

# Customer Orders Data

## Cleaning Project

**Prepared By:** SOURABH ARYA

**Date:** 27 December 2025

**Tool Used:** MySQL

### Introduction

This project focuses on cleaning and standardizing a customer orders dataset. The goal was to ensure consistency across names, emails, mobile numbers, order details, ratings, and customer categories. A total of 15 steps were performed, and finally a cleaned view was created for analysis.

### ◆ STEP 0: Inspect Raw Data

```
SELECT *
```

```
FROM customer_orders
```

```
LIMIT 10;
```

```

136
137
138 •   SELECT *
139     FROM customer_orders
140     LIMIT 10;
141
142

```

customer_id	first_name	last_name	email	mobile_number	order_id	order_date	delivery_date	order_amt
1001	Anita	SHARMA	anita1@GMAIL.COM	919110053353	ORD-2022-0001	2022-07-18	2022-07-20	3662.377
1002	KIRAN	Gupta	kiran2@GMAIL.COM	0091-9749110053353	P21-0002	2021-01-14	2021-01-21	10669.92
1003	Vikas	PATEL	vikas3@yahoo.com	0091-9664130526	ORD-2024-0003	2024-02-05	2024-02-08	23175.87
1004	POOJA	SINGH	pooja4@yahoo.com	919654049436	ORD-2022-0004	2022-07-01	2022-07-07	38255.81

customer orders 34 x

Action Output

#	Time	Action	Message	Duration / Fetch
40	14:26:03	SELECT signup_date, DATEDIFF(NOW(),signup_date) AS days_with_compan...	350 row(s) returned	0.000 sec / 0.000 sec
41	14:28:01	SELECT order_amount, CASE WHEN order_amount >= 50000 THEN Hig...	350 row(s) returned	0.000 sec / 0.000 sec
42	14:30:10	SELECT signup_date, CASE WHEN DATEDIFF(NOW(),signup_date) <= ...	350 row(s) returned	0.000 sec / 0.000 sec
43	14:33:05	CREATE OR REPLACE VIEW customer_orders_cleaned AS SELECT customer_...	0 row(s) affected	0.031 sec
44	14:36:02	SELECT * FROM customer_orders_cleaned LIMIT 10	10 row(s) returned	0.031 sec / 0.000 sec
45	14:38:43	SELECT * FROM customer_orders LIMIT 10	10 row(s) returned	0.000 sec / 0.000 sec

```

136
137
138 •   SELECT *
139     FROM customer_orders
140     LIMIT 10;
141
142

```

id	mobile_number	order_id	order_date	delivery_date	order_amount	city	signup_date	rating
a1@GMAIL.COM	919110053353	ORD-2022-0001	2022-07-18	2022-07-20	3662.377	PUNE	2021-11-22	1.882
n2@GMAIL.COM	0091-9749110053353	P21-0002	2021-01-14	2021-01-21	10669.923	delhi	2018-10-02	4.892
is3@yahoo.com	0091-9664130526	ORD-2024-0003	2024-02-05	2024-02-08	23175.878	bangalore	2021-09-29	1.651
ja4@yahoo.com	919654049436	ORD-2022-0004	2022-07-01	2022-07-07	38255.816	MUMBAI	2020-07-22	4.287

customer orders 34 x

Action Output

#	Time	Action	Message	Duration / Fetch
40	14:26:03	SELECT signup_date, DATEDIFF(NOW(),signup_date) AS days_with_compan...	350 row(s) returned	0.000 sec / 0.000 sec
41	14:28:01	SELECT order_amount, CASE WHEN order_amount >= 50000 THEN Hig...	350 row(s) returned	0.000 sec / 0.000 sec
42	14:30:10	SELECT signup_date, CASE WHEN DATEDIFF(NOW(),signup_date) <= ...	350 row(s) returned	0.000 sec / 0.000 sec
43	14:33:05	CREATE OR REPLACE VIEW customer_orders_cleaned AS SELECT customer_...	0 row(s) affected	0.031 sec
44	14:36:02	SELECT * FROM customer_orders_cleaned LIMIT 10	10 row(s) returned	0.031 sec / 0.000 sec
45	14:38:43	SELECT * FROM customer_orders LIMIT 10	10 row(s) returned	0.000 sec / 0.000 sec

