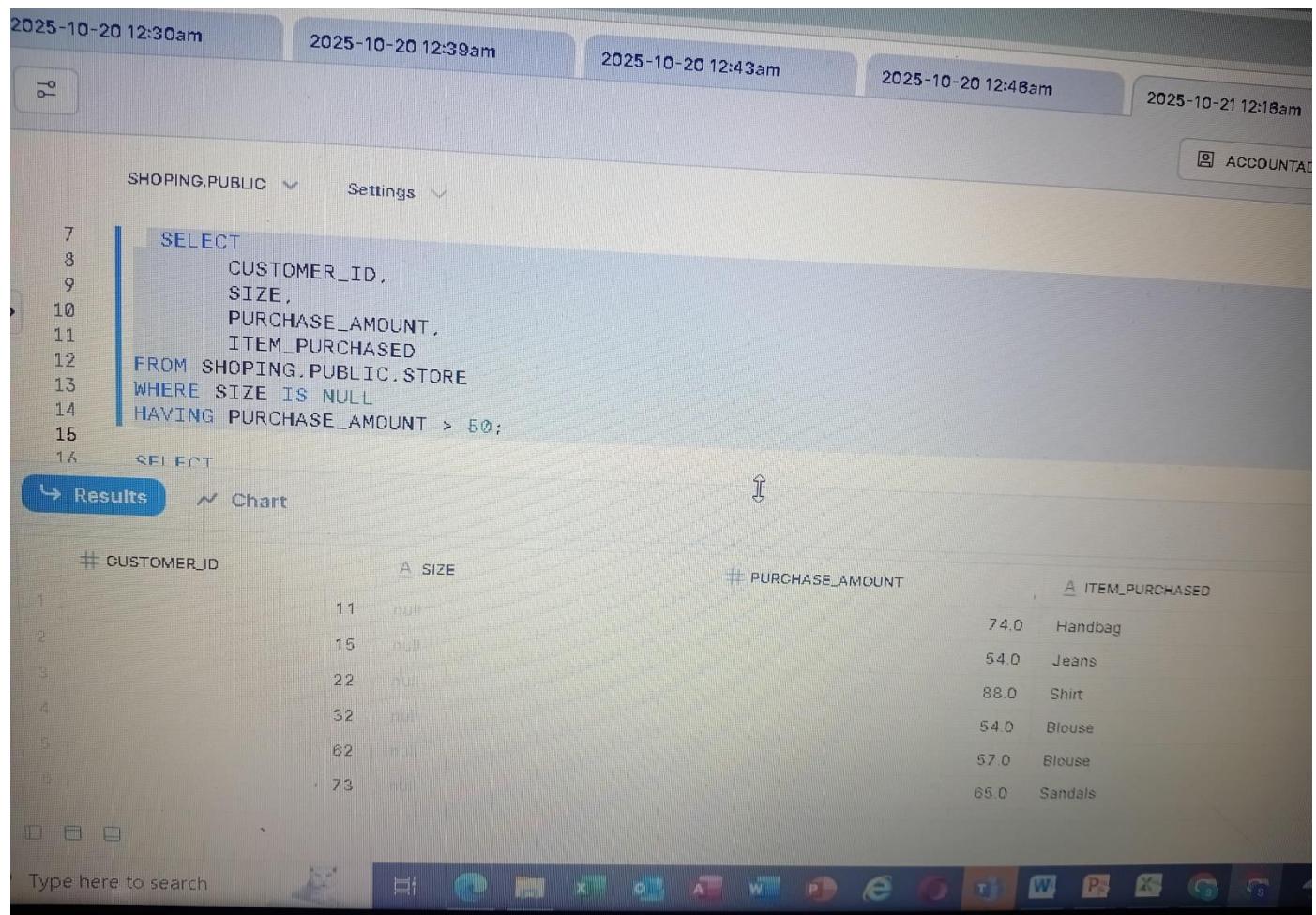


# Advanced SQL NULL FUNCTIONS

1 Find all records where size is missing and the purchase\_amount is greater than 50.

EXPECTED COLUMNS : customer\_ID, size, purchase\_amount, item\_purchased



The screenshot shows a SQL query being run in a database interface. The query is:

```
7  SELECT
8      CUSTOMER_ID,
9          SIZE,
10         PURCHASE_AMOUNT,
11            ITEM_PURCHASED
12   FROM SHOPING.PUBLIC.STORE
13 WHERE SIZE IS NULL
14 HAVING PURCHASE_AMOUNT > 50;
15
16 SELECT
```

