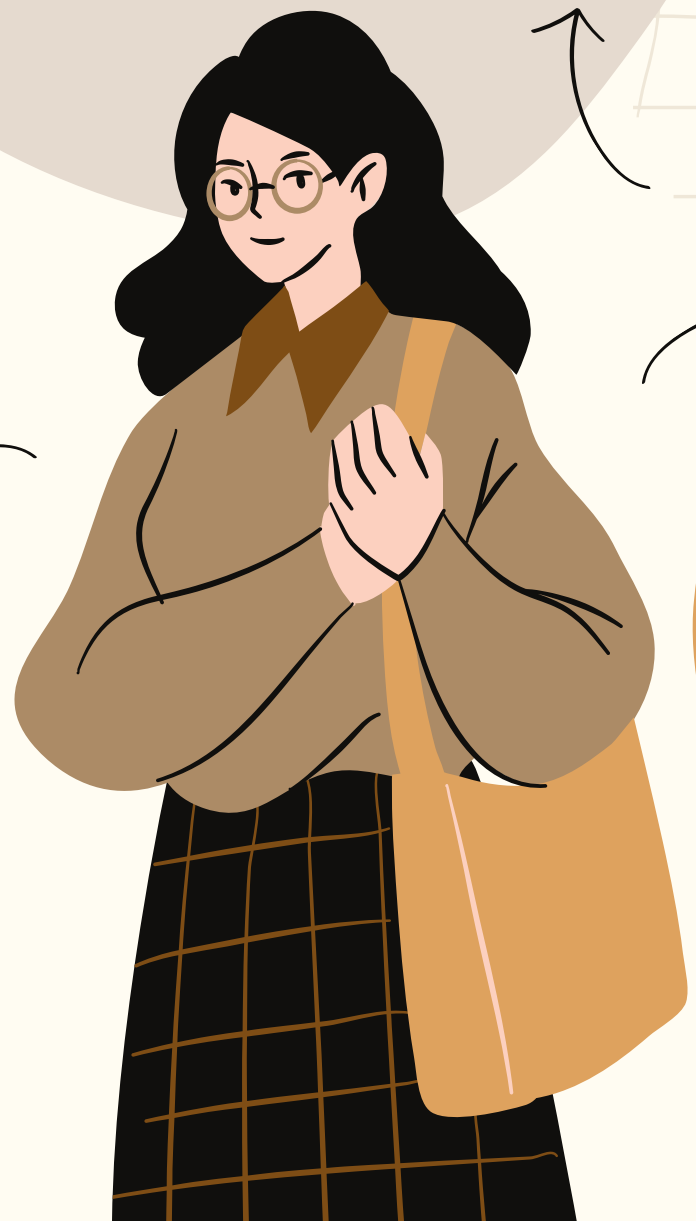
Tip 1: Select Only What you need

Best Practice:

"FETCHING DATA"

```sql
-- Bad Practice
SELECT * FROM Sales.Customers

-- Good Practice
SELECT CustomerID, FirstName,
LastName FROM Sales.Customers
```



Bad Practice

Results | Messages

	CustomerID	FirstName	LastName	Country	Score
1	1	Jossef	Goldberg	Germany	350
2	2	Kevin	Brown	USA	900
3	3	Mary	NULL	USA	750
4	4	Mark	Schwarz	Germany	500
5	5	Anna	Adams	USA	0

Good Practice

Results | Messages

	CustomerID	FirstName	LastName
1	1	Jossef	Goldberg
2	2	Kevin	Brown
3	3	Mary	NULL
4	4	Mark	Schwarz
5	5	Anna	Adams