**Explanation: This step was performed to quickly view the raw dataset before applying any cleaning. By selecting the first 10 rows, we can understand the structure of the table, the columns available (like first\_name, last\_name, email, mobile\_number, order\_date, etc.), and identify issues such as inconsistent formatting, extra spaces, mixed cases, or irregular values. This inspection helps plan the cleaning steps systematically.**

## ◆ STEP 1: Clean `first_name` (Spaces + Case)

```
SELECT first_name, TRIM(first_name) AS trimmed,  
       UPPER(TRIM(first_name)) AS cleaned_first_name  
  FROM customer_orders;
```

The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the database structure with the schema `customer_orders_csv` selected.
- SQL Editor:** Contains the following T-SQL code:

```
7  
8 •  SELECT first_name,  
9      TRIM(first_name) AS trimmed,  
10     UPPER(TRIM(first_name)) AS cleaned_first_name  
11    FROM customer_orders;  
12  
13
```
- Result Grid:** Displays the results of the query:

first_name	trimmed	cleaned_first_name
Anita	Anita	ANITA
KIRAN	KIRAN	KIRAN
Vikas	Vikas	VIKAS
POOJA	POOJA	POOJA
POOJA	POOJA	POOJA
- Output:** Shows the execution log with the following entries:

#	Time	Action	Message
22	13:26:33	SELECT signup_date, DATEDIFF(NOW(), signup_date) AS days_with_compan...	350 row(s) returned
23	13:27:14	SELECT order_amount, CASE WHEN order_amount >= 50000 THEN 'Hig...	350 row(s) returned
24	13:27:43	SELECT signup_date, CASE WHEN DATEDIFF(NOW(), signup_date) <= ...	350 row(s) returned
25	13:28:24	CREATE OR REPLACE VIEW customer_orders_cleaned AS SELECT customer_j...	0 row(s) affected
26	13:29:20	SELECT * FROM customer_orders_cleaned LIMIT 10	10 row(s) returned
27	13:44:45	SELECT first_name, TRIM(first_name) AS trimmed, UPPER(TRIM(first_nam...	350 row(s) returned

Explanation: Converted first names to uppercase and removed extra spaces.

## ◆ STEP 2: Clean last\_name

```
SELECT last_name,
```

```
    UPPER(TRIM(last_name)) AS cleaned_last_name
```

```
FROM customer_orders;
```

The screenshot shows the SSMS interface with the following details:

- Schemas:** The current schema is set to `customer_orders_csv`.
- Query Editor:** The query is:13  
14  
15  
16 •     SELECT last\_name,  
17        UPPER(TRIM(last\_name)) AS cleaned\_last\_name  
18    FROM customer\_orders;  
19
- Result Grid:** The results show the conversion of last names to uppercase and removing spaces:

	last_name	cleaned_last_name
▶	SHARMA	SHARMA
	Gupta	GUPTA
	PATEL	PATEL
	SINGH	SINGH
	VERMA	VERMA
- Output Grid:** The history of actions and their duration:

#	Time	Action	Message	Duration / Fetch
23	13:27:14	SELECT order_amount, CASE WHEN order_amount >= 50000 THEN 'High' ELSE 'Low' END AS category FROM customer_orders	350 row(s) returned	0.000 sec / 0.000 sec
24	13:27:43	SELECT signup_date, CASE WHEN DATEDIFF(NOW(),signup_date) <= 30 THEN 'New' ELSE 'Old' END AS status FROM customer_orders	350 row(s) returned	0.000 sec / 0.000 sec
25	13:28:24	CREATE OR REPLACE VIEW customer_orders_cleaned AS SELECT * FROM customer_orders	0 row(s) affected	0.031 sec
26	13:29:20	SELECT * FROM customer_orders_cleaned LIMIT 10	10 row(s) returned	0.015 sec / 0.000 sec
27	13:44:45	SELECT first_name, TRIM(first_name) AS trimmed, UPPER(TRIM(first_name)) AS cleaned_first_name FROM customer_orders	350 row(s) returned	0.000 sec / 0.000 sec
28	13:51:06	SELECT last_name, UPPER(TRIM(last_name)) AS cleaned_last_name FROM customer_orders	350 row(s) returned	0.000 sec / 0.000 sec

Explanation: Converted last names to uppercase and removed extra spaces.

