Database Using SQL Server & Raw Data available over my GitHub Profile

MySQL

PostgreSQL

SQL Temporary Tables With 12 Most Used Queries

AnshLibrary

What is a Temporary Tables

Temporary Tables in SQL Server are used to store intermediate results or small subsets of data temporarily during a session, procedure, or batch.

They're created with a prefix # or ##.

Syntax :

TEMPORARY TABLE

```
SELECT ...
INTO #New-Table
FROM ...
WHERE ...
```

Sql Server

PERMENANT CREATE TABLE

```
CREATE TABLE TABLE-NAME AS
(
  SELECT ...
  FROM ...
  WHERE ...
)
```

TEMPORARY TABLE

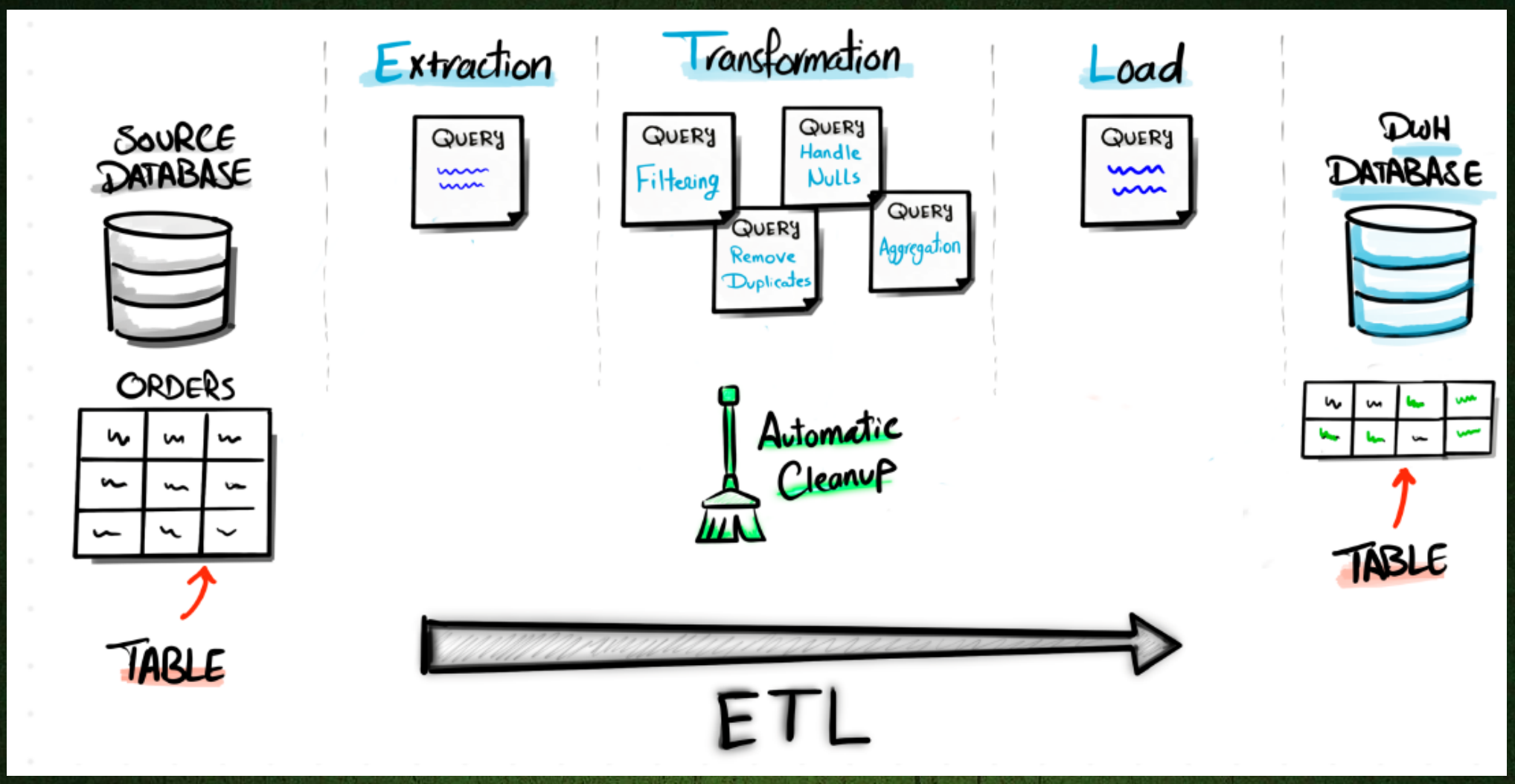
```
CREATE TEMPORARY TABLE TABLE-NAME AS
(
  SELECT ...
  FROM ...
  WHERE ...
)
```