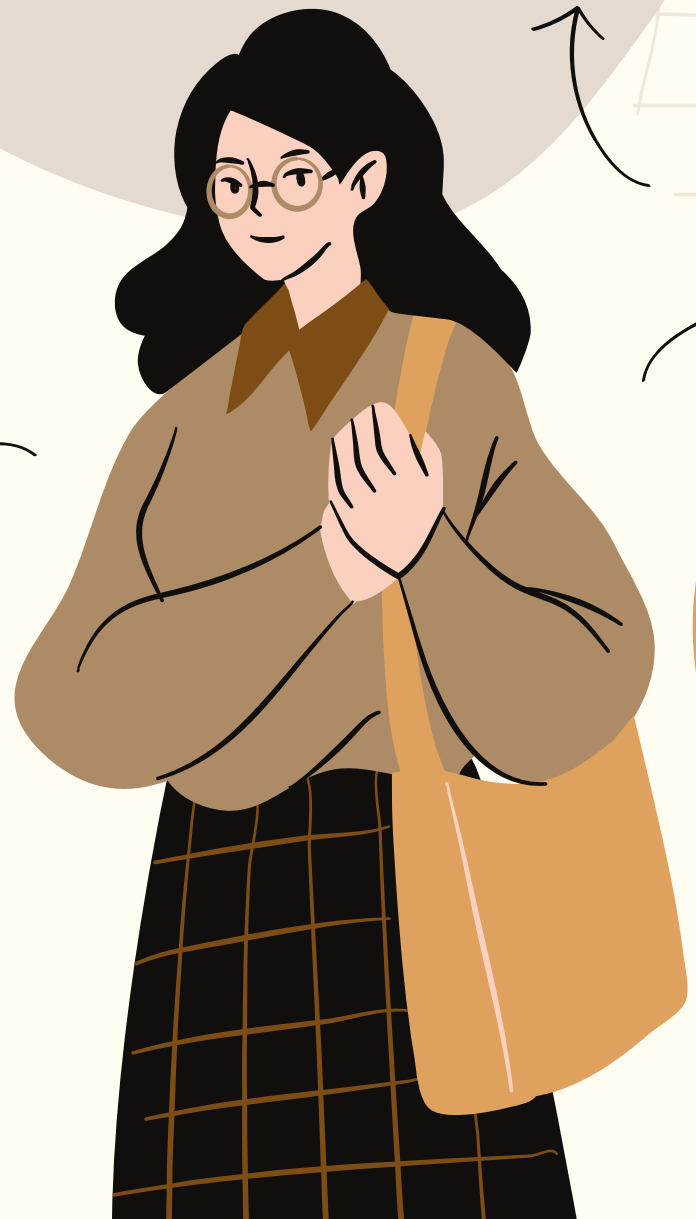
Tip 2: Avoid Unnecessary DISTINCT & ORDER BY

```sql
-- Bad Practice
 SELECT DISTINCT
   FirstName
FROM Sales.Customers
ORDER BY FirstName

-- Good Practice
SELECT FirstName
FROM Sales.Customers
```

Best Practice:

"FETCHING DATA"



Bad Practice

Results | Messa

	FirstName
1	Anna
2	Jossef
3	Kevin
4	Mark
5	Mary

Good Practice

Results | Messa

	FirstName
1	Jossef
2	Kevin
3	Mary
4	Mark
5	Anna

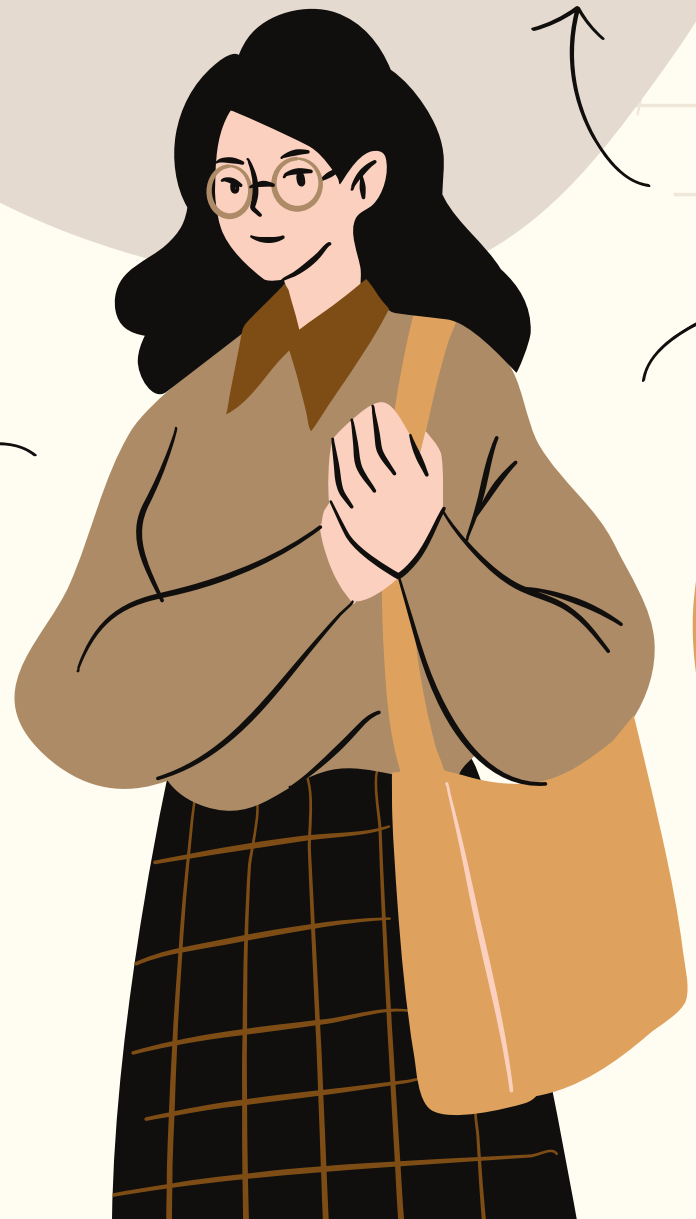
Tip 3: FOR Exploration Purpose, Limit Rows