## ◆ STEP 3: Create full\_name (CONCAT)

SELECT CONCAT(

    UPPER(TRIM(first\_name)),



    UPPER(TRIM(last\_name))

) AS full\_name

FROM customer\_orders;

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the schema 'customer\_orders\_csv' selected. The right pane shows the results of the following query:

```
19
20 • SELECT CONCAT(UPPER(TRIM(first_name)), ' ', UPPER(TRIM(last_name))) AS full_name
21 FROM customer_orders
22
23
24
25
```

The results grid shows the following data:

full_name
ANITA SHARMA
KIRAN GUPTA
VIKAS PATEL
POOJA SINGH
POOJA VERMA

The status bar at the bottom indicates the duration of the query execution.

Explanation: Combined first and last names into a standardized full name.

## ◆ STEP 4: Clean email (Standardization)

**SELECT email,**

**LOWER(email) AS cleaned\_email**

**FROM customer\_orders;**

The screenshot shows the SSMS interface with the following details:

- Schemas:** The current schema is set to `customer_orders_csv`.
- Query Editor:** The query is:25  
26 • SELECT email,  
27 LOWER(email) AS cleaned\_email  
28 FROM customer\_orders;  
29  
30  
31
- Result Grid:** The results show the conversion of five email addresses to lowercase:email cleaned\_email  
arita1@GMAIL.COM arita1@gmail.com  
kiran2@GMAIL.COM kiran2@gmail.com  
vikas3@yahoo.com vikas3@yahoo.com  
pooja4@yahoo.com pooja4@yahoo.com  
pooja5@yahoo.com pooja5@yahoo.com
- Action Output:** The history of actions shows the execution of the query and the subsequent concatenation of names:# Time Action Message Duration / Fetch  
27 13:44:45 SELECT first\_name, TRIM(first\_name) AS trimmed, UPPER(TRIM(first\_name)) AS cleaned\_first\_name FROM customer\_orders LIMIT 1 350 row(s) returned 0.000 sec / 0.000 sec  
28 13:51:06 SELECT last\_name, UPPER(TRIM(last\_name)) AS cleaned\_last\_name FROM customer\_orders LIMIT 1 350 row(s) returned 0.000 sec / 0.000 sec  
29 14:00:23 SELECT CONCAT(UPPER(TRIM(first\_name)), '.', UPPER(TRIM(last\_name))) AS full\_name FROM customer\_orders LIMIT 1 350 row(s) returned 0.016 sec / 0.000 sec  
30 14:07:26 SELECT email, LOWER(email) AS cleaned\_email FROM customer\_orders LIMIT 1 350 row(s) returned 0.000 sec / 0.000 sec  
31 14:07:38 SELECT email, LOWER(email) AS cleaned\_email FROM customer\_orders LIMIT 1 350 row(s) returned 0.000 sec / 0.000 sec  
32 14:13:34 SELECT email, LOWER(email) AS cleaned\_email FROM customer\_orders LIMIT 1 350 row(s) returned 0.000 sec / 0.000 sec

Explanation: Converted all emails to lowercase.

## ◆ STEP 5: Clean mobile\_number (Extract last 10 digits)

```
SELECT mobile_number,
```

```
    SUBSTR(mobile_number, LENGTH(mobile_number) - 9, 10)  
AS cleaned_mobile
```

```
FROM customer_orders;
```

The screenshot shows the SSMS interface with the following details:

- Schemas:** customer\_orders\_csv is selected.
- Tables:** customer\_orders is selected.
- Query Editor:** The following T-SQL code is executed:

```
31  
32 •  SELECT mobile_number,  
33     SUBSTR(mobile_number, LENGTH(mobile_number) - 9, 10) AS cleaned_mobile  
34  FROM customer_orders;
```
- Result Grid:** The output shows the original mobile number and the extracted cleaned mobile number.

mobile_number	cleaned_mobile
919110053353	9110053353
0091-9749621470	9749621470
0091-9664130526	9664130526
919654049436	9654049436
919940992571	9940992571
- Action Output:** The log shows the execution steps and their durations.