MySQL | Postgres | Oracle

Type	Syntax	Scope	Lifetime
Local Temp Table	#TableName	Current session only	Until the session is closed
Global Temp Table	##TableName	All sessions (globally)	Until all sessions are closed

Why We Using :

- Load Data to TEMP Table
- Transform Data in TEMP Table
- Load TEMP Table into Permanent Table



Empty Table	View	Permenant Table	Temporary Table
`CREATE TABLE Table-Name ( ID INT, Name VARCHAR (50) )`	`CREATE VIEW View-Name AS ( SELECT … FROM … WHERE … )`	`SELECT … INTO New-Table FROM … WHERE …`	`SELECT … INTO #New-Table FROM … WHERE …`

01

Stage Filtered Orders for ETL

--Created Like This

```
SELECT *
INTO #StagedOrders
FROM orders
WHERE OrderDate >= DATEADD(DAY, -7, GETDATE());
```

--USED BY This Select Query

```
SELECT * FROM #StagedOrders
```

Results | Messages

	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate	ShipDate	OrderStatus	ShipAddress	BillAddress	Quantity	Sales	CreationTime
1	13	102	3	5	2025-07-03	2025-07-07	In Transit	1200 Sunset Way	4001 Clear Lake	6	66.00	2025-07-03 12:15:15.0000000
2	14	103	5	5	2025-07-04	2025-07-08	Packed	305 Mission Dr.	8321 Breeze Ave	7	77.00	2025-07-04 14:20:20.0000000
3	15	104	3	4	2025-07-05	2025-07-09	Preparing	991 Ocean Ct.	2271 River Bend	8	88.00	2025-07-05 15:30:25.0000000
4	16	105	4	4	2025-07-06	2025-07-10	Hold	3433 Palm Point	6902 Drift St.	9	99.00	2025-07-06 16:40:35.0000000
5	17	106	5	2	2025-07-07	2025-07-11	Dispatched	7282 Rain Tree Ln	3082 Sunshine Blvd	10	110.00	2025-07-07 17:50:45.0000000
6	18	105	4	1	2025-07-08	2025-07-12	Unconfirmed	1223 Ash Way	1811 Winter Oak Rd	11	121.00	2025-07-08 18:00:55.0000000
7	19	105	1	1	2025-07-09	2025-07-13	Awaiting Payment	5312 Laurel Ridge	4447 Warm Springs	12	132.00	2025-07-09 19:10:00.0000000
8	20	101	1	1	2025-07-10	2025-07-14	Refunded	6820 Moonlight Rd	8400 Crystal Ln	13	143.00	2025-07-10 20:20:20.0000000
9	21	106	4	2	2025-07-11	2025-07-15	Delivered	7449 Green Leaf Ct.	1222 Northgate Ave	2	90.00	2025-07-11 10:10:10.0000000
10	22	106	2	3	2025-07-12	2025-07-16	Shipped	3333 Violet Blvd	6777 Oak Hollow Dr	3	120.00	2025-07-12 11:20:30.0000000
11	23	104	3	4	2025-07-13	2025-07-17	Delivered	200 Lake Shore Dr	9012 Elm Heights	4	140.00	2025-07-13 12:30:40.0000000
12	24	107	5	4	2025-07-14	2025-07-18	Shipped	1111 Breeze Dr	3000 Hill Valley Rd	5	160.00	2025-07-14 13:40:50.0000000
13	25	107	6	2	2025-07-15	2025-07-19	Delivered	5000 West Palm Ln	1044 Sunny Dr	6	180.00	2025-07-15 14:51:55.0000000
14	26	102	4	2	2025-07-16	2025-07-20	Dispatched	8778 Lavender Loop	3339 Dusty Trail	7	200.00	2025-07-16 15:55:05.0000000
15	27	101	1	3	2025-07-17	2025-07-21	On Hold	6464 Crystal River	7779 Old Hollow Dr	8	220.00	2025-07-17 16:05:15.0000000
16	28	103	4	2	2025-07-18	2025-07-22	Packed	3922 Amber Crest	1100 Jade Hill Blvd	9	240.00	2025-07-18 17:15:25.0000000