Best Practice:

"FETCHING DATA"

```sql
-- Bad Practice
SELECT OrderID,Sales
FROM Sales.Orders

-- Good Practice
SELECT TOP 5  OrderID,Sales
FROM Sales.Orders
```



Bad Practice

Results | Messages

	OrderID	Sales
1	1	10
2	2	15
3	3	20
4	4	60
5	5	25
6	6	50
7	7	30
8	8	90
9	9	20
10	10	60

Good Practice

Results | Messages

	OrderID	Sales
1	1	10
2	2	15
3	3	20
4	4	60
5	5	25

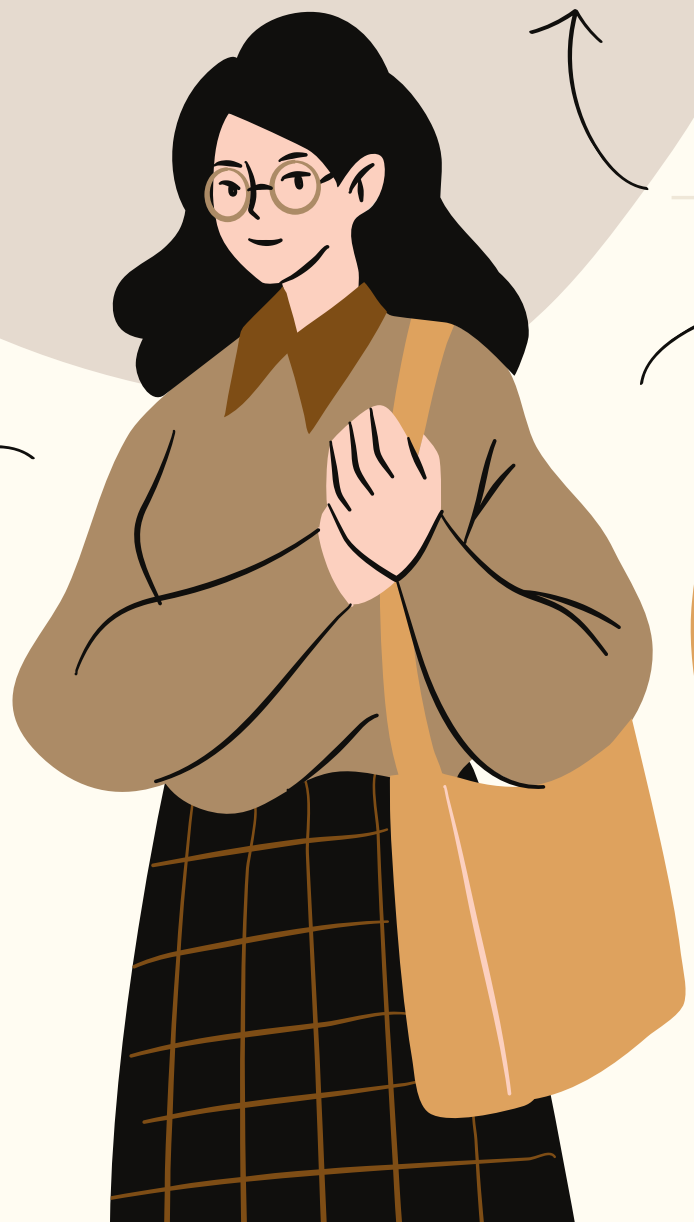
Tip 4: Create Nonclustered Index on Frequently used Columns in WHERE Clause

Best Practice:

"FILTERING DATA"

```sql
CREATE NONCLUSTERED INDEX Idx_Orders_OrderStatus
ON Sales.Orders(OrderStatus)

SELECT * FROM Sales.Orders WHERE
OrderStatus = 'Delivered'
```



Same Practice

Results | Messages

	OrderID	ProductID	CustomerID	SalesPersonID	Orde
1	1	101	2	3	2025
2	3	101	1	5	2025
3	5	104	2	5	2025
4	6	104	3	5	2025
5	7	102	1	1	2025

Good Practice

Results | Messages

	OrderID	ProductID	CustomerID	SalesPersonID	Orde
1	1	101	2	3	2025
2	3	101	1	5	2025
3	5	104	2	5	2025
4	6	104	3	5	2025
5	7	102	1	1	2025