#	Time	Action	Message	Duration / Fetch
28	13:51:06	SELECT last_name, UPPER(TRIM(last_name)) AS cleaned_last_name FROM c...	350 row(s) returned	0.000 sec / 0.000 sec
29	14:00:23	SELECT CONCAT(UPPER(TRIM(first_name)), '.', UPPER(TRIM(last_name))) AS full...	350 row(s) returned	0.016 sec / 0.000 sec
30	14:07:26	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
31	14:07:38	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
32	14:13:34	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
33	14:16:22	SELECT mobile_number, SUBSTR(mobile_number, LENGTH(mobile_number) - ...	350 row(s) returned	0.000 sec / 0.000 sec

Explanation: Extracted last 10 digits of mobile numbers.

## ◆ STEP 6: Extract Year from order\_id

```
SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year  
FROM customer_orders;
```

The screenshot shows the SSMS interface with the following details:

- Schemas:** The current schema is set to `customer_orders_csv`.
- SQL Editor:** The query window contains the following T-SQL code:

```
37  
38 •  SELECT order_id,  
39      SUBSTR(order_id, 5, 4) AS order_year  
40  FROM customer_orders;  
41  
42  
43
```
- Result Grid:** The results of the query are displayed in a grid:

order_id	order_year
ORD-2022-0001	2022
ORD-2021-0002	2021
ORD-2024-0003	2024
ORD-2022-0004	2022
ORD-2022-0005	2022
- Action Output:** A table showing the history of actions taken during the session:

#	Time	Action	Message	Duration / Fetch
30	14:07:26	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
31	14:07:38	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
32	14:13:34	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
33	14:16:22	SELECT mobile_number, SUBSTR(mobile_number, LENGTH(mobile_number) - ...	350 row(s) returned	0.000 sec / 0.000 sec
34	14:18:01	SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
35	14:18:01	SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec

Explanation: Extracted year from order ID.

## ◆ STEP 7: Round order\_amount

**SELECT order\_amount,**

**ROUND(order\_amount, 2) AS cleaned\_order\_amount**

**FROM customer\_orders;**

The screenshot shows the SSMS interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard toolbar icons.
- Navigator:** Shows the database schema. Under the 'customer\_orders\_csv' schema, there is a 'Tables' node expanded, showing 'customer\_orders' with its columns, indexes, foreign keys, and triggers.
- Query Editor:** Displays the following T-SQL code:

```
43  
44 •  SELECT order_amount,  
45      ROUND(order_amount, 2) AS cleaned_order_amount  
46  FROM customer_orders;  
47  
48  
49
```
- Result Grid:** Shows the output of the query:

order_amount	cleaned_order_amount
3662.377	3662.38
10669.923	10669.92
23175.878	23175.88
38255.816	38255.82
91497.485	91497.48
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
31	14:07:38	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
32	14:13:34	SELECT email, LOWER(email) AS cleaned_email FROM customer_orders LIMIT...	350 row(s) returned	0.000 sec / 0.000 sec
33	14:16:22	SELECT mobile_number, SUBSTR(mobile_number, LENGTH(mobile_number) - ...)	350 row(s) returned	0.000 sec / 0.000 sec
34	14:18:01	SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
35	14:18:01	SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
36	14:19:38	SELECT order_amount, ROUND(order_amount, 2) AS cleaned_order_amount F...	350 row(s) returned	0.000 sec / 0.000 sec

Explanation: Rounded order amounts to 2 decimal places.