The results table has four columns: CUSTOMER\_ID, SIZE, PURCHASE\_AMOUNT, and ITEM\_PURCHASED. The data is as follows:

CUSTOMER_ID	SIZE	PURCHASE_AMOUNT	ITEM_PURCHASED
11	null	74.0	Handbag
15	null	54.0	Jeans
22	null	88.0	Shirt
32	null	54.0	Blouse
62	null	57.0	Blouse
73	null	65.0	Sandals

List the total number of purchases grouped by Season, treating NULL values as 'Unknown Season'. Expected Columns: Season, Total Purchases

The screenshot shows a SQL query being run against a database named 'SHOPING.PUBLIC'. The query is:

```
15  
16 | SELECT  
17 |     COUNT(ITEM_PURCHASED) AS TOTAL_PURCHASES,  
18 |     COALESCE(SEASON, 'UNKNOWN') AS SEASON  
19 | FROM "SHOPING"."PUBLIC"."STORE"  
20 | GROUP BY COALESCE(SEASON, 'UNKNOWN');
```

The results are displayed in a table with two columns: 'TOTAL\_PURCHASES' and 'SEASON'. The data is:

TOTAL_PURCHASES	SEASON
58	Summer
26	UNKNOWN
71	Winter
66	Spring
50	Fall

Count how many customers used each Payment Method, treating NULLs as 'Not Provided'. Expected Columns:  
Payment Method, Customer Count

The screenshot shows a database interface with a query editor and a results table.

**Query Editor:**

```
21
22  SELECT
23      COALESCE(PAYMENT_METHOD, 'NOT PROVIDED') AS PAYMENT_METHOD,
24      COUNT(CUSTOMER_ID) AS TOTAL_CUSTOMERS
25  FROM SHOPING.PUBLIC.STORE
26  GROUP BY COALESCE(PAYMENT_METHOD, 'NOT PROVIDED');
```

**Results:**

PAYMENT_METHOD	TOTAL_CUSTOMERS
1 PayPal	51
2 NOT PROVIDED	30
3 Credit Card	44
4 Venmo	53
5 Debit Card	42
6 Bank Transfer	38
7 Cash	42

The results table lists payment methods and their corresponding customer counts. The columns are labeled PAYMENT\_METHOD and TOTAL\_CUSTOMERS. The data rows are numbered 1 through 7.

Show customers where Promo Code Used is NULL and Review Rating is below 3.0. Expected Columns: Customer ID, Promo Code Used, Review Rating, Item Purchased

The screenshot shows a Snowflake query interface with the following details:

Query Editor:

- Session: 2025-10-21 12:16am - Snowflake
- URL: app.snowflake.com/af-south-1.aws/du04063/w2p4su7kiMSM#query
- Timestamps: 2025-10-20 12:30am, 2025-10-20 12:39am, 2025-10-20 12:43am, 2025-10-20 12:48am, 2025-10-21 12:16am
- User: ACCOUNTADMIN

Query:

```
27
28 | SELECT
29 |     CUSTOMER_ID,
30 |     PROMO_CODE_USED,
31 |     REVIEW_RATING
32 | FROM SHOPING.PUBLIC.STORE
33 | WHERE PROMO_CODE_USED IS NULL
34 | HAVING REVIEW_RATING < 3.0;
35 |
```

Results:

CUSTOMER_ID	PROMO_CODE_USED	REVIEW_RATING
1	null	2.5
2	null	2.6
3	null	2.5
4	null	2.6
5	null	2.8
6	null	2.5
7	null	2.5

Group customers by Shipping Type, and return the average purchase\_amount, treating missing values as 0. Expected Columns: Shipping Type, Average purchase\_amount

The screenshot shows a Snowflake query interface with the following details:

Query URL: app.snowflake.com/af-south-1.aws/du04063/v2p4su7kiM5M#query

Timestamps at the top: 2025-10-20 12:30am, 2025-10-20 12:39am, 2025-10-20 12:43am, 2025-10-20 12:46am, 2025-10-21 12:18am.