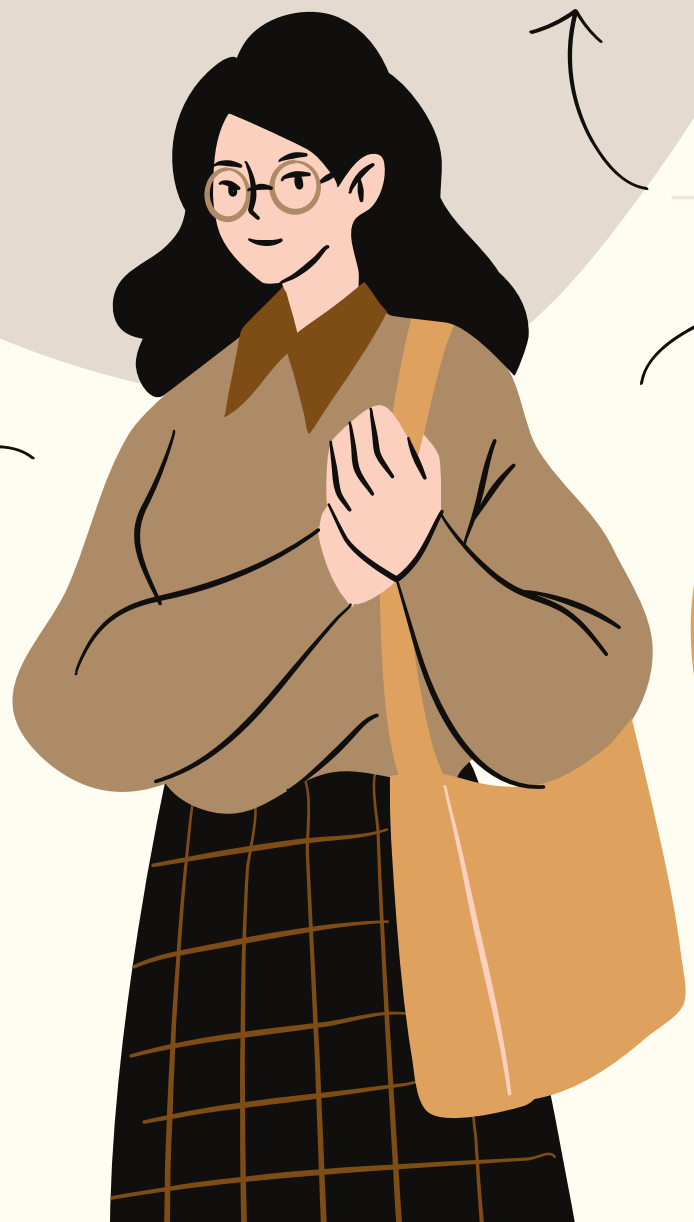
Tip 5: Avoid applying functions to columns in WHERE Clauses

Best Practice:

"FILTERING DATA"

```sql
-- Bad Practice
SELECT * FROM Sales.Orders
WHERE LOWER(OrderStatus) = 'delivered'

-- Good Practice
SELECT * FROM Sales.Orders
WHERE OrderStatus = 'Delivered'
```



Bad Practice

SalesPersonID	OrderDate	ShipDate	OrderStatus
3	2025-01-01	2025-01-05	Delivered
5	2025-01-10	2025-01-25	Delivered
5	2025-02-01	2025-02-05	Delivered
5	2025-02-05	2025-02-10	Delivered
1	2025-02-15	2025-02-27	Delivered

Good Practice

SalesPersonID	OrderDate	ShipDate	OrderStatus
3	2025-01-01	2025-01-05	Delivered
5	2025-01-10	2025-01-25	Delivered
5	2025-02-01	2025-02-05	Delivered
5	2025-02-05	2025-02-10	Delivered
1	2025-02-15	2025-02-27	Delivered