## ◆ STEP 8: Round rating

**SELECT rating,**

**ROUND(rating, 1) AS cleaned\_rating**

**FROM customer\_orders;**

The screenshot shows the SSMS interface with the following details:

- Schemas:** customer\_orders\_csv
- Tables:** customer\_orders
- Query:** SELECT rating, ROUND(rating, 1) AS cleaned\_rating FROM customer\_orders;
- Result Grid:** Shows the output of the query with the following data:

rating	cleaned_rating
1.882	1.9
4.892	4.9
1.651	1.7
4.287	4.3
2.598	2.6

- Action Output:** Shows the history of actions taken during the session, including the execution of the current query.

Explanation: Rounded ratings to 1 decimal place.

## ◆ STEP 9: Standardize city

**SELECT city,**

**UPPER(city) AS cleaned\_city**

**FROM customer\_orders;**

The screenshot shows the SSMS interface with the following details:

- Schemas:** The current schema is set to `customer_orders_csv`.
- Query Editor:** The query window contains the following T-SQL code:

```
55
56 •  SELECT city,
57      UPPER(city) AS cleaned_city
58  FROM customer_orders;
```
- Result Grid:** The results show the conversion of city names to uppercase:

city	cleaned_city
PUNE	PUNE
delhi	DELHI
bangalore	BANGALORE
MUMBAI	MUMBAI
hyderabad	HYDERABAD
- Action Output:** The log shows the following actions:

#	Time	Action	Message	Duration / Fetch
33	14:16:22	SELECT mobile_number, SUBSTR(mobile_number, LENGTH(mobile_number) - ...	350 row(s) returned	0.000 sec / 0.000 sec
34	14:18:01	SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
35	14:18:01	SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
36	14:19:38	SELECT order_amount, ROUND(order_amount, 2) AS cleaned_order_amount F...	350 row(s) returned	0.000 sec / 0.000 sec
37	14:21:03	SELECT rating, ROUND(rating, 1) AS cleaned_rating FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
38	14:22:53	SELECT city, UPPER(city) AS cleaned_city FROM customer_orders LIMIT 0, 50...	350 row(s) returned	0.000 sec / 0.000 sec

Explanation: Converted city names to uppercase.

## ◆ STEP 10: Delivery Time Calculation (DATEDIFF)

```
SELECT order_date, delivery_date,  
       DATEDIFF(delivery_date, order_date) AS delivery_days  
FROM customer_orders;
```

The screenshot shows the SSMS interface with the following details:

- Schemas:** customer\_orders\_csv
- Tables:** customer\_orders
- Query:** SELECT order\_date, delivery\_date, DATEDIFF(delivery\_date, order\_date) AS delivery\_days FROM customer\_orders;
- Result Grid:** Displays the following data:

order_date	delivery_date	delivery_days
2022-07-18	2022-07-20	2
2021-01-14	2021-01-21	7
2024-02-05	2024-02-08	3
2022-07-01	2022-07-07	6
2022-07-03	2022-07-04	1

- Action Output:** Shows the history of actions taken during the session.

Explanation: Calculated delivery days between order and delivery dates.

## ◆ STEP 11: Customer Tenure Calculation

**SELECT signup\_date,**

**DATEDIFF(NOW(), signup\_date) AS days\_with\_company**

**FROM customer\_orders;**

The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the database schema with the **customer\_orders\_csv** schema expanded, revealing tables like **customer\_orders**.
- SQL Editor:** Displays the T-SQL query:

```
67
68 •  SELECT signup_date,
69      DATEDIFF(NOW(), signup_date) AS days_with_company
70  FROM customer_orders
71
72
73
```
- Result Grid:** Shows the output of the query:

signup_date	days_with_company
2021-11-22	1496
2018-10-02	2643
2021-09-29	1550
2020-07-22	1984
2021-11-19	1499
- Action Output:** Shows the execution history with the following rows:

#	Time	Action	Message	Duration / Fetch
35	14:18:01	SELECT order_id, SUBSTR(order_id, 5, 4) AS order_year FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
36	14:19:38	SELECT order_amount, ROUND(order_amount, 2) AS cleaned_order_amount F...	350 row(s) returned	0.000 sec / 0.000 sec
37	14:21:03	SELECT rating, ROUND(rating, 1) AS cleaned_rating FROM customer_order...	350 row(s) returned	0.000 sec / 0.000 sec
38	14:22:53	SELECT city, UPPER(city) AS cleaned_city FROM customer_orders LIMIT 0, 50...	350 row(s) returned	0.000 sec / 0.000 sec
39	14:24:31	SELECT order_date, delivery_date, DATEDIFF(delivery_date, order_date) AS...	350 row(s) returned	0.000 sec / 0.000 sec
40	14:26:03	SELECT signup_date, DATEDIFF(NOW(), signup_date) AS days_with_compan...	350 row(s) returned	0.000 sec / 0.000 sec

**Explanation: Calculated how many days each customer has been with the company.**

## ◆ STEP 12: CASE WHEN – Order Value Category

```
SELECT order_amount,
```

```
CASE
```

```
WHEN order_amount >= 50000 THEN 'High Value'
```

```
WHEN order_amount >= 20000 THEN 'Medium Value'
```

```
ELSE 'Low Value'
```

```
END AS order_category
```

```
FROM customer_orders;
```

```
File Edit View Query Database Server Tools Scripting Help  
Navigator  
SCHEMAS  
Filter objects  
acciojob_da  
batch6  
college  
customer_orders_csv  
Tables  
customer_orders  
Columns  
Indexes  
Foreign Keys  
Triggers  
Views  
Stored Procedures  
Functions  
empl  
emplo  
Administration Schemas  
Information  
Schema:  
customer_orders_csv  
Object Info Session  
SQL File 10* SQL File 11* SQL File 12* SQL File 14* SQL File 15* employees_salary SQL File 17* SQLAdditions  
77  
78 WHEN order_amount >= 20000 THEN 'Medium Value'  
79  
80 ELSE 'Low Value'  
81  
82  
83  
FROM customer_orders;  
Result Grid Filter Rows Export Wrap Cell Content  
order_amount order_category  
3662.377 Low Value  
10669.923 Low Value  
23175.878 Medium Value  
38255.816 Medium Value  
91497.485 High Value  
Result Grid Read Only Context Help Snippets  
Action Output  
# Time Action Message Duration / Fetch  
36 14:19:38 SELECT order_amount, ROUND(order_amount, 2) AS cleaned_order_amount F... 350 row(s) returned 0.000 sec / 0.000 sec  
37 14:21:03 SELECT rating, ROUND(rating, 1) AS cleaned_rating FROM customer_orders U... 350 row(s) returned 0.000 sec / 0.000 sec  
38 14:22:53 SELECT city, UPPER(city) AS cleaned_city FROM customer_orders LIMIT 0, 50... 350 row(s) returned 0.000 sec / 0.000 sec  
39 14:24:31 SELECT order_date, delivery_date, DATEDIFF(delivery_date, order_date) AS d... 350 row(s) returned 0.000 sec / 0.000 sec  
40 14:26:03 SELECT signup_date, DATEDIFF(NOW(), signup_date) AS days_with_compan... 350 row(s) returned 0.000 sec / 0.000 sec  
41 14:28:01 SELECT order_amount, CASE WHEN order_amount >= 50000 THEN 'High V... 350 row(s) returned 0.000 sec / 0.000 sec
```

Explanation: Categorized orders into High, Medium, or Low value.

## ◆ STEP 13: CASE WHEN – Customer Type

SELECT signup\_date,

CASE

WHEN DATEDIFF(NOW(), signup\_date) <= 30 THEN 'New'

WHEN DATEDIFF(NOW(), signup\_date) <= 180 THEN 'Regular'

ELSE 'Loyal'

END AS customer\_type

FROM customer\_orders;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following SQL code:

```
84 •    SELECT signup_date,
85   CASE
86     WHEN DATEDIFF(NOW(), signup_date) <= 30 THEN 'New'
87     WHEN DATEDIFF(NOW(), signup_date) <= 180 THEN 'Regular'
88     ELSE 'Loyal'
89   END AS customer_type
90
91 FROM customer_orders;
```

The results grid shows the output of the query:

signup_date	customer_type
2021-11-22	Loyal
2018-10-02	Loyal
2021-09-29	Loyal
2020-07-22	Loyal
2021-11-19	Loyal

The status bar at the bottom right indicates "Result Grid" and "Read Only". The context help pane on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Explanation: Classified customers as New, Regular, or Loyal.