Use by: Used in ETL pipelines for recent transactions..

AnshLibrary

02

Calculate Customer Order Count for Loyalty Tiering

```
SELECT
    CustomerID,
    COUNT(*) AS TotalOrders
INTO #CustomerOrderCount
FROM Orders
GROUP BY CustomerID;

--------------------------------------------------------------------------------
SELECT * FROM #CustomerOrderCount
WHERE TotalOrders > 5;
```

Results | Messages

	CustomerID	TotalOrders
1	1	6
2	2	6
3	3	7
4	4	6

Use by: Classify customers into loyalty tiers.

AnshLibrary

03

Temporary Table to Store Failed API Response Data

```
CREATE TABLE #FailedAPIOrders
(OrderID INT, ErrorMessage NVARCHAR(255));

-- later insert failed records in ETL flow
INSERT INTO #FailedAPIOrders VALUES (1012, 'Timeout occurred');

--------------------------------------------------------------------------------
SELECT * FROM  #FailedAPIOrders
```

Results | Messages

	OrderID	ErrorMessage
1	1012	Timeout occurred

Use by: Log temporary failures in automation pipelines.

AnshLibrary

04

Prepare Summary Before Final Reporting

```
SELECT

    ProductID,
    COUNT(OrderID) AS TotalOrders
INTO #EmpOrderSummary
FROM Orders
GROUP BY ProductID;

-------------------------------------------------------------------------------

SELECT p.ProductID, p.Product, TotalOrders
FROM Products p
JOIN #EmpOrderSummary s ON p.ProductID =
s.ProductID;
```

Results | Messages

	ProductID	Product	TotalOrders
1	101	Bottle	7
2	102	Tire	6
3	103	Socks	2
4	104	Caps	5
5	105	Gloves	5
6	106	Helmet	3
7	107	Water Bottle Cage	2

Use by: Simplifies final reporting logic.

AnshLibrary

05

Store Large Joins Temporarily for Reuse

```
SELECT

  o.OrderID,
  c.Customer_id,
  COALESCE(c.First_Name, '') + ' ' + COALESCE(c.Last_Name, '')
AS FullName,
  p.ProductID

INTO #JoinedData
FROM Orders o
JOIN Customers c ON o.CustomerID = c.Customer_id
JOIN Products p ON p.ProductID = o.ProductID;
-------------------------------------------------------------------------------------------
```

```
SELECT * FROM #JoinedData
WHERE Customer_id NOT IN(2,4,6,8) ;
```

Results | Messages

	OrderID	Customer_id	FullName	ProductID
1	2	3	Mary	102
2	3	1	Jossef Goldberg	101
3	4	1	Jossef Goldberg	105
4	6	3	Mary	104
5	7	1	Jossef Goldberg	102
6	10	3	Mary	102
7	13	3	Mary	102
8	14	5	Anna Adams	103
9	15	3	Mary	104
10	17	5	Anna Adams	106
11	19	1	Jossef Goldberg	105
12	20	1	Jossef Goldberg	101
13	23	3	Mary	104
14	24	5	Anna Adams	107
15	27	1	Jossef Goldberg	101
16	29	3	Mary	104
17	30	9	Carlos Ramirez	105

```
SELECT * FROM #JoinedData
WHERE Customer_id IN(2,4,6) ;
```

Results | Messages

	OrderID	Customer_id	FullName	ProductID
1	1	2	Kevin Brown	101
2	5	2	Kevin Brown	104
3	8	4	Mark Schwarz	101
4	9	2	Kevin Brown	101
5	11	2	Kevin Brown	102
6	12	2	Kevin Brown	101
7	16	4	Mark Schwarz	105
8	18	4	Mark Schwarz	105
9	21	4	Mark Schwarz	106
10	22	2	Kevin Brown	106
11	25	6	Emily Clark	107
12	26	4	Mark Schwarz	102
13	28	4	Mark Schwarz	103

Used For : Avoid recalculating large joins.