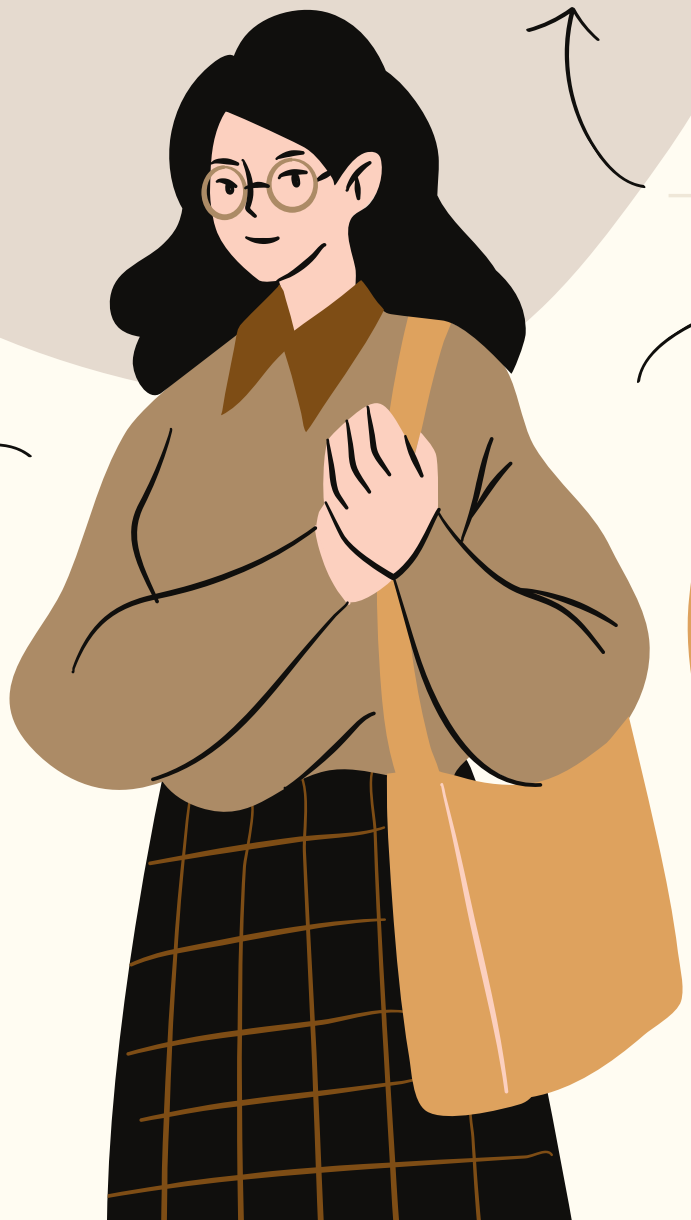
Tip 6: Avoid leading wildcards as they prevent index usage

Best Practice:

"FILTERING DATA"

```sql
-- Bad Practice
SELECT * FROM Sales.Customers
WHERE LastName LIKE '%Gold%'

-- Good Practice
SELECT * FROM Sales.Customers
WHERE LastName LIKE 'Gold%'
```