## ◆ STEP 14: FINAL CLEANED VIEW (Industry Practice)

```
CREATE OR REPLACE VIEW customer_orders_cleaned AS  
  
SELECT  
  
    customer_id,  
  
    UPPER(TRIM(first_name)) AS first_name,  
  
    UPPER(TRIM(last_name)) AS last_name,  
  
    CONCAT(UPPER(TRIM(first_name)), ' ', UPPER(TRIM(last_name))) AS full_name,  
  
    LOWER(email) AS email,  
  
    SUBSTR(mobile_number, LENGTH(mobile_number) - 9, 10) AS mobile_number,  
  
    order_id,  
  
    SUBSTR(order_id, 5, 4) AS order_year,  
  
    order_date,  
  
    delivery_date,  
  
    DATEDIFF(delivery_date, order_date) AS delivery_days,  
  
    ROUND(order_amount, 2) AS order_amount,  
  
    UPPER(city) AS city,  
  
    signup_date,  
  
    DATEDIFF(NOW(), signup_date) AS customer_tenure_days,
```

```
CASE
```

```
    WHEN order_amount >= 50000 THEN 'High Value'
```

```
    WHEN order_amount >= 20000 THEN 'Medium Value'
```

```
    ELSE 'Low Value'
```

```
END AS order_category,
```

```
ROUND(rating, 1) AS rating,
```

```
CASE
```

```
    WHEN DATEDIFF(NOW(), signup_date) <= 30 THEN 'New'
```

```
    WHEN DATEDIFF(NOW(), signup_date) <= 180 THEN 'Regular'
```

```
    ELSE 'Loyal'
```

```
END AS customer_type
```

```
FROM customer_orders;
```

```
CREATE OR REPLACE VIEW customer_orders_cleaned AS
SELECT
    customer_id,
    UPPER(TRIM(first_name)) AS first_name,
    UPPER(TRIM(last_name)) AS last_name,
    CONCAT(UPPER(TRIM(first_name)), ' ', UPPER(TRIM(last_name))) AS full_name,
    LOWER(email) AS email,
    SUBSTR(mobile_number, LENGTH(mobile_number) - 9, 10) AS mobile_number,
    order_id,
    SUBSTR(order_id, 5, 4) AS order_year,
```

#	Time	Action	Message	Duration / Fetch
38	14:22:53	SELECT city, UPPER(city) AS cleaned_city FROM customer_orders LIMIT 0, 50...	350 row(s) returned	0.000 sec / 0.000 sec
39	14:24:31	SELECT order_date, delivery_date, DATEDIFF(delivery_date, order_date) AS d...	350 row(s) returned	0.000 sec / 0.000 sec
40	14:26:03	SELECT signup_date, DATEDIFF(NOW(), signup_date) AS days_with_compan...	350 row(s) returned	0.000 sec / 0.000 sec
41	14:28:01	SELECT order_amount, CASE WHEN order_amount >= 50000 THEN 'High ...	350 row(s) returned	0.000 sec / 0.000 sec
42	14:30:10	SELECT signup_date, CASE WHEN DATEDIFF(NOW(), signup_date) <= 30 THEN 'New' ...	350 row(s) returned	0.000 sec / 0.000 sec
43	14:33:05	CREATE OR REPLACE VIEW customer_orders_cleaned AS SELECT customer_id, ...	0 row(s) affected	0.031 sec

Explanation: Created a view combining all transformations.