User: ACCOUNTADMIN

Database: SHOPING.PUBLIC

Table: STORE

Code (Line numbers 35-41):

```
35
36 | SELECT
37 |   COALESCE(SHIPPING_TYPE, '0') AS SHIPPING_TYPE,
38 |   AVG(COALESCE(PURCHASE_AMOUNT, '0')) AS AVERAGE_PURCHASE_AMOUNT
39 | FROM SHOPING.PUBLIC.STORE
40 | GROUP BY COALESCE(SHIPPING_TYPE, '0');
```

Results tab selected.

SHIPPING_TYPE	AVERAGE_PURCHASE_AMOUNT
Free Shipping	50.2142857
Store Pickup	55.3333333
Express	53.4545455
Standard	47.6666667
Next Day Air	54.8866667
0	52.7037037
2-Day Shipping	51.5576923

Type here to search

Display the number of purchases per Location only for those with more than 5 purchases and no NULL Payment Method. Expected Columns: Location, Total Purchases

The screenshot shows a Snowflake query interface with the following details:

URL: app.snowflake.com/af-south-1.aws/du04063/w2p4su7kiM5M#query

Timeline: 5-10-20 12:30am, 2025-10-20 12:39am, 2025-10-20 12:43am, 2025-10-20 12:48am, 2025-10-21 12:18am

User: ACCOUNTADMIN

Database: SHOPING.PUBLIC

Table: STORE

Query:

```
35
36  SELECT
37      COALESCE(SHIPPING_TYPE, '0') AS SHIPPING_TYPE,
38      AVG(COALESCE(PURCHASE_AMOUNT, '0')) AS AVERAGE_PURCHASE_AMOUNT
39  FROM SHOPING.PUBLIC.STORE
40  GROUP BY COALESCE(SHIPPING_TYPE, '0');
```

Results:

A SHIPPING_TYPE	# AVERAGE_PURCHASE_AMOUNT
1 Free Shipping	50.2142857
2 Store Pickup	55.3333333
3 Express	53.4545455
4 Standard	47.6666667
5 Next Day Air	54.8888887
6 0	52.7037037
7 2-Day Shipping	51.5576923

Create a column Spender Category that classifies customers using CASE: 'High' if amount > 80, 'Medium' if BETWEEN 50 AND 80, 'Low' otherwise. Replace NULLs in purchase\_amount with 0. Expected Columns: Customer ID, purchase\_amount, Spender Category

10-20 12:30am	2025-10-20 12:39am	2025-10-20 12:43am	2025-10-20 12:46am	2025-10-21 12:18am																				
				ACCOUNTADMIN																				
SHOPING.PUBLIC				Settings																				
+1																								
41																								
42																								
43																								
44																								
45																								
46																								
SELECT LOCATION, COUNT(ITEM_PURCHASED) AS ITEM_PURCHASED, FROM SHOPING.PUBLIC.STORE WHERE PAYMENT_METHOD IS NOT NULL																								
<a href="#">Results</a> <a href="#">Chart</a>																								
<table><thead><tr><th>LOCATION</th><th>ITEM_PURCHASED</th></tr></thead><tbody><tr><td>null</td><td>20</td></tr><tr><td>Maine</td><td>36</td></tr><tr><td>Oregon</td><td>26</td></tr><tr><td>Kentucky</td><td>28</td></tr><tr><td>Florida</td><td>31</td></tr><tr><td>Massachusetts</td><td>31</td></tr><tr><td>Texas</td><td>21</td></tr><tr><td>Rhode Island</td><td>27</td></tr><tr><td>New York</td><td>28</td></tr></tbody></table>					LOCATION	ITEM_PURCHASED	null	20	Maine	36	Oregon	26	Kentucky	28	Florida	31	Massachusetts	31	Texas	21	Rhode Island	27	New York	28
LOCATION	ITEM_PURCHASED																							
null	20																							
Maine	36																							
Oregon	26																							
Kentucky	28																							
Florida	31																							
Massachusetts	31																							
Texas	21																							
Rhode Island	27																							
New York	28																							
Type here to search																								

Create a column Spender Category that classifies customers using CASE: 'High' if amount > 80, 'Medium' if BETWEEN 50 AND 80, 'Low' otherwise. Replace NULLs in purchase\_amount with 0.Expected Columns: Customer ID, purchase\_amount, Spender Category

The screenshot shows a database interface with a query editor and a results viewer.