Bad Practice

Results | Messages

	CustomerID	FirstName	LastName	Country
1	1	Jossef	Goldberg	Germany

Good Practice

Results | Messages

CustomerID	FirstName	LastName	Country
1	Jossef	Goldberg	Germany

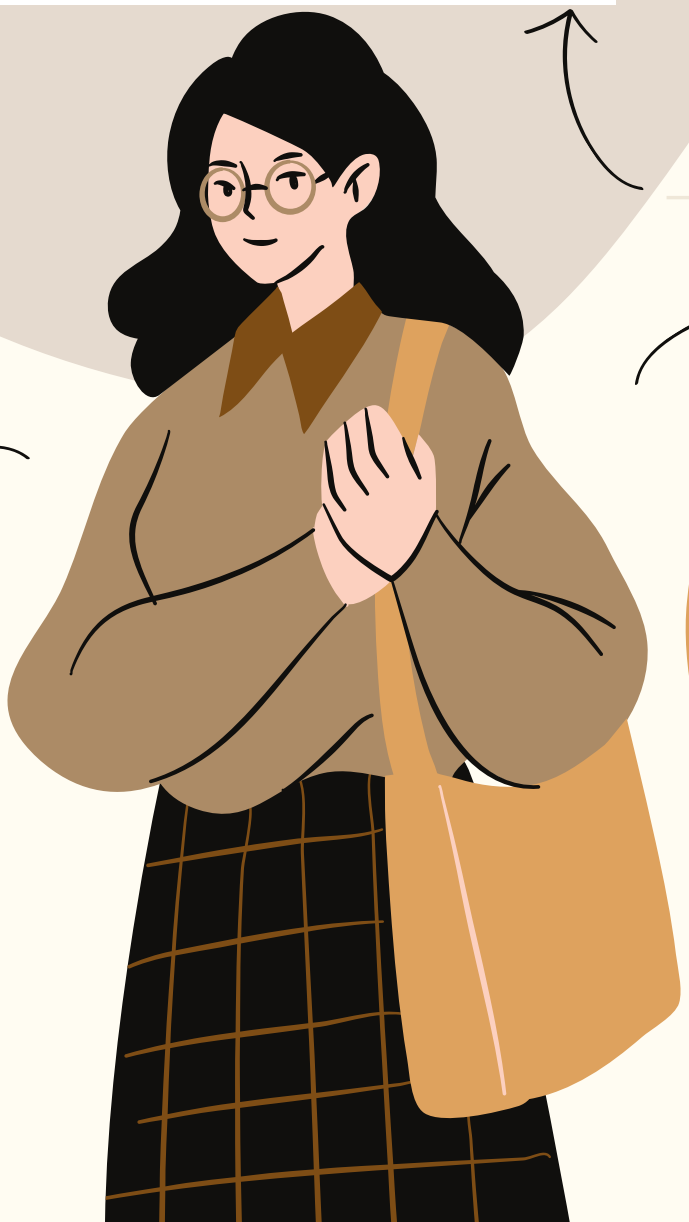
Tip 7: Use IN instead of Multiple OR

Best Practice:

"FILTERING DATA"

```sql
-- Bad Practice
SELECT * FROM Sales.Orders
WHERE CustomerID = 1 OR
CustomerID = 2 OR CustomerID = 3

-- Good Practice
SELECT * FROM Sales.Orders
WHERE CustomerID IN (1,2,3)
```



Bad Practice

Results | Messages

	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate
1	1	101	2	3	2025-01-01
2	2	102	3	3	2025-01-05
3	3	101	1	5	2025-01-10
4	4	105	1	3	2025-01-20
5	5	104	2	5	2025-02-01
6	6	104	3	5	2025-02-05
7	7	102	1	1	2025-02-15
8	9	101	2	3	2025-03-10
9	10	102	3	5	2025-03-15

Good Practice

Results | Messages

	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate
1	1	101	2	3	2025-01-01
2	2	102	3	3	2025-01-05
3	3	101	1	5	2025-01-10
4	4	105	1	3	2025-01-20
5	5	104	2	5	2025-02-01
6	6	104	3	5	2025-02-05
7	7	102	1	1	2025-02-15
8	9	101	2	3	2025-03-10
9	10	102	3	5	2025-03-15

Tip 8: Understand The Speed of Joins & Use INNER JOIN when possible

Best Practice:

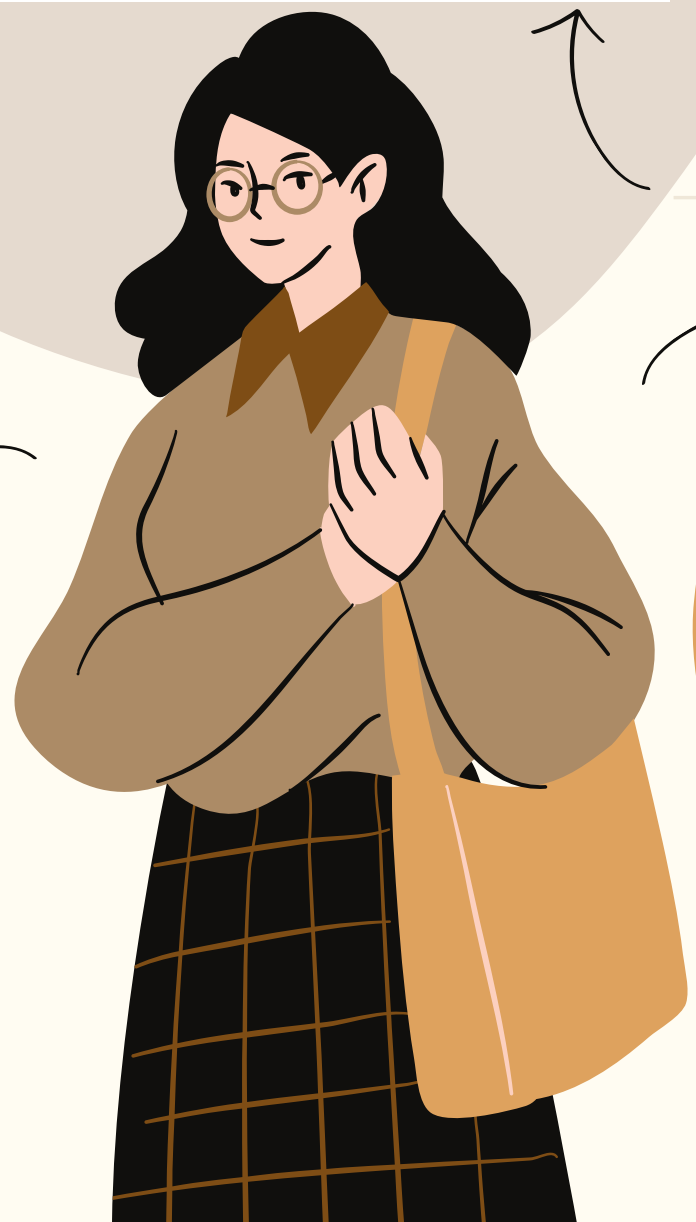
"FILTERING DATA"

```sql
-- Best Performance
SELECT c.FirstName, o.OrderId FROM Sales.Customers c
INNER JOIN Sales.Orders o ON c.CustomerID = o.CustomerID

-- Slightly Slower Performance
SELECT c.FirstName, o.OrderID FROM Sales.Customers c
RIGHT JOIN Sales.Orders o ON c.CustomerID = o.CustomerID

SELECT c.FirstName, o.OrderID FROM Sales.Customers c
LEFT JOIN Sales.Orders o ON c.CustomerID = o.CustomerID

-- Worst Performance
SELECT c.FirstName, o.OrderID FROM Sales.Customers c
OUTER JOIN Sales.Orders o ON c.CustomerID = o.CustomerID
```