06

Track Duplicate Customers (Audit)

```sql
SELECT
  Email,
  COUNT(*) AS Cnt
INTO #DuplicateEmails
FROM Customers
GROUP BY Email
HAVING COUNT(*) > 1;

-------------------------------------------------------------------------------

select * from #DuplicateEmails
```

Results | Messages

	Email	Cnt
1	david.lee@anshulcorp.com	4
2	jossef.goldberg@anshulcorp.com	4
3	liam.obrien@anshulcorp.com	3
4	mary@anshulcorp.com	2

Use by: Pre-cleaning step before marketing export..

AnshLibrary

07

Session-Based Work in Stored Procedures

```
CREATE PROCEDURE sp_TempAnalysis
AS
BEGIN
    SELECT *
    INTO #RecentOrders
    FROM Orders
    WHERE OrderDate >= DATEADD(DAY, -30, GETDATE());

END
-------------------------------------------------------------------------------------------
SELECT COUNT(*)Total_Order_Count
FROM #RecentOrders;
```

Results | Messages

	Total_Order_Count
1	20

```
SELECT *  FROM #RecentOrdersWHERE OrderStatus
IN ( 'Failed', 'Returned','Hold');
```

Results | Messages

	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate	ShipDate	OrderStatus	ShipAddress	BillAddress	Quantity	Sales	CreationTime
1	11	102	2	2	2025-07-01	2025-07-05	Returned	9877 Redwood Dr.	1340 Birch Hollow	4	44.00	2025-07-01 09:01:01.0000000
2	12	101	2	2	2025-07-02	2025-07-06	Failed	2640 Oak Circle	3124 Pine Vista	5	55.00	2025-07-02 10:11:12.0000000
3	16	105	4	4	2025-07-06	2025-07-10	Hold	3433 Palm Point	6902 Drift St.	9	99.00	2025-07-06 16:40:35.0000000

Use by: Temp tables help modularize stored procedure logic.

08

Temp Table for Pagination Results

```
SELECT

ROW_NUMBER() OVER (ORDER BY OrderDate
DESC) AS RowNum, *
INTO #PagedOrders
FROM Orders;

-----------------------------------------------------------------------
SELECT * FROM #PagedOrders
WHERE RowNum BETWEEN 11 AND 20;
```

Results | Messages

	RowNum	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate	ShipDate	OrderStatus	ShipAddress	BillAddress	Quantity	Sales	CreationTime
1	11	20	101	1	1	2025-07-10	2025-07-14	Refunded	6820 Moonlight Rd	8400 Crystal Ln	13	143.00	2025-07-10 20:20:20.0000000
2	12	19	105	1	1	2025-07-09	2025-07-13	Awaiting Payment	5312 Laurel Ridge	4447 Warm Springs	12	132.00	2025-07-09 19:10:00.0000000
3	13	18	105	4	1	2025-07-08	2025-07-12	Unconfirmed	1223 Ash Way	1811 Winter Oak Rd	11	121.00	2025-07-08 18:00:55.0000000
4	14	17	106	5	2	2025-07-07	2025-07-11	Dispatched	7282 Rain Tree Ln	3082 Sunshine Blvd	10	110.00	2025-07-07 17:50:45.0000000
5	15	16	105	4	4	2025-07-06	2025-07-10	Hold	3433 Palm Point	6902 Drift St.	9	99.00	2025-07-06 16:40:35.0000000
6	16	15	104	3	4	2025-07-05	2025-07-09	Preparing	991 Ocean Ct.	2271 River Bend	8	88.00	2025-07-05 15:30:25.0000000
7	17	14	103	5	5	2025-07-04	2025-07-08	Packed	305 Mission Dr.	8321 Breeze Ave	7	77.00	2025-07-04 14:20:20.0000000
8	18	13	102	3	5	2025-07-03	2025-07-07	In Transit	1200 Sunset Way	4001 Clear Lake	6	66.00	2025-07-03 12:15:15.0000000
9	19	12	101	2	2	2025-07-02	2025-07-06	Failed	2640 Oak Circle	3124 Pine Vista	5	55.00	2025-07-02 10:11:12.0000000
10	20	11	102	2	2	2025-07-01	2025-07-05	Returned	9877 Redwood Dr.	1340 Birch Hollow	4	44.00	2025-07-01 09:01:01.0000000