**Query Editor:**

```
50
51  SELECT
52    CASE
53      WHEN PURCHASE_AMOUNT IS NULL THEN '0'
54      WHEN PURCHASE_AMOUNT > 80 THEN 'HIGH'
55      WHEN PURCHASE_AMOUNT BETWEEN 50 AND 80 THEN 'MEDIUM'
56      WHEN PURCHASE_AMOUNT < 50 THEN 'LOW'
57      ELSE 'OTHERWISE'
58    END AS SPENDER_CATEGORY
59
60  FROM SHOPING.PUBLIC.STORE;
61
```

**Results View:**

SPENDER_CATEGORY
LOW
LOW
LOW
LOW
MEDIUM

The results show five rows, each containing the value 'LOW' except for the last row which contains 'MEDIUM'.

Find customers who have no Previous Purchases value but whose Color is not NULL.Expected Columns:  
Customer ID, Color, Previous Purchases

The screenshot shows a database interface with a query editor and a results table.

**Query Editor:**

```
SELECT
    CUSTOMER_ID,
    COLOR,
    PREVIOUS_PURCHASES
FROM SHOPING.PUBLIC.STORE
WHERE PREVIOUS_PURCHASES IS NULL AND COLOR IS NOT NULL;
```

**Results Table:**

#	CUSTOMER_ID	A COLOR	# PREVIOUS_PURCHASES
8		Green	null
21		Yellow	null
25		White	null
37		Maroon	null
40		Gray	null
43		Black	null
44		Green	null
70		Yellow	

**Interface Elements:**

- Top right: Date: 2025-10-21 12:18am, User: ACCOUNTADMIN
- Left sidebar: SHOPING.PUBLIC, Settings
- Bottom: Type here to search, Taskbar icons (e.g., File Explorer, Microsoft Edge, File Manager, etc.)

Group records by frequency of purchases and show the total amount spent per group, treating NULL frequencies as 'Unknown'. Expected Columns : Frequency of Purchases, Total purchases amount

The screenshot shows a database interface with a query editor and a results table.

**Query Editor:**

```
70 | SELECT
71 |     COALESCE(FREQUENCY_OF_PURCHASE, 'UNKNOWN') AS FREQUENCY_OF_PURCHASE,
72 |     SUM(PURCHASE_AMOUNT) AS TOTAL_PURCHASE_AMOUNT
73 | FROM SHOPING.PUBLIC.STORE
74 | GROUP BY ALL;
```

**Results Table:**

FREQUENCY_OF_PURCHASE	TOTAL_PURCHASE_AMOUNT
Annually	1765.0
Monthly	1780.0
UNKNOWN	1518.0
Bi-Weekly	2099.0
Quarterly	2541.0
Every 3 Months	1749.0
Weekly	2184.0
Fortnightly	2033.0

**Bottom Bar:**

- Type here to search
- Icons for various applications: File, Edge, Mail, Excel, OneDrive, Access, Word, Power BI, Edge, OneDrive, Trello, Word, Power BI, OneDrive, Google Sheets, Rain

Display a list of all category values with the number of times each was purchased, excluding rows where categories is NULL. Expected columns: category, Total purchases

The screenshot shows a database interface with a query editor and a results table.

**Query Editor:**

```
75  
76 SELECT  
77 COALESCE(CATEGORY, '0') AS CATEGORY,  
78 COUNT(ITEM_PURCHASED) AS TOTAL_PURCHASES  
79 FROM SHOPING.PUBLIC.STORE  
80 GROUP BY ALL;  
81
```

**Results Table:**

CATEGORY	TOTAL_PURCHASES
Footwear	63
Outerwear	56
Clothing	55
Accessories	68
0	29

**System Status:**

- 2025-10-20 12:39am
- 2025-10-20 12:43am
- 2025-10-20 12:48am
- 2025-10-21 12:18am

**User Information:**

- ACCOUNTADMIN

**Taskbar:**

- Type here to search
- File
- Recycle Bin
- OneDrive
- OneNote
- PowerPoint
- Edge
- OneDrive
- Teams
- Word
- PowerPoint
- Kinect
- Cloud
- Rain

Return the top 5 location with highest total purchases amount, replacing NULL in amount with 0. Expected columns: locations, Total purchases amount.