Bad Practice

```
Msg 156, Level 15, State 1, Line 121
Incorrect syntax near the keyword 'JOIN'.

Completion time: 2025-06-12T16:25:25.2382
```

Good Practice

Results | Messages

	FirstName	LastName	OrderId
1	Jossef	Goldberg	3
2	Jossef	Goldberg	4
3	Jossef	Goldberg	7
4	Kevin	Brown	1
5	Kevin	Brown	5
6	Kevin	Brown	9
7	Mary	NULL	2
8	Mary	NULL	6
9	Mary	NULL	10
10	Mark	Schwarz	8

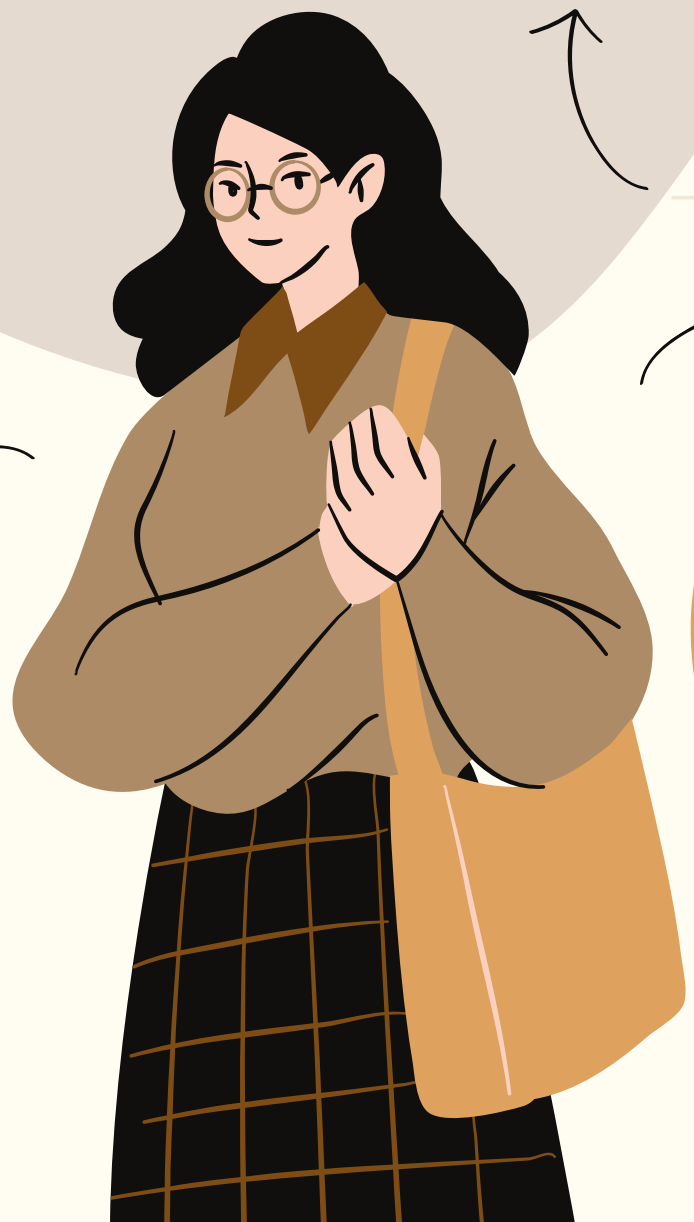
Tip 9: Use Explicit Join (ANSI Join) Instead of Implicit Join (Non-ANSI Join)

Best Practice:

"FILTERING DATA"

```sql
-- Bad Practice
SELECT o.OrderID, c.FirstName
FROM Sales.Customers c, Sales.Orders o
WHERE c.CustomerID = o.CustomerID

-- Good Practice
SELECT o.OrderID, c.FirstName
FROM Sales.Customers c
INNER JOIN Sales.Orders o
ON c.CustomerID = o.CustomerID
```