Using: Manual pagination logic for APIs

AnshLibrary

09

Join Order & Archive to Detect Duplicates

```
SELECT
OrderID
INTO #AllOrders
FROM Orders

UNION ALL

SELECT
OrderID
FROM OrdersArchive;

-------------------------------------------------------------------------------------

SELECT TOP 10 OrderID
FROM #AllOrders
GROUP BY OrderID
HAVING COUNT(*) > 1;
```

Results | Messages

	OrderID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Use by: Archive audit and cleanup task.

10

Merge Customer Details for Upload

```
SELECT

c.Customer_id, c.First_Name + ' ' + c.Last_Name AS FullName,
o.OrderID

INTO #MergedUpload
FROM Customers c
JOIN Orders o ON c.Customer_id = o.CustomerID;

-- Export #MergedUpload to external system.
-------------------------------------------------------------------------------------------

SELECT Distinct FullName FROM #MergedUpload
ORDER BY FullName DESC
```

Results | Messages

	FullName
1	Mark Schwarz
2	Jossef Goldberg
3	Emily Clark
4	Carlos Ramirez
5	Anna Adams
6	Kevin Brown
7	NULL

Use by: Prepare CSV upload format temporarily..

AnshLibrary

11

Top Employee Order Volume Summary

```
SELECT

    SalesPersonID,
    COUNT(*) AS OrdersHandled

INTO #TopEmployees
FROM Orders
GROUP BY SalesPersonID
HAVING COUNT(*) > 2;

-------------------------------------------------------------------------------
```

Results | Messages

	SalesPersonID	OrdersHandled
1	1	4
2	2	7
3	3	7
4	4	5
5	5	7

Use Case: Used in internal leaderboard or reward system

12

Customer Order Gap Analysis

```
SELECT
    CustomerID,
    OrderDate,
    LAG(OrderDate) OVER(PARTITION BY CustomerID ORDER BY OrderDate) AS PrevOrderDate

INTO #CustomerOrderGaps
FROM Orders;

-------------------------------------------------------------------------------------
SELECT *, DATEDIFF(DAY, PrevOrderDate, OrderDate) AS GapDays
FROM #CustomerOrderGaps
WHERE PrevOrderDate IS NOT NULL;
```

Results | Messages

	CustomerID	OrderDate	PrevOrderDate	GapDays
1	1	2025-01-20	2025-01-10	10
2	1	2025-02-15	2025-01-20	26
3	1	2025-07-09	2025-02-15	144
4	1	2025-07-10	2025-07-09	1
5	1	2025-07-17	2025-07-10	7
6	2	2025-02-01	2025-01-01	31
7	2	2025-03-10	2025-02-01	37
8	2	2025-07-01	2025-03-10	113
9	2	2025-07-02	2025-07-01	1
10	2	2025-07-12	2025-07-02	10
11	3	2025-02-05	2025-01-05	31
12	3	2025-03-15	2025-02-05	38
13	3	2025-07-03	2025-03-15	110
14	3	2025-07-05	2025-07-03	2
15	3	2025-07-13	2025-07-05	8
16	3	2025-07-19	2025-07-13	6
17	4	2025-07-06	2025-03-18	138

Use Case: Tracks customer inactivity periods.