The screenshot shows a data analysis interface with a query editor and a results table. The query is:

```
SELECT
    LOCATION,
    SUM(COALESCE(PURCHASE_AMOUNT, '0')) AS PURCHASE_AMOUNT
FROM SHOPING.PUBLIC.STORE
GROUP BY LOCATION
```

The results table displays the following data:

LOCATION	PURCHASE_AMOUNT
Maine	2294.0
Florida	1980.0
Massachusetts	1899.0
Rhode Island	1876.0
Kentucky	1798.0

Group customer by Gender and Size and count how many entries have a NULL in color

Expected columns: Gender, Size and count color null

The screenshot shows a Microsoft Power BI environment. At the top, there's a navigation bar with 'SHOPING.PUBLIC' and 'Settings'. On the right, it displays the date and time as '2025-10-21 12:16am' and the user as 'ACCOUNTADMIN'. Below the navigation bar is a code editor window containing the following SQL query:

```
90  
91 SELECT  
92     COUNT(COLOR) AS NULL_COLOR,  
93     GENDER,  
94     SIZE  
95 FROM SHOPING.PUBLIC.STORE  
96 WHERE COLOR IS NULL  
97 GROUP BY GENDER, SIZE;
```

Below the code editor is a results grid. The columns are labeled '# NULL\_COLOR', 'GENDER', and 'SIZE'. The data shows five rows of male customers with null colors, grouped by size S, L, and XL. The results grid has a header row and five data rows.

# NULL_COLOR	GENDER	SIZE
0	Male	S
0	Male	L
0	Male	null
0	Male	XL
0	Male	M

At the bottom of the screen, there's a taskbar with various application icons, including Microsoft Edge, File Explorer, and other Office applications like Word, Excel, and PowerPoint. A search bar at the bottom left says 'Type here to search'.

Identify all the purchased where more than 3 purchases had null shipping type. Expected columns: item purchased, null shipping type count null.

The screenshot shows a database interface with a query editor and a results table. The query is:

```
99 SELECT
100     ITEM_PURCHASED,
101     COUNT(SHIPPING_TYPE) AS NULL_SHIPPING_COUNT
102    FROM SHOPING.PUBLIC.STORE
103   WHERE SHIPPING_TYPE IS NULL
104 GROUP BY ITEM_PURCHASED;
```

The results table has two columns: ITEM\_PURCHASED and NULL\_SHIPPING\_COUNT. The data is:

ITEM_PURCHASED	NULL_SHIPPING_COUNT
Shoes	0
Shirt	0
null	0

At the bottom, there is a search bar and a taskbar with various application icons.

Show a count of how many customers per payment method have null review rating. Expected columns: payment method , missing review rating count.

The screenshot shows a database interface with a query editor and a results table. The query editor displays the following SQL code:

```
--  
115  
116    SELECT  
117        CATEGORY,  
118        AVG(COALESCE(REVIEW_RATING, '0')) AS REVIEW_RATING  
119    FROM SHOPING.PUBLIC.STORE  
120    GROUP BY ALL  
121    HAVING AVG(COALESCE(REVIEW_RATING, '0')) > 3.0;  
122
```

The results table has two columns: 'CATEGORY' and 'REVIEW\_RATING'. The data is as follows:

CATEGORY	REVIEW_RATING
Footwear	3.1871429
Outerwear	3.3083333
Clothing	3.0152542
Accessories	3.2551282
null	3.5000000

At the bottom of the screen, there is a taskbar with various icons and a search bar.

List all colors that are missing (Null) in at least 2 rows and the average age of customers for those rows.

The screenshot shows a Snowflake query editor interface. At the top, there are tabs for 'Recovery Options' and 'Activate your Snowflake account'. A session bar indicates '2025-10-21 12:16am - Snowflake'. Below the header, the URL is 'app.snowflake.com/af-south-1.aws/du04063/w2p4su7kiM5M#query'. The timeline shows several recent sessions: '2025-10-20 12:30am', '2025-10-20 12:39am', '2025-10-20 12:43am', '2025-10-20 12:48am', and '2025-10-21 12:16am'. On the right, there's an 'ACCOUNTADMIN' role indicator. The main area displays a query:

```
123 SELECT
124   COLOR,
125   AVG(AGE) AS AVERAGE_AGE
126 FROM SHOPING.PUBLIC.STORE
127 WHERE COLOR IS NULL
```

Below the query, there are two buttons: 'Results' (highlighted in blue) and 'Chart'. The results section shows a single row:

COLOR	AVERAGE_AGE
null	47.8461538

At the bottom of the screen, there's a search bar with the placeholder 'Type here to search' and a taskbar with various application icons.