Bad Practice

Results | Messages

	OrderID	FirstName
1	3	Jossef
2	4	Jossef
3	7	Jossef
4	1	Kevin
5	5	Kevin
6	9	Kevin
7	2	Mary

Good Practice

Results | Messages

	OrderID	FirstName
1	3	Jossef
2	4	Jossef
3	7	Jossef
4	1	Kevin
5	5	Kevin
6	9	Kevin
7	2	Mary

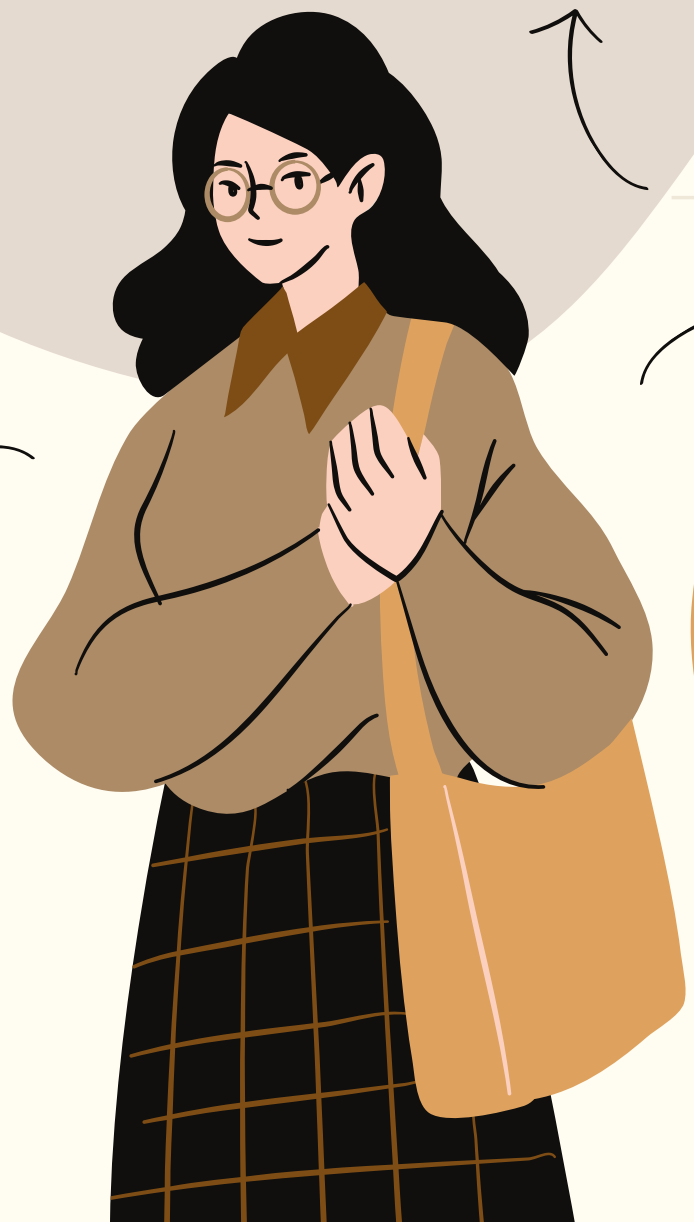
Tip 10: Make sure to Index the columns used in the ON clause

Best Practice:

"FILTERING DATA"

```sql
SELECT c.FirstName, o.OrderID
FROM Sales.Orders o
INNER JOIN Sales.Customers c
ON c.CustomerID = o.CustomerID

CREATE NONCLUSTERED INDEX IX_Orders_CustomerID
ON Sales.Orders(CustomerID)
```



Same Practice

Results | Messages

	FirstName	OrderID
1	Jossef	3
2	Jossef	4
3	Jossef	7
4	Kevin	1
5	Kevin	5
6	Kevin	9
7	Mary	2
8	Mary	6
9	Mary	10

Good Practice

Results | Messages

	FirstName	OrderID
1	Jossef	3
2	Jossef	4
3	Jossef	7
4	Kevin	1
5	Kevin	5
6	Kevin	9
7	Mary	2
8	Mary	6
9	Mary	10

🧡 Thank You for Exploring These SQL Tips!

These insights are based on real-world best practices to help you write smarter and faster queries.

Keep learning, stay curious, and build great data solutions! 🚀

🤝 Let's Connect & Grow Together

💼 Follow me for more dashboards, SQL tricks, and data analysis content.

📍 Scan the QR code to visit my GitHub or LinkedIn profile.

Github👇

📌 Created by: Sudhanshu Kumar

🎓 BBA (International Business) Aspiring Data Analyst