## ◆ STEP 15: Validate Cleaned Data

```
SELECT * FROM customer_orders_cleaned LIMIT 10;
```

The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the database schema, including the `customer_orders_csv` schema which contains the `customer_orders` table.
- SQL Editor:** Contains the following T-SQL code:
 

```
123     END AS customer_type
124     FROM customer_orders;
125
126
127
128
129 •   SELECT * FROM customer_orders_cleaned LIMIT 10;
```
- Result Grid:** Displays the results of the query, showing 10 rows of customer data:
 

customer_id	first_name	last_name	full_name	email	mobile_number	order_id	order_year	order_date
1001	ANITA	SHARMA	ANITA SHARMA	anita1@gmail.com	9110053353	ORD-2022-0001	2022	2022-07-18
1002	KIRAN	GUPTA	KIRAN GUPTA	kiran2@gmail.com	9749621470	ORD-2021-0002	2021	2021-01-14
1003	VIKAS	PATEL	VIKAS PATEL	vikas3@yahoo.com	9664130526	ORD-2024-0003	2024	2024-02-05
1004	POOJA	SINGH	POOJA SINGH	pooja4@yahoo.com	9654049436	ORD-2022-0004	2022	2022-07-01
- Output Window:** Shows the execution log with the following entries:
 

#	Time	Action	Message	Duration / Fetch
39	14:24:31	SELECT order_date, delivery_date, DATEDIFF(delivery_date, order_date) AS d...	350 row(s) returned	0.000 sec / 0.000 sec
40	14:26:03	SELECT signup_date, DATEDIFF(NOW(), signup_date) AS days_with_compan...	350 row(s) returned	0.000 sec / 0.000 sec
41	14:28:01	SELECT order_amount, CASE WHEN order_amount >= 50000 THEN High...	350 row(s) returned	0.000 sec / 0.000 sec
42	14:30:10	SELECT signup_date, CASE WHEN DATEDIFF(NOW(), signup_date) <= ...	350 row(s) returned	0.000 sec / 0.000 sec
43	14:33:05	CREATE OR REPLACE VIEW customer_orders_cleaned AS SELECT customer_...	0 row(s) affected	0.031 sec
44	14:36:02	SELECT * FROM customer_orders_cleaned LIMIT 10	10 row(s) returned	0.031 sec / 0.000 sec

The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the database schema, including the `customer_orders_csv` schema which contains the `customer_orders` table.
- SQL Editor:** Contains the same T-SQL code as the first screenshot:
 

```
123     END AS customer_type
124     FROM customer_orders;
125
126
127
128
129 •   SELECT * FROM customer_orders_cleaned LIMIT 10;
```
- Result Grid:** Displays the results of the query, showing 10 rows of customer data with different columns:
 

order_date	delivery_date	delivery_days	order_amount	city	signup_date	customer_tenure_days	order_category	rate
2022-07-18	2022-07-20	2	3662.38	PUNE	2021-11-22	1496	Low Value	1.9
2021-01-14	2021-01-21	7	10669.92	DELHI	2018-10-02	2643	Low Value	4.9
2024-02-05	2024-02-08	3	23175.88	BANGALORE	2021-09-29	1550	Medium Value	1.7
2022-07-01	2022-07-07	6	38255.82	MUMBAI	2020-07-22	1984	Medium Value	4.3
- Output Window:** Shows the execution log with the same entries as the first screenshot.

The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the schema structure, including the 'customer\_orders\_csv' schema and its 'customer\_orders' table.
- SQL Editor:** Displays the T-SQL code for creating a view named 'customer\_orders\_cleaned'.
- Result Grid:** Shows the output of the query 'SELECT \* FROM customer\_orders\_cleaned LIMIT 10'. The results are as follows:

order_date	delivery_days	order_amount	city	signup_date	customer_tenure_days	order_category	rating	customer_type
-07-20	2	3662.38	PUNE	2021-11-22	1496	Low Value	1.9	Loyal
-01-21	7	10669.92	DELHI	2018-10-02	2643	Low Value	4.9	Loyal
-02-08	3	23175.88	BANGALORE	2021-09-29	1550	Medium Value	1.7	Loyal
-07-07	6	38255.82	MUMBAI	2020-07-22	1984	Medium Value	4.3	Loyal

- Action Output:** Shows the history of actions taken during the session, including the creation of the view and the execution of the SELECT statement.

Explanation: Verified that all cleaned columns are correctly displayed.