Use a case to create a column Delivery speed ‘fast’ if shipping is ‘Express’ or ‘next day air’, slow if ‘standard’

Other for all else including Null. Then count how many customers fall into each category.

Expected column: delivery speed , customer count

2025-10-20 12:43am 2025-10-20 12:46am 2025-10-21 12:18am

ACCOUNTAD

SHOPING.PUBLIC Settings

```
130
131 SELECT
132     COUNT(CUSTOMER_ID) AS CUSTOMER_COUNT,
133
134     CASE
135         WHEN SHIPPING_TYPE IN ('EXPRESS', 'NEXT DAY AIR') THEN 'FAST'
136         WHEN SHIPPING_TYPE = 'STANDARD' THEN 'SLOW'
137         ELSE 'OTHER'
138     END AS SPEED_DELIVERY
139     FROM SHOPING.PUBLIC.STORE
140     GROUP BY ALL;
```

141

↳ Results ⚡ Chart

↑ ↓

CUSTOMER_COUNT	SPEED_DELIVERY
1	300 OTHER

□ □ □

Type here to search



Find customers whose purchase amount is Null and whose promo code used is 'yes'

Expected column: customerID , Purchase amount ,  
Promo code used.

The screenshot shows a database interface with a query editor and a results table. The query is:

```
SELECT
    CUSTOMER_ID,
    PURCHASE_AMOUNT,
    PROMO_CODE_USED
FROM SHOPING.PUBLIC.STORE
WHERE PURCHASE_AMOUNT IS NULL AND PROMO_CODE_USED = 'YES';
```

The results table has three columns: CUSTOMER\_ID, PURCHASE\_AMOUNT, and PROMO\_CODE\_USED. The data is as follows:

CUSTOMER_ID	PURCHASE_AMOUNT	PROMO_CODE_USED
13	null	TRUE
30	null	TRUE
78	null	TRUE
95	null	TRUE
124	null	TRUE
129	null	TRUE
130	null	TRUE
130	100	TRUE

The taskbar at the bottom of the screen shows various application icons.

Group by location and show and show maximum previous purchases , replacing Nulls with 0, only where average rating is above 4.0

Expected: Location ,max previous purchases, average review rating.

The screenshot shows a database interface with a query editor and a results viewer.

**Query Editor:**

```
2025-10-20 12:43am 2025-10-20 12:48am 2025-10-21 12:18am
ACCOUNTADMIN

SHOPING.PUBLIC Settings

WHERE PURCHASE_AMOUNT IS NULL AND PROMO_CODE_USED = YES;

SELECT
    AVG(REVIEW_RATING) AS AVERAGE_RATING,
    LOCATION,
    MAX(COALESCE(PREVIOUS_PURCHASES, '0')) AS MAX_PREVIOUS_PURCHASES
FROM SHOPING.PUBLIC.STORE
GROUP BY LOCATION
HAVING AVG(REVIEW_RATING) > 4.0;
```

**Results View:**

AVERAGE_RATING	LOCATION	MAX_PREVIOUS_PURCHASES
Query produced no results		

Show customers who have a Null shipping type but made a purchase in the range of 30 to 70 USD.

Expected: CustomerID , Shipping type , Purchase amount, item purchased.

The screenshot shows a database interface with a query editor and a results table. The query is:

```
158 | SELECT
159 |     CUSTOMER_ID,
160 |     SHIPPING_TYPE,
161 |     PURCHASE_AMOUNT,
162 |     ITEM_PURCHASED
163 | FROM SHOPING.PUBLIC.STORE
164 | WHERE SHIPPING_TYPE IS NULL AND PURCHASE_AMOUNT BETWEEN 30 AND 70;
```

The results table has four columns: CUSTOMER\_ID, SHIPPING\_TYPE, PURCHASE\_AMOUNT, and ITEM\_PURCHASED. The data is as follows:

CUSTOMER_ID	SHIPPING_TYPE	PURCHASE_AMOUNT	ITEM_PURCHASED
15	null	54.0	Jeans
105	null	43.0	Shirt
141	null	37.0	Shorts
196	null	68.0	Coat
213	null	36.0	Shirt
235	null	38.0	Sandals
293	null	35.0